Exploiting Latent Features of Text and Graphs

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Computer Science

---

by
Justin Sybrandt
May 2020

Accepted by:
Dr. Ilya Safro, Committee Chair
Dr. Amy Apon
Dr. Sez Atamturktur
Dr. Brian Dean
Dr. Alexander Herzog

Abstract

As the size and scope of online data continues to grow, new machine learning techniques become necessary to best capitalize on the wealth of available information. However, the models that help convert data into knowledge require nontrivial processes to make sense of large collections of text and massive online graphs. In both scenarios, modern machine learning pipelines produce embeddings — semantically rich vectors of latent features — to convert human constructs for machine understanding. In this dissertation we focus on information available within biomedical science, including human-written abstracts of scientific papers, as well as machine-generated graphs of biomedical entity relationships. We present the Moliere system, and our method for identifying new discoveries through the use of natural language processing and graph mining algorithms. We propose heuristically-based ranking criteria to augment Moliere, and leverage this ranking to identify a new gene-treatment target for HIV-associated Neurodegenerative Disorders. We additionally focus on the latent features of graphs, and propose a new bipartite graph embedding technique. Using our graph embedding, we advance the state-of-the-art in hypergraph partitioning quality. Having newfound intuition of graph embeddings, we present Agatha, a deep-learning approach to hypothesis generation. This system learns a data-driven ranking criteria derived from the embeddings of our large proposed biomedical semantic graph. To produce human-readable results, we additionally propose CBAG,

a technique for conditional biomedical abstract generation.

Dedication

To Emma Cinatl,

You're the most loving, caring, affectionate, and supportive person I've ever encountered. I couldn't have asked for a better partner to persevere with through the last four years. Thank you for reminding me to see the beauty in the ordinary, for teaching me to enjoy a good hike, for encouraging me when times were tough, and for celebrating with me now that we've made it through.

With love,

Justin Sybrandt

Acknowledgments

I would like to thank Ilya Safro, as well as the members of my committee, for guidance over these last four years. I am grateful to have had mentors so amenable and accomplished. I would also like to acknowledge my collaborators: Michael Shtutman on Chapters 5, 6, and 8, Angelo Carrabba on Chapter 7, Ruslan Shaydulin on Chapter 4, and Ilya Tyagin on Chapter 8.

Furthermore, I would like to thank Lisa and Larry Sybrandt, my parents, for raising me to give my best to everything I do. Thank you to my siblings, Jennifer and Joseph Sybrandt, for keeping me grounded. To Marilyn and Darrel Apps, my grandparents, for providing ever-present encouragement and enthusiasm. To my grandmother, Kayleen Sybrandt, for mailing countless comic strips full of three-toed sloths and road-crossing chickens, for long conversations full of Seinfeld references, and for never-wavering love full of optimism.

Lastly, I would like to thank all of my friends and colleagues, of which there are too many to list here. Your contributions, both social and scholarly, were crucial. I would specifically like to thank everyone who I had the pleasure of working alongside in our lab group: Angelo Carrabba, Varsh Chauhan, Joey Liu, Korey Palmer, Zirou Qiu, Ehsan Sadrfaridpour, Ruslan Shaydulin, Ilya Tyagin, and Hayato Ushijima-Mwesigwa. What started as a beige windowless office grew into a home away from home.

Funding

Justin Sybrandt has received funding through both the US Department of Education through the GAANN DAISE fellowship program, as well through the National Science Foundation, Award #1633608, through the NRT RIES fellowship program. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or the US Department of Education.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Data on the internet grows exponentially, and over 80% of this data is unstructured [177]. By "unstructured," we refer to data that typically requires human understanding to make use of. For instance, the written posts published to social media are full of context, humor, inside jokes, and personal information. Similarly, the social networks that emerge on these websites represent an online mapping of the complex, messy, and unpredictable set of human relationships. Human users often struggle to understand these nuances, and many times the creator's original intent is lost in translation when encoded online. How then can we expect a machine to be able to leverage unstructured data algorithmically?

The modern way to handle unstructured data for the purpose of machine learning is to first identify an *embedding*. Embeddings are learned sets of latent variables that describe the underlying qualities of a dataset that could not be observed directly. One clear example of a latent feature in the domain of natural language is that of sentiment. Natural language's only observable information is the letters and words that make up a particular writing sample. However, human readers can often easily intuit whether the author feels positively or negatively about the discussed subject. This

understanding of sentiment *emerges* from the text, because most authors convey this sort of emotion through subtle or indirect means. Therefore, a high quality algorithmic analysis of sentiment needs to both understand the meaning behind each word, but also the *latent* reasons that the observed words were selected and particularly ordered.

In this thesis we explore latent features of text and graphs, with a focus on understanding the content present in biomedical scientific literature. This analysis includes a new proposed bipartite graph embedding (Chapter 3), and technique that uses graph embeddings as a heuristic for better solving the NP-hard problem of hypergraph partitioning (Chapter 4). Then, we present significant advances in biomedical hypothesis generation, starting with the proposed Moliere system (Chapter 5). To quantify Moliere system performance and improve system usability, we present a new large-scale validation technique for hypothesis generation (Chapter 6), and apply that technique to determine whether full-text papers are worth their associated computational overhead (Chapter 7). Then, using more modern text and graph embedding techniques, we present Agatha, a deep-learning approach to hypothesis generation (Chapter 8). Finally, we propose a new technique for conditional biomedical abstract generation, which can enable hypothesis generation techniques to output human-understandable summaries of new findings (Chapter 9).

1.1 Research Objectives

This dissertation explores the following questions pertaining to both text and graph embeddings:

- Do traditional graph embedding techniques erode key variance of bipartite graphs? (Chapter 3)

- Can the global structural features learned by bipartite graph embedding improve upon existing node-similarity measures in the problem of multilevel hypergraph partitioning? (Chapter 4)
- Can text embeddings help identify useful papers related to user-supplied hypothesis generation queries, and can topic modeling provide hypothesis-related insights to domain scientists? (Chapter 5)
- Can we use text embeddings and topic models to automate the analysis of generated hypotheses? (Chapter 6)
- Is the content present in scientific abstracts sufficient for hypothesis generation, or is it worth processing full-text papers? (Chapter 7)
- In what ways can large scale graph embedding and deep learning techniques improve hypothesis generation? (Chapter 8)
- Can we model scientific language such that we can generate new language discussing user-defined keywords? (Chapter 9).

1.2 Contributions in Summary

Each of the seven research questions posed above is explored in depth in the following chapters. We summarize the key findings of each here.

1.2.1 Bipartite Graph Embedding

We present First- and High-Order Bipartite Embeddings (FOBE and HOBE). These techniques produce embeddings for the nodes of each half of a bipartite graph such that key variance within each half can be better encoded. This process involves

decomposing edges of the original graph into sets of node interactions that occur within each half. Using these techniques, we demonstrate competitive performance on the task of link prediction, as well as improved performance for recommendation.

1.2.2 Embedding-based Coarsening for Hypergraph Partitioning

A hypergraph is a generalization of a classical graph wherein "hyperedges" may contain any subset of nodes. There is a one-to-one relationship between hypergraphs and bipartite graphs, which allows us to apply bipartite graph embedding techniques like FOBE and HOBE in a new context. The problem of hypergraph partitioning is to divide nodes into similarly-sizes subsets in order to minimize the number of hyperedges that span more than one subset. This problem is NP-Hard, so most modern solvers use the multilevel algorithm that solves an similarly-structured small problem who's solution can be interpolated and refined onto the input hypergraph. In order to find these sub-problems, solvers iteratively *coarsen* the input hypergraph by merging similar nodes and hyperedges. The coarsening process has an outsized impact on overall solution quality, and is primarily determined by node-similarity measures. We use bipartite embeddings of the original hypergraph to improve this node-similarly measure through a process called *embedding-based coarsening*, which we implement in two modern partitioners. As a result, we demonstrate an average improvement of approximately 10% for low partition counts across a large benchmark. In some cases, this improvement is as significant as 400%.

1.2.3 Hypothesis Generation with Moliere

We propose Moliere, an automatic biomedical hypothesis generation system, which compiles all publicly available biomedical abstracts into a large semantic network. Edges within this network indicate similarity between elements. For instance, two keywords with similar biomedical text embeddings are more likely to receive a strongly weighted edge. Using this network we can perform shortest-path queries between two user-specific elements of interest. The shortest-path identifies a region within the network that is more likely to contain relevant information. From this region, we extract a subset of biomedical abstracts on which we perform topic modeling. These topics can be analyzed by biomedical experts to determine the quality and strength of potential connections bridging the queried entities. Using this system trained on historical information, we rediscover a number of more recent new connections.

1.2.4 Validation of Hypothesis Generation Systems

The originally proposed Moliere analysis of topic models requires significant human oversight. As a result, the validation of hypothesis generation systems like Moliere is slow, biased, and does not scale beyond a handful of queries. In fact, no prior large scale validation of hypothesis generation systems has the necessary speed, breadth, or scalability needed to evaluate Moliere. To address these concerns, we propose a new validation technique based on the ranking process performed by scientists during the drug discovery process. This analysis requires that a system rank a set of recently published connections with a set of randomly sampled negative connections. From there, a hypothesis generation system is scored by its ability to rank published results above noise. To perform this ranking, however, we propose a number of

embedding- and topic-model-based heuristic criteria. We not only demonstrate that our proposed ranking criteria results in high scores in our proposed benchmark, but also that these scores can be used in real-world applications to find gene-treatment targets. Specifically, we identify the inhibition of DDX3 as a likely treatment for HIV-associated neurodegenerative disease, which is originally discovered by Moliere, and later confirmed in a laboratory experiment.

1.2.5 Full Text Papers for Hypothesis Generation

Using our validation method, we can ask questions regarding hypothesis generation system parameters. Specifically, we can retrain various instances of Moliere and perform our large-scale variation to quantify the differences in performance given different inputs and hyperparameters. The main question we answer is whether there is an added benefit to using full-text papers, as opposed to shorter abstract summaries of the same work. Through our analysis, we demonstrate that full-text papers inside Moliere can lead to a 10% improvement in quality, but increases runtime by forty-five times.

1.2.6 Deep Learning Hypothesis Generation with Agatha

In order to address a number of assumptions present in Moliere we present Agatha, a deep-learning approach to hypothesis generation that begins by constructing and embedding a large semantic graph built around *sentences* (as opposed to abstracts). Then, using a *transformer encoder* model, we learn a ranking criteria for hypotheses directly from embeddings. Because the Agatha system does not rely on any heuristically-backed ranking criteria we observe a substantial increase in result quality when validating on the same hypothesis set as Moliere. Additionally, because

we no longer perform expensive shortest-path queries, the Agatha system can process many hundreds of hypotheses in the same amount of time it takes Moliere to process just one. Increased query speed enables new many-to-many queries, which we evaluate by exploring multiple popular medical sub-domains. We identify that within particular hypothesis types, such as hypotheses between genes and cell functions, that Agatha can recommend new research directions with a top-10 precision of over 0.5. Therefore, the Agatha system is capable of recommending fruitful new research ideas *without* explicit input from human scientists.

1.2.7 Conditional Biomedical Language Generation

A key problem with existing hypothesis generation systems is a lack of interpretable output. While numeric scores are valuable for large-scale queries, biomedical researchers need rich information at the small-scale in order to act on specific automatically generated hypotheses. To begin to address this limitation, we turn to conditional language modeling, a process wherein we can generate new text provided a user-specified prior. For the purpose of this analysis we use author-supplied metadata as conditional data for generating the body of biomedical abstracts. However, future applications could supply any embedding, such as the hidden units of the Agatha model, to condition text generation. Using this technique we demonstrate an ability to generate full sensible biomedical texts, recover nontrivial in-domain keywords, and outperform more general-purpose language models. Additionally, we demonstrate that the model is very sensitive to the conditioning metadata, which enables useful controls over generated text.

Chapter 2

Background

This chapter presents fundamental prior work needed to understand the contributions in this thesis. Broadly, these topics encompass text embeddings, graph embeddings, and hypothesis generation. However, specific prior works that apply to particular chapters appear in those chapters independent background sections.

2.1 Latent Features

In order to understand the intuitions behind text and graph embeddings, one should first explore more historical instances of latent feature analysis. Latent features are those that describe the underlying distribution of an observable variable. In the introduction, we explain that *sentiment* in the domain of natural language processing is an easily comprehensible example of a latent feature. Sentiment is considered latent because the positive or negative view of the author is rarely encoded explicitly, but rather more subtly alters the choice and order of words. Therefore in this casein this case the distribution and ordering of words are the observable variable, and sentiment is the unobservable latent feature.

However, latent feature analysis is far more fundamental to the process of machine learning and data mining than the sentiment example may lead a reader to believe. For instance, when one applies principal-component analysis (PCA) [250] in order to visualize a dataset, they are also exploring latent features. In this case, each of the two or three principal components indicates a direction of maximal variance of the dataset. While these vectors often do not have as clear of an interpretation as sentiment does for text, their analysis can lead to useful conclusions about a dataset.

Matrix factorization leads to another form of latent feature analysis. One famous example is the use of non-negative matrix factorization [134] for the analysis of human faces [133]. In this case, square grayscale images of faces are decomposed into latent features through the matrix factorization process. The observable variables, pixels within each image as represented by a number between zero and one, begin in a large N by M matrix. Each of N rows represents a different pixel location, and each of M columns represents a different image of a face. Non-negative matrix factorization decomposes the large N by M matrix into two smaller-rank matrices of sizes N by K and K by M. In this manner, we identify K latent features, both in the domain of pixels (the $N \times K$ matrix), and in the domain of images (the $K \times M$ matrix). What Lee and Seung find is that the first set of features indicates the primary components of each face, such as eyes, the node, the mouth, and various shadows that each could cast on the other. Meanwhile, the second set of features indicates the makeup of a specific face in terms of the components. These latent facial features arise from a simple matrix of pixels intuitively, because they help describe the distribution of lightness values across a grayscale image. Importantly though, the process of matrix factorization is unaware that it is processing faces. These latent properties are discovered from the very simple set of grayscale values.

Back in the domain of natural language processing, topic modeling through

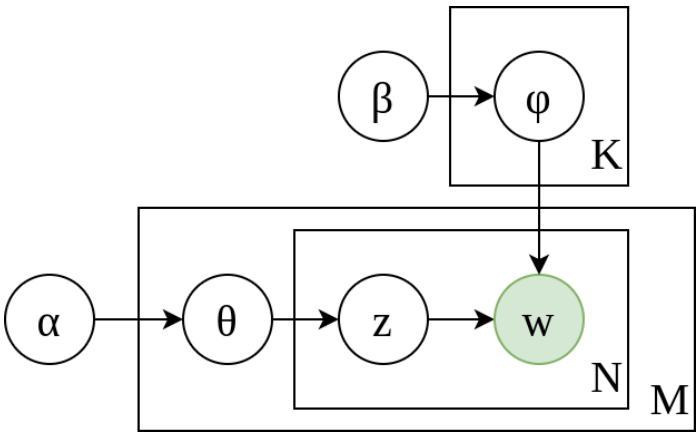


Figure 2.1: A summary of the LDA topic model.

Latent Dirichlet Allocation (LDA) is a common process to uncover the latent distribution of topics across a corpus of words. Proposed by Blei [31], and depicted in Figure 2.1. This model fits a latent distribution of topics over the observed words per document. Here, the words within each document w is the only variable that can be directly observed. However, this model assumes that there exist K topics, each defined as a probability distribution over words, and that each document can be described as a mixture θ over these topics. In practice, this model learns topic mixtures that tend to comport with an intuitive clustering of words with respect to the corpus at hand. One common example is to present the word probabilities of topics resent in New York Times articles. Unsurprisingly, the high-probability words in each distribution look surprisingly like the papers subsection titles. In this case, we can recover the latent distribution of articles, which follow the key topics over which the New York Times reports, simply by studying word co-occurrence patterns.

A key limitation of LDA, and other similar models, is the reliance on hand-crafted statistical models to understand an underlying distribution. In the case of LDA, we have to take for granted that words in our corpus actually do follow a topic distribution that maps into a Dirichlet distribution for instance. In contrast to these

crafted approaches, there are a range of neural network methods for identifying latent features that make fewer assumptions about the underlying distribution of data. These models instead learn to find sets of latent features given models of many parameters, sometimes millions or billions, in any way that best optimizes some objective. The clearest view into this sort of model is the auto encoder, however one should note that *every* neural network model (really, any Bayesian statistical model) includes the discovery of latent features as one of its most elementary operations.

The auto-encoder is a simple unsupervised neural network architecture wherein the model must reconstruct its input through a low-rank approximation [146]. A simple example of this model is defined as followed, and might optimize the following objective function:

$$\mathcal{L}(x) = \frac{1}{n}\sum_{i=1}^{n}\left(x_i - \mathcal{H}(x)_i\right)^2 \tag{2.1}$$

Here $\mathcal{L}(x)$ corresponds to the loss associated with training example $x \in \mathbb{R}^n$, and $\mathcal{H}$ represents the parameterized auto encoder. The loss here is mean-squared error, and while this is a common case, we only present this particular loss for the purpose of following an easy example. A simple "one-layer" definition of H would be:

$$\begin{gathered} H(x) = \max(0, x\theta)\theta' \\ \text{where } \theta \in \mathbb{R}^{n\times k} \\ \text{and } \theta' \in \mathbb{R}^{k\times n} \end{gathered} \tag{2.2}$$

Here θ and θ' are learned parameters that are updated when minimizing $\mathcal{L}$, often through gradient decent. The first set of parameters projects the input data into a low-rank approximation, and in doing so must combine various signals present in the input in order to preserve the overall structure of x. The second set of parameters, θ',

then expands the k-dimensional approximation of x back to the input domain. The loss is minimized when the difference between the input and recreated version of x is minimal. Intuitively, therefore, the set of k underlying features that correspond to x should be the latent variables associated with the distribution of training data.

Using this intuition, one can see how all neural network models perform embeddings. For instance, transfer learning relies on this property in image processing [247]. Transfer learning is the process of applying weights that were learned in one context to a different problem in a similar domain. For instance, one might want to use weights from a deep-learning model trained to classify images in the popular ImageNet dataset [60] in order to help perform object detection. In practice, one would construct a model that has the same initial neural-network architecture as the first layers of a pretrained network, and then has a new special purposed "head" that is selected to model a new particular task. While the weights in the head are assigned randomly, the weights in the rest of the model can be copied from the pretrained model. Then, after freezing the pretrained component, the head can be "fine-tuned" for the new task, often with increased performance as opposed to training a new model whole cloth.

The reason transfer learning works is because the pretrained models have picked up on real-world latent variables associated with the problem at hand. For instance, early layers might quantify broad shapes and colors, while intermediate layers might represent more fine-grained patterns and shadings. While the interpretation and analysis of these weights remains an open problem — many ImageNet model contain many millions of parameters — these discovered latent features can be directly applied to new contexts, especially those with less available data.

While all neural network models identify latent features, there are some that are specifically designed to find low-rank representations of specific kinds of high-

dimensional data. These *embedding* models typically act as a special pre-processing phase to begin more domain-specific neural network problems. As the title of this dissertation suggests, we will focus on embedding techniques associated with text and graphs.

2.2 Text Embedding

The most straightforward representation of text is colloquially known as the "one-hot" vector. A each word in a vocabulary of size V is ordered and assigned a V-dimensional representation. The i^{th} word's representation consists of a value of one in the i^{th} position, and zeros elsewhere. Thus, the name "one-hot." We refer to this approach as a "representation" (not an embedding) as no latent information is captured by these vectors. All words are equally dissimilar, and there are no insights to be gained from studding the V-dimensional space.

While the dimensionality and sparsity of one-hot embeddings are challenges for many machine learning models, they did facilitate a significant amount of early text mining [156, 56]. Simple calculations, such as term-frequency inverse-document-frequency (TF-IDF) [175], expose some nontrivial properties from one-hot embeddings, such as the relative importance of words within a corpus. Other early text representation methods include using hand-crafted features [70], or using hashing to find lower-rank representations [74]. More recent work uses recurrent neural networks to identify relational trends within text [154], however more complex models are able to uncover latent spaces rich with semantic meaning.

Mikolov et al. present two models that set a new baseline for text embeddings in [153, 155]. Known colloquially as "word2vec," the Bag-of-Words and hierarchical Skip-Gram models learn semantic embeddings that transfer across a number of

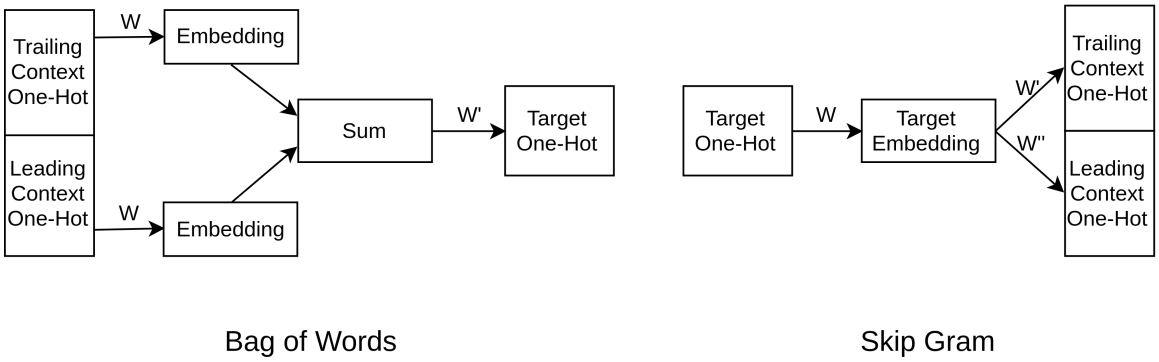


Figure 2.2: Word2vec architectures. Here each W corresponds to weight matrices. Note that the training and leading context is typically comprised of many words, and only one is shown above.

machine learning applications. Each method begins by sampling "windows" from a corpus of text. A window is simply a short string of words centered around a particular target. Typically a window contains an equal number of "context" words both leading and following the target. For example, the window *"quick brown fox jumped over"* contains the target word "fox," and two words for both leading and trailing context. Note that windows can be sampled in parallel, and do not contain additional information regarding where the sample originated. After sampling, the two models diverge. We depict a summary of each model architecture in Figure 2.2.

The Bag-of-Words model learns to predict the target word given its context. Each word is initially represented as a one-hot vector and the corresponding weights of this model are later interpreted as the resulting embeddings. The Bag-of-Words model first maps each words from a sample's context their corresponding embeddings. From there, it uses the summation of context embeddings as a way to predict the target one-hot embedding. Error from the prediction back-propagates to the embeddings [94], which lead to an updated feature space during training. The name "Bag-of-Words" comes from the summation at the center of this model. By collapsing the context

vectors into a single sum, this model discards word-order information found in the context. However, by only maintain two weight matrices, the one that maps context to embedding, and the other that maps summation to target, the Bag-of-Words model is one of the fastest to train.

The Skip-Gram model adds additional weights in order to retain word-order information. Specifically, this model swaps input and output to instead learn a mapping from target word to context. The target one-hot vector is first mapped to an embedding, which is then transformed to one-hot representations independently for each word in the context. This means that different weights will learn the relationship between "fox" to "quick" and "brown," from the above window. Error originating at each word in the context is summed during back-propagation in order to update only the target word's embedding. This model can learn higher-quality latent features, but does so at a steeper training cost. Now, instead of two weight matrices, the number of proportional to the window size. However, due to parallel training and typically small window sizes, the added complexity does not make the Skip-Gram model infeasible for real-world applications.

Building off the word2vec models of Mikolov et al., Joulin et al. (with many of the same authors) leverage character-group embeddings to improve the quality of textual latent spaces in the "fasttext" model [113, 114, 34]. This model extends Skip-Gram to include sub-word information. The authors note that long rare words often share common roots with better-known terms. This word decomposition is referred to as morphology in linguistics. The intuition is often seen in humans who encounter unknown words. For instance, when a hypothetical person encounters a new word, such as "neurodegenerative" (as the fasttext model will later in this thesis), it is reasonable that a human observer would deconstruct this longer word into known sub-components: "neuro-de-generat-ive." She could then reasonably guess that neu-

rodegenerative refers to something that lessens the brain, assuming they knew that "neuro" often related to the brain, "de" negates, "generat" refers to growth, and "ive" implies that the term is an adjective. While this approximation may not convey the exact intended meaning, these sub-word approximations filled in a significant amount of unobserved information. The fasttext model captures this process mathematically. Each individual word is represented as a "bag of character n-grams" with added special symbols ("¡" and "¿") to denote the start and end of a word. These n-grams are taken through a sliding window approach, using a range of window sizes. In addition, the whole term is added to the bag. To recreate the example found in for a sliding window of size 3 [34]:

$$\text{where} \rightarrow (\text{<where>}, \text{<wh, whe, her, ere, re>})$$

Using this decomposition approach, fasttext first embeds each set of characters and derives word embeddings through the sum of its character embedding. That summed embedding is then used to predict the remaining context one-hot vectors, in the same manner found in the Skip-Gram model. Using this sub-word approach, in addition to implementation details found in [113, 114], the fasttext method achieves higher quality latent spaces while remaining an efficient tool for embeddings.

One limitation of the above text embedding models is posed by homographs — words that are spelled the same but have different meanings. In our work on full-text papers we identified many such words, such as "fig," which on its own may refer to a tree, fruit, gene, or "figure." The above-listed methods each identify a single vector representation for each observed word, meaning that "fig" would have a single representation regardless of context. The ELMo (Embedding from Language Models) model addresses this by adding sequence-based machine learning techniques. The

Lost Short Term Memory (LSTM) unit captures recurrent properties of sequences, and can identify trends in natural language sentences [217]. The hidden state of each LSTM unit is conditioned both on a current input as well as the previous state of the unit. As a result, the LSTM model has the ability to distinguish homographs from context. For instance, the sentences "I ate a fig" and "I sat under the fig" each contain information prior to the homograph that disambiguates its meaning. An LSTM model will therefore have different internal state when considering the homograph during the embedding. The ELMo model specifically uses a bidirectional approach wherein two LSTM models each consider the sequence starting from the front and end respectively. Their joint features are then understood by a second layer of bidirectional LSTMs. Training uses sub-sequence information to predict the following term. For instance, the forward-facing LSTM will predict the i^{th} word given words $1, 2, ..., i-1$, and the backward facing LSTM will predict the same word given $i+1, i+2, ..., n$, where n is the length of the sentence.

2.2.1 The Transformer

Modern advances in deep-learning architectures has enabled a new wave of text embedding models. The Transformer [236], a sequence-to-sequence model built through multi-headed attention layers, has been customized for a number of NLP tasks, as best demonstrated by BERT [63], GPT-2 [173], and a range of notable follow-ups [174, 216, 147]. Conceptually, the attention mechanism works by learning multiple weighted averages per-element of the input sequence. Specifically, this includes three projections of each element's embedding, represented as packed matrices: Q, K, and V. Each projection functions differently, with Q acting as a "query" that is compared against "keys" K and "values" V. The specific mechanism is defined as follows, with

d_k representing the dimensionality of each Q and K embedding:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\intercal}{\sqrt{d_k}}\right)V \tag{2.3}$$

The "multi-headed" aspect of the transformer indicates that the self-attention mechanism is applied multiple times per-layer, per-element of the sequence. These multiple heads are then recombined through a feed-forward layer:

$$\begin{aligned} \text{MultiHead}(X, Y) &= [h_1; \ldots; h_k]W^{(4)} \\ \text{where } h_i &= \text{Attention}\left(XW_i^{(1)}, YW_i^{(2)}, YW_i^{(3)}\right) \end{aligned} \tag{2.4}$$

The transformer model presented by Vaswani et al. [236] use the attention attention mechanism in three different ways. Within the encoder stack, which processes the input sequence in their proposed sequence-to-sequence model, the K, Q, and V embeddings all come from the same sequence of tokens. This is referred to all "self attention." In the decoder stack, the part of the model that uses the encoder output to generate a new sequence, these embedding matrices are masked during the attention function such that the output embedding for position i can only depend on prior elements. This is called "masked self attention". Following this operation, each decoder embedding is attended with all of the encoder embeddings. Specifically, Q values are derived from the decoder, while K and V values depend on the encoder. We refer to this operation as "Encoder-Decoder Attention." Note that BERT [240] uses only the encoder self-attention layers, while GPT-2 [173] uses the decoder's masked self-attention layers. The work presented here uses all three.

The multi-head components are combined with a feed-forward operation, denoted FF, that projects the concatenated embedding into a larger dimensionality, applies the ReLU activation function, and then reduces back to the set embedding

rank:

$$\mathrm{FF}(X) = \max(0, XW)W' \tag{2.5}$$

Then, combined with a learned layer-wise normalization, these components combine to form encoder and decoder blocks. Omitting the standard dropout between each operation, the encoder block is defined as:

$$\begin{aligned} \mathcal{E}(X) &= \mathrm{LayerNorm}(\mathrm{FF}(\alpha) + \alpha) \\ \alpha &= \mathrm{LayerNorm}(\mathrm{MultiHead}(X, X) + X) \end{aligned} \tag{2.6}$$

while the decoder block is defined as:

$$\begin{aligned} \mathcal{D}(X, Y) &= \mathrm{LayerNorm}(\mathrm{FF}(\alpha) + \alpha) \\ \alpha &= \mathrm{LayerNorm}(\mathrm{MultiHead}(\beta, Y) + \beta) \\ \beta &= \mathrm{LayerNorm}(\mathrm{MultiHead}(X, X) + X) \end{aligned} \tag{2.7}$$

We depict the transformer architecture in Figure 2.3.

Tokenization chunks an input sequence of characters into input for a transformer-based model. BERT leverages the WordPiece algorithm [252], which first learns to identify a predetermined number of character-groups from a sample of text in order to minimize the expected number of character groups per sentence. The fact that practitioners can tune the number of tokens in a WordPiece tokenization of critical for lowering the overall vocabulary words, and ultimately the size of the model. This approach also allows the model to more easily adapt to out-of-vocabulary words, as infrequent words can simply be constructed by assembling smaller word-chunks (often the chunks containing a single character) [191]. While the WordPiece algorithm itself is proprietary, SentencePiece is an official open-source implementation.

Many groups have worked to endow transformer-based language models with

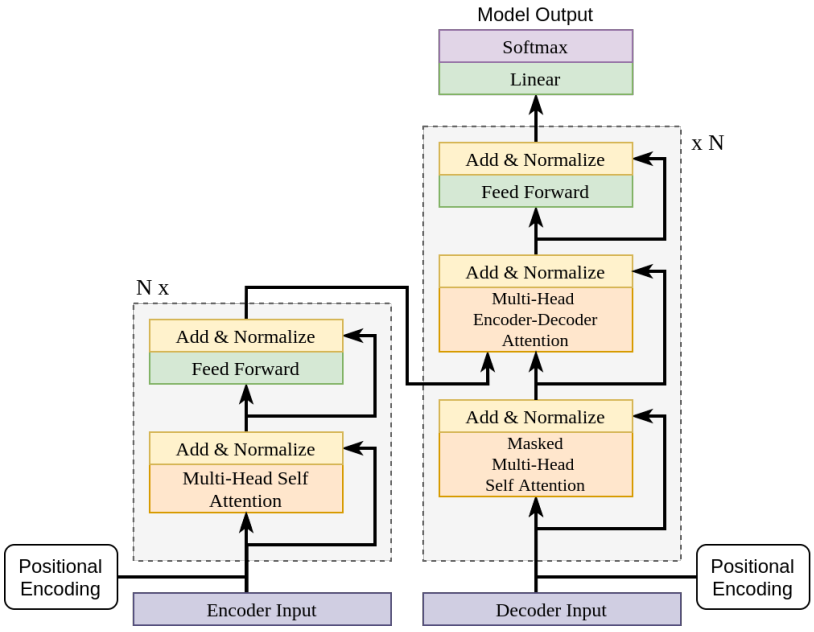


Figure 2.3: The Transformer Architecture. Note that BERT uses only the encoder half, while GPT-2 uses only the decoder half.

domain-based information. In the field of scientific language, two major models have been proposed: SciBERT from AllenNLP [22], and BioBERT from Korea University in Seoul [135]. SciBERT is trained on over one-million papers from Semantic-Scholar.org, and constructed to completed named entity recognition, PICO Extraction, Text Classification, Relation Classification, and Dependency Parsing. For each of these tasks, training data is provided by relatively small human annotated datasets. Improved performance comes from initial pretraining done on the base of the model, in the same manner as was performed for the original BERT. From there, the base model can be used to instantiate fine-tuned version of SciBERT, each with different "task-heads," which learn to associate the fundamental semantic content of the base SciBERT model with the particular task at hand. BioBERT performs a similar procedure, focusing on texts available from Medline and PubMedCentral, as well as English Wikipedia and the Books Corpus. Then, after being pretrained on all four

datasets, BioBERT fine-tunes for named entity recognition, relation extraction, and question answering. Again, the datasets used for fine-tuning are significantly smaller than the datasets used for the BioBERT pretraining phase. In both cases, SciBERT and BioBERT demonstrate superior performance in their respective tasks.

2.3 Graph Embedding

Inspired by the Skip-Gram method for embedding text, Perozzi et al. demonstrate that for a similar method can capture latent structural features of traditional graphs [168]. Their approach, Deepwalk, reduces the graph problem into a text problem by performing a large number of random walks. Each walk produces a sequence of nodes forming a "sentence," which they then input to a similar Skip-Gram model [157]. This model learns to predict a target node given its sampled context within an individual walk.

An alternative approach, LINE by Tang et al., models first- and second-order node relationships directly [228]. This process samples from the local neighborhoods of each node explicitly, and learns an embedding such that the dot product of embeddings correlates with the observed similarities. The bipartite embedding method we present in Chapter 3 is most directly similar to LINE. This model optimizes to reduce the KL-divergence between the set of node samples and the estimated probabilities calculated through embeddings.

Grover et al. observe that Deepwalk and LINE each capture a different set of latent features [88]. The depth-first random walks performed in Deepwalk capture homophilic equivalences, while LINE's breadth-first similarities capture structural equivalences. Homophily describes the tendency for similar nodes to be densely connected [77], while structural similarity descries the roles played by nodes that may

be in desperate regions of a network. This is best described in a corporate social network. Two members of the same project group share homophilic similarity — they are likely both connected to each other member of their group, and all work toward a common task. Meanwhile, team leaders across the company each share structural similarities. The team leaders may not be directly connected, and may work on very different projects, but they are all identified by their managerial relationship bridging the rest of the group to upper management.

Grover et al. propose the "node2vec" model, which learns both homophilic and structural similarities to form a richer latent space [88]. This model uses a biased random walk that has tunable parameters corresponding to the return- and out-step weights during each sample. Formally, if the model begins a walk at node n_1 it takes its first step through a uniform random sample of n_1's neighborhood to neighbor n_2. From there, the algorithm selects n_3 by a weighted random sample. Node2vec assigns a return weight of $1/p$ to n_1. It assigns an in-step weight 1 to all neighbors of n_2 that are also neighbors of n_1 ($\Gamma(n_1) \cap \Gamma(n_2)$). It assigns an out-step weight of $1/q$ to the remaining unweighted neighbors. Using this method, one may tune node2vec by selecting values of p and q that prioritize structural and homophilic similarities accordingly. Upon collecting random walks, this approach learns embeddings through the same Skip-Gram model used by Perozzi et al. [168].

Many additional embedding methods tailor to specific graph subclasses. One of particular interest is the heterogeneous information network (HIN) [199]. A HIN is a generalization to the traditional graph wherein each node is assigned a type. For instance, a citation network may consists of types: author, paper, and venue. Each HIN also has a corresponding schema that describes the connection patterns between types. The citation network contains links between authors and papers (authorship) and between papers and venues (publishing). The resulting schema, therefore, would

be the line graph: author – paper – venue. A metapath is a path template described as a walk through the schema. For instance, co-authorship is defined as the metapath: author – paper – author. Metapath2Vec++, presented by Dong et al., learns to embed a HIN given a predefined set of metapaths [66]. Sampled walks are restricted to only metapath descriptions, but are sampled similarly to Deepwalk. These samples are fed into a modified Skip-Gram that incorporates type information by using different weight matrices for each node type.

Further tailored embedding methods are described in Chapter 3, which outlines our proposed bipartite graph embedding.

2.4 Automatic Hypothesis Generation

A significant portion of this thesis centers around the application of biomedical automatic hypothesis generation. Sometimes called literate-based discovery, this process consists of collecting scientific knowledge, typically encoded as research papers or summaries, and using that information to predict upcoming novel and fruitful research directions. We focus our analysis in the domain of biomedical literature due to the vast quantity of publicly available datasets, such as the MEDLINE database [162] provided by the US National Library of Medicine (NLM). This single database contains nearly 30-million citations at the time of writing, has approximately 1-million citations added yearly, and this rate is increasing [1].

Within this large collection of publicly available research exist *undiscovered public knowledge*, as Swanson described in [221]. Put simply, imagine two research labs are working on related information, one establishing an $A-B$ connection and the other establishing a $B-C$ connection. When both teams publish their findings, the dataset of known entity relationships will contain the *implicit* $A-B-C$ connection,

but until that information is presented to a human actor, we cannot make use of this information. Here, automatic hypothesis generation systems aim to discover these implicit connections in order to provide useful information to domain researchers.

The ABC pattern of automated discovery begins in earnest with Swanson's ARROWSMITH system [219]. By running database queries for titles similar to keywords of interest, this 1986 system identified lists of relevant terms for both user-supplied keywords A and C. From there, the overlap of lists forms the candidate B set. This simple paradigm established real-world medical discoveries, including the connection between fish oil and blood viscosity [219], migraines and magnesium [220], and around a dozen similar findings.

While there have been many contemporary hypothesis generation systems since ARROWSMITH, and in the following chapters we detail many of them, there are a number of overarching strategies worth summarizing here. The first, as seen by ARROWSMITH, is keyword retrieval. The second, popularized by Spanger's work [208], is to produce recommendations through co-occurrence matrices. His described system, implemented in IBM's Watson for Drug Discovery [18], creates a large term-document matrix related to a particular set of user-supplied interests. From there, Watson perform matrix decomposition in a way similar to the non-negative matrix factorization example from Section 2.1. This technique allows their proposed system to perform recommendation by comparing similarities of elements in the low-rank latent space. What makes the Spanger approach different from other co-occurrence methods is their heavy use of visualizations. In [208], Spanger depicts a range of hierarchies and graphs that a biomedical researcher could explore in order to come to their own conclusions about the queried objects of interest.

A key limitation of many hypothesis generation systems we explore later, including ARROWSMITH and Watson, is a reliance on human interpretation of results

in order to evaluate the plausibility of a single hypothesis. One strategy for hypothesis generation that can overcome this issue is link prediction. Many groups maintain biomedical graphs, where nodes correspond to medical entities and edges correspond to known relationships of correlations, such as OMIM [80] and GWAS [21]. Using these networks, link-prediction methods such as DiseaseConnect [145] can identify missing connections through a range of graph analysis techniques in order to estimate the likelihood of a new result. This class of methods slightly removes the human from the query loop because the plausibility of a new connection requires significantly less analysis, when compared to that of studying visualizations. However, the breadth of information contained in these graphs is typically limited to a specific type of relationship, such as that between genes and diseases. Therefore, while the link-prediction strategy overcomes some limitations of the keyword retrieval and the visualization approaches, it imposes a new limitation on scope.

Across the above strategies, the field of hypothesis generation is also restricted on validation strategies [41]. As opposed to other data mining tasks, hypothesis generation seeks to uncover novel information that is unknown to everyone, including those who are designing hypothesis generation systems. Therefore, it is particularly challenging to validate an up-to-date system that proposes an unexpected connection between two elements. In order to determine whether or not the hypothesis generation system is performing correctly, the obvious test is to perform laboratory experiments. However, this process is time consuming and expensive. The second-best test is often historical analysis. However, many systems require expert input, or are limited to small domains when performing historical tests. Both limiting factors cause problems for broad and large-scale testing.

Given that the amount of published information continues to accelerate each yet, it is likely that these implicit findings are more plentiful, useful, and hard to find.

As a result, we propose a number of advances, beyond the limited "ABC" model [4], to create more sophisticated hypothesis generation systems. In the latter chapters of this dissertation we explore new hypothesis generation systems that extend the ABC model. We also propose new ways to quantify hypothesis plausibility that does not rely on expert input, and ways to perform historical validation that overcome previous limitations.

Chapter 3

FOBE and HOBE: First- and High-Order Bipartite Embeddings

Abstract

Typical graph embeddings may not capture type-specific bipartite graph features that arise in such areas as recommender systems, data visualization, and drug discovery. Machine learning methods utilized in these applications would be better served with specialized embedding techniques. We propose two embeddings for bipartite graphs that decompose edges into sets of indirect relationships between node neighborhoods. When sampling higher-order relationships, we reinforce similarities through algebraic distance on graphs. We also introduce ensemble embeddings to combine both into a "best of both worlds" embedding. The proposed methods are evaluated on link prediction and recommendation tasks and compared with other state-of-the-art embeddings. Our embeddings are found to perform better on recommendation tasks and equally competitive in link prediction. Although all considered embeddings are beneficial in particular applications, we demonstrate that none of

those considered is clearly superior (in contrast to what is claimed in many papers). Therefore, we discuss the trade offs among them, noting that the methods proposed here are robust for applications relying on same-typed comparisons.

Reproducibility: Our code, data sets, and results are all publicly available online at: `https://sybrandt.com/2019/fobe_hobe`.

3.1 Background and Motivation

Graph embedding methods place nodes into a continuous vector space in order to capture structural properties that enable machine learning tasks [83]. While many have made significant progress embedding general graphs [168, 228, 88, 232], we find that bipartite graphs have received less study [79], and that the field is far from settled on this interesting case. There exist a variety of special algorithmic cases for bipartite graphs, which are utilized in applications such as user-product or user-group recommender systems [258], hypergraph based load balancing and mapping [161], gene-disease relationships [19], and drug-to-drug targets [254], to mention just a few.

We define a simple, undirected, and unweighted bipartite graph to be $G = (V, E)$ where $V = \{v_1, v_2, \ldots, v_{n+m}\}$ is composed of the *disjoint* subsets $A = \{\alpha_1, \ldots, \alpha_n\}$ and $B = \{\beta_1, \ldots, \beta_n\}$ ($V = A \cup B$). Here, A and B represent the two halves of the network, and are sometimes called "types." We use v_i to indicate any node in V, α_i for nodes in A, and β_i for those in B. In a bipartite graph, edges only occur across types, and $E \subseteq \{A \times B\}$ indicates those connections within G. A single edge is notated as $\alpha_i\beta_j \in E$, and because our graph is undirected, $\alpha_i\beta_j = \beta_j\alpha_i$. The neighborhood of a node is indicated by the function $\Gamma(\cdot)$. If $\alpha_i \in A$ then $\Gamma(\alpha_i) = \{\beta_j | \alpha_i\beta_j \in E\}$, and vice-versa for nodes in B. In order to sample an element from a set, such as selecting

a random α_i from A with uniform probability, we notate $\alpha_i \sim A$. The problem of graph embedding is to determine a representation of the nodes in G in a vector space of r dimensions such that $r << |V|$ and that a select node-similarity measure defined on V is encoded by these vectors [232]. We notate this embedding as the function $\epsilon(\cdot) : V \to \mathbb{R}^r$, that maps each node to an embedding.

We propose two methods for embedding bipartite graphs. These methods fit embeddings by optimizing nodes of each type separately, which we find can lead to higher quality type-specific latent features. Our first method, First-Order Bipartite Embedding (FOBE), samples for the existence of direct, and first-order similarities within the bipartite structure. This approach maintains the separation of types by reformulating edges in E into indirect same-typed observations. For instance, the connection $\alpha_i\beta_j \in E$ decomposes into a set of observed pairs $(\alpha_i, \alpha_k \sim \Gamma(\beta_j))$ and $(\beta_j, \beta_k \sim \Gamma(\alpha_i))$.

Our second method, High-Order Bipartite Embedding (HOBE), samples direct, first-, and second-order relationships, and weighs samples using algebraic distance on bipartite graphs [50]. Again, we represent sampled relationships between nodes of different types by decomposing them into collections of same-typed relationships. While this sampling approach is similar to FOBE, algebraic distance allows us to improve embedding quality by accounting for broader graph-wide trends. Algebraic distance on bipartite graphs has the effect of capturing strong local similarities between nodes, and reduces the effect of less meaningful relationships. This behavior is beneficial in many applications, such as shopping, where two users are likely more similar if they both purchase a niche hobby product, and may not be similar even if they both purchase a generic cleaning product.

Because FOBE and HOBE each make different prior assumptions about the relevance of bipartite relationships, we propose a method for combining bipartite em-

beddings to get "best of both worlds" performance. This ensemble approach learns a joint representation from multiple pre-trained embeddings. The "direct" combination method fits a non-linear transformation of the original embeddings into a fixed-size hidden layer in accordance to sampled similarities. The "auto-regularized" combination extends the direct method by introducing a denoising-autoencoder layer in order to regulate the learned joint embedding [238]. The architecture of both approaches maintains a separation between nodes of different types, which allows for type-specific embeddings, without the constraint of a shared global structure. Evaluation of all proposed embeddings is performed on link prediction reinforced with holdout experiments and recommender system tasks.

Our contribution in summary: (1) We introduce First- and High-Order Bipartite Embeddings that learn representations of bipartite structure that retaining type-specific semantic information. (2) We present the direct and the auto-regularized methods to leverage multiple pre-trained graph embeddings to produce a "best of both words" embedding. (3) We discuss the strengths and weaknesses of our proposed methods as they compare to a range of graph embedding techniques. We identify certain graph properties that suit different graph types, and report that none of the proposed embeddings is clearly superior. However, we find that applications wanting to make many same-typed comparisons are often best suited by a type-sensitive embedding.

3.1.1 Related Work

Low-rank embeddings project high-order data into a compressed real-valued space, often for the purpose of facilitating machine learning models. Inspired by the Skip-Gram approach[153], Perozzi et al. demonstrate that for a similar method

can capture latent structural features of traditional graphs [168]. An alternative approach, LINE by Tang et al., models first- and second-order node relationships explicitly [228]. Node2Vec blends the intuitions behind both LINE and Deepwalk by combining homophilic and structural similarities through a biased random walk [88]. Our proposed methods are certainly influenced by LINE's approach, but differ in a few key areas. Firstly, we split our model in order to only make same-typed comparisons. Furthermore, we introduce terms that compare nodes with relevant negihborhoods, and can weigh different samples with algebraic distance [50].

While the three previously listed embedding approaches are designed for traditional graphs, Metapath2Vec++ by Dong et al. presents a heterogeneous approach using extended type-sensitive skip-gram model [66]. Our method differs from Dong et al.'s in a number of ways. Again, we do not apply random walks or the skip-gram model. Furthermore, the Metapath2Vec++ model implicitly asserts that output type-specific embeddings be a linear combination of the same hidden layer. In contrast, we create entirely separate embedding spaces for the nodes of different types. BiNE by Gao et al. focuses directly on the bipartite case [79]. This approach uses the biased random-walks described in Node2Vec, and samples these walks in proportion to each node's HITS centrality [123]. While our methods differ, again, in the use of skip-gram, BiNE also fundamentally differs from our proposed approaches by enforcing global structure through cross-type similarities. Tsitsulin et al. present VERSE, a versatile graph embedding method that allows multiple different node-similarity measures to be captured by the same overarching embedding technique [232]. This method requires that the user specify a node-similarity measure that will be encoded in the dot product of resulting embeddings. A key difference between the methods presented here, and the methods presented in VERSE, come from differences in objective values when training embeddings. VERSE uses a range of methods to sample node-pairs,

from direct sampling to Noise Contrastive Estimation [89], and updates embeddings according to their observed similarity or dissimilarity (in the case of negative samples). However, the optimization method proposed here enforces only same-typed comparisons.

3.2 Methods and Technical Solutions

We present two sibling strategies for learning bipartite embeddings. First-Order Bipartite Embedding (FOBE) samples direct links from E and first-order relationships between nodes sharing common neighbors. We then fit embeddings to minimize the KL-Divergence between our observations and our embedding-based estimations. The second method, High-Order Bipartite Embedding (HOBE), begins by computing algebraic similarity estimates for each edge [50, 195]. Using these heuristic weights, HOBE samples direct, first- and second-order relationships, to which we fit embeddings using mean-squared error. We implement both methods in Python using Keras [53] and Tensorflow [8].

At a high level, both embedding methods begin by observing structural relationships within a graph G and then fitting an embedding ϵ in order to encode structural features via dot product of embeddings. We combine three types of observations for a single graph These observations are represented through the functions $\mathbb{S}_A(\cdot,\cdot)$, $\mathbb{S}_B(\cdot,\cdot)$, and $\mathbb{S}_V(\cdot,\cdot)$. Each function maps two nodes to an observed similarity: $V \times V \to \mathbb{R}$. The result of $\mathbb{S}_A$ is nonzero only if both arguments are in A, $\mathbb{S}_B$ is similarly nonzero only if both arguments are in B. In this manner, these functions capture type-specific similarities. The $\mathbb{S}_V$ function, in contrast, captures cross-typed observations, and is nonzero if its arguments are of different types. We define a reciprocal set of functions to model these similarities: $\widetilde{\mathbb{S}}_A(\cdot,\cdot)$, $\widetilde{\mathbb{S}}_B(\cdot,\cdot)$, and $\widetilde{\mathbb{S}}_V(\cdot,\cdot)$. These

functions are defined in terms of $\epsilon(\cdot)$, and each method must select some embedding such that the difference between each corresponding set of $\mathbb{S}, \widetilde{\mathbb{S}}$ pairs. However, the specifics of each observation, estimation, and objective differs across methods.

Because we estimate similarities within type-specific subsets of ϵ separately, we can better preserve typed latent features. This is important for many applications. Consider an embedding of the bipartite graph of viewers and movies, often used for applications such as video recommendations. Within "movie space" one would expect to uncover latent features such as genre, budget, or the presence of high-profile actors. These features are undefined within "viewer space," wherein one would expect to observe latent features corresponding to demographics and viewing preferences. Clearly these two spaces are correlated in a number of ways, such as the alignment between viewer tastes and movie genres. However, we find methods that enforce direct comparisons between viewer and movie embeddings can result in an erosion of type-specific features, which can lead to lower downstream performance. In contrast, the methods proposed here never make a direct assertion of cross-type similarity, and allow implicit relationships to govern any key correlations across spaces.

3.2.1 First-Order Bipartite Embedding

The goal of FOBE is to model direct and first-order relationships from the original structure. This very simple method only detects the existence of a relationship between two nodes, and therefore does not distinguish between two nodes that share only one neighbor from two nodes that share many. However, we find that this simplicity enables scalability at little cost to quality. Here, a direct relationship is any edge from the original bipartite graph, while a first-order relationship is defined as $\{(\alpha_i, \alpha_j) \mid \Gamma(\alpha_i) \cap \Gamma(\alpha_j) \neq \emptyset\}$. Note that nodes in a first-order relationship share

the same type. We define observations corresponding with each relationship. Direct observations simply detect the presence of an edge, while first-order relationships similarly detect a common neighbor. Formally:

$$\mathbb{S}_A(\alpha_i, \alpha_j) = \begin{cases} 1 & \alpha_i, \alpha_j \in A \ \ \& \ \ \Gamma(\alpha_i) \cap \Gamma(\alpha_j) \neq \emptyset \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

$$\mathbb{S}_B(\beta_i, \beta_j) = \begin{cases} 1 & \beta_i, \beta_j \in B \ \ \& \ \ \Gamma(\beta_i) \cap \Gamma(\beta_j) \neq \emptyset \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

$$\mathbb{S}_V(\alpha_i, \beta_j) = \begin{cases} 1 & \alpha_i\beta_j \in E \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

By sampling γ neighbors, we allow our later embedding model to approximate the effects of Γ, similar to the k-ary set sampling in [160]. Note also that each sample contains one nonzero $\mathbb{S}$ value. By fitting all three observations simultaneously, we implicitly generate two negative samples for each positive sample. Furthermore, we generate a fixed number of samples for each node's direct and first-order relationships.

Given these observations $\mathbb{S}_A$, $\mathbb{S}_B$, and $\mathbb{S}_V$, we fit the ϵ embedding according to corresponding estimation functions $\widetilde{\mathbb{S}}_A$, $\widetilde{\mathbb{S}}_B$, $\widetilde{\mathbb{S}}_V$. To estimate a first-order relationship ($\widetilde{\mathbb{S}}_A$ and $\widetilde{\mathbb{S}}_B$) we calculate the sigmoid of the dot product of embeddings (3.5), namely,

$$\sigma(x) = \frac{1}{1 + e^{-x}}. \tag{3.4}$$

$$\widetilde{\mathbb{S}}_A(\alpha_i, \alpha_j) = \sigma\left(\epsilon(\alpha_i)^\intercal \epsilon(\alpha_j)\right) \tag{3.5}$$

$$\widetilde{\mathbb{S}}_B(\beta_i, \beta_j) = \sigma\left(\epsilon(\beta_i)^\intercal \epsilon(\beta_j)\right) \tag{3.6}$$

Building from this, we train embeddings based on direct relationships by composing relevant first-order relationships. Specifically, if $\alpha_i\beta_j \in E$ then we would expect α_i to be similar to $\alpha_k \in \Gamma(\beta_j)$ and vice-versa. Intuitively, a viewer has a higher chance of watching a movie if they are similar to others that have. We formulate our direct relationship estimate to be the product of each node's average first-order estimate to the other's neighborhood. Formally:

$$\widetilde{\mathbb{S}}_V(\alpha_i, \beta_j) = \mathop{\mathbb{E}}_{\alpha_k \in \Gamma(\beta_j)} \left[\widetilde{\mathbb{S}}_A(\alpha_i, \alpha_k)\right] \mathop{\mathbb{E}}_{\beta_k \in \Gamma(\alpha_i)} \left[\widetilde{\mathbb{S}}_B(\beta_j, \beta_k)\right] \tag{3.7}$$

In order to train our embedding function ϵ for the FOBE method, we minimize the KL-Divergence [128] between our observed similarities $\mathbb{S}$ and our estimated similarities $\widetilde{\mathbb{S}}$. We minimize for each simultaneously, for both direct and first-order similarities, using the Adagrad optimizer [68], namely, we solve:

$$\min_{\epsilon} \sum_{v_i, v_j \in V \times V} \begin{bmatrix} \widetilde{\mathbb{S}}_A(v_i, v_j) \log\left(\dfrac{\mathbb{S}_A(v_i, v_j)}{\widetilde{\mathbb{S}}_A(v_i, v_j)}\right) \\ +\widetilde{\mathbb{S}}_B(v_i, v_j) \log\left(\dfrac{\mathbb{S}_B(v_i, v_j)}{\widetilde{\mathbb{S}}_B(v_i, v_j)}\right) \\ +\widetilde{\mathbb{S}}_V(v_i, v_j) \log\left(\dfrac{\mathbb{S}_V(v_i, v_j)}{\widetilde{\mathbb{S}}_V(v_i, v_j)}\right) \end{bmatrix} \tag{3.8}$$

3.2.2 High-Order Bipartite Embedding

The goal of HOBE is to capture distant relationships between nodes that are related, but may not share an edge or a neighborhood. In order to differentiate the meaningful distant connections from those that are spurious, we turn to algebraic

distance on graphs [195]. This method is fast to calculate and provides a strong signal for *local similarity*. For example, algebraic distance can tell us which neighbor of a high-degree node is the most similar to the root. As a result, we can utilize this signal to estimate which multi-hop connections are the most important to preserve in our embedding.

Algebraic distance is a measure of dependence between variables popularized in algebraic multigrid (AMG) [179, 40, 149]. Later, it has been shown to be a reliable and fast way to capture implicit similarities between nodes in graphs [111, 141] and hypergraphs that are represented as bipartite graphs [195] (which is leveraged in this work) taking into account distant neighborhoods. Technically, it is a process of relaxing randomly initialized test vectors using stationary iterative relaxation applied on graph Laplacian homogeneous system of equations, where in the end the algebraic distance between system's variables x_i and x_j (that correspond to linear system's rows i and j) is defined as an maximum absolute value between the ith and jth components of the test vectors (or, depending on application, as sum or sum of squares of them).

In our context, a variable is a node, and we apply K iterations of Jacobi over-relaxation (JOR) on the bipartite graph Laplacian as in [179] ($K = 20$ typically ensures good stabilization as we do not need full convergence, see Theorem 4.2 [50]). Initially, each node's coordinate is assigned a random value, but on each iteration a node's coordinate is updated to move it closer its neighbors' average. Weights corresponding to each neighbor are inversely proportional their degree in order to increase the "pull" of small communities. Intuitively, this acknowledges that two viewers who both watch a niche new-wave movie are more likely similar than two viewers who watched a popular blockbuster. We run JOR on R independent trials (called test vectors in AMG works, convergence proven in [50]). Formally, for rth test vector a_r the update step of JOR is performed as follows, where $a_r^{(t)}(v_i)$ represents

node v_i's algebraic coordinate on iteration $t \in \{1, .., K\}$, and λ is a damping factor (suggested $\lambda = 0.5$ in [195]).

$$\mathbf{a}_r^{(t+1)}(v_i) = \lambda \mathbf{a}_r^{(t)}(v_i) + (1 - \lambda) \frac{\sum_{v_j \in \Gamma(v_i)} \mathbf{a}_r^{(t)}(v_j) |\Gamma(v_j)|^{-1}}{\sum_{v_j \in \Gamma(v_i)} |\Gamma(v_j)|^{-1}} \tag{3.9}$$

We use the l^2-norm in order to summarize the algebraic distance of two nodes across R trails with different random initializations. As a result, two nodes will be close in our distance calculation if they remain nearby across many trials, which lessens the effect of too slow convergence in a single trial. For our purposes we select $R = 10$. Additionally, we define "algebraic similarity", $s(i, j)$, as a closeness across trials. We subtract the distance between two embeddings from the maximum distance in our space, and rescale the result to the unit interval. Because we know that the maximum distance between any two coordinates in the same trial is 1, we can compute this in constant time:

$$d(v_i, v_j) = \sqrt{\sum_{r=1}^{R} \left(\mathbf{a}_r^{(K)}(v_i) - \mathbf{a}_r^{(K)}(v_j) \right)^2} \tag{3.10}$$

$$s(v_i, v_j) = \frac{\sqrt{R} - d(v_i, v_j)}{\sqrt{R}} \tag{3.11}$$

After calculating algebraic similarities for pairs of nodes of all edges, we begin to sample direct, first-order, and second-order similarities from the bipartite structure. Here, a second-order connection is one wherein α_i and β_j share a neighbor that shares a neighbor: $\alpha_i \in \Gamma(\Gamma(\Gamma(\beta_j)))$. Note that the set of second-order relationships is a superset of the direct relationships. We can extend to these higher-order connections with HOBE, as opposed to FOBE, because of the information provided in algebraic distances. Many graphs contain a small number of high degree nodes, which creates a

very dense second-order graph. Algebraic distances are therefore needed to distinguish which of the sampled second-order connections are meaningful, especially when the refinement is normalized by $|\Gamma(v_i)|^{-1}$.

We formulate our first-order observations to be equal to the strongest shared bridge between two nodes. This indicates that both nodes are closely related to something that is mutually representative, such as two viewers that watch new-wave cinema. Formally:

$$\mathbb{S}_A^{'}(\alpha_i, \alpha_j) = \begin{cases} \max\limits_{\beta_k \in \Gamma(\alpha_i) \cap \Gamma(\alpha_j)} \min\left(s(\alpha_i, \beta_k), s(\alpha_j, \beta_k)\right) \\ \quad \text{if } \alpha_i, \alpha_j \in A \\ 0 \text{ otherwise} \end{cases} \tag{3.12}$$

$$\mathbb{S}_B^{'}(\beta_i, \beta_j) = \begin{cases} \max\limits_{\alpha_k \in \Gamma(\beta_i) \cap \Gamma(\beta_j)} \min\left(s(\alpha_k, \beta_i), s(\alpha_k, \beta_j)\right) \\ \quad \text{if } \beta_i, \beta_j \in B \\ 0 \text{ otherwise} \end{cases} \tag{3.13}$$

When observing second-order relationships between nodes α_i and β_j if different types, we again construct a measurement from shared first-order relationships. Specifically, we are looking for the strongest first-order connection between i and j's neighborhood, and vice-versa. In the context of viewers and movies this represents the similarity between a viewer and a movie watched by a friend. Formally:

$$\mathbb{S}_V^{'}(\alpha_i, \beta_j) = \max \begin{pmatrix} \max\limits_{\alpha_k \in \Gamma(\beta_j)} \mathbb{S}_A^{'}(\alpha_i, \alpha_k), \\ \max\limits_{\beta_k \in \Gamma(\alpha_i)} \mathbb{S}_B^{'}(\beta_j, \beta_k) \end{pmatrix} \tag{3.14}$$

We again collect a fixed number of samples for each relationship type: direct,

first- and second-order. We then train embeddings using cosine similarities, however we select the ReLU activation function to replace sigmoid in order to capture the weighted relationships. We optimize for all three observations simultaneously, which again has the effect of creating negative samples for non-observed phenomena. Our estimated similarities are defined as follows:

$$\widetilde{\mathbb{S}}'_A(\alpha_i, \alpha_j) = \max\left(0, \epsilon(\alpha_i)^\intercal \epsilon(\alpha_j)\right) \tag{3.15}$$

$$\widetilde{\mathbb{S}}'_B(\beta_i, \beta_j) = \max\left(0, \epsilon(\beta_i)^\intercal \epsilon(\beta_j)\right) \tag{3.16}$$

$$\widetilde{\mathbb{S}}'_B(\alpha_i, \beta_j) = \mathop{\mathbb{E}}_{\alpha_k \in \Gamma(\beta_j)} \left[\widetilde{\mathbb{S}}'_A(\alpha_i, \alpha_k)\right] \mathop{\mathbb{E}}_{\beta_k \in \Gamma(\alpha_i)} \left[\widetilde{\mathbb{S}}'_B(\beta_j, \beta_k)\right] \tag{3.17}$$

We use the same model as FOBE to train HOBE, but with our new estimation functions and a new objective. We now optimize for the mean-squared error between our observed and estimated samples, as KL-Divergence is ill-defined for the weighted samples we collect. Formally, we minimize:

$$\min_{\epsilon} \mathop{\mathbb{E}}_{v_i, v_j \in V \times V} \begin{bmatrix} (\mathbb{S}'_A(v_i, v_j) - \widetilde{\mathbb{S}}'_A(v_i, v_j))^2 \\ +(\mathbb{S}'_B(v_i, v_j) - \widetilde{\mathbb{S}}'_B(v_i, v_j))^2 \\ +(\mathbb{S}'_V(v_i, v_j) - \widetilde{\mathbb{S}}'_V(v_i, v_j))^2 \end{bmatrix} \tag{3.18}$$

3.2.3 Combination Bipartite Embedding

In order to unify our proposed approaches, we present a method to create a joint embedding from multiple pre-trained bipartite embeddings. This combination method maintains our initial assertion that nodes of different types ought to partic-

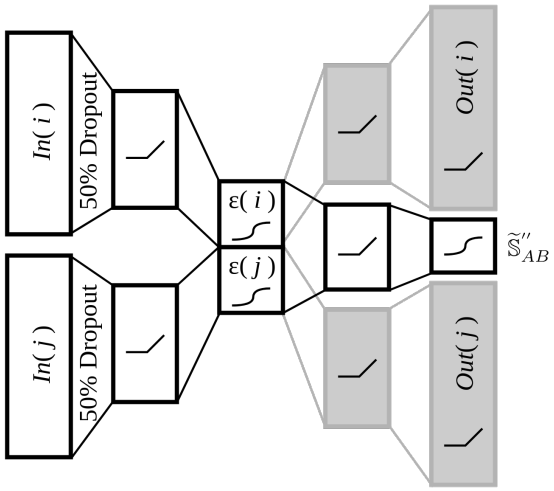


Figure 3.1: **Combination Neural Network Models.** Boxes correspond to dense neural network layers, each depicts its activation function. Grey layers only used for the auto-regularized case.

ipate in different global embedding structures. We fit a non-linear projection of the input embeddings such that an intermediate embedding can accurately uncover direct relationships. This raises a question as to whether it is better to create an intermediate that succeeds in this training task, or whether it is better to fully encode the input embeddings. To address this concern we propose two flavors of our combination method: the "direct" approach maximizes performance on the training task, while the "auto-regularized" approach enforces a full encoding of input embeddings. The models used to generate these embeddings is summarized in Figure 3.1.

We begin by taking the edge list of the original bipartite graph E as our set of positive samples. We then generate five negative samples for each node by selecting random pairs $\alpha_i\beta_j \notin E$. For each sample, we create an input vector by concatenating each of the e' pre-trained embeddings.

$$In(v_i) = [\epsilon_1(v_i) \;\; \epsilon_2(v_i) \;\; ... \;\; \epsilon_{e'}(v_i)] \tag{3.19}$$

After generating $In(\alpha_i)$ and $In(\beta_j)$, our models assert 50% dropout in these input vectors [212]. We do so in the auto-regularized case so that we follow the pattern of denoising auto-encoders, which have shown high performance in robust dimensionality reductions [238]. However, we also find that this dropout increases performance in the direct combination model as well. This is because in either case, we anticipate both redundant and noisy signals to be present across the concatenated embeddings. This is especially necessary for larger values of k and e', where the risk of overfitting increases.

We then project $In(\alpha_i)$ and $In(\beta_j)$ separately onto two hidden layers of size ${}^{d(In)+k'}/_{2}$ where $d(\cdot)$ indicates the dimensionality of the input, and k' represents the desired dimensionality of the combined embeddings. By separating these hidden layers, we only allow signals from within embeddings of the same node to affect its combination. We then project down to two combination embeddings of size k', which act as input to both the joint link-prediction model, as well as to the optional auto-encoder layers.

In the direct case, we simply minimize the mean-squared error between the predicted links and the observed links. Formally, let $\mathbb{S}''(\alpha_i, \beta_j) \to \{0, 1\}$ equal the sampled value, and let $\widetilde{\mathbb{S}}''(\alpha_i, \beta_j) \to \mathbb{R}$ be combination estimate. In the auto-regularized case we introduce a factor to enforce that the original (pre-dropout) embeddings can be recovered from the combined embedding. We weight these factors so they are half as important as performing the link prediction training task. The neural architecture used to learn these combination embeddings is depicted in the supplemental information. If Θ is the set of free parameters of our neural network model, N is the set of negative samples, and $Out(v_i)$ is the output of the auto-encoder corresponding to

$In(v_i)$, then we optimize the following (direct followed by auto-regularized):

$$\min_{\Theta} \mathop{\mathbb{E}}_{\alpha_i,\beta_j \in (E+N)} \left(\mathbb{S}''(\alpha_i, \beta_j) - \widetilde{\mathbb{S}}''(\alpha_i, \beta_j) \right)^2 \tag{3.20}$$

$$\min_{\Theta} \mathop{\mathbb{E}}_{\alpha_i,\beta_j \in (E+N)} \begin{pmatrix} 4 \left(\mathbb{S}''(\alpha_i, \beta_j) - \widetilde{\mathbb{S}}''(\alpha_i, \beta_j) \right)^2 \\ +||In(\alpha_i) - Out(\alpha_i)||_2 \\ +||In(\beta_j) - Out(\beta_j)||_2 \end{pmatrix} \tag{3.21}$$

3.3 Algorithmic Analysis

In order to efficiently compute FOBE and HOBE, we collect a fixed number of samples per node for each of the observation functions, $\mathbb{S}$. As later explored in Table 3.5, we find that the performance of our proposed methods does not significantly increase beyond a relatively small, fixed sampling rate s_r, where $s_r << |V|$. Using this observation, we can efficiently minimize the FOBE and HOBE objective values by approximating the expensive $O(n^2)$ set of comparisons ($v_i, v_j \in V \times V$) with a linear number of samples (specifically $O(|V|s_r)$). Furthermore, we can estimate the effect of each node's neighborhood in observations $\mathbb{S}_V$ and $\mathbb{S}'_V$ by following a similar approach. Instead of considering each node's total $O(V)$-sized neighborhood, we can randomly sample s_γ neighboring nodes with replacement. These specifically samples nodes are recorded during the sampling procedure so that they may be referenced during training. Algorithm 1 describes the sampling algorithm formally.

Procedure 1 Sampling for FOBE and HOBE. Note that when a single sample is recorded, unobserved values are recorded as either zero or empty.

function SameTypeSample($v_i, s_r, \mathbb{S}$)
    $v_j \sim \Gamma(\Gamma(v_i))$
    Record v_i,v_j, and $\mathbb{S}(v_i, v_j)$
function DiffTypeSample($v_i, s_r, s_\gamma, G, \mathbb{S}$)
    $v_j \sim G(v_i)$
    Let γ_α and γ_β be sets of size s_γ sampled with replacement from the neighborhoods $\Gamma(v_i)$ and $\Gamma(v_j)$ according to the types of v_i and v_j.
    Record $v_i, v_j, \gamma_\alpha, \gamma_\beta$, and $\mathbb{S}(v_i, v_j)$.
function FobeSampling(G, s_r, s_γ)
    for all $v_i \in V$ **do**
        for s_r samples **do**
            SameTypeSample($v_i, s_r, \mathbb{S}_A$)
            SameTypeSample($v_i, s_r, \mathbb{S}_B$)
            DiffTypeSample($v_i, s_r, s_\gamma, \Gamma(\cdot), \mathbb{S}_V$)
function HobeSampling(G, s_r, s_γ)
    for all $v_i \in V$ **do**
        for s_r samples **do**
            SameTypeSample($v_i, s_r, \mathbb{S}'_A$)
            SameTypeSample($v_i, s_r, \mathbb{S}'_B$)
            DiffTypeSample($v_i, s_r, s_\gamma, \Gamma(\Gamma(\Gamma(\cdot))), \mathbb{S}'_V$)

3.4 Empirical Evaluation

Link Prediction We evaluate the performance of our proposed embeddings across three link prediction tasks and a range of training-test splits. When removing edges, we visit each in random order and remove them with probability h provided the removal does not disconnect the graph. This additional check ensures all nodes appear in all experimental embeddings. The result is the subgraph $G' = (V, E', h)$. Deleted edges form the positive test-set examples, and we generate set of negative samples (edges not present in original graph) of equal size. These samples are used to train three sets of link-prediction models: the A-Personalized, B-Personalized (where A and B are parts of V), and unified models.

The A-personalized model is a support vector machine trained on the neighborhood of a particular node. A model personalized to $i \in A$ learns to identify a region in B-space corresponding to its neighborhood in G'. We use support vector machines with the radial basis kernel ($C = 1, \gamma = 0.1$) because we find these models result in robust performance given limited training data, and because the chosen kernel function allows for non-spherical decision boundaries. We additionally generate five negative samples for each positive sample (a neighbor of i in G'). In doing so we evaluate the ability to capture type-specific latent features, as each personalized model only considers one-type's embeddings. While the personalized task may not be typical for production link-prediction systems, it is an important measure of latent features found in each space. In many bipartite applications, such as the six we have selected for evaluation, $|A|$ and $|B|$ may be drastically different. For instance, there are typically more viewers than movies, or more buyers than products. Therefore it becomes important to understand the differences in quality between the latent spaces of each type, which we evaluate through these personalized models.

The unified link-prediction model, in contrast, learns to associate $\alpha_i\beta_j \in E'$ with a combination of $\epsilon(\alpha_i)$ and $\epsilon(\beta_j)$. This model attempts to quantify global trends across embedding spaces. We use a hidden layer of size k with the ReLU activation function, and a single output with the sigmoid activation. We fit this model against mean-squared error using the Adagrad optimizer [68].

Datasets. We evaluate each embedding across six datasets detailed in Table 3.1. The Amazon, YouTube, DBLP, Friendster, and Livejournal graphs are all taken from the Stanford Large Network Dataset Collection (SNAP) [140]. We select the distribution of each under the listing "Networks with Ground-Truth Communities." Furthermore, we collect the MadGrades graph, from an online source provided by the University of Wisconsin at Madison [3]. This graph consists of teachers and

Graph	$^{\lvert A\rvert}/_{\lvert B\rvert}$	$\Gamma(\alpha_i)$ md	$\Gamma(\alpha_i)$ max	$\Gamma(\beta_j)$ md	$\Gamma(\beta_j)$ max	SR	LCP
Amazon	$^{16{,}716}/_{5{,}000}$	3	49	8	328	75.8	1.6
DBLP	$^{93{,}432}/_{5{,}000}$	1	12	8	7,556	174.7	81.7
Friendster	$^{220{,}015}/_{5{,}000}$	1	26	133	1,612	80.3	58.3
Livejournal	$^{84{,}438}/_{5{,}000}$	1	20	16	1,441	100.9	27.0
MadGrades	$^{11{,}951}/_{6{,}462}$	3	39	4	393	57.3	99.7
YouTube	$^{39{,}841}/_{5{,}000}$	1	54	4	2,217	113.3	80.6

Table 3.1: **Graph Summary.** We report the median (md) and max degree for each node set, as well as the Spectral Radius (SR) and the percentage of the largest connected component (LCP).

course codes, wherein an edge signifies that teacher α_i has taught course code β_j. We clean this dataset by iteratively deleting any instructor or course with degree 1 until none remain.

Experimental Parameters. We evaluate the performance of our proposed methods: FOBE and HOBE, as well as our two combination approaches: Direct and Auto-Regularized Combination Bipartite Embedding. We compare against all methods described in Section 3.1.1. We evaluate each across the six above graphs and nine training-test splits $h = 0.1, 0.2, ..., 0.9$. For all embeddings we select dimensionality $k = 100$. For Deepwalk, we select a walk length of 10, a window size of 5, and 100 walks per node. For LINE we apply the model that combines both first- and second-order relationships, selecting 10,000 samples total and 5 negative samples per node. For Node2Vec we select 10 walks per node, walk length of 7 and a window size of 3. Furthermore, we select default parameters for BiNE and Metapath2Vec++. For the latter, we supply the metapath of alternating $A - B - A$ nodes, the only metapath in our bipartite case. For FOBE and HOBE we generate 200 samples per node, and when sampling neighborhoods we select 5 nodes with replacement upon each observation. After training both methods, we fit the Direct and Auto-Regularized Combination methods, each trained using *only* the results of FOBE and HOBE.

Recommendation: We follow the procedure originally described by Gao et al. and evaluate our proposed embeddings through the task of recommendation [79]. Recommendation systems propose products to users in order to maximize the overall interaction rate. These systems fit the bipartite graph model because they are defined on the set of user-product interactions. While many such systems could be reformulated as operations on bipartite networks, methods such as matrix factorization and user-user nearest neighbors do not capture granular local features to the same extent as modern graph embeddings [79, 33]. In contrast, bipartite graph embedding provides a framework to often learn richer latent representations for both users and products. These representations can then be used directly through simple similarity measures, or added to existing solution archetypes, such as k-nearest neighbors, which often provides significant quality benefits.

While there are many similarities between recommendation and link prediction, the key difference is the introduction of weighted connections. As a result, recommendation systems are evaluated based on their ability to rank products in accordance to held-out user supplied rankings. This is quantified through a number of metrics defined on the top k system-supplied recommendation for each user. When using embeddings to make a comparison, Gao et al. rank products by their embedding's dot product with a given user. However, our proposed methods relax the constraint that products and users be directly comparable. As a result, when ranking products for a particular user for our proposed embeddings we must first define a product-space representation. For each user we collect the set of known product ratings, and calculate a product centroid weighted by those ratings.

Experimental Procedure. We present a comparison between our proposed

methods and all previously discussed embeddings across the DBLP[1] and LastFM[2] datasets. Note that this distribution of DBLP is the bipartite graph of authors and venues, and is different from the community-based version distributed by SNAP. The LastFM dataset consists of listeners and musicians, where an edge indicates listen count, which we log-scale to improve convergence for all methods. We start by splitting each rating set into training- and test-sets with a 40% holdout. In the case of DBLP we use the same split as Gao et al. We use embeddings from the training bipartite graph to perform link prediction. We then compare the ranked list of training-set recommendations for each user, truncated to 10 items, to the test-set rankings. We calculate 128-dimensional embeddings for each method, and report F1, Normalized Discounted Cumulative Gain (NDCG), Mean Average Precision (MAP) and Mean Reciprocal Rate (MRR).

3.5 Significance and Impact

In contrast to what is typically claimed in papers, we observe that the link prediction data (Table 3.2) demonstrates that different graphs lead to very different performance results for the existing state-of-the-art and proposed embeddings. Moreover, their behavior is changed with different holdouts when the size of training set is smaller. For instance, our methods are above the state of the art in the Youtube and MadGrades graphs, but Metapath2Vec++, Node2Vec, and LINE each have scenarios wherein they outperform the field. Additionally, while there are scenarios where the combination methods perform as expected, such as in the Youtube, MadGrades, and DBLP B-Personalized cases, we observe that variability in the other proposed embeddings can disrupt this performance gain.

[1] `https://github.com/clhchtcjj/BiNE/tree/master/data/dblp`
[2] `https://grouplens.org/datasets/hetrec-2011/`

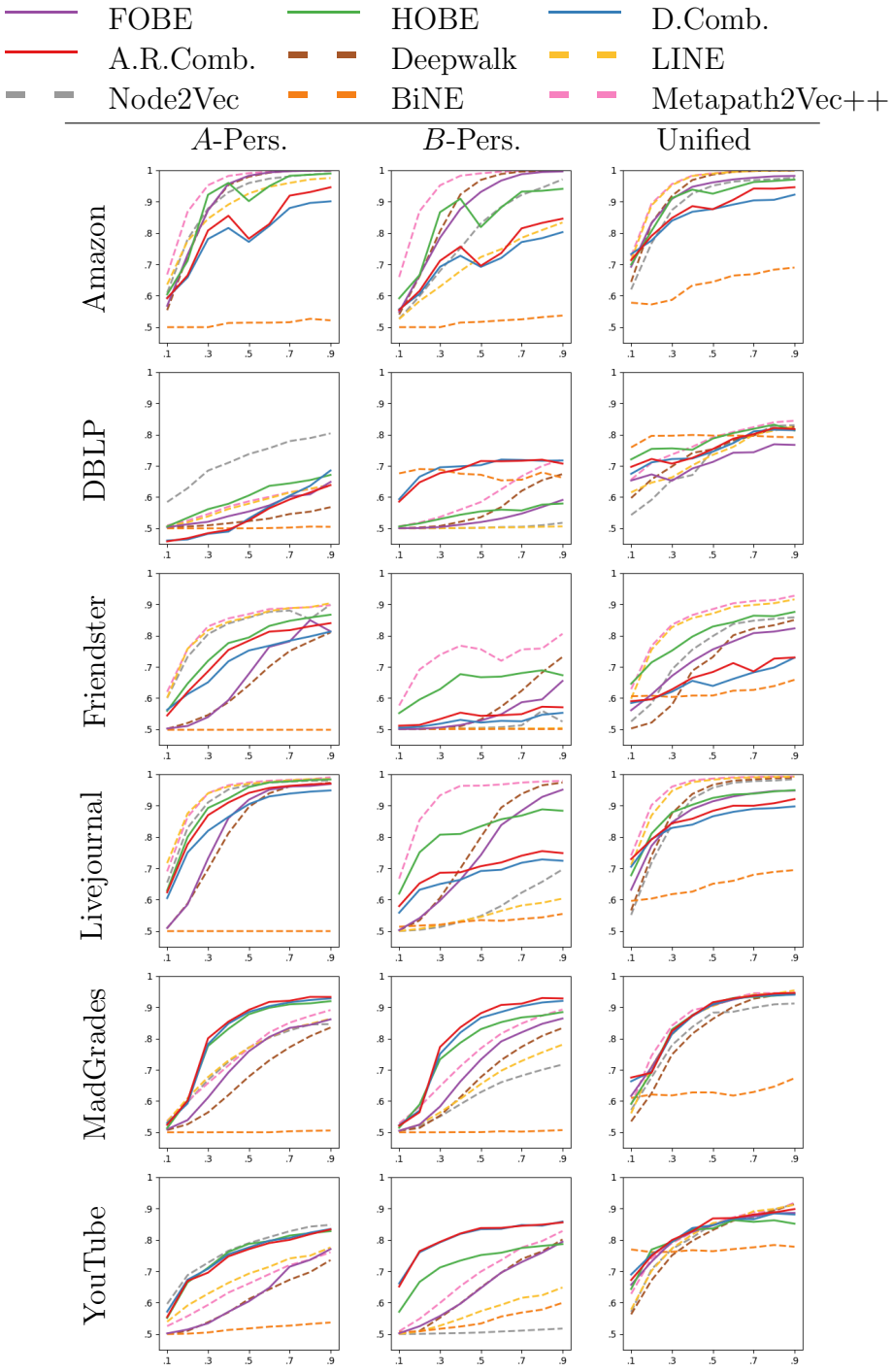


Table 3.2: Link Prediction Accuracy vs. Training-Test Ratio. Methods represented by dashed lines indicate the state-of-the-art, while solid lines indicate methods presented in this work.

When comparing the A- and B-Personalized results, its is important to keep in mind that for all considered graphs there are more A nodes ($|A| > |B|$), and therefore these nodes tend to have fewer neighbors ($\mathbb{E}[\Gamma(\alpha)] < \mathbb{E}[\Gamma(\beta)]$). For this reason, we find that different embedding methods can exhibit significantly different behavior across both personalized tasks. Intuitively, performing well on the A-Personalized set indicates an ability to extrapolate connections between elements with significantly more sparse attachments, such as selecting a new movie given a viewer's limited history. In contrast, performance on the B-Personalized set indicates an ability to uncover trends among relatively larger sets of connections, such as determining what patterns are common across all the viewers of a particular movie. While these two tasks are certainly related, we observe that the B-Personalized evaluation appears to be significantly more challenging for a number of embedding methods, such as Node2Vec on Lovejournal and YouTube. In contrast, HOBE succeeds in this evaluation for both cases, as well as Friendster and MadGrades. Metapath2Vec++ additionally is superior on LiveJournal and Friendster, but falls behind on DBLP, MadGrades, and Youtube.

In the recommendation results (Table 3.3 and 3.4), our methods improve the state-of-the art. This is further evidence that our sampling decompositions are better able to capture product-specific features. Our biggest increase is in MRR for DBLP, indicating that the first few suggestions from our embeddings are often more relevant. The performance of HOBE, demonstrates the ability for algebraic distance to estimate useful local similarity measures. Interestingly, in the LastFM dataset, FOBE outperforms HOBE. One reason for this is that LastFM contains significantly more artists-to-user than DBLP contains venues-to-author. As a result the amount of information present when estimating algebraic similarities is different across datasets, and insufficient to boost HOBE above FOBE.

Metric@10:	F1	NDCG	MAP	MRR
DeepWalk	.0850	.2414	.1971	.3153
LINE	.0899	.1441	.0962	.1713
Node2Vec	.0854	.2389	.1944	.3111
MP2V++	.0865	.2514	.1906	.3197
BINE	**.1137**	.2619	.2047	.3336
FOBE	.1108	.3771	.2382	.4491
HOBE	.1003	**.4054**	**.3156**	**.6276**
D.Comb.	.0753	.2973	.2362	.5996
A.R.Comb.	.0667	.2359	.1730	.5080

Table 3.3: DBLP Recommendation. Note: result numbers from prior works are reproduced from [79].

Metric@10:	F1	NDCG	MAP	MRR
DeepWalk	.0027	.0153	.0069	.1844
LINE	.0067	.0435	.0229	.2477
Node2Vec	.0279	.1261	.0645	.2047
MP2V++	.0024	.0153	.0088	.2677
BINE	.0227	.1551	.0982	.3539
FOBE	**.0729**	**.3085**	**.1997**	.3778
HOBE	.0195	.1352	.0789	.3400
D.Comb.	.0243	.1285	.0795	.3520
A.R.Comb.	.0388	.1927	.1249	**.3915**

Table 3.4: LastFM Recommendations.

When looking at both link prediction and recommendation tasks, we observe a highly variable performance of the combination methods. In some cases, such as the MadGrades and YouTube link prediction tasks, as well as the LastFM recommendation task, these combinations are capable of learning a joint representation from FOBE and HOBE that can improve overall performance. However, in other cases, such as the Amazon link prediction task, the combination method appears to have significantly decreased performance. This effect is due to the increased number of hyperparameters introduced by the combination approach, which are determined not by the complexity of a given dataset, but are instead determined by the number and size of input embeddings. In the Amazon dataset, these free parameters lead to overfitting the combination embeddings.

3.6 Sensitivity Study

We select the MadGrades network to demonstrate how our proposed methods are effected by the sampling rate. We run ten trials for each experimental sampling rate, consisting of powers of 2 from 1 to 1024. Each trial represents an independent 50% holdout experiment. We present min, mean, and max observed link prediction accuracy.

To continue comparing FOBE and HOBE, it would appear that higher-order sampling is often able to produce better results, but that the algebraic distance heuristic introduces added variability that occasionally reduces overall performance. In some applications it would appear that this variability is manageable, as seen in our DBLP recommendation results. However in the case of link prediction on Amazon communities, this caused an unintentional drop when FOBE remained more consistent. Overall, FOBE and HOBE are fast methods that broaden the array of embed-

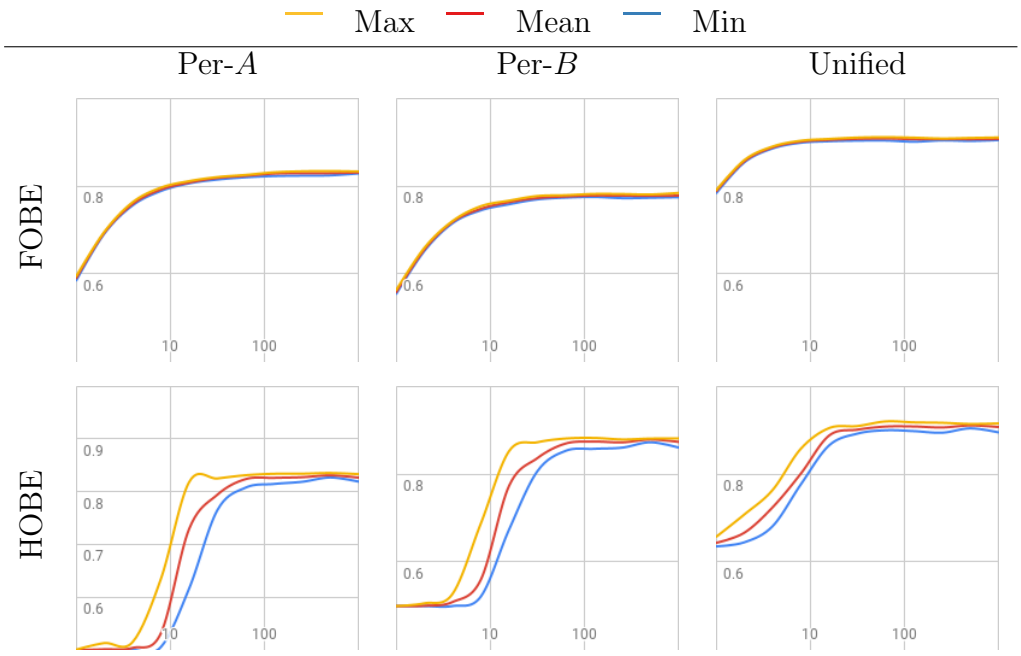


Table 3.5: Link Prediction Accuracy vs. Sampling Rate. Depicts the effect of increasing s_r from 2 to 1024 on the MadGrades dataset, running 10-trials of the 50% holdout experiment per value of s_r.

ding techniques available for bipartite graphs. While no method is clearly superior in every case, there exist a range of graphs and applications that are better suited by these methods.

Looking to the sensitivity study (Tables 3.5), we see the variability of HOBE is significantly larger for small sampling rates. However, we do observe that after approximately 32 samples per node, in the case of MadGrades, this effect is reduced. Still, considering FOBE does not exhibit this same quality, it is likely the variability of the algebraic similarity measure that ultimately leads to otherwise unexpected reductions in HOBES performance.

3.7 Conclusions

In this work we present FOBE and HOBE, two strategies for modeling bipartite networks that are designed to capture type-specific structural properties. FOBE, which captures first-order relationships, samples nodes in small local neighborhoods. HOBE, in contrast, captures higher-order relationships that are prioritized by a heuristic signal provided by algebraic distance on graphs. In addition we present two variants on an approach to learn joint representations that are designed to identify a "best of both worlds" embedding. We evaluate these methods against the state-of-the-art via a set of link prediction and recommendation tasks.

Our results indicate that none of the considered embeddings are clearly superior in every downstream embedding task, therefore we advocate for combination approaches, which can adapt to many different scenarios. This result is significant as practitioners often rely on a single embedding technique, and reuse embeddings across a wide range of tasks. In Table 3.2 we identify methods such as Deepwalk [168] may have significantly reduced performance on important sub-tasks, such as the B-Personalized task in our case, even if their overall performance is still strong. While the FOBE and HOBE methods are not a cure-all for embedding tasks, we do observe that they are consistently capable of capturing both A- and B-specific features for applications that rely on many same-typed comparisons. These methods are fast, easily parallizable, and capable of exceeding state-of-the-art performance on a range of downstream embedding tasks. While the bipartite graph embeddings remain an understudied problem, FOBE and HOBE can provide higher-quality type-specific latent features.

Chapter 4

Partition Hypergraphs with Embeddings

Abstract

Problems in scientific computing, such as distributing large sparse matrix operations, have analogous formulations as hypergraph partitioning problems. A hypergraph is a generalization of a traditional graph wherein "hyperedges" may connect any number of nodes. As a result, hypergraph partitioning is an NP-Hard problem to both solve or approximate. State-of-the-art algorithms that solve this problem follow the multilevel paradigm, which begins by iteratively "coarsening" the input hypergraph to smaller problem instances that share key structural features. Once identifying an approximate problem that is small enough to be solved directly, that solution can be interpolated and refined to the original problem. While this strategy represents an excellent trade off between quality and running time, it is sensitive to coarsening strategy. In this work we propose using graph embeddings of the initial hypergraph in order to ensure that coarsened problem instances retrain key structural

features. Our approach prioritizes coarsening within self-similar regions within the input graph, and leads to significantly improved solution quality across a range of considered hypergraphs.

Reproducibility: All source code, plots and experimental data are available at sybrandt.com/2019/partition .

4.1 Introduction

Hypergraphs provide the formalism needed to solve problems consisting of interconnected item sets. Similar to a traditional graph, the hypergraph has the added generalization that "hyperedges" may connect any number of nodes. Domains such as very-large-scale integration for creating integrated circuits [116], machine learning [261, 95, 259], parallel algorithms [48], combinatorial scientific computing [161], and social network analysis [197, 260] all contain significant and challenging instances of hypergraph problems. One important problem, *Hypergraph partitioning*, involves dividing the nodes of a hypergraph among k similarly-sized disjoint sets while reducing the number of hyperedges that span multiple partitions. In the context of load balancing, this is the problem of dividing logical threads (nodes) that share data dependencies (hyperedges) among available machines (partitions) in order to balance the number of threads per machine and minimize communication overhead. However, hypergraph partitioning is both NP-Hard to solve [138] and approximate [43].

Therefore, state-of-the-art partitioners apply heuristically-backed algorithms to overcome these inherent computational imitations [189]. The most common and effective technique is the *multilevel paradigm* [15, 195, 116, 37, 62]. Multilevel partitioners consist of three phases, referred to collectively as the *V-Cycle*: coarsening, the initial solution, and uncoarsening. We depict these phases in Figure 4.1. The

overarching idea behind this technique is to find a problem instance that shares key structural features with the input hypergraph, but is small enough to be partitioned directly. The initial solution to this small analogous problem can then be interpolated and refined to apply to the input hypergraph.

The small analogous problem is identified through an iterative *coarsening* process consisting of many levels. At each level, groups of similar nodes are identified, and each is "contracted" into a single merged node at the next more-coarse level. While grouping nodes, the goal is to identify self-similar regions of the current hypergraph so that the more coarse problem instances retain key structural features. Most commonly, these coarsening groups are formed by pairing nodes due to a similarity measure [62, 195]. An n-level algorithm is one that identifies only one pair of coarsening partners at each level [189], while a $\log n$-level algorithm pairs almost all nodes each time [62]. Coarsening stops once identifying a sufficiently small subproblem based on some threshold. The *initial solution* can then be identified directly using a typically-inefficient classical algorithm. Now, the solution is *uncoarsened* back through the levels in order to identify a solution to the original problem. Uncoarsening consists of three sub-phases: expansion, interpolation, and refinement. Expansion undoes the coarsening at a given level by "expanding" the current level's coarsened nodes with those contracted in the prior. Next, interpolation assigns each expanded node the partition label assigned to their corresponding coarse representation. Then, local refinement cheaply updates the partition labels among the expanded nodes in order to improve the overall solution quality for the next level. This process is repeated from the initial solution through all coarsening levels and back to the original hypergraph, which is accepted as the solution to the partitioning problem.

Because the strategy used to contract nodes determines the coarsening at each level, the quality of the initial solution, and the behavior of interpolation and

refinement during uncoarsening, we find that this single factor can dramatically effect partitioning quality. Other work exploring coarsening strategies, such as relaxation-based [195] or community-aware [100] coarsening, arrives with a similar conclusion.

4.1.1 Our Contribution

We propose *embedding-based coarsening*, a novel coarsening strategy that leverages graph embeddings to prioritize the contraction of self-similar regions of the input hypergraph in order to retain global structural features. This approach augments the existing strategy that contracts nodes based on their co-participation in small hyperedges by adding an embedding-based term that can break ties among similarly ranked coarsening pairs. A toy example of this phenomena is depicted in Figure 4.2, wherein three potential coarsening pairs are equally ranked by the traditional scheme, but embedding-based signals favor the pair that retains both key clusters.

The field of graph embedding is evolving rapidly, and the proposed embedding-based coarsening is designed to be agnostic with respect to any particular technique, provided that similarities between nodes are encoded via the dot product of embedding vectors. Specifically, our proposed technique accepts a precomputed embedding as an auxiliary input per-hypergraph, and we demonstrate that a wide range of existing embedding techniques improve partitioning performance similarly. Decoupling partitioning from embedding enables embedding-based coarsening to more easily benefit from future advances in machine learning techniques. In order to apply embedding techniques designed for classical graphs, we need a classical representation of each input hypergraph. The star-expansion [9] represents a hypergraph as an undirected bipartite graph wherein hyperedges from the original structure form a new layer of nodes. An edge between two nodes i and j in the bipartite structure indicates that

node i participated in hyperedge j within the original structure. As opposed to other classical representations like the clique-expansion, the star-expansion retains all relevant hypergraph information, and is scalable for large graphs [9]. Furthermore, existing embedding techniques specifically designed for bipartite graphs [224] apply to star-expanded graphs directly.

Given an input hypergraph and an embedding for each node of the input structure, embedding-based coarsening follows this outline at each coarsening level. First, each node is assigned a score equal to the highest dot product between its embedding and each of its neighbors. Nodes are visited in decreasing order by score. A visited node is matched from among its neighbors based on the product of their classical edge-wise score, and the dot product of each node's embedding. After matching nodes, based on whether we are performing n- or $\log n$-level coarsening, mated nodes are contracted. Newly coarsened nodes are assigned an embedding equal to the average embedding of all initial embeddings contained within the coarse representation.

We implement our proposed coarsening strategy in both KaHyPar [189], which is a n-level partitioner with state-of-the-art solution quality, as well as Zoltan [62], which is a parallel $\log n$-level partitioner with high quality and state-of-the-art speed. Furthermore, we compare the effect of various different embedding techniques, including Node2Vec [88], Metapath2Vec++ [66], and FOBE/HOBE [224], which were designed specifically for bipartite graphs. We additionally compare the effect of each embedding-based coarsening strategy with hMetis [115], Zoltan [62], PaToH [47], KaHyPar (with community-based coarsening [100]), and KaHyPar Flow (with both community-based coarsening and flow-based refinement [99])[1]. We compare performance of each partitioner across 96 hypergraph from the SuiteSparse Matrix Collec-

[1]Neither hMetis nor PaToH provide source code that would allow us to implement embedding-based coarsening for comparison. Instead, we can only use pre-compiled binaries for comparison purposes.

tion [58]. For each graph, we compute one embedding using each of the proposed techniques to serve as an auxiliary input across all trial. We compare quality across both the "cut" and "connectivity" objectives as well as for partition counts from 2 to 128 for each partitioner[2]. For each combination of experimental parameters we run 20 trials in order to compare the variance and establish significance with respect to different random seeds. Overall, we produce over 500,000 individual trials.

We find that embedding-based coarsening has a significant improvement over the state of the art that is especially pronounced for smaller partition counts (¡32). In some cases, this leads to a solution quality that is improved by as much as 400%. Because embedding-based coarsening replaces the traditionally random visit order with one that prioritizes self-similar regions of the hypergraph, we also observe a standard deviation of quality that is a small fraction of baseline methods. All experimental code, data, visualization scripts, and a database of all experimental results, including all hyperparameters per-trial, can be found at: sybrandt.com/2019/partition .

4.2 Notation and Preliminary Concepts

A hypergraph $H = (V, E)$ consists of nodes $v \in V$ and hyperedges $e \in E$. As opposed to a traditional graph, each hyperedge may contain any non-empty subset of V. The hypergraph partitioning problem is to divide V into k disjoint subsets of similar size while minimizing a given objective function. Two common objectives considered here are "cut" and "connectivity." Cut measures the number of hyperedges spanning more than one partition. If $\lambda(e)$ is the number of partitions spanned by edge

[2]hMetis does not supply a "connectivity" objective and is therefore omitted from that set of comparisons.

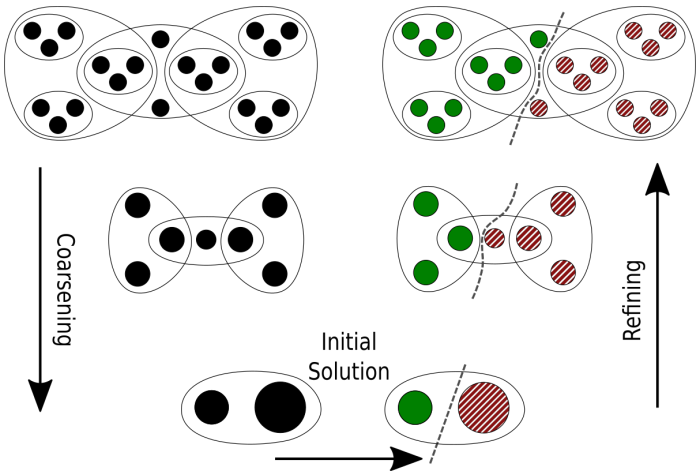


Figure 4.1: A standard V-cycle, consisting of coarsening, and initial partition, and uncoarsening. Node size corresponds to the weight of hypothetical coarse nodes. The dashed line demonstrates the initial partition and iterative local searches at each uncoarsening level. In this example, the multilevel hierarchy consists of three levels.

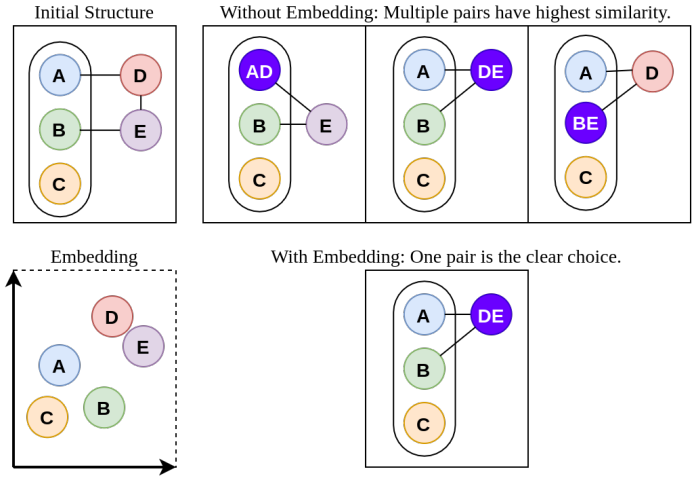


Figure 4.2: An example where embedding-based coarsening improves quality. Above we depict an example hypergraph and the set of coarsening pairs that would all receive the highest similarity score through the traditional edge-wise similarity function. When the embedding is introduced, we can prioritize the coarsening pair that best retains the initial global structure. In this case, we select the DE pair, as this still consists of a cluster of weight 3 connected to a cluster of weight 2.

e, then the cut objective is defined as:

$$\text{cut} = \sum_{e \in E, \lambda(e) > 1} 1 \tag{4.1}$$

The "connectivity" objective, also commonly referred to as "$k-1$," penalizes each edge by the number of spanned partitions. In the case where $k = 2$, this is equivalent to cut. Formally, the connectivity objective is defined as:

$$\text{connectivity} = \sum_{e} \lambda(e) - 1 \tag{4.2}$$

Weights. Although we consider unweighted input hypergraphs (all nodes and hyperedges count the same towards the objectives), the multilevel paradigm introduces weights to intermediate sub-problems. Each node and hyperedge has a corresponding weight (w_v and w_e) equal to one for the input hypergraph. During coarsening, if two nodes v_i and v_j are contracted into a new coarse node v', then $w_{v'} = w_{v_i} + w_{v_j}$. This new node v' will also be added to all edges originally containing either v_i or v_j, before removing those original nodes from the resulting coarse hypergraph. If two edges are "parallel" in the resulting coarse structure, meaning they contain the same subset of nodes ($e_1 = e_2$), they will be replaced with a new hyperedge e' containing the same nodes but with added weights: $w_{e'} = w_{e_1} + w_{e_2}$. When solving for cut and connectivity for intermediate sub-problems during the multilevel strategy, we introduce these weights into the objective:

$$\text{weighted cut} = \sum_{e \in E, \lambda(e) > 1} w_e \tag{4.3}$$

$$\text{weighted connectivity} = \sum_{e} (\lambda(e) - 1) w_e \tag{4.4}$$

One issue during coarsening is the potential for individual nodes to accumulate a disproportionate amount of weight. When this occurs, balanced partitioning can become impossible at the coarsest level, especially if one node's weight exceeds $|V|/k$. In order to avoid this negative effect, multilevel partitioners enforce a weight tolerance $w^{(T)}$, which is parameterized by the user. No coarsening partners may be contracted if their resulting coarse node would exceed this limit.

Imbalance Constraint. It is important to balance the number of nodes in the resulting partitions. Therefore, partitioners include an optimization constraint to determine how uneven the resulting partitions are allowed to be. For each partition $V_i \subset V$, given a predefined imbalance tolerance α, this constraint is defined as:

$$\sum_{v' \in V_i} w_{v'} \leq (1+\alpha) \left\lceil \frac{1}{k} \sum_{v \in V} w_v \right\rceil \tag{4.5}$$

Embeddings. We use the function $\epsilon : V \to \mathbb{R}^n$ to denote a pre-trained embedding. Conceptually, this is a lookup table that assigns a node in the input hypergraph to a real-valued n-dimensional vector. In our experiments we select $n = 100$.

4.3 Background and Related Work

Multilevel Partitioning. First introduced to speed up existing algorithms [20] and inspired by multigrid and multiscale optimization strategies [39], the multilevel method was quickly recognized as an effective method to improve the quality of hypergraph partitioning [117], and is currently considered to be one of the state-of-the-art methods for this problem [45]. As introduced in Section 4.1,the multilevel paradigm solves problems by following the v-cycle pattern that consists of coarsening, the initial solution, and uncoarsening. This approach is effective because coarse sub-problems

are easier to solve yet they retain global structural features of the original. Coarse sub-problems are created by iteratively merging multiple nodes at the current "finer" level into single nodes at the "coarser" level. Once sufficiently small, a partitioner can directly solve the coarsest problem instance using an algorithm that would normally be infeasible for large problems. The uncoarsening process then applies that solution through iteratively finer problem instances by expanding contracted nodes, interpolating coarse solutions onto finer problem instances, and refining intermediate solutions at each level. Once the uncoarsening process reaches the most-fine problem instance, the refined solution is accepted as the partitioning of the input hypergraph. Usually, at each level of the coarsening process all or almost all nodes have at least one merging partner, resulting in $\log n$ levels. This is the approach used by Mondriaan [235], hMetis2 [116], Zoltan [62], and PaToH [47]. However, KaHyPar [189] implements an n-level approach where at each level only one pair of nodes is contracted. The multilevel paradigm is the current gold-standard for hypergraph partitioning, having achieved an excellent trade off between time and quality. Unsurprisingly, most practical and state-of-the-art partitioners follow this paradigm, including all methods considered in this work. For an extensive review of both classical and hypergraph partitioning methods, we refer the reader to [45, 25].

Coarsening Strategies. State-of-the-art partitioners follow heuristic strategies to identify groups of nodes to contract during coarsening. A good coarsening strategy is one that groups together nodes that will ultimately share the same partition label, meaning that the coarser solution can be interpolated to the finer solution without a loss of quality. In practice, this loss of quality is to be expected, which is why local search refinement is common during uncoarsening. However, if global structural features are not preserved, the loss of quality during interpolation cannot be rectified through the fast local refinement process. Therefore, the choice of coarsening heuristic

is paramount.

Most heuristics used to identify nodes for contraction do so by scoring node pairs, and most partitioners, including Mondriaan [235], hMetis2 [116] and Zoltan [62], measure the edge-wise inner product, or some variation. The edge-wise inner-product is the Euclidean inner product of the weighted hyperedge incidence vectors [62]. Edge weights are defined formally in Section 4.2. Specifically, if w_e is the weight of hyperedge e, then the edge-wise inner product of nodes u and v is defined as:

$$\sum_{u,v \in e \in E} w_e \tag{4.6}$$

Although this approach is simplistic, it is also very computationally inexpensive and has provided a firm baseline. As mentioned, many variations exist, such as *absorption*, implemented in PaToH [47], and *heavy edge*, implemented in hMetis2 [116], Parkway [231], and KaHyPar [100], as well as a number of other normalization techniques, often based on node or hyperedge degree. Heavy edge, which is of particular interest due to its simple formulation and high performance, simply normalizes hyperedge weight by the expected degree of the resulting hyperedge following contraction. If $|e|$ is the number of nodes present in hyperedge e, then this score is:

$$S_E(u,v) = \sum_{u,v \in e \in E} \frac{w_e}{|e|-1} \tag{4.7}$$

One key limitation to the edge-wise score heuristics is that each only considers local information around each node. Therefore, global structural features can be collapsed during coarsening. This work seeks to use graph embeddings to provide this global information, however prior work has attempted to provide similar signals in alternate ways. Shaydulin et al. introduce *algorithmic distance* for hypergraphs, a

relaxation-based similarity measure that extends a similar approach from traditional graphs [50]. This measure treats nodes as entities in a mutually-reinforcing environment, which enables this technique to apply a fast relaxation-based approach to supply a coordinate per node. Conceptually this acts as a one-dimensional embedding, wherein two nodes receive a similar coordinate if their neighborhoods are similar. This similarity measure is used to quantify node similarities and assign weights to hyperedges.

Another approach to incorporate global information is *community-aware* coarsening, which uses clustering information to restrict matching between communities. This approach, which is implemented in KaHyPar, makes the assumption that nodes belonging to different clusters of the input hypergraph should never be contracted. The proposed clustering is performed by a fast global modularity-maximizing algorithm, leveraging the connection between partitioning and clustering. This modularity-based clustering, which groups star-expanded nodes within a bipartite representation of a hypergraph, identifies communities are internally dense and externally sparse [164], which is desired for a good partitioning. We note, and discuss further in Section 4.6, that the clusters found by this modularity-maximizing approach are similar to the self-similar regions within a graph embedding. However, in some scenarios the hard restriction to never merge nodes across communities appears to be too restrictive. Instead, embedding-based coarsening simply penalizes the contraction of nodes across clusters, allowing more flexible decisions for nodes along the periphery.

Refinement. While this work proposes a new coarsening strategy, important work also explores the refinement stage of uncoarsening, wherein each partitioner performs local search in order to improve the interpolated coarser solution on the finer level. The typical strategy is the *node-moving heuristic*, wherein each expanded node at

the newly refined level is given the option of switching partition label. A majority of hypergraph partitioners use a variation of Fiduccia-Mattheyses [75] or Kernighan-Lin [118] to perform these local searches [99, 235, 116, 62, 47, 231]. Recently, Heuer et al. introduced a flow-based refinement scheme for k-way hypergraph partitioning [99], extending similar approaches from graph partitioning [187]. This flow-based refinement, which is implemented in KaHyPar, is considered as a temperate case within our benchmark. As a result, we can compare the performance of embedding-based coarsening without flow-based regiment, and vice-versa.

Additional Partitioning Strategies. There are a few coarsening and partitioning strategies that are not included in our benchmark, but are worth additional discussion. *Memetic partitioning*, also proposed for KaHyPar, uses the principles of genetic algorithms to discover improved partitioning solutions [15]. This approach creates high quality partitions by iterating through different "generations" of solutions, starting with an initial generation produced by KaHyPar run multiple times with different seeds. From the initial set, multiple combination operators "breed" new solutions by combining some number of "parents" to form new solutions. Each iteration is designed to improve the population's average connectivity metric. Combination operators are specifically posed such that offspring solutions perform at least as good as its corresponding parents. While this approach is demonstrated to improve overall hypergraph partitioning quality, it does so by adding a meta process to the set of initial hypergraph solutions. We anticipate that adding embedding-based coarsening as a method for generating a high quality initial solution population may be a complimentary way to improve the overall process. *Aggregative coarsening* [196] uses ideas from algebraic multigrid, extending an unfinished attempt published in Sandia Summer Reports [44]. At each step of the coarsening process a set of seed vertices is selected. Each seed then becomes a center of an aggregate, with non-seeds assigned

to seeds using different aggregation rules. An aggregate at finer level forms a vertex at coarser level. Two aggregation rules, based on inner product matching and stable matching were explored. Our embedding-based coarsening can be used within the aggregative coarsening to inform the aggregation rules.

4.3.1 Graph Embeddings

Our embedding-based coarsening accepts embeddings for each node of the input hypergraph as an auxiliary input. While we make the assumption that node similarity is encoded through the dot product of embeddings, we do not depend on any particular embedding technique. However, there are a range of embedding methods that we consider in our benchmark due to their scalability and applicability to the bipartite graphs produced by the star-expansion process. At a high level, graph embeddings assign a real-valued vector of fixed size to each node (and sometimes each edge) of an input graph. Therefore, techniques such as non-negative matrix factorization, principal component analysis, or even algebraic distance [195] can all apply as embeddings from the perspective of embedding-based coarsening. While our early experiments explored all of these and more, we found the most significant improvements when using neural-network-based embeddings.

In all of the considered embedding techniques, various node features are encoded via the dot product of node embeddings. Furthermore, new techniques are published frequently that identify new ways to encode latent node footers. Rather than depend on a particular graph embedding technique, this work simply assumes that some measure of global graph structure is encoded via the dot product of embeddings, meaning that two nodes with a higher dot product of embeddings will be more similar. In this manner, new advances in graph embedding, or fine-tuned versions of

existing algorithms for particular graphs, can be introduced into our proposed strategy. Importantly, this work does not seek to establish any graph embedding technique as inherently better for hypergraph partitioning. Instead, we find that all considered embeddings greatly improve solution quality.

Neural Graph Embedding. The Deepwalk graph embedding [168], which applies the skip-gram model [155] to random walks of nodes, marks the beginning of neural network graph embeddings. The node2vec approach [88] modifies Deepwalk to parameterize random walk behavior, allowing walks to explore local regions or broad swaths of a graph. In doing so, Grover et al. identify that node2vec graph embeddings can encode both homophilic and structural latent features. Tsitsulin et al. generalize the formalism across a range of random-walk based graph embedding techniques, noting that community-based, role-based, and structural features of nodes can all be encoded in a single unified framework [232].

Bipartite Embeddings. The above graph embeddings were designed with general graphs in mind. However, a few works have specifically proposed embedding techniques adapted for bipartite graphs. These techniques receive specific focus in this work as we produce bipartite graphs when performing the star-expansion of input hypergraphs. Sybrandt et al. [224] explore a number of such techniques when presenting First- and Higher-Order Bipartite Embedding (FOBE and HOBE), including BiNE [79], and Metapath2Vec++ [66]. Bipartite Network Embeddings (BiNE) generates walks that are weighted by a network centrality measure, and uses these walks to fit both explicit and implicit relationships simultaneously. However, this method performs similar to random embeddings for a range of link prediction tasks in [224], and for this reason is omitted from the benchmark in this work. Metapath2Vec++ extends the Deepwalk framework in two ways. First, random walks are restricted to follow particular patterns of nodes. Second, node embeddings are computed using

different parameters based on that node's type. In the case of bipartite graphs, the only random walk pattern is that of alternating node types, but the added flexibility gained by parameterizing each side of a bipartite graph separately leads to improved performance in [224].

Because most readers will not be familiar with FOBE and HOBE, and because we find these techniques are very beneficial to embedding-based coarsening, we summarize these techniques here. FOBE samples all edges in a bipartite graph, as well as every indirect connection between two nodes on the same side. Formally, if $G = (V, E)$ is a bipartite graph, $\Gamma(x)$ is the neighborhood of node x, and $u, v \in V$ are nodes, then FOBE assigns a sampled score for the u, v pair as follows:

$$\mathbb{S}^{(\text{FOBE})}(u, v) = \begin{cases} 1 & \Gamma(u) \cap \Gamma(v) \neq \emptyset \\ 1 & uv \in E \\ 0 & \text{otherwise} \end{cases} \tag{4.8}$$

Embeddings are fit such that the sigmoid of the dot product of embeddings match these binary samples. Specifically, this technique minimizes the following loss associated with a single u, v pair where $\epsilon(x)$ indicates the learned embedding of node x:

$$\begin{aligned} \mathcal{L}^{(\text{FOBE})}(u, v) &= \sigma(\epsilon(u), \epsilon(v)) \log \left(\frac{\mathbb{S}^{(\text{FOBE})}(u, v)}{\sigma(\epsilon(u), \epsilon(v))} \right) \\ \text{where } \sigma(x) &= \frac{1}{1 + e^{-x}} \end{aligned} \tag{4.9}$$

HOBE, in contrast, learns higher-ordered relationships that are weighted using algebraic distance, the same underlying technique used within relaxation-based coarsening [195]. Algebraic distance is an iterative relation process that places all nodes on the unit interval, such that similar nodes are more likely to share similar

algebraic coordinates. Specifically, the algebraic coordinate of node u is determined by this iterative process:

$$\mathbf{a}_{i+1}(u) = \frac{1}{2}\left(\mathbf{a}_i(u) + \frac{\sum\limits_{v\in\Gamma(u)} \mathbf{a}_i(v)|\Gamma(v)|^{-1}}{\sum\limits_{v\in\Gamma(u)} |\Gamma(v)|^{-1}}\right) \tag{4.10}$$

Here, $\mathbf{a}_0$ is randomly initialized, and the algebraic coordinate for u is determined after a fixed number of steps t. We summarize the algebraic distance between two nodes across multiple instances of the above relaxation process started using various random seeds. If we run $R = 10$ random restarts, then the similarity between nodes u and v is:

$$\begin{aligned} s(u,v) &= \frac{\sqrt{R} - d(u,v)}{\sqrt{R}} \\ \text{where } d(u,v) &= \sqrt{\sum_{r=1}^{R}\left(\mathbf{a}_t^{(r)}(u) - \mathbf{a}_t^{(r)}(v)\right)^2} \end{aligned} \tag{4.11}$$

Using algebraic distance, HOBE weights similarities between nodes in the following manner:

$$\begin{aligned} \mathbb{S}^{(\text{HOBE})}(u,v) &= \begin{cases} \alpha(u,v) & \Gamma(u)\cap\Gamma(v) \neq \emptyset \\ \max\left(\begin{array}{l}\max\limits_{x\in\Gamma(v)} \alpha(u,x), \\ \max\limits_{x\in\Gamma(u)} \alpha(x,v)\end{array}\right) & uv \in E \\ 0 & \text{otherwise} \end{cases} \\ \text{where } \alpha(u,v) &= \max_{x\in\Gamma(u)\cap\Gamma(v)} \min\left(s(u,x), s(v,x)\right) \end{aligned} \tag{4.12}$$

Then, HOBE learns embeddings to match the above samples to the dot product of embeddings through the mean-squared-error objective. The loss associated

with a u, v pair is:

$$\mathcal{L}^{(\text{HOBE})}(u, v) = \left(\mathbb{S}^{(\text{HOBE})}(u, v) - \max(0, \epsilon(u)^\intercal \epsilon(v))\right)^2 \tag{4.13}$$

Combination Embeddings. To demonstrate the ability for embedding-based coarsening to apply to any given embeddings, we explore the combination approach also presented by Sybrandt et al. in [224]. This method learns a joint representation for each node given multiple pretrained embeddings. This technique does not rely on any random walk strategy, and instead learns a unified embedding per-node given the edge list as a set of embeddings per node. The particular model combines a link-prediction objective with an auto-encoding objective, and in doing so ensures that the resulting joint embedding captures relevant structural signals that are needed to reproduce both the edge list as well as the input embeddings. This technique is very similar to that presented by Wang et al. [242] in that it consists of two connected auto encoders. The result of this method is an embedding that merges the structural features present in a range of embeddings while preserving any useful distinct features from across the set. We direct the reader to [224] to find the specifics of this approach.

Deep Learning Graph Embedding. In addition to the above techniques, which are generally fast, scalable, and parallel sizable, there are another set of deep-learning embedding techniques that apply larger models to the problem of graph embedding. One popular technique, the graph convolutional network [122], constructs a neural network in the same structure as the input graph, and embeddings are derived by a "message-passing" function that distributes node features among neighborhoods. Another technique by Cao et al. learns deep representation by first constructing a large co-occurrence matrix from a process of "random-surfing" following by deep auto encoders [46]. A similar auto-encoder-based approach is presented by Wang et

al. [242], wherein a pair of deep auto-encoders both encode nodes independently, as well as ensure that similar nodes are assigned similar embedding. While these deep-learning techniques do achieve high quality results for relatively small graphs, these techniques are less scalable than the previously discussed class of algorithms, due to their larger model structure and the accompanying need for more graph samples. While these techniques could certainly improve the quality of embedding-based coarsening for some hypergraphs, we designed our proposed technique to be independent of any particular embedding, and evaluated our technique over a large collection of hypergraphs and scenarios. As a result, the analysis of deep-learning graph embedding techniques was infeasible for this work.

4.4 Embedding-Based Coarsening

Embedding-based coarsening begins with a user-supplied hypergraph as well as an embedding of each node. For instance, we use the star-expansion [9] of the hypergraph in order to apply a range of embedding techniques designed for classical graphs. During coarsening, nodes are visited in an order determined by the embeddings of each node's neighborhood. When visited, an unmatched node is paired with whichever neighbor maximizes a combined measure of edge-wise inner product as well as embedding dot product. After identifying matches, paired nodes are contracted into new coarse nodes, which are assigned an embedding equal to the average of all its contracted embeddings. Because embedding-based coarsening preserves more global structural features than other methods, the initial partitioning solution is more applicable to the large-scale graph, resulting in higher partitioning quality. We implement embedding-based coarsening in both Zoltan [62] and KaHyPar [189], and explore a range of embedding techniques, including node2vec [88], MetaPath2Vec++ [66],

FOBE and HOBE [224], as well as merged embeddings from among this set.

Node Visit Order. We begin matching nodes in an order that tries to prioritize self-similar regions of the input hypergraph. Specifically, a node is a good candidate for being contracted at the current level if it shares a hyperedge with a partner that has a very similar embedding. This indicates that both nodes share many global structural features that would be preserved in their coarsened replacement. However, it is also important to reduce the weight of the resulting coarse nodes. While we also apply more explicit weight-based limitations below, maintaining the balance of coarse node weights begins with adding a weight normalization to this embedding-based similarity score. Otherwise, very dense regions of the network will be contracted into extremely imbalanced and heavy nodes before the rest of the hypergraph, which can eventually invalidate the imbalance constraint. Note that Section 4.2 contains more thorough definitions for the embedding function ϵ and node weight w, as well as the rest of the notation used in this section. Using these concepts, we can order nodes based on how similar each is to its closest neighbor. Specifically, we order each node u with respect to the following:

$$S_O(u) = \max_{v \in \Gamma(u), u \neq v} \frac{\epsilon(u)^\intercal \epsilon(v)}{w_u w_v} \tag{4.14}$$

Scoring Contraction Partners. When visiting node u at a given level of coarsening, we must select a neighbor v with which it will contract into a new coarse node in the following level. To do so, we assign a score to each neighbor of u, and select the node with the highest score to match with. We assign scores based on a combination of the KaHyPar "heavy edge" scoring function [100], as summarized in Section 4.3, as well as the dot product of embeddings. The heavy edge scoring function increases the score of hyperedges with fewer nodes. In real-world applications, this can correspond to "niche" communities that tend to carry more meaning for those involved. We ad-

ditionally penalize this score by the node's weights in order to reduce the imbalance of the resulting coarse nodes. Specifically, we assign a score to neighboring nodes u and v during the matching process equal to:

$$S_\epsilon(u,v) = \left(\frac{\epsilon(u)^\intercal \epsilon(v)}{w_u w_v}\right)\left(\sum_{e\in\Gamma(u)\cap\Gamma(v)} \frac{w_e}{|e|-1}\right) \tag{4.15}$$

Note that in order for a node pair to receive a high S_ϵ score, they must both share low-participation hyperedges as well as global structural embedding-based features. This way, embedding-based coarsening allows us to break ties between multiple nodes that all co-occur in similar intersections of similarly weighted hyperedges, which has the effect of breaking ties, as depicted in Figure 4.2. Additionally, this measure provides a sorting criteria who's relative values is more important than its absolute value. For this reason we observe a significant benefit by *not* normalizing the dot product value. While some embedding techniques encode node similarity through cosine similarity, which normalizes the dot products between nodes, others do not. In these cases, the relative magnitudes of embedding dot products is a valuable signal for determining coarsening partners.

Imbalance Constraint. As previously stated, it is also important to ensure that the weight of coarsened nodes remains reasonably balanced so that no coarse node becomes so "heavy" that the overall partitioning becomes imbalanced. To address this, we only match nodes that will produce coarse nodes below a given weight tolerance. Different partitioners have different strategies for selecting an imbalance constraint, but typical values are between $0.5\% - 1\%$ of the input graph, normalized by the desired number of partitions. We accept the weight tolerance $w^{(T)}$ to be a hyperparameter determined by the partitioner. Then, when matching nodes u and v, we disqualify any pair such that $w_u + w_v > w^{(T)}$.

Embeddings. We explore a range of embedding techniques to produce a vector per node that functions as ϵ in the above scoring functions. For the sake of comparison, we choose 100-dimensional embeddings for all cases and for all hypergraphs. We only embed each considered hypergraph once per method, and interpolate intermediate representations for coarsened nodes. This interpolation consists of the average of all initial node embeddings present in the coarsened node. For instance, if the coarse node u has weight w_u, then that number of nodes from the input hypergraph have been accumulated into u. These initial nodes, $v_1, \ldots, v_{w_u}$ each have embeddings that were supplied in the initial hypergraph embedding. Therefore, we define $\epsilon(u)$ to be the following in the case where u is a coarse node that *does not appear* in the initial embedding:

$$\epsilon(u) = \frac{2}{w_u} \sum_{i=0}^{w_u} \epsilon(v_i) \tag{4.16}$$

Implementations. We implement our algorithm in both KaHyPar [189], the n-level partitioner, as well as Zoltan [62], the $\log n$-level partitioner. These two partitioners are considered as other alternatives such as PaToH [47] and hMetis [115] do not provide open source implementations. In each, embedding-based coarsening consists of only a few hundred lines of code, demonstrating that both partitioners are easily expandable for new coarsening algorithms. For ease of development, we use singletons to manage the state of the embedding, and overwrite functions related to scoring neighboring nodes during the coarsening process. We additionally implement a partitioner independent prepossessing step to convert a hypergraph into a star-expanded classical graph in order to apply existing graph embedding techniques. The output of these graph embeddings is supplied as an auxiliary input to the singleton embedding manager. Overall, this separation of embedding and partitioning allows easy experimentation and adaptation with respect to new and constantly changing embedding

techniques. The specific algorithm we implement, which summarizes the above steps, is summarized in Procedure 2

Procedure 2 Embedding-based Coarsening.

Output: Produces a set of (u, v) pairs to be contracted in the next level of coarsening.

$M_u \leftarrow \emptyset \; \forall u \in V$ ▷ M is the matching array.
Sort $u \in V$ in decreasing order by $S_O(u)$. ▷ Eq. (4.14)
for $u \in V$ **do**
    if $M_u = \emptyset$ **then**
        $p \leftarrow \emptyset$ ▷ p will be matched with u.
        $s \leftarrow -\infty$ ▷ s is the score associated with p.
        for $v \in \Gamma(u)$ **do**
            if $v \neq u$**and** $M_v = \emptyset$**and** $w_u + w_v < w^{(T)}$ **then**
                $t \leftarrow S_\epsilon(u, v)$ ▷ Eq. (4.15)
                if $t > s$ **then**
                    $s \leftarrow t$
                    $p \leftarrow v$
        if $p \neq \emptyset$ **then**
            $M_p \leftarrow u$ ▷ Match u and p.
            $M_u \leftarrow p$
Contract nodes according to M.

An additional implementation note is necessary for managing embedding behavior during the recursive bisection process of Zoltan, which identifies larger numbers of partitions by iteratively solving the 2-partition problem on iterative halves of the input hypergraph. Simply put, node indices during the recursive bisection process are remapped to start at zero for each recursive partitioning call, which requires the embedding singleton to access the logic of this routine. KaHyPar, in contrast, solves the k-partitioning problem directly at the point of initial solution, and does not perform recursive bisection, and therefore does not require extra engineering. While important for any wishing to implement embedding-based coarsening in the context of recursive bisection, this technical detail does not modify the overall behavior of the proposed algorithm.

Runtime Impacts. Embedding-based coarsening comes with two runtime increases that are not present in the fast edge-wise coarsening that is typically used by KaHyPar and Zoltan. Firstly, one must perform a graph embedding to learn ϵ. Secondly, at each level of coarsening, we sort V in accordance to embedding-based signals. Graph embedding, in general, is an expensive machine learning operation, requiring significant time and memory to sample a graph and learn embeddings for each node. However, because the proposed embedding-based coarsening algorithm is independent of any particular embedding technique, the specific resources and time needed to produce a graph embedding are subject to change. However, there are a few broad patterns that most embedding methods follow. Graph embeddings are learned from a set of samples. These samples can be the edges of the graph itself [139], observations determined based on first- or second-order relationships [224, 228], or random-walks of the graph [66, 88, 168]. In each case, the observation capturing process is linear with respect to the size of the graph. Additionally, these observations can often be collected in parallel. Next, the observations are formulated into batches for a neural network to learn embeddings. Each observations is viewed once-per-epoch, and effects the learned weights of a gain model. Therefore, the complexity of training is equal to the size of the graph times the complexity of performing back-propagation of a particular model. Embeddings can also be paralleled, both by GPU acceleration, as well as through multi-node computation [176]. While the embedding process overall is certainly expensive, the coarsening algorithm proposed in this work only requires one embedding of the input graph as a prepossessing step. This may not be feasible for applications that must partition thousands of midsize hypergraphs daily, but is likely worth it for any application the relies more on the quality of the resulting partition.

The second difference in runtime comes from the sorting used to prioritize

coarsening partners during each step of the proposed algorithm. At each iteration, we visit each node to find its most-similar neighbor in terms of node embedding, and then order nodes by this measure. In contrast, classical coarsening randomly orders nodes before identifying partners. While this process introduces the overhead of sorting, we find that removing randomness to prioritize self-similar hypergraph regions can significantly improve partitioning quality while decreasing the quality variance. In practice, many practitioners re-partition a hypergraph many times in order to find the highest-quality partition. By incurring the cost of sorting, these practitioners save the cost of multiple trials.

4.5 Experimental Design

We implement embedding-based coarsening in both KaHyPar [189] and Zoltan [62], and compare the result quality against KaHyPar with community-based coarsening [100], KaHyPar with community-based coarsening and flow-based refinement [99], Zoltan with standard coarsening [62], PaToH [47], and hMetis [115]. For both KaHyPar and Zoltan with embedding-based coarsening, we compare embeddings produced by Node2Vec [88], Metapath2Vec++ [66], FOBE and HOBE [224], as well as a combined FOBE+HOBE embedding, and a combined Node2Vec, Metapath2Vec++, FOBE, and HOBE embedding. The combinations are trained using the semi-supervised joint embedding technique also presented in [224], which merges retrained embeddings through a combination of auto-encoding and link-predictive objectives. We selected the FOBE and HOBE combination as this produces a high quality embedding in prior work [224]. We then wanted to explore a new combination with the whole range of considered embeddings. Additionally, when comparing performance of embedding-based coarsening within KaHyPar, we compare both with

and without flow-based refinement. Overall, we explore 18 different partitioning settings with embedding-based coarsening, and five different partitioners with traditional coarsening strategies.

For each of the 23 total partitioner configurations, we explore 96 total hypergraphs. Eighty-six of these are supplied by the SuiteSparse Matrix Collection [58]. These matrices span a range of domains including social networks, power grids, and linear systems. We interpret each matrix $\mathcal{H}$ as the incidence matrix of a hypergraph. In doing so, we consider each row to represent a node, each column to be a hyperedge, and a nonzero value in $\mathcal{H}_{ij}$ to indicate node j participates in hyperedge i. We additionally include ten synthetic hypergraphs that were designed to test the robustness of the coarsening process, extending a similar approach from graphs [184]. These graphs are a mixture of graphs that are weakly connected between each other, with less than 1% of edges connecting different graphs in the mixture. In multilevel setting, this can cause the coarsening process to incorrectly contract edges between different graphs in the mixture, resulting in uneven coarsening, overloaded refinement and worse quality of the final solution. This structure can be found in many real-world graphs, including multi-mode networks [229] and logistics multi-stage system networks [213]. We introduce additional complexity by adding additional $< 1\%$ random edges (denoted in the online appendix as "W/ Noise"). Full graphs, as well as scripts used to generate them are available in the online appendix. Summary statistics for each graph are supplied in Table A.1.

For each partitioner and hypergraph combination, we explore both the "cut" and the "connectivity" objective, which can influence the initial solution, as well as some decisions during refinement across the considered benchmark[3]. Additionally,

[3]hMetis cannot optimize the "connectivity" objective, and is therefore omitted from that portion of the analysis.

we explore a number of partitions (k) for powers of 2 from 2 to 128. For each partitioner, objective, and k-value combination, we run at least twenty trials with different random seeds in order to explore the stability of each scenario. Overall, we compute over 500,000 different experimental trials across our wide benchmark, and for each trial we record all relevant hyperparameters and quality results in a database download supplied in our online appendix.

Metrics. In order to understand aggregate system performance, we report a range of summary statistics for each proposed method. We are primarily concerned with partitioning performance, as quantified by the value of the considered objective value at the end of the multilevel paradigm. However, different hypergraphs have substantially different optimal objective values. Therefore, we report *improvement* statistics between two considered partitioners, with one acting as a baseline for the consideration of the other. A value greater than 1 indicates a *reduction* in the considered partitioner when compared to the baseline across the same hypergraphs.

Formally, if P is a partitioner configuration, including algorithm, embedding method (if applicable), k, and objective function, and H is a hypergraph then let $P(H)$ be the resulting value of the objective function given H and a new random seed. Then, let G be a summary statistic, such as mean, min, max, or standard deviation. We apply G over τ trials of a given partitioner with the same input and different random seeds. The improvement of P with respect to baseline method P_B for a single hypergraph is determined to be:

$$I(P, P_B, G, H) = \frac{G(P_B(H)_1, \ldots, P_B(H)_\tau)}{G(P(H)_1, \ldots, P(H)_\tau)} \tag{4.17}$$

Note that the formulation above places the baseline partitioner in the numerator because an "improvement" is quantified as a *decrease* in objective value.

Therefore, if the proposed partitioner P produces consistently *lower* objective values than P_B, then I will be a number greater than 1.

When comparing two partitioners across the entire benchmark of hypergraphs D, we compute the macro-summary. This means that we first apply the summary statistic G to each hypergraph's trials separately, before averaging the results together. Formally, the macro-summary is defined as:

$$\mathcal{I}(P, P_B, G) = \frac{1}{|D|} \sum_{H \in D} I(P, P_B, G, H) \tag{4.18}$$

When making these comparisons, we select P and P_B pairs such that both partitioners are optimizing the same number of partitions using the same objective. Additionally, we explore summary functions G including mean, min, max, and standard deviation. While mean indicates average quality, min and max indicate worse- and best-case performance, while the standard deviation explores the variance in the resulting partitioners with respect to the random seed.

4.6 Results

We present a range of result summaries following the experimental design discussed above. For a more in-depth look at our results, we present additional data regarding each hypergraph, and each trial in our online appendix.

To begin our analysis, we summarize the performance improvement of each embedding-based coarsening implementation compared to its respective baseline. For instance, we compare KaHyPar with embedding-based coarsening, using FOBE embeddings, against KaHyPar using community-aware coarsening. We also compare KaHyPar with flow-based refinement, as well as Zoltan with and without embedding-

(a) Average connectivity improvement.

# Parts(k):	2	4	8	16	32	64	128
KaHyPar	8%	13%	10%	6%	4%	3%	1%
KaHyPar(flow)	9%	11%	4%	2%	3%	2%	0%
Zoltan	48%	28%	15%	14%	9%	5%	3%

(b) Average cut improvement.

# Parts(k):	2	4	8	16	32	64	128
KaHyPar	8%	16%	9%	1%	3%	1%	0%
KaHyPar(flow)	10%	11%	3%	1%	1%	1%	-1%
Zoltan	51%	45%	51%	41%	31%	14%	8%

Table 4.1: Improvement for each implementation of embedding-based coarsening when compared to its corresponding baseline for both the "cut" and "connectivity" objectives. Results each use the FOBE embedding instance of embedding-based coarsening. Performance numbers correspond to $\mathcal{I}$ macro-summaries (Eq. 4.18) where $G = \text{mean}$.

based coarsening. These results are summarized using the macro-improvement statistic $\mathcal{I}$ (Eq. 4.18), and with the trial summary statistic $G = mean$. Improvements as a percentage for both "cut" and "connectivity" objectives for all considered numbers of partitions (k) in Table 4.1.

The most striking result in this small collection of summaries is the inverse relationship between improvement and k. As the number of partitions increases, the advantage of embedding-based coarsening decreases. This is due to the manner that we create interpolated embeddings for coarse nodes. As detailed in Section 4.4, when a newly coarsened node is introduced at a new level of the coarsening process, it is assigned an embedding equal to the average of the initial embeddings it contains. This has the effect of "smoothing" the embedding space at the coarse level. As a result of this smoothing, only major variances between nodes will be captured at the point of initial solution. For instance, if a hypergraph structure has a set number of key clusters, it is hard for embedding-based coarsening to identify anything else at the coarsest level. Higher values of k, larger than the number of identified clusters,

therefore do not benefit from this technique.

Future work looking creating more useful coarse node representations is likely to address this problem. However, simple solutions such as embedding coarse graph instances has significant challenges. For instance, we find that small problem instances result in poor embedding convergence across all considered embedding techniques. Therefore we observed in initial trials, re-embedding coarse graphs dramatically *decreased* result quality. Additionally, graph embeddings are computationally expensive, and performing any non-constant number of embeddings is likely to be infeasible for any real-world problem instance.

We continue our comparison of embedding-based coarsening across a range of baselines in Figure 4.3. Here we compare Zoltan with embedding-based coarsening, KaHyPar with embedding-based coarsening and flow-based refinement (the better performing KaHyPar implementation), against all baseline methods. We additionally explore a range of summary statistics G including mean, best-case (min), worse-case (max), and standard deviation. To easily compare all partitioners, we use KaHyPar with flow-based refinement as the baseline (P_B) for all methods. Therefore, KaHyPar with flow always scores a one, denoted by the dashed line in each plot, and an improvement over this baseline is indicated by a macro-improvement $\mathcal{I}$ greater than one.

We observe a similar negative relationship between k and improvement across the benchmark in Figure 4.3 as was seen in Table 4.1. When considering the connectivity objective for k values of 2 and 4, we interestingly observe that both the KaHyPar and Zoltan implementations with embedding-based coarsening outperform the baseline. This is especially important for Zoltan, which greatly under performs the baseline without our proposed coarsening. When looking at best-base performance ($G = \max$) we observe that the negative trends with respect with k is less

pronounced for KaHyPar and the connectivity objective. This trend demonstrates the consistent ability of embedding-based coarsening to identify solutions that are of a higher quality than any found by any considered baseline, and suggests that practitioners willing to accept a quality-for-speed trade-off can find substantial performance gains with our proposed technique.

Examining the standard deviation results shown in Figure 4.3, we observe that embedding-based coarsening greatly improves the standard deviation of possible results for a given hypergraph (shown where $G = \text{std}$). This decrease in variance comes from the deterministic node-visit order, which replaces a typically random ordering. As a result, the standard deviation of KaHyPar with embedding-based coarsening can be reduced by over an order-of-magnitude in some cases. Because many applications run multiple partitioning trials with various random seeds in order to find a top-performing result [230], we find that this decrease in variance enables these applications to run fewer trials while retaining the same confidence in their performance.

Comparison with Community-aware Coarsening. Embedding-based coarsening attempts to merge together self-similar regions of the input hypergraph with respect to the structural signals provided by node embeddings. In contrast, community-aware coarsening restricts the contraction of nodes that do not share a cluster assignment in the original hypergraph. While these two approaches are very similar, they both promote contractions within self-similar regions of the original hypergraph, we find that embedding-based coarsening is a more flexible constraint. Embedding-based coarsening simply penalizes nodes that do not share structural features, but may still merge seemingly dissimilar neighbors if no better options are found. Because of this relaxation, we find that embedding-based coarsening outperforms community-aware coarsening in a range of scenarios. This behavior, which we first report in aggregate

in Table 4.1, is explored in depth in Figure 4.4. In this example, each considered hypergraph is listed, and graph-wise performance summaries I (Eq. 4.17) are depicted for each. The specific properties of each graph are briefly summarized in Table A.1, and more information regarding each hypergraph is available online.

In the summary table, we demonstrate that for low k-values, that embedding-based coarsening can improve result quality over community-aware coarsening by around 10% for $k = 2, 4, 8$. When viewing the per-hypergraph results, we see a more detailed picture. Some hypergraphs with particularly useful structural features, such as then hypergraph constructed from the enron email dataset, the eu email dataset, or the difficult and noisy merged hypergraphs, *can find partitioning solutions with a connectivity objective that is between one half and one fourth of the community-aware baseline.* For many other graphs this improvement is a modest few percentage points, while other graphs are relatively unchanged. For these graphs, we find that the community-detection solution found by KaHyPar provides nearly the same information as the selected graph embedding, leading to no improvement. Only a small handful of graphs are substantially worsened by this proposed technique when compared to the community-aware baseline. For instance, Nemsemm2, a sparse matrix corresponding to a linear program, is partitioned almost three-times worse using embedding-based coarsening. The incidence matrix of this hypergraph is nearly block-diagonal, which results in significant hyperedge-wise features that are not translated into an embedding, as disjoint graph regions are often embedded in overlapping spaces. In contrast, Nemswrld is another linear-program sparse matrix published by the same group, but is less block-diagonal and receives an statistically significant average improvement of about 33%.

Comparison Across All Partitioners. Our large table in Figure A.2 depicts the average improvement of each proposed embedding-based coarsening partitioner con-

figuration against each baseline for the connectivity objective. The numbers in each cell correspond to the macro-summary $\mathcal{I}$ using G = mean to summarize trials. For space limitations, we only show this one large table, but online we present similar tables for the cut objective, as well for the min, max, and standard-deviation summarizes. Additionally, we include a per-hypergraph plot similar to Figure 4.4 for each cell. When examining the included table, however, we see clear trends that are replicated in each online table. All considered embeddings improve performance similarly, with FOBE and HOBE performing marginally above the other methods in some cases. We observe that Zoltan is the "easiest" baseline partitioner, while KaHyPar with flow-based refinement is the most challenging. Because KaHyPar with flow-based refinement produces higher quality partitions than Zoltan in prior work [99], it is notable that some instances of Zoltan with embedding-based coarsening can achieve similar quality.

When looking across the KaHyPar trials, we see that embedding-based coarsening without flow-based refinement can outperform community-aware coarsening with the most expensive flow-based refinement. This result confirms the intuition, initially discussed in Section 4.1, that the coarsening process one of the most fundamental operations in multilevel partitioning.

Across all considered implementations of embedding-based coarsening we still observe a decrease in performance for larger values of k. As previously discussed, this derives from the smoothed embedding space produced by iterative averages of coarse nodes. It is worth noting that this smoothing effect produces results that are most similar to KaHyPar with its broad community-aware coarsening. In contrast, embedding-based coarsening still outperforms Zoltan and PatoH, partitioners that do not account for global structural properties in a similar way.

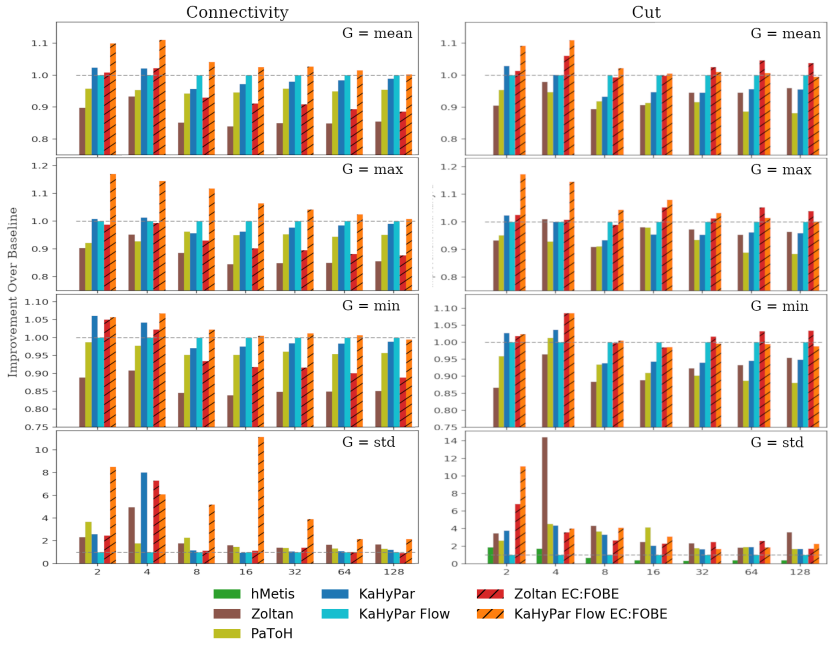


Figure 4.3: The above depicts the relative performance of various partitioners, each using KaHyPar with flow-based refinement as a baseline. The results correspond to macro-summaries $\mathcal{I}$ (Eq. 4.18), where a value of 1, indicated by the horizontal dashed line, is baseline performance of P_B. We explore different summary statistics G, including mean, max, min, and standard deviation.

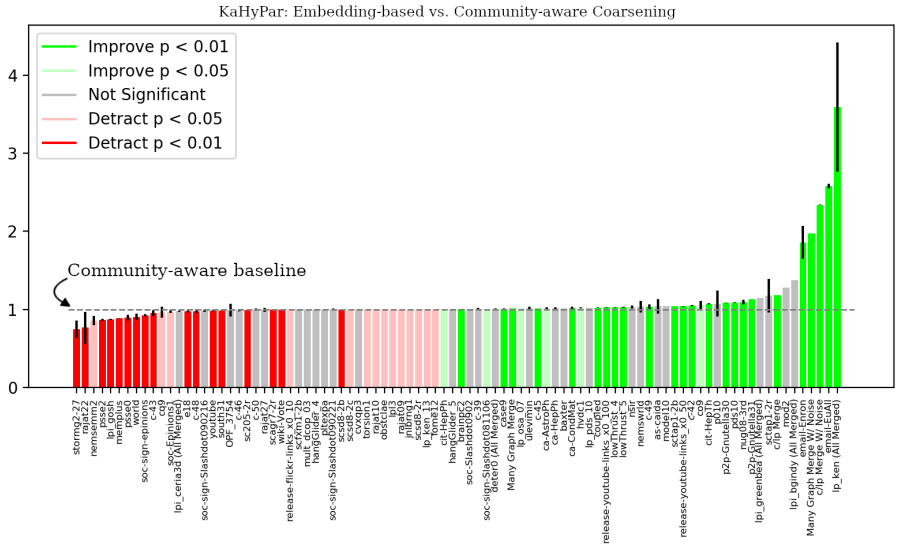


Figure 4.4: The above depicts per-hypergraph summary statistics, I from Eq. 4.17, comparing KaHyPar with embedding-based coarsening (P) to KaHyPar with community-based coarsening (P_B). We use the mean over trails as our summary statistic G, as denoted by the height of each bar. A value higher than 1, which is emphasized by the dashed line, indicates better solution quality. The small black bar at the top of each graph indicates the standard deviation of trials, and the color of each bar indicates the statistical significance, where a more saturated color indicates a lower p-value. Hypergraph names are supplied across the horizontal axis, and graphs are ordered by relative improvement.

4.7 Conclusion

We propose embedding-based coarsening, and approach that leverages global structural features present in a pretrained hypergraph embedding in order improve the solution quality of multilevel hypergraph partitioning. This approach prioritizes self-similar regions of the hypergraph by visiting nodes in a deterministic order based on the embedding properties of each node's neighborhood. From there, embedding-based coarsening matches nodes by a score that combines a more traditional edge-wise inner-product with the dot product of node embeddings. We observe that the introduction of embedding-based features provides a "time-breaking" mechanism that ultimately preserves global structural features at the coarsest level in the V-cycle. We implement our proposed coarsening strategy in both KaHyPar [189] and Zoltan [62].

We evaluate this approach over multiple trials per combination of 96 graphs, 7 partition counts, 6 pretrained embedding methods, 5 baseline partitioners, 3 implementations, 2 objective functions and at least twenty trials per partitioner combination. Overall this benchmark consists of over 500,000 experimental trials. All experiments, plots and code are available in our online appendix at `sybrandt.com/2019/partitioning`.

We observe a significant increase in quality for small values of k (from 2 to 16) gained from embedding-based coarsening. For higher values of k we observe overall quality that returns to the state-of-the-art baseline. Furthermore, we find that embedding-based coarsening improves partitioning quality significantly across a range of scenarios in both the KaHyPar and Zoltan frameworks. *Specifically, KaHyPar with flow-based refinement [99] and embedding-based coarsening, using either FOBE or HOBE [224] to produce node embedding, scores consistently higher on average than all considered baselines.* Furthermore, we find that by replacing the random

node visit order in many coarsening algorithms with a deterministic strategy that prioritizes self-similar node pairs, we both improve solution quality while drastically reducing solution variance, often by an order of magnitude. Large scale results for all benchmarks and considered metrics is also available in the online appendix.

Chapter 5

Moliere: Automatic Biomedical Hypothesis Generation System

Abstract

Hypothesis generation is becoming a crucial time-saving technique which allows biomedical researchers to quickly discover implicit connections between important concepts. Typically, these systems operate on domain-specific fractions of public medical data. Moliere, in contrast, utilizes information from over 24.5 million documents and does not limit the document vocabulary. At the heart of our approach lies a multi-modal and multi-relational network of biomedical objects extracted from several heterogeneous datasets from the National Center for Biotechnology Information (NCBI). These objects include but are not limited to scientific papers, keywords, genes, proteins, diseases, and diagnoses. We model hypotheses using Latent Dirichlet Allocation applied on abstracts found near shortest paths discovered within this network. We demonstrate the effectiveness of Moliere by performing hypothesis generation on historical data. Our network, implementation, and resulting data are

all publicly available for the broad scientific community.

5.1 Introduction

Vast amounts of biomedical information accumulate in modern databases such as MEDLINE [162], which currently contains the bibliographic data of over 24.5 million medical papers. These ever-growing datasets impose a great difficulty on researchers trying to survey and evaluate new information in the existing biomedical literature, even when advanced ranking methods are applied. On the one hand, the vast quantity and diversity of available data has inspired many scientific breakthroughs. On the other hand, as the set of searchable information continues to grow, it becomes impossible for human researchers to query and understand all of the data relevant to a domain of interest.

In 1986 Swanson hypothesized that novel discoveries could be found by carefully studying the existing body of scientific research [221]. Since then, many groups have attempted to mine the wealth of public knowledge. Efforts such as Swanson's own ARROWSMITH generate hypotheses by finding concepts which implicitly link two queried keywords. His method and others are discussed at length in Section 5.1.3. Ideally, an effective hypothesis generation system greatly increases the productivity of researchers. For example, imagine that a medical doctor believed that stem cells could be used to repair the damaged neural pathways of stroke victims (as some did in 2014 [92]). If no existing research directly linked stem cells to stroke victims, this doctor would typically have no choice but to follow his/her intuition. Hypothesis generation allows this researcher to quickly learn the likelihood of such a connection by simply running a query. Our hypothetical doctor may query the topics *stem cells* and *stroke* for example. If the system returned topics such as *paralysis* then not only

would the doctor's intuition be validated, but he/she would be more likely to invest in exploring such a connection. In this manner, an intelligent hypothesis generation system can increase the likelihood that a researcher's study yields usable new findings.

5.1.1 Our Contribution

We introduce a deployed system, MOLIERE, with the goal of generating more usable results than previously proposed hypothesis generation systems. We develop a novel method for constructing a large network of public knowledge and devise a query process which produces human readable text highlighting the relationships present between nodes.

To the best of our knowledge, MOLIERE is the first hypothesis generation system to utilize the entire MEDLINE data set. By using state-of-the-art tools, such as TOPMINE [69] and FASTTEXT [34], we are able to find novel hypotheses without restricting the domain of our knowledge network or the resulting vocabulary when creating topics. As a result, MOLIERE is more generalized and yet still capable of identifying useful hypotheses.

We provide our network and findings online for others in the scientific community. Additionally, to aid interested biomedical researchers, we supply an online service where users can request specific query results at `http://jsybran.people.clemson.edu/mForm.php`. Furthermore, MOLIERE is entirely open-source in order to facilitate similar projects. See `https://github.com/JSybrandt/MOLIERE` for the code needed to generate and query the MOLIERE knowledge network.

In the following chapter we describe our process for creating and querying a large knowledge network built from MEDLINE and other NCBI data sources. We use natural language processing methods, such as Latent Dirichlet Allocation (LDA) [31]

and topical phrase mining [69], along with other data mining techniques to conceptually link together abstracts and biomedical objects (such as biomedical keywords and n-grams) in order to form our network. Using this network we can run shortest path queries to discover a pathway between two concepts which are non-trivially connected. We then find clouds of documents around these pathways which contain knowledge representative of the path as a whole. PLDA+, a scalable implementation of LDA [148], allows us to quickly find topic models in these clouds. Unlike similar systems, we do not restrict PLDA+ to any set vocabulary. Instead, by using topical phrase mining, we identify meaningful n-grams in order to improve the performance, flexibility, and understandability of our LDA models. These models result in both quantitative and qualitative connections which human researchers can use to inform their decision making.

We evaluate our system by running queries on historical data in order to discover landmark findings. For example, using data published on or before 2009, we find strong evidence that the protein Dead Box RNA Helicase 3 (DDX3) can be applied to treat cancer. We also verify the ability of MOLIERE to make predictions similar to previous systems with restricted LDA [243].

5.1.2 Our Method in Summary

We focus on the domain of medicine because of the large wealth of public information provided by the National Library of Medicine (NLM). MEDLINE is a database containing over 24.5 million references to medical publications dating all the way back to the late 1800s [162]. Over 23 million of these references include the paper's title and abstract text. In addition to MEDLINE, the NLM also maintains the Unified Medical Language System (UMLS) which is comprised of three main

resources: the metathesaurus, the semantic network, and the SPECIALIST natural language processing (NLP) tools. These resources, along with the rest of our data, are described in section 5.2.1.

Our knowledge base starts as XML files provided by MEDLINE, from which we extract each publication's title, document ID, and abstract text. We first process these results with the SPECIALIST NLP toolset. The result is a corpus of text which has standardized spellings (for example "colour" becomes "color"), no stop words (including medical specific stop words such as *Not Otherwise Specified (NOS)*), and other characteristics which improve later algorithms on this corpus. Then we use ToPMine to identify multi-word phrases from that corpus such as "asthma attack," allowing us to treat phrases as single tokens [69]. Next, we send the corpus through FastText, the most recent word2vec implementation, which maps each unique token in the corpus to a vector [153]. We can then fit a centroid to each publication and use the Fast Library for Approximate Nearest Neighbors (FLANN) to generate a nearest neighbors graph [159]. The result is a network of MEDLINE papers, each of which are connected to other papers sharing a similar topic. This network, combined with the UMLS metathesaurus and semantic network, constitutes our full knowledge base. The network construction process is described in greater detail in Section 5.2.

With our network, a researcher can query for the connections between two keywords. We find the shortest path between the two keywords in the knowledge network, and extend this path to identify a significant set of related abstracts. This subset contains many documents which, due to our network construction process, all share common topics. We perform topic modeling on these documents using PLDA+ [148]. The result is a set of plain text topics which represent different concepts which likely connect the two queried keywords. More information about the query process is detailed in Section 5.3.

We use landmark historical findings in order to validate our methods. For example, we show the implicit link between Venlafaxine and HTR1A, and the involvement of DDX3 on Wnt signaling. These queries and results are detailed in Section 5.4. In Sections 5.5 and 5.6 we discuss challenges and open research questions we have uncovered during our work.

5.1.3 Related Work

The study and exploration of undiscovered public knowledge began in 1986 with Swanson's landmark paper [221]. Swanson hypothesized that fragments of information from the set of public knowledge could be connected in such a way as to shed light on new discoveries. With this idea, Swanson continued his research to develop ARROWSMITH, a text-based search application meant to help doctors make connections from within the MEDLINE data set [203, 218, 222]. To use ARROWSMITH, researchers supply two UMLS keywords which are used to find two sets of abstracts, A and C. The system then attempts to find a set $B \approx A \cap C$. Assuming sets A and C do not overlap initially, implicit textual links are used to expand both sets until some sizable set B is discovered. The experimental process was computationally expensive, and queries were typically run on a subset of the MEDLINE data set (according to [222] around 1,000 documents).

Spangler has also been a driving force in the field of hypothesis generation and mining undiscovered public knowledge. His textbook [208] details many text mining techniques as well as an example application related to hypothesis generation in the MEDLINE data set. His research in this field has focused on p53 kinases and how these undiscovered interactions might aid drug designers [209, 208]. His method leverages unstructured text mining techniques to identify a network entities

and relationships from medical text. Our work differs from this paradigm by utilizing the structured UMLS keywords, their known connections, and mined phrases. We do, however, rely on similar unstructured text mining techniques, such as FastText and FLANN, to make implicit connections between the abstracts.

Rzhetsky and Evans notice that current information gathering methods struggle to keep up with the growing wealth of forgotten and hard to find information [72]. Their work in the field of hypothesis generation has included a study on the assumptions made when constructing biomedical models [64] and digital representations of hypothesis [206].

Divoli et al. analyze the assumptions made in medical research [64]. They note that scientists often reach contradictory conclusions due to differences in each person's underly assumptions. The study in [64] highlights the variance of these preconceptions by surveying medical researchers on the topic of cancer metastasis. Surprisingly, 27 of the 28 researchers surveyed disagree with the textbook process of cancer metastasis. When asked to provide the "correct" metastasis scenario, none of the surveyed scientists agree. Divoli's study highlights a major problem for hypothesis generation. Scientists often disagree, even in published literature. Therefore, a hypothesis generation system must be able to produce reliable results from a set of contradicting information.

In [206], Soldatova and Rzhetsky describe a standardized way to represent scientific hypotheses. By creating a formal and machine readable standard, they envision a collection of hypotheses which clearly describes the full spectrum of existing theories on a given topic. Soldatova and Rzhetsky extend existing approaches by representing hypotheses as logical statements which can be interpreted by *Adam*, a robot scientist capable of starting one thousand experiments a day. Adam is successful, in part, because they model hypotheses as an ontology which allows for Bayesian

inference to govern the likelihood of a specific hypothesis being correct.

DiseaseConnect, an online system that allows researchers to query for concepts intersecting two keywords, is a notable contribution to hypothesis generation [145]. This system, proposed by Liu et al., is similar to both our system and ARROWSMITH [204] in its focus on UMLS keywords and MEDLINE literature mining. Unlike our system, Liu et al. restrict DiseaseConnect to simply 3 of the 130 semantic types. They supplement this subset with concepts from the OMIM [80] and GWAS [21] databases, two genome specific data sets. Still, their network size is approximately 10% of the size of MOLIERE. DiseaseConnect uses its network to identify diseases which can be grouped by their molecular mechanisms rather than symptoms. The process of finding these clusters depends on the relationships between different types of entities present in the DiseaseConnect network. Users can view sub-networks relevant to their query online and related entities are displayed alongside the network visualization.

Barabási et al. improve upon the network analytic approach to understand biomedical data in both their work on the disease network [80] as well as their more generalized symptoms-disease network [262]. In the former [80], the authors construct a bipartite network of disease phonemes and genomes to which they refer to as the *Diseasome*. Their inspiration is an observation that genes which are related to similar disorders are likely to be related themselves. They use the *Diseasome* to create two projected networks, the human disease network (HDN), and the Disease Gene Network (DGN). In the latter [262], they construct a more generalized human symptoms disease network (HSDN) by using both UMLS keywords and bibliographic data. HSDN consists of data collected from a subset of MEDLINE consisting of only abstracts which contained at least one disease as well as one symptom, a subset consisting of approximately 850,000 records. From this set, Goh et al. calculated

keyword co-occurrence statistics in order to build their network. They validate their approach using 1,000 randomly selected MEDLINE documents and, with the help of medical experts, manually confirm that the relationship described in a document is reflected meaningfully in HSDN. Ultimately, Goh et al. find strong correlations between the symptoms and genes shared by common diseases.

Bio-LDA is a modification of LDA which limits the set of keywords to the set present in UMLS [243]. This reduction improves the meaning and readability of topics generated by LDA. Wang et al. also show in this work that their method can imply connections between keywords which do not show up in the same document. For example, they note that Venlafaxine and HTR1A both appear in the same topic even though both do not appear in the same abstract. We explore and repeat these findings in Section 5.4.2.

5.1.4 Related and Incorporated Technologies

FastText is the most recent implementation of word2vec from Milkolov et al. [153, 155, 114, 34]. Word2vec is a method which utilizes the skip-gram model to identify the relationships between words by analyzing word usage patterns. This process maps plain text words into a high dimensional vector space for use in data mining applications. Similar words are often grouped together, and the distances between words can reveal relationships. For example, the distance between the words "Man" and "Woman" is approximately the same as the distance between "King" and "Queen". FastText improves upon this idea by leveraging sub-strings in long rarely occurring words.

ToPMine, a project from El-Kishky et al., is focused on discovering multi-word phrases from a large corpus of text [86]. This project intelligently groups un-

igrams together to create n-gram phrases for later use in text mining algorithms. By using a bag-of-words topic model, ToPMine groups unigrams based on their co-occurrence rate as well as their topical similarity using a process they call Phrase LDA.

Latent Dirichlet Allocation [31] is the most common topic modeling process and PLDA+ is a scalable implementation of this algorithm [86, 148]. Developed by Zhiyuan Liu et al., PLDA+ quickly identifies groups of words and phrases which all relate to a similar concept. Although it is an open research question as to how best to interpret these results, simple qualitative analysis allows for "ballpark" estimations. For instance, it may take a medical researcher to wholly understand the topics generated from abstracts related to two keywords, but anyone can identify that all words related to a concept of interest occur in the same topic. Results like this, show that LDA has distinguished the presence of a concept in a body of text.

5.2 Knowledge Network Construction

In order to discover hypotheses we construct a large weighted multi-layered network of biomedical objects extracted from NLM data sets. Using this network, we run shortest-centroid-path queries (see Section 5.3) whose results serve as an input for hypothesis mining. The wall clock time needed to complete this network construction pipeline is depicted in Figure 5.1 (see details in Section 5.4.4). Omitted from this figure is the time spent preprocessing the initial abstract text due to its embarrassingly parallel nature.

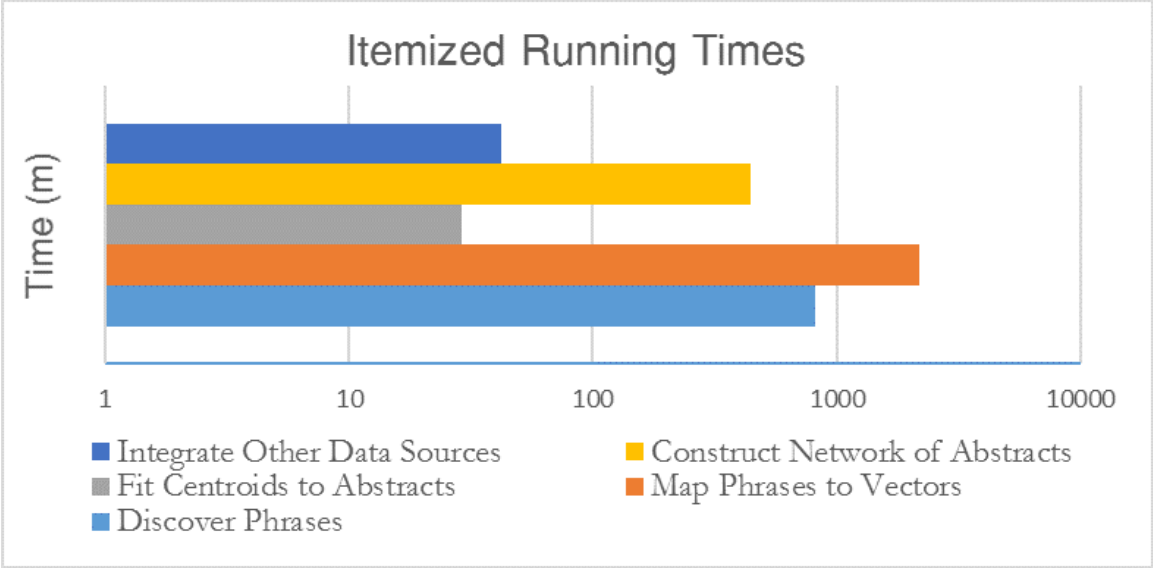


Figure 5.1: Running times of each network construction phase. All phases run on a single node described in section 5.4.4. Not shown:moliere: Initial text processing which was handled by a large array of small nodes.

5.2.1 Data Sources

The NLM maintains multiple databases of medical information which are the main source of our data. This includes MEDLINE [162], a source containing the metadata of approximately 24.5 million medical publications since the late 1800's. Most of these MEDLINE records include a paper's title, authors, publication date, and abstract text.

In addition to MEDLINE, the NLM maintains UMLS [7], which in turn provides the metathesaurus as well as a semantic network. The metathesaurus contains two million keywords along with all known synonyms (referred to as "atoms") used in medical text. For example, the keyword "RNA" has many different synonyms such as "Ribonucleinicum acidum", "Ribonucleic Acid", and "Gene Products, RNA" to name a few. These metathesaurus keywords form a network comprised of multi-typed edges. For example, an edge may represent a *parent - child* or a *boarder concept - narrower*

concept relationship. RNA has connections to terms such as "Nucleic Acids" and "DNA Synthesizers". Lastly, each keyword holds a reference to an object in the semantic network. RNA is an instance of the "Nucleic Acid, Nucleoside, or Nucleotide" semantic type.

The UMLS semantic network is comprised of approximately 130 semantic types and is connected in a similar manner as the metathesaurus. For example, the semantic type "Drug Delivery Device" has an "is a" relationship with the "Medical Device" type, and has a "contains" relationship with the "Clinical Drug" type.

MEDLINE, the metathesaurus, and the semantic network are represented in our network as different layers. Articles which contain full text abstracts are represented as the abstract layer nodes $\mathcal{A}$, keywords from the metathesaurus are represented as nodes in the keyword layer $\mathcal{K}$, and items from the semantic network are represented as nodes in the semantic layer $\mathcal{S}$.

5.2.2 Network Topology

We define a weighted undirected graph underlying our network $\mathcal{N}$ as $G = (V, E)$, where $V = \mathcal{A} \cup \mathcal{K} \cup \mathcal{S}$. The construction of G was governed by two major goals. Firstly, the shortest path between two indirectly related keywords should likely contain a significant number of nodes in $\mathcal{A}$. If instead, this shortest path contained only $\mathcal{K} - \mathcal{K}$ edges, we would limit ourselves to known information contained within the UMLS metathesaurus. Secondly, conceptual distance between topics should be represented as the distance between two nodes in $\mathcal{N}$. This implies that we can determine the similarity between $i, j \in V$ by the weight of their shortest path. If $ij \in E$, this would imply that exists a previously known relationship between i and j. We are instead interested in connections between distant nodes, as these potentially represent

unknown information. Below we describe the construction of each layer in $\mathcal{N}$.

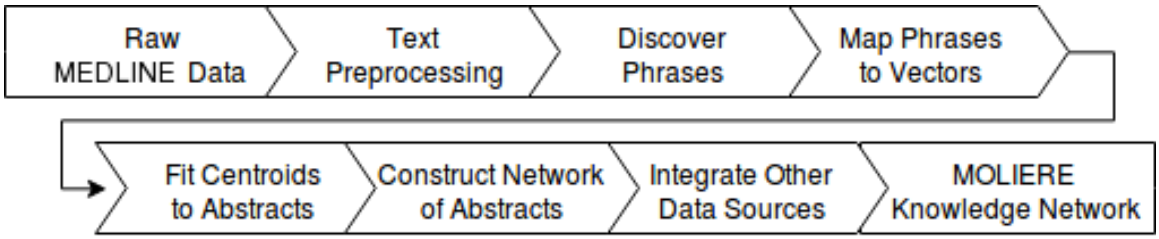


Figure 5.2: MOLIERE network construction pipeline.

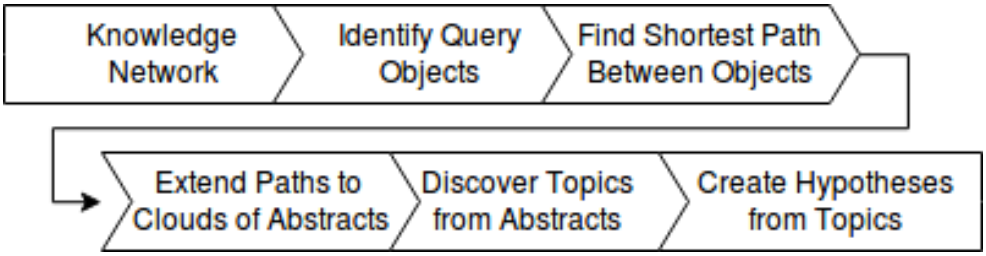


Figure 5.3: MOLIERE query pipeline.

5.2.3 Abstract Layer $\mathcal{A}$

When connecting abstracts ($\mathcal{A} - \mathcal{A}$ edges), we want to ensure that two nodes $i, j \in \mathcal{A}$ with similar content are likely neighbors in the $\mathcal{A}$ layer. In order to do this, we turned to the UMLS SPECIALIST NLP toolset [6] as well as TOPMINE [69] and FASTTEXT [34, 114]. Our process for constructing $\mathcal{A}$ is summarized in Figure 5.2.below.

First, we extract all titles, abstracts, and associated document ID (referred to as PMID within MEDLINE) from the raw MEDLINE files. We then process these combined titles and abstracts with the SPECIALIST NLP toolset to standardize spelling, strip stop words, convert to ASCII, and perform a number of other data cleaning processes. We then use TOPMINE to generate meaningful n-grams and

further clean the text. This process finds tokens that appear frequently together, such as *newborn* and *infants* and combines them into a single token *newborn_infants*. Cleaning and combining tokens in this manner greatly increases the performance of FastText, the next tool in our pipeline.

When running ToPMine, we keep the minimum phrase frequency and the maximum number of words per phrase set to their default values. We also keep the topic modeling component disabled. On our available hardware, the MEDLINE data set can be processed in approximately thirteen hours without topic modeling, but does not finish within three days if topic modeling is enabled. Because the resulting phrases are of high quality even without the topic modeling component, we accept this quality vs. time trade off. It is also important to note that we modify the version of ToPMine distributed by El-Kishky in [69] to allow phrases containing numbers, such as gene names like p53.

Next, FastText maps each token in our corpus to a vector $v \in \mathbb{R}^d$, allowing us to fit a centroid per abstract $i \in \mathcal{A}$. Using a sufficiently high-dimensional space ensures a good separation between vectors. In other words, each abstract $i \in \mathcal{A}$ is represented in $\mathbb{R}^d$ as $c_i = 1/k \cdot \sum_{j=1}^{k} x_j$, where x_j are FastText vectors of k keywords in i.

We choose to use the skipgram model to train FastText and reduce the minimum word count to zero. Because our data preprocessing and ToPMine have already stripped low support words, we accept that any n-gram seen by FastText is important. Following examples presented in [153, 155, 114] and others, we set the dimensionality of our vector space d to 500. This is consistent with published examples of similar size, for example the Google news corpus processed in [153]. Lastly, we increase the word neighborhood and number of possible sub-words from five to eight in order to increase data quality.

Finally, we used FLANN [159] to create nearest neighbors graph from all $i \in \mathcal{A}$ in order to establish $\mathcal{A} - \mathcal{A}$ edges in E. This requires that we presuppose a number of expected nearest neighbors per abstract k. We set this tunable parameter to ten initially and noticed that this value seemed appropriate. By studying the distances between connected abstracts, we observed that most abstracts had a range of very close and relatively far "nearest neighbors". For our purposes in these initial experiments, we kept $k = 10$ and saw promising results. Due to time and resource limitations, we were unable to explore higher values of k in this study, but we are currently planning experiments where $k = 100$ and $k = 1000$. It is important to note that the resulting network will have $\approx k(2.3 \times 10^7)$ edges, so there is a considerable trade-off between quality vs. space and time complexity.

After experimenting with both L_2 and normalized cosine distances, we observed that L_2 distance metric performs significantly better for establishing connections between centroids. Unfortunately, we cannot utilize the k-tree optimization in FLANN along with non-normalized cosine distance, making it computationally infeasible a dataset of our size. This is because the k-tree optimization requires an agglomerative distance metric. Lastly, we scale edges to the $[0, 1]$ interval in order to relate them to other edges within the network.

5.2.4 Keyword Layer $\mathcal{K}$

The $\mathcal{K}$ layer is imported from the UMLS metathesaurus. Each keyword is referenced by a CUI number of UMLS. This layer links keywords which share already known connections. These known connections are $\mathcal{K} - \mathcal{K}$ edges. The metathesaurus connections link related words; for example, the keyword "Protine p53" *C0080055* is related to "Tumor Suppressor Proteins" *C0597611* and "Li-Fraumeni Syndrome"

C0085390 among others. There exist 14 different types of connections between keywords representing relationships such as *parent - child* or *broader concept - narrower concept*. We assign each a weight in the $[0, 1]$ interval corresponding to its relevance, and then scale all weights by a constant factor σ so the average $\mathcal{A} - \mathcal{A}$ edge are is stronger than the average $\mathcal{K} - \mathcal{K}$ edge. The result is that a path between two indirectly related concepts will more likely include a number of abstracts. We selected $\sigma = 2$, but more study is needed to determine the appropriate edge weights within the keyword layer.

5.2.5 $\mathcal{A} - \mathcal{K}$ Connections

In order to create edges between $\mathcal{A}$ and $\mathcal{K}$, we used a simple metric of term frequency-inverse document frequency (tf-idf). UMLS provides not only a list of keywords, but all known synonyms for each keyword. For example, the keyword Color *C0009393* has the American spelling, the British spelling, and the pluralization of both defined as synonyms. Therefore we used the raw text abstracts and titles (before running the SPECIALIST NLP tools) to calculate tf-idf. In order to quickly count all occurrences of UMLS keywords across all synonyms, we implemented a simple parser. This was especially important because many keywords in UMLS are actually multi-word phrases such as "Clustered Regularly Interspaced Short Palindromic Repeats" (a.k.a. CRISPR) *C3658200*.

In order to count these keywords, we construct a parse tree from the set of synonyms. Each node in the tree contains a word, a set of CUIs, and a set of children nodes, with the exception of the root which contains the null string. We build this tree by parsing each synonym word by word. For each word, we either create a new node in the tree, or traverse to an already existing child node. We store each synonym's

CUI in the last node in its parse path. Then, to parse a document, we simply traverse the parse tree. This can be done in parallel over the set of abstracts. For each word in an abstract, we move from the current tree node to a child representing the same word. If none exists, we return to the root node. At each step of this traversal, we record the CUIs present at each visited node. In this manner, we get a count of each CUI present in each abstract. Our next pass aggregates these counts to discover the total number of usages per keyword across all abstracts. We calculate tf-idf per keyword per abstract. Because our network's weights represent distance, we take the inverse of tf-idf to find the weight for an $\mathcal{A}-\mathcal{K}$ edge. This is done simply by dividing a CUI's count across all abstracts by its count in a particular abstract. By calculating weights this way, abstracts which use a keyword more often will have a *lower* weight, and therefore, a shorter distance. We scale the edge weights to the $[0, \sigma]$ interval so that these edges are comparable to those within the $\mathcal{A}$ and $\mathcal{K}$ layers.

5.2.6 Semantic Layer $\mathcal{S}$

The UMLS supplies a companion network referred as the semantic network. This network consists of semantic types, which are overarching concepts. These "types" are similar to the function of a "type" in a programming language. In other words, it is a conceptual entity embodied by instantiations of that type. In the UMLS network, elements of $\mathcal{K}$ are analogous to the instantiations of semantic types. While there are over two million elements of $\mathcal{K}$, there are approximately 130 elements in $\mathcal{S}$. For example, the semantic type Disease or Syndrome *T047* is defined as "A condition which alters or interferes with a normal process, state, or activity of an organism" [7]. There are thousands of keywords, such as "influenza" *C0021400* that are instances of this type.

The $\mathcal{S}-\mathcal{S}$ edges are connected similarly to $\mathcal{K}-\mathcal{K}$ edges. The overall structure is hierarchical with "Event" *T051* and "Entity" *T071* being the most generalized semantic types. Cross cutting connections are also present and can take on approximately fifty different forms. These cross cutting relations also form a hierarchy of relationship types. For example, "produces" *T144* is a more specific relation than its parent "brings about" *T187*.

We initially included $\mathcal{S}$ in our network by linking each keyword to its corresponding semantic type. Unfortunately, in our early results we found that many shortest paths traversed through $\mathcal{S}$ rather than through $\mathcal{A}$. For example, if we were interested in two diseases, it was possible for the shortest path would simply travel to the "Disease or Syndrome" *T047* type. This ultimately degraded the performance of our hypothesis generation system. As a result we removed this layer, but that further study may find that careful choice of $\mathcal{S}-\mathcal{S}$ and $\mathcal{K}-\mathcal{S}$ connection weights may make $\mathcal{S}$ more useful. This is further discussed in Section 5.5.

5.3 Query Process

The process of running a query within MOLIERE is summarized in Figure 5.3. Running a query starts with the user selecting two nodes $i, j \in V$ (typically, but not necessarily, $i, j \in \mathcal{K}$). For example, a query searching for the relationship between "stem cells" and "strokes" would be input as keyword identifiers *C0038250* and *C1263853*, respectively. This process simplifies our query process, but determining a larger set of keywords and abstracts which best represents a user's search query is a future work direction.

After receiving two query nodes i and j, we find a shortest path between them, $(ij)_{\mathbf{s}}$, using Dijkstra's algorithm. These paths typically are between three and five

nodes long and contain up to three abstracts (unless the nodes are truly unrelated, see Section 5.4.1). We observed that when $(ij)_{\mathsf{s}}$ contains only two or three nodes in $\mathcal{K}$, that the ij relationship is clearly well studied because it was solely supplied by the UMLS layer $\mathcal{K}$. We are more interested in paths containing abstracts because these represent keyword pairs whose relationships are less well-defined. Still, the abstracts we find along these shortest paths alone are not likely to be sufficient to generate a hypothesis.

5.3.1 Hypothesis Modeling

Broadening $(ij)_s$ consists of two main phases, the results of which are depicted in Figure 5.4. First, we select all nodes $S = (ij)_s \cap \mathcal{A}$. These abstracts along the path $(ij)_s$ represent papers which hold key information relating two unconnected keywords. We find a neighborhood around S using a weighted breadth-first traversal, selecting the closest 1,000 abstracts to S. We will call this set N. Because $\mathcal{A}$ was constructed as a nearest neighbors graph, it is likely that the concepts contained in N will be similar to the concepts contained in S, which increases the likelihood that important concepts will be detected by PLDA+ later in the pipeline.

Next, we identify abstracts with contain information pertaining to the $\mathcal{K} - \mathcal{K}$ connections present in $(ij)_s$. We do so in order to identify abstracts which likely contain concepts which a human reader could use to understand the known relationship between two connected keywords. We start by traversing $(ij)_s$ to find $\alpha, \beta \in \mathcal{K}$ such that α and β are adjacent in $(ij)_s$. From there, we find a set of abstracts $C = \{c : c\alpha \in E \wedge c\beta \in E\}$. That is, C is a subset of abstracts containing both keywords α and β. Because $(ij)_s$ can have many edges between keywords, and because thousands of abstracts can contain the same two keywords, it is important to limit

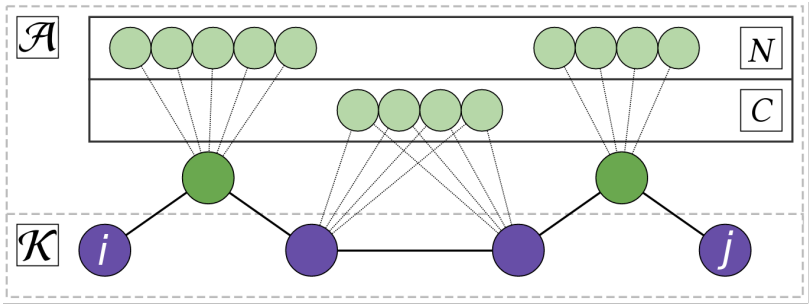


Figure 5.4: Process of extending a path to a cloud of abstracts.

the size of C.

This process creates a set of around 1,300 $\mathcal{A}$-nodes. This set will typically contain around 15,000-20,000 words and is large enough for PLDA+ to find topics. We run PLDA+ and request 20 topics. We find this provides a sufficient spread in our resulting data sets. The trained model generated by PLDA+ is what is eventually returned by our query process.

For our experiments, we often must process tens of thousands of results and thus must train topic models quickly. This is most apparent when running a one-to-many query such as the drug repurposing example in 5.4.3. Additionally, the training corpus returned from a Moliere query is often only a couple thousand documents large. As a result, we set the number of topics and the number of iterations to relatively small values, 20 and 100 respectively. Because we store intermediary results, it is trivial to retrain a topic model if the preliminary result seems promising.

The process of analyzing a topic model and uncovering a human interpretable sentence to describe a hypothesis is still a pressing open problem. The process as stated here does have some strong benefits which are apparent in Section 5.4. These include the ability to find correlations between medical objects, such as between a drug and multiple genes. In Section 5.6 we explain our initial plans to improve the quality of results which can be deduced from these topic models.

5.4 Experiments

We conduct two major validation efforts to demonstrate our system's potential for hypothesis generation. For each of these experiments we use the same set of parameters for our trained model and network weights. Our initial findings show our choices, detailed in Section 5.2, to be robust. We plan to refine these choices with methods described in Section 5.6.

We repeat an experiment done by Wang et al. in [243] wherein we discover the implicit connections between the drug Venlafaxine and the genes HTR1A and HTR2A. We also perform a large scale study of Dead Box RNA Helicase 3 (DDX3) and its connection to cancer adhesion and metastasis. Each of these experiments is described in greater detail in the following sections. *In this chapter, we deliberately do not evaluate our experiments with extremely popular objects such as p53. These objects are so highly connected within $\mathcal{K}$ that hypothesis generation involving these keywords is easy for many different methods.*

5.4.1 Network Profile

We conduct our experiments on a very large knowledge graph which has been constructed according to Section 5.2. We initially created a network $\mathcal{N}$ containing information dating up to and including 2016. This network consists of 24,556,689 nodes and 989,169,295 edges. The network overall consists of largest strongly connected component containing 99.8% of our network. The average degree of a node in $\mathcal{N}$ is 79.65, and we observe a high clustering coefficient of 0.283. These metrics cause us to expect that the shortest path between two nodes will be very short. Our experiments agree, showing that most shortest paths are between three and six nodes long.

5.4.2 Venlafaxine to HTR1A

Wang et al. in [243] use a similar topic modeling approach, and find during one of their experiments that Venlafaxine *C0078569* appears in the same topic as the HTR1A and HTR2A genes (*C1415803* and *C1825553* respectively). When looking into these results, they find a stronger association between Venlafaxine and HTR1A. This finding is important because Venlafaxine is used to treat depressive disorder and anxiety, which HTR1A and HTR2A have been thought to affect, but as of 2009 no abstract contains this link. As a result, this implicit connection is difficult to detect with many existing methods.

Results: As a result of running two queries, Venlafaxine to HTR1A, and Venlafaxine to HTR2A, we can corroborate the findings of Wang et al. in [243]. We find that neither pair of keywords is directly connected or connected through a single abstract. Nevertheless, phrases such as "long term antidepressant treatment," "action antidepressants," and "antidepressant drugs" are all prominent keywords in the HTR1A query. Meanwhile, the string "depress" only occurs four times in unrelated phrases with the HTR2A results. The distribution of depression related keywords from both queries can be see in figure 5.5.

Similarly, our results for HTR1A contain a single topic holding the phrases "anxiogenic," "anxiety disorders," "depression anxiety disorders," and "anxiolytic response." In contrast, our HTR2A results do not contain any phrases related to anxiety. The distribution of anxiety related keywords from both queries can be see in figure 5.6.

Our findings agree with those of Wang et al. which were that a small association score of 0.34 between Venlafaxine and HTR1A indicates a connection which is likely related to depressive disorder and anxiety. The association score between

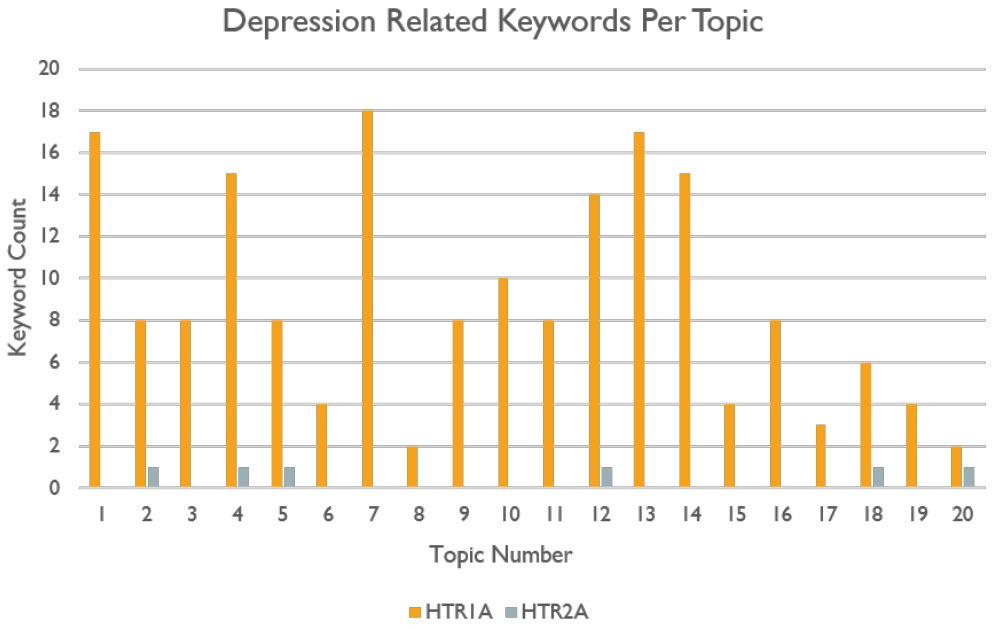


Figure 5.5: Distribution of n-grams having to do with depression from Venlafaxine queries.

Venlafaxine and HTR2A, in contrast, is a much higher 4.0. This indicates that the connection between these two keywords is much weaker.

5.4.3 Drug Repurposing and DDX3's Anti-Tumor Applications

Many genes are active in multiple cellular processes and in many cases they are found to be active outside of the original area in which the gene was initially discovered. The prediction of new processes is especially important for repurposing existing drugs (or drug target genes) to a new application [16, 165, 12]. As an example, the drugs developed for the treatment of infectious diseases were recently repurposed for cancer treatment. Extending applications of existing drugs provides a tremendous opportunity for the development of cost-effective treatments for cancers and other

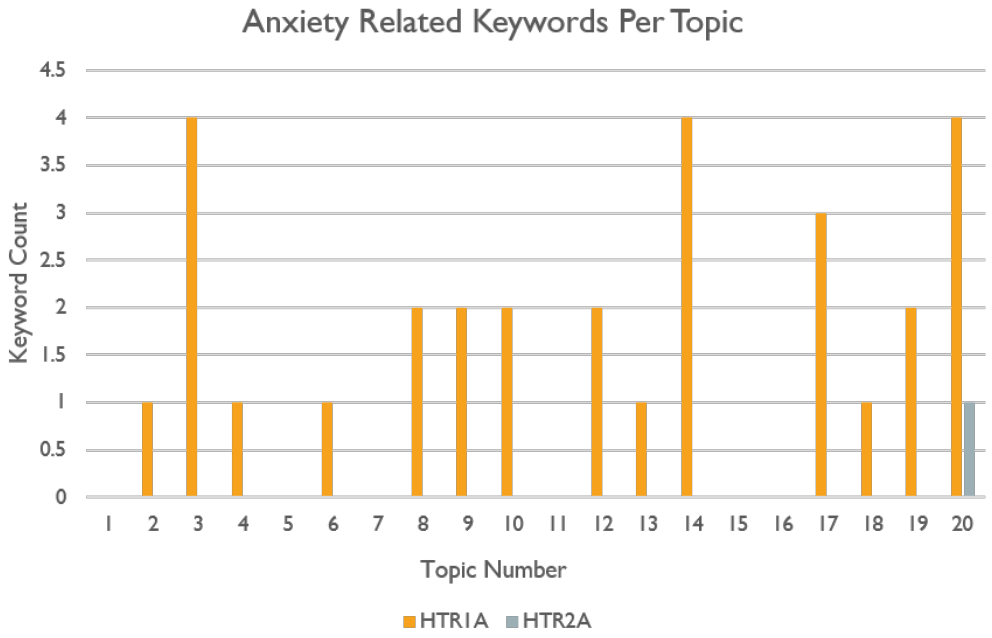


Figure 5.6: Distribution of n-grams having to do with anxiety from Venlafaxine queries.

life-threatening diseases.

To estimate the predictive value of our system for the discovery of new applications of small molecules we select Dead Box RNA Helicase 3 (DDX3) *C2604356*. DDX3 is the member of Dead-box RNA helicase and was initially discovered to be a regulator of transcription and propagation of Human Immunodeficiency Virus (HIV) as well as ribosomal biogenesis. Initially, DDX3 was a target for the development of anti-viral therapy for the AIDS treatment [129, 151].

More recently, DDX3 activity was found to be involved cancer development and progression mainly through regulation of the Wnt signaling pathway [57, 255] and associated regulation of Cell-cell and Cell-matrix adhesion, tumor cells invasion, and metastasis [49, 215, 234, 126]. Currently, DDX3 is an established target for anti-tumor drug development [35, 186, 36] and represents a case for repurposing target

anti-viral drugs into the application area of anti-tumor therapy.

To test this hypothesis, we analyze the data available on and before 12/31/2009, when no published indication of links in between DDX3 and the Wnt signaling were available. We compare DDX3 to all UMLS keywords containing the text "signal transduction", "transcription", "adhesion", "cancer", "development", "translation", or "RNA" in their synonym list. This search results in 9,905 keywords over which we query for relationships to DDX3. From this large set of results we personally analyze a subset of important pairs.

Results: In our generated dataset, we found following text grouping within topics: "substrate adhesion," "RGD cell adhesion domain," "cell adhesion factor," "focal adhesion kinase" which are indicative for the cell-matrix adhesion. The topics "cell-cell adhesion," "regulation of cell-cell adhesion," "cell-adhesion molecules" indicate the involvement of DDX3 into cell-cell adhesion regulation. The involvement of adhesion is associated with topics related to tumor dissemination: " Collaborative staging metastasis evaluation Cancer," "metastasis adhesion protein, human," "metastasis associated in colon cancer 1" (selected in between others similar topics).

The results above suggested that through analysis of the $\leq$2009 dataset we can predict the involvement of DDX3 in tumor cell dissemination through the effects of Cell-cell and cell-matrix adhesion. Next, we analyzed, whether it will be possible to made inside of the mechanisms of DDX3-dependent regulation of Wnt signaling. As shown recently, DDX3 involvement on Wnt signaling is based on the regulated Casein kinase epsilon, to affect phosphorylation of the disheveled protein. Although we cannot predict the exact mechanism of DDX3 based on $\leq$2009 dataset, the existence of multiple topics of signal-transduction associated kinases, like "CELL ADHESION KINASE", "activation by organism of defense-related host MAP kinase-mediated signal transduction pathway", "modulation of defense-related symbiont mitogen-activated

protein kinase-mediated signal transduction pathway by organism", suggested the ability of DDX3 to regulate kinases activities and kinase-regulated pathways.

5.4.4 Experimental Setup

We performed all experiments on a single node within Clemson's Palmetto supercomputing cluster. To perform our experiments and construct our network, we use an HP DL580 containing four Intel Xeon x7542 chips. This 24 core node has 500 GB of memory and access to a large ZFS-based file system where we stored experimental data.

For the DDX3 queries, we initially searched for all $(ij)_s$ where i = DDX3 and $j \in \mathcal{K}$. This resulted in 1,350,484 shortest paths with corresponding abstract clouds. We used PLDA+ to construct models for all of these paths. Discovering all $(ij)_s$ completed in almost 10 hours of CPU time, and training the respective models completed in slightly over 68 hours of CPU time. We ran PLDA+ in parallel, resulting in a wall time of only 12 hours. As mentioned previously, this large dataset was filtered to the 9,905 paths we are interested in.

We generate the results for the Venlafaxine experiments in one hour of CPU time, which is mostly spent loading our very large network and then running Dijkstra's algorithm. After this, the two resulting PLDA+ models were trained in parallel within a minute.

5.5 Deployment Challenges

In the following section we detail the challenges which we have faced and are expecting to encounter while creating our system and deploying it to the research community.

Dynamic Information Updates The process of creating our network is computationally expensive and for the purposes of validation we must create multiple instances of our network representing different points in time. Initially we would have liked to create these multiple instances from scratch, starting from the MEDLINE archival distribution and rebuilding the network from there. Unfortunately, this proved infeasible because creating a single network is a time consuming process. Instead, we filter our network by removing abstracts and keywords which were published after our select date. Additionally, the act of adding information to our network, such as extending the 2016 network to create a 2017 network, is not straightforward. Ideally, adding a small number abstracts or keywords should be a fast and dynamic process which only affects localized regions of the network. If this were so, our deployed system could take advantage of new ideas and connections as soon as they are published.

A deployed system could support dynamic updates with an amortized approach. Using previously created FastText and ToPMine models, new documents could be fitted into an existing network with suitably high performance. Of course, if a new document introduced a new keyword or phrase, we would be unable to detect it initially. After some threshold of new documents had been added to the network, we could then rerun the entire network construction process to ensure that new keywords, phrases, and concepts would be properly placed in the network.

Query Platform and Performance: Initially, we expected to use a graph database to make the query process easier. We surveyed a selection of graph databases and found that Neo4j [61] provides a powerful query language as well as a platform capable of holding our billion-edge network. Unfortunately, Neo4j does not easily support weighted shortest path queries. Although some user suggestions did hint that it may be possible, the process requires leveraging edge labels and custom java procedures in a way that did not seem scalable. In place of Neo4j, we implemented Dijkstra's

shortest path algorithm in C++ using skew heaps as the internal priority queue. This implementation was chosen to minimize memory usage while maximizing speed and readability. Because we implemented Dijkstra's algorithm ourselves, we can also combine the process of finding a shortest path and finding all neighboring abstracts for all keywords from a specific source. With only these high level optimizations, we were able to generate over 1,350,000 shortest paths and abstract neighborhoods in under ten hours, but generating a single result takes slightly over one hour.

5.6 Lessons Learned and Open Problems

Specialized LDA: During last two decades there has been a number of significant attempts to design automatic hypothesis generation systems [208, 218, 243]. However, most of these improve their performance by restricting either their information space or the size of their dictionary. For example, specialized versions of LDA such as Bio-LDA [243] uncover latent topics using a dictionary that gives a priority to special terms. We find that such approaches are helpful when general language may significantly over weigh a specialized language. However, phrase mining approaches that recover n-grams, such as [69], produce accurate methods without limiting the dictionary.

Hypothesis Viability and Novelty Assessment: Intuitively, a strong connection between two concepts in $\mathcal{N}$ means that there exist a significant amount of research that covers a path between them. Similar observations are valid for LDA, i.e., latent topics are likely to describe well known facts. As a result, the most meaningful connections and interpretable topical inference are discovered with latent keywords that are among the most well known concepts. However, real hypotheses are not necessarily described using the most latent keywords in such topic models. In many

cases, the keywords required for a successful and interpretable hypothesis start to appear among 20-30 most latent topical keywords. Thus, a major open problem related is the process to which one should select a combination of keywords and topics in order to represent a viable hypothesis. This problem is also linked to the problem of assessing the viability of a generated hypothesis.

These problems, as well as the problem of hypothesis novelty assessment, can be partially addressed by using the Dynamic Topic Modeling (DTM) [30]. Our preliminary experiments with scalable time-dependent clustered LDA [87] that significantly accelerates DTM demonstrate a potential to discover dynamic topics in MEDLINE. The dynamic topics are typically more realistic than those that can be discovered in the static network. This significantly simplifies the assessment of viability and topic noise elimination.

Incorporating the Semantic Layer $\mathcal{S}$: In section 5.2.6 we describe the process in which we evaluated the UMLS semantic network and found that it worsened our resulting shortest path queries. Further work could improve the contribution that $\mathcal{S}$ has on our overall network, possibly allowing $\mathcal{S}$ to define the overall structure of our knowledge graph. In order to do this, one would likely need to take into account the hierarchy of relationship types present in this network, as well as the relative relationship each element in $\mathcal{K}$ has with its connection in $\mathcal{S}$. Ultimately, these different relationships would need to inform a weighting scheme that balances the over generalizations that $\mathcal{S}$ introduces. For example, it may be useful to understand that two keywords are both diseases, but it is much less useful to understand that two keywords are "entities".

Learning the Models of Hypothesis Generation: There is surprisingly little research focused on addressing the process of biomedical research and how that process evolves over time. We would like to model the process of discovery formation, tak-

ing into account the information context surrounding and preceding a discovery. We believe we could do so by reverse engineering existing discoveries in order to discover factors which altered the steps in a scientist's research pipeline. Several promising observations in this direction have been done by Foster et al. [78] who examined this through Bourdieu's field theory of science by developing a typology of research strategies on networks extracted from MEDLINE. However, instead of reverse engineering their models, they separate innovation steps from those that are more traditional in the research pipeline.

Dynamic Keyword Discovery: One of the limitations we found when performing our historical queries is the delay between the first major uses of a keyword and its appearance in the UMLS metathesaurus. Initially, we planned to study the relationship between "CRISPR" *C3658200* and "genome editing" *C4279981*. To our surprise, many keywords related to this query did not exist in our historical networks between 2009 and 2012, despite their frequent usage in cutting-edge research during that time. To further confuse the issue, although the keyword "CRISPR" did not appear in the UMLS releases on or before 2012, keywords containing "CRISPR" as a substring, such as "CRISPR element metabolism" *C1752766*, do appear. We find this to be contradictory and that these inconsistencies highlight the limitation of relying on so strongly on keyword databases. Going forward, we plan to devise a way to extend a provided keyword network, utilizing semantic connections we can find within the MEDLINE document set. Projects like [209] have already shown this method can work in domains of smaller scales with good results. The challenge will be to extend this method to perform well when used on the entire MEDLINE data set.

Improving Performance of Algorithms with Graph Reordering Techniques: Cache-friendly layouts of graphs are known to generally accelerate the performance of the path and abstract retrieval algorithms which we apply. Moreover, it is desirable to

consider this type of acceleration in order to make our system more suitable for regular modern desktops. This is an important consideration as memory is not expected to be a major bottleneck after the network is constructed. We propose to rearrange the network nodes by minimizing such objectives as the minimum logarithmic or linear arrangements [185, 183]. On a mixture of $\mathcal{K}-\mathcal{K}$, $\mathcal{A}-\mathcal{K}$, and $\mathcal{A}-\mathcal{A}$ edges we anticipate an improvement of at least 20% in the number of cache misses according to [182].

Mass Evaluation: We note that evaluation techniques are largely an issue in the state of the art of hypothesis generation. While some works feature large scale evaluation performed by many human experts, a majority, this work included, are restricted to only a couple of promising results to justify the system. In order to better evaluate and compare hypothesis generation techniques we must devise a common and large scale suite of historical hypotheses. We are currently evaluating whether a ground-truth network, like the drug-side-effect network SIDER [136], can be a good source of such hypotheses. For example, if we identify a set of recently added connections within SIDER, and predict a substantial percentage of those connections using MOLIERE, then we may be more certain of our performance.

New Domains of Interest: We have considered other domains on which MOLIERE may perform well. These include generating hypotheses regarding economics, patents, narrative fiction, and social interactions. These are all domains where a hypothesis would involve finding new relationships between distinct entities. We contrast this with domains such as mathematics where the entity-relationship network is much less clear, and logical approaches from the field of automatic theorem proving are more applicable.

5.7 Conclusions

In this study we describe a deployed biomedical hypothesis generation system, Moliere, that can discover relationship hypotheses among biomedical objects. This system utilizes information which exists in MEDLINE and other NLM datasets. We validate Moliere on landmark discoveries using carefully filtered historical data. Unlike several other hypothesis generation systems, we do not restrict the information retrieval domain to a specific language or a subset of scientific papers since this method can lose an unpredictable amount of information. Instead, we use recent text mining techniques that allow us to work with the full heterogeneous data at scale. We demonstrate that Moliere successfully generates hypotheses and recommend using it to advance biomedical knowledge discovery. Going forward, we note a number of directions along which we can improve Moliere as well as many existing hypothesis generation systems.

5.8 Acknowledgments

We would like to thank Dr. Lihn Ngo for his help in using the Palmetto supercomputer which ran our experiments, and Cong Qiu for initial experiments with Neo4j.

Chapter 6

Large-Scale Validation of Hypothesis Generation Systems via Candidate Ranking

Abstract

The first step of many research projects is to define and rank a short list of candidates for study. In the modern rapidity of scientific progress, some turn to automated hypothesis generation (HG) systems to aid this process. These systems can identify implicit or overlooked connections within a large scientific corpus, and while their importance grows alongside the pace of science, they lack thorough validation. Without any standard numerical evaluation method, many validate general-purpose HG systems by rediscovering a handful of historical findings, and some wishing to be more thorough may run laboratory experiments based on automatic suggestions. These methods are expensive, time consuming, and cannot scale. Thus, we present a numerical evaluation framework for the purpose of validating HG systems that lever-

ages thousands of validation hypotheses. This method evaluates a HG system by its ability to rank hypotheses by plausibility; a process reminiscent of human candidate selection. Because HG systems do not produce a ranking criteria, specifically those that produce topic models, we additionally present novel metrics to quantify the plausibility of hypotheses given topic model system output. Finally, we demonstrate that our proposed validation method aligns with real-world research goals by deploying our method within MOLIERE, our recent topic-driven HG system, in order to automatically generate a set of candidate genes related to HIV-associated neurodegenerative disease (HAND). By performing laboratory experiments based on this candidate set, we discover a new connection between HAND and Dead Box RNA Helicase 3 (DDX3).

6.1 Introduction

In the early stages of a research project, biomedical scientists often perform "candidate selection," wherein they select potential targets for future study [107]. For instance, when exploring a certain cancer, scientists may identify a few dozen genes on which to experiment. This process relies on the background knowledge and intuitions held by each researcher, and higher-quality candidate lists often lead to more efficient research results. However, the rate of scientific progress has been increasing steadily [233], and occasionally scientists miss important findings. for instance, was the case regarding the missing connection between Raynaud's Syndrome and fish oil [219], and in the case of five genes recently linked to Amyotrophic Lateral Sclerosis [18]. Hypothesis Generation (HG) systems allow scientists to leverage the cumulative knowledge contained across millions of papers, which lead to both above findings, among many others. The importance of these systems rises along-

side the pace of scientific output; an abundance of literature implies an abundance of overlooked connections. While many propose techniques to understand potential connections [243, 225, 145, 221, 208], few *automated* validation techniques exist [41] for general-purpose HG systems (not designed for specific sub-domains or types of queries such as OHSUMED [98] or BioCreative datasets). Often, subject-matter experts assist in validation by running laboratory experiments based on HG system output. This process is expensive, time consuming, and does not scale beyond a handful of validation examples.

HG systems are hard to validate because they attempt to uncover novel information, unknown to even those constructing or testing the system. For instance, how are we to distinguish a bizarre generated hypothesis that turns out to produce important results from one that turns out to be incorrect? Furthermore, how can we do so at scale or across fields? While there are verifiable models for novelty in specific contexts, each is trained to detect patterns similar to those present in a training set, which is conducive to traditional cross-validation. Some examples include using non-negative matrix factorization to uncover protein-protein interactions [84], or to discover mutational cancer signatures [11]. However, HG is unlike the above examples as it strives to detect novel patterns that are a) *absent* from a dataset, b) may be wholly unknown or even currently counterintuitive, and c) not necessarily outliers as in traditional data mining.

Our contribution: In this chapter we propose novel hypothesis ranking methods and a method to validate HG systems that does not require expert input and allows for large validation sets. This method judges a system by its ability to rank hypotheses by plausibility, similarly to how a human scientist must rank potential research directions during candidate selection. We start by dividing a corpus based on a "cut date," and provide a system only information that was priorly available.

Then, we identify predicates (clauses consisting of subject, verb, and object) whose first co-occurrence in a sentence is after the cut date. Because typical corpora contain only titles and abstracts, these recently introduced connections represent significant findings that were not previously formulated, thus we can treat them as surrogates for plausible hypotheses from the perspective of the system under evaluation. To provide implausible hypotheses, we randomly generate predicates that do not occur in the corpus as a whole. Then, the HG system must rank both the plausible and implausible predicates together by evaluating the predicted connection strength between each predicate's subject and object. The system's evaluation is based on the area under this ranking's Receiver Operating Characteristic (ROC) curve, wherein the highest area under curve (AUC) of 1 represents a ranking that places all plausible connections above the implausible, and the lowest AUC of 0.5 represents an even mixture of the two.

We note that many HG systems do not typically produce a ranking criteria for potential hypotheses. Particularly, we find that those systems that produce topic model output, such as Moliere [225] or BioLDA [243], lack this criteria, but present promising results through expert analysis. Therefore, we additionally developed a number of novel metrics for topic-driven HG systems that quantify the plausibility of potential connections. These metrics leverage word embeddings [153] to understand how the elements of a hypothesis relate to its resulting LDA topic model [31]. Through our experiments, described below, we identify that a polynomial combination of five different metrics allows for the highest-scoring ranking (0.834). This result is especially significant given that the main validation methods available, to both Moliere and other similar systems (see survey in [225]), were expert analysis and replicating the results of others [41]. Still, while the systems mentioned above focus on the medical domain, we note that neither our metrics, nor our validation

methodology, are domain specific.

To demonstrate that our proposed validation process and new metrics apply to real-world applications, we present a case study wherein our techniques validate an open-source HG system as well as identify a novel gene-disease connection. We modify Moliere to support our new metrics, and we perform our validation process. This system is trained on Medline [162], a database containing over 27 million papers (titles and abstracts) maintained by the National Library of Health. We use SemMedDB [120], a database of predicates extracted from Medline, in order to identify the set of "published" (plausible) and "noise" (implausible) hypotheses. This database represents its connections in terms of codified entities provided by the Unified Medical Language System (UMLS), which enables our experimental procedure to be both reproducible and directly applicable to many other medical HG systems. This evaluation results in an ROC AUC of 0.834, and when limiting the published set to only predicates occurring in papers that received over 100 citations, this rises to 0.874. Then, we generate hypotheses, using up-to-date training data, which attempt to connect HIV-associated neurodegenerative disease (HAND) to over 30,000 human genes. From there, we select the top 1,000 genes based on our ranking metrics as a large and rudimentary "candidate set." By performing laboratory experiments on select genes within our automatically generated set, we discover a new relation between HAND and Dead Box RNA Helicasee 3 (DDX3). Thus, demonstrating the practical utility of our proposed validation and ranking method.

6.2 Technical Background

Extracting Information from Hypothesis Generation Systems Swanson and Smalheiser created the first HG system Arrowsmith [203], and in doing so outlined

the ABC model for discovery [223]. Although this approach has limitations [201], its conventions and intuitions remain in modern approaches [208].

In the ABC model, users run queries by specifying two keywords a and c. From there, the goal of a HG system is to discover some entity b such that there are known relationships "$a \rightarrow b$" and "$b \rightarrow c$," which allow us to infer the relationship between a and c. Because many connections may require more than one element b to describe, researchers apply other techniques, such as topic models in our case, to describe these connections.

We center this work around the Moliere HG system [225]. Once a user queries a and c, the system identifies a relevant region within its multi-layered knowledge network, which consists of papers, terms, phrases, and various types of links. The system then extracts abstracts and titles from this region and creates a sub-corpus upon which we generate a topic model (Note that in [226] we address trade-offs of using full text). This topic model describes groups of related terms, which we study to understand the quality of the a-to-c connection. Previously, these results were compared biased on those words that co-occur with high probability in prominent topics. Without clear metrics, or a validation framework, experts could only help evaluate a select handful of a, c pairs.

6.2.1 Topic Modeling

Originally presented by Blei et al. [31], Latent Dirichlet Allocation (LDA) is a generative probabilistic model used to interpret large text corpora. This model represents each document as a mixture of "topics," which themselves are probability distributions over the corpus's vocabulary. In practice, each topic is a fuzzy cluster of terms, which we can interpret to help us understand the overall document set. But,

TOPIC 0	TOPIC 1	TOPIC 2	TOPIC 3
tobacco	patient	find	control
lung_cancer	normal	level	find
health	select	activity	observe
cancer	therapy	study	strain

Table 6.1: We generated 20 topics on documents related to tobacco and lung cancer. Here four top words of the four most relevant topics.

a key limitation is that the number of topics must be specified a priori.

In most applications, the true number of topics is unknown. This is especially true in the case of HG, where the ultimate goal is to uncover hidden topical information. Many algorithms overcome this challenge through expensive model selection methods such as 10-fold cross validation [91]. But, cross validation can only be accomplished when some known training data is available.

Although techniques such as hierarchical topic models [85] provide a method to remove this limitation, they are computationally infeasible at our scale. Instead, we leverage an observation that across models, prevalent topics will stay consistent [87]. When combined with the metrics we present in Section 6.4, we are safe to generate a sufficiently large number of topics with the assurance that our methods will help filter any extra noisy topics. Additionally, we could compare each model's performance with our metrics across a number of hyperparameter settings, but this falls outside the scope of this work.

In Table 6.1, we show an example of topic models as they relate to HG. Using the Moliere query process, we select documents relevant to both tobacco and lung cancer in order to generate a topic model. In order to efficiently generate topic models in parallel, we leverage PLDA+, a scalable parallel implementation of the LDA algorithm [148].

Word and Phrase Embedding The method of finding dense vector representa-

tions of words is often referred to as "WORD2VEC." In reality, this umbrella term references two different algorithms, the Continuous BOW (CBOW) method and the Skip-Gram method [153]. Both rely on shallow neural networks in order to learn vectors through word-usage patterns. We provide a diagram outlining these methods in Figure 6.1.

Both methods take as input a corpus, represented as sentences, and an integer representing the desired dimensionality of the resulting vectors. Additionally, both methods begin by assigning random vectors to each word present in the training corpus. Next, both methods select random windows, consisting of at most k words from sampled sentences, in order to train on.

The CBOW method takes each window of k words and trains a shallow neural net to predict the centered word from the $k-1$ surrounding words. Thus, the $k-1$ surrounding corresponding vectors are averaged together and fed as input into the perceptron. The averaging process discards word-order information, reducing the input effectively to the BOW model. This method compares the predicted vector against the vector corresponding the centered word resulting in error calculations leading to back-propagation.

The Skip-Gram model takes as input the centered word, and attempts to predict the $k-1$ surrounding words in each window. Like above, we compare the output of the perceptron with the existing vectors corresponding to the surrounding words. Because the order of predicted words is used to calculate error and back-propagation updates, this model takes word order into account. Because the output of this model is a whole window, as opposed to a single vector, error calculations and back-propagation are substantially more expensive to compute.

Iteratively, WORD2VEC refines its predictor and then uses it to predict a higher quality vector space. This new vector space is used to continue refining the

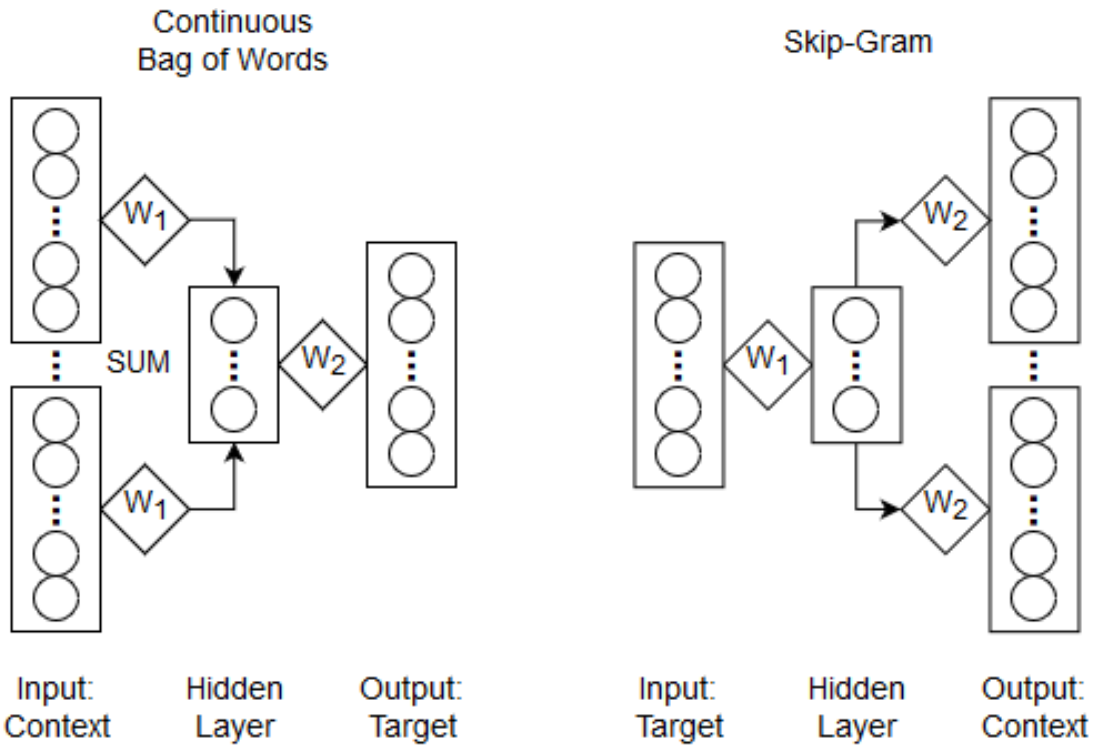


Figure 6.1: Mikolov et al. presented two methods for discovering word embeddings in [153, 155]. This diagram depicts the CBOW method, highlighting the intermediate layer. In this diagram, each rectangle represents a vector, with its internal circles representing that vector's dimensions. The diamonds represent the transformation matrices which map input vectors to a hidden layer, and the hidden layer to the output. Note how each dimension in the output vectors correspond to a linear combination of hidden layer features. Additionally, note how the features discovered in the hidden layer corresponds closely to a topic model.

prediction model, and so on until convergence.

Moliere uses FastText [113], a similar tool under the Word2Vec umbrella, to find high-quality embeddings of medical entities. By preprocessing Medline text with the automatic phrase mining technique ToPMine [69], we improve these embeddings while finding multi-word medical terms such as "lung cancer" or "benign tumor." We see in Figure 6.2 that FastText clusters similar biological terms, an observation we later leverage to derive a number of metrics. We also see

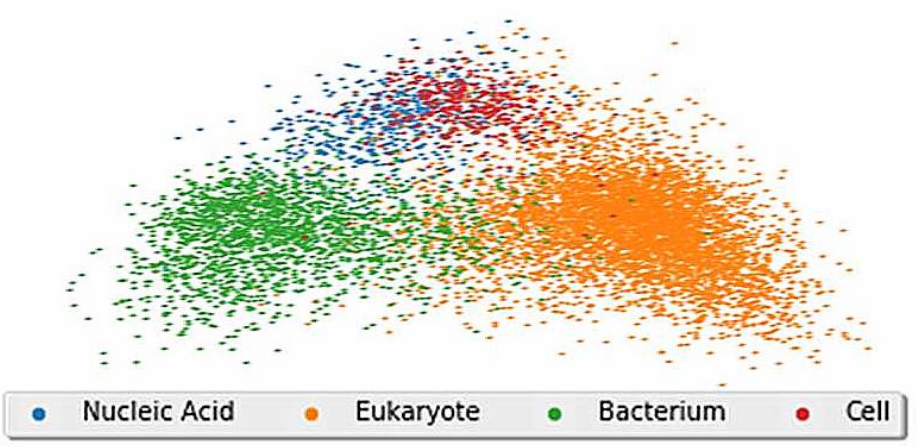


Figure 6.2: The above diagram shows a 2-D representation of the embeddings for over 8 thousand UMLS keywords within MOLIERE. We used singular value decomposition to reduce the dimensionality of these vectors from 500 to 2.

this property in a number of other word-embedding methods, such as DOC2VEC [132], LINE [228], and FASTTEXT [34, 114, 113]. We use FASTTEXT in our methodology, and note that the specific embedding method should not change this core principle, but additional exploration of each method's clustering may reveal insights into performance benefits.

Topic Models Latent Dirichlet Allocation (LDA) [31], the classical topic modeling method, groups keywords based on their document co-occurrence rates in order to describe the set of trends that are expressed across a corpus. A topic is simply a probability distribution over a vocabulary, and each document from the input corpus is assumed to be a mixture of these topics. For instance, a topic model derived from New York Times articles would likely find one topic containing words such as "computer," "website," and "Internet," while another topic may contain words such as "money," "market," and "stock."

In the medical domain, some use topic models to understand trends across scientific literature. We look for groupings of entities such as genes, drugs, and

diseases, which we then analyze to find novel connections. While LDA is the classical algorithm, MOLIERE uses a parallel technique, PLDA+ [148] to quickly find topics from documents related to a and c. Additionally, because MOLIERE preprocess's MEDLINE articles with TOPMINE, its resulting topic models include both words and phrases. This often leads to more interpretable results, as a topic containing an n-gram, such as "smoking induced asthma," is typically easier to understand than a topic containing each unigram listed separately with different probabilities.

We additionally can use the probabilities of each word to represent a topic within an embedding space created with WORD2VEC. For instance, we can take a weighted average over the embeddings for each topic to describe each topics's "center." Additionally, we can simply treat each topic as a weighted point cloud for the purposes of typical similarity metrics. We leverage both representations later in our metrics.

6.3 Validation Methodology

In order to unyoke automatic HG from expert analysis, we propose a method that any system can leverage, provided it can rank its proposed connections. A successful system ought to rank published connections higher than those we randomly created. We train a system given historical information, and create the "published," "highly-cited," and "noise" query sets. We pose these connections to an HG system, and rank its outputs in order to plot ROC curves, which determine whether published predicates are preferred to noise. Through the area under these ROC curves, a HG system demonstrates its quality at a large scale without expert analysis.

Our challenge starts with the Semantic Medical Database (SEMMEDDB) [120] that contains predicates extracted from MEDLINE defined on the set of UMLS terms [7]. For instance, predicate "C1619966 TREATS C0041296" represents a discovered fact

"abatacept treats tuberculosis." Because MOLIERE does not account for word order or verb, we look for distinct unordered word-pairs a–c instead. In Section 6.8, we discuss how we may improve MOLIERE to include this unused information.

From there, we select a "cut year." Using the metadata associated with each predicate, we note the date each unordered pair was first published. For this challenge, we train MOLIERE using only information published before the cut year. We then identify the set of SEMMEDDB unordered pairs a–c first published after the cut year provided a and c both occur in that year's UMLS release. This "published set" of pairs represent new connections between existing entities, from the perspective of the HG system. We select 2010 as the cut year for our study in order to create a published set of over 1 million pairs. (Due to practical limitations, our evaluation consists of a randomly chosen subset of 4,319 pairs.)

Additionally, we create a set of "highly-cited" pairs by filtering the published set by citation count. We use data from SEMMEDDB, MEDLINE, and Semantic Scholar to identify 1,448 pairs from the published set that first occur in a paper cited over 100 times. We note that this set is closer to the number of landmark discoveries since the cut-date, given that the published set is large and likely contains incidental or incorrect connections.

To provide negative examples, we generate a "noise set" of pairs by sampling the cut-year's UMLS release, storing the pair only if it does not occur in SEMMEDDB. These pairs represent nonsensical connections between UMLS elements. Although it is possible that we may stumble across novel findings within the noise set, we assume this will occur infrequently enough to not affect our results. We generate two noise pair sets of equal size to both the published and highly-cited sets.

We run a–c queries from each set through MOLIERE and create two ranked lists: published vs. noise (PvN) (8,638 total pairs) and highly-cited vs. noise (HCvN)

(2,896 total pairs). After ranking each set, we generate ROC curves [91], which allow us to judge the quality of an HG system. If more published predicates occur earlier in the ranking than noise, the ROC area will be close to 1; otherwise it will be closer to 0.5.

6.4 New Ranking Methods for Topic Model Driven Hypotheses

Because many HG systems do not currently produce a ranking criteria, such as those systems that instead return topic models [225, 243], we propose here a number of metrics to numerically evaluate the plausibility of potential connections. We implement these metrics within Moliere [225]. This system is open source, and already leverages word embeddings in order to produce topic model output for potential connections — all of which are properties our metrics exploit. Put simply, Moliere takes as input two keywords (a and c), and produces a topic model (T) that describes the structure of relevant documents.

While these metrics are proposed in the context of validation, another extremely important use case is that of the *one-to-many* query. Often during candidate selection, scientists may have a large list of initial potential targets — such as 30,000 genes in the human genome — that they wish to consider. For this, one may run a large set of queries between some disease a, and each target c_i. However, without a ranking criteria, the analysis of each a–c_i connection is left to experts, which is untenable for most practical purposes.

To begin, we note a key intuition underpinning the following metrics, depicted in Figure 6.3. Not only are related objects grouped in a word embedding space, but

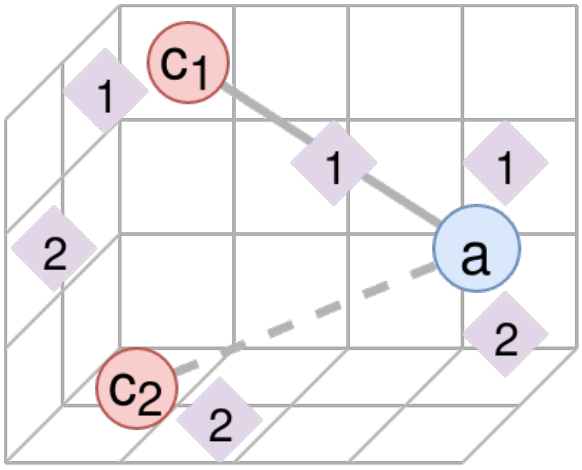


Figure 6.3: The above depicts two queries, a–c_1 and a–c_2, where a–c_1 is a published connection and a–c_2 is a noise connection. We see topics for each query represented as diamonds via $\textsc{Centr}(T_i)$. Although both queries lead to topics which are similar to a, c_1, or c_2, we find that the the presence of some topic which is similar to *both* objects of interest may indicate the published connection.

the distances between words are also meaningful. For this reason we hypothesize, and later show through validation experiments, that one can estimate the strength of an a–c connection by comparing the distance of topics to the embeddings of each a, c, and their midpoint. Note, we use $\epsilon(x)$ to map a text object x into this embedding space, as described in [153]. But, because not all hypotheses or topic models exhibit the same features, we quantify this "closeness" in eleven ways, and then train a polynomial to weight the relevance of each proposed metric.

6.4.1 Similarity Between Query Words

As a baseline, we first consider two similarity metrics that do not include topic information: cosine similarity (CSim) and Euclidean distance (L_2):

$$\textsc{CSim}(a,c) = \frac{\epsilon(a)\cdot\epsilon(c)}{||\epsilon(a)||_2 \times ||\epsilon(c)||_2}\ ,\ L_2(a,c) = ||\epsilon(a)-\epsilon(c)||_2,$$

where a and c are the two objects of interest, and $\epsilon(x)$ is an embedding function (see Section 6.2). Note that when calculating ROC curves for the L_2 metric, we will sort in reverse, meaning smaller distances ought to indicate published predicates.

These metrics indicate whether a and c share the same cluster with respect to the embedding space. Our observation is that this can be a good indication that a and c are of the same kind, or are conceptually related. This cluster intuition is shared by others studying similar embedding spaces [244].

6.4.2 Topic Model Correlation

The next metric attempts to uncover whether a and c are mutually similar to the generated topic model. This metric starts by creating vectors $v(a, T)$ and $v(c, T)$ which express each object's similarity to topic model $T = \{T_i\}_{i=1}^{k}$ derived from an $a - c$ query. We do so by calculating the weighted cosine similarity $\textsc{TopicSim}(x, T_i)$ between each topic T_i and each object $x \in \{a, c\}$, namely,

$$\textsc{TopSim}(x, T_i) = \sum_{(w,p)\in T_i} p \cdot \textsc{CSim}(x, w),$$

where a probability distribution over terms in T_i is represented as word-probability pairs (w, p). This metric results in a value in the interval [-1, 1] to represent the weighted similarity of x with T_i. The final similarity vectors $v(a, T)$ and $v(c, T)$ in $\mathbb{R}^k$ are defined below.

$$\forall x \in \{a, c\} \quad v(x, T) = \begin{bmatrix} \textsc{TopSim}(x, T_1) \\ \textsc{TopSim}(x, T_2) \\ \vdots \\ \textsc{TopSim}(x, T_k) \end{bmatrix}$$

Finally, we can see how well T correlates with both a and c by taking another cosine similarity

$$\textsc{TopicCorr}(a, c, T) = \frac{v(a, T) \cdot v(c, T)}{||v(a, T)||_2 \times ||v(c, T)||_2} \in [-1, 1].$$

If $\textsc{TopicCorr}(a, c, T)$ is close to 1, then topics that are similar or dissimilar to a are also similar or dissimilar to c. Our preliminary results show that if some explanation of the $a - c$ connection exists within T, then many $T_i \in T$ will likely share these similarity relationships.

6.4.3 Similarity of Best Topic Centroid

While the above metric attempts to find a trend within the entire topic model T, this metric attempts to find just a single topic $T_i \in T$ that is likely to explain the $a - c$ connection. This metric is most similar to that depicted in Figure 6.3. Each T_i is represented in the embedding space by taking a weighted centroid over its word probability distribution. We then rate each topic by averaging its similarity with both queried words. The score for the overall hypothesis is simply the highest score among the topics.

We define the centroid of T_i as

$$\textsc{Centr}(T_i) = \sum_{(w,p) \in T_i} \epsilon(w) \cdot p,$$

and then compare it to both a and c through cosine similarity and Euclidean distance. When comparing with CSim, we highly rank T_i's with centroids located within the arc between $\epsilon(a)$ and $\epsilon(c)$. Because our embedding space identifies dimensions that help distinguish different types of objects, and because we trained a 500-dimensional

embedding space, cosine similarity intuitively finds topics that share similar characteristics to both objects of interests. We define the best centroid similarity for CSim as

$$\textsc{BestCentrCSim}(a, c, T) = \max_{T_i \in T} \frac{\textsc{CSim}(a, T_i) + \textsc{CSim}(c, T_i)}{2}.$$

What we lose in the cosine similarity formulation is that clusters within our embedding space may be separate with respect to Euclidean distance but not cosine similarity. In order to evaluate the effect of this observation, we also formulate the best centroid metric with L_2 distance. In this formulation we look for topics that occur as close to the midpoint between $\epsilon(a)$ and $\epsilon(c)$ as possible. We express this score as a ratio between that distance and the radius of the sphere with diameter from $\epsilon(a)$ to $\epsilon(c)$. In order to keep this metric in a similar range to the others, we limit its range to [0, 1], namely, for the midpoint $m = (\epsilon(a) + \epsilon(c))/2$.

$$\textsc{BestCentrL}_2(a, c, T) = \max_{T_i \in T} \left\{ 1 - \frac{\|\textsc{Centr}(T_i) - m\|_2}{\|m\|_2} \right\}$$

6.4.4 Cosine Similarly of Best Topic Per Word

In a similar effort to the above centroid-based metric, we attempt to find topics which are related to a and c, but this time on a per-word (or phrase) basis using $\textsc{TopicSim}(x, T_i)$ from Section 6.4.2. Now instead of looking across the entire topic model, we attempt to identify a single topic which is similar to both objects of interest. We do so by rating each topic by the lower of its two similarities, meaning the best topic overall will be similar to both query words.

$$\textsc{BestTopPerWord}(a, c, T) = \max_{T_i \in T} \min \begin{pmatrix} \textsc{TopSim}(a, T_i), \\ \textsc{TopSim}(c, T_i) \end{pmatrix}$$

6.4.5 Network of Topic Centroids

A majority of the above metrics rely on a single topic to describe the potential connection between a and c, but as Smalheizer points out in [4], a hypothesis may be best described as a "story" — a series of topics in our case. To model semantic connections between topics, we induce a nearest-neighbors network $\mathcal{N}$ from the set of vectors $V = \epsilon(a) \cup \epsilon(b) \cup \{\textsc{Centr}(T_i) | T_i \in T\}$ which form the set of nodes for $\mathcal{N}$. In this case, we set the number of neighbors per node to the smallest value (that may be different for each query) such that there exists a path from a to c. Using this topic network, we attempt to model the semantic differences between published and noise predicates using network analytic metrics.

We depict two such networks in Figure 6.4, and observe that the connectivity between a and c from a published predicate is substantially stronger and more structured. In order to quantify this observed difference, we measure the average betweenness and eigenvector centrality [164] of nodes along a shortest path from a to c (denoted by $a \sim c$) within $\mathcal{N}$ to reflect possible information flow between $T_i \in T$. This shortest path represents the series of links between key concepts present within our dataset that one might use to explain the relationship between a and c. We expect the connection linking a and c to be stronger if that path is more central to the topic network. Below we define metrics to quantify the differences in these topic networks. Such network analytic metrics are widely applied in semantic knowledge networks [207].

$\textsc{TopWalkLength}(a, c, T)$: Length of shortest path $a \sim c$

$\textsc{TopWalkBtwn}(a, c, T)$: Avg. $a \sim c$ betweenness centrality

$\textsc{TopWalkEigen}(a, c, T)$: Avg. $a \sim c$ eigenvalue centrality

$\textsc{TopNetCCoef}(a, c, T)$: Clustering coefficient of $\mathcal{N}$

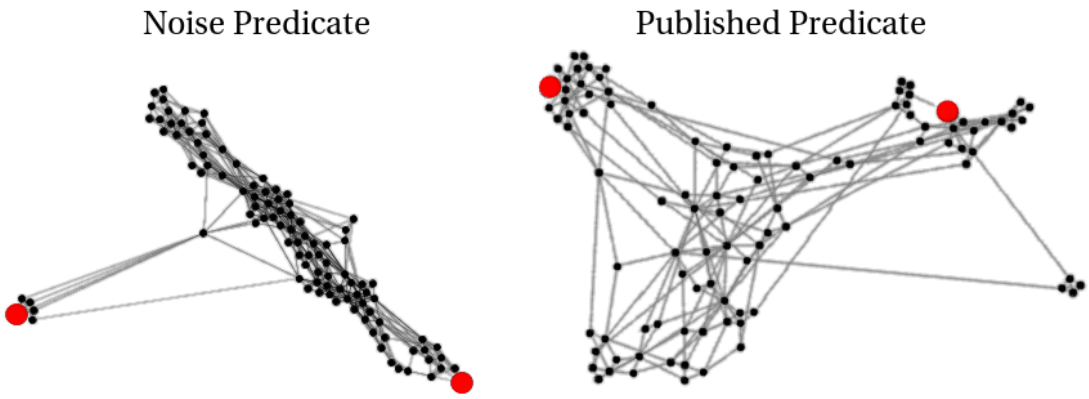


Figure 6.4: Above depicts two topic networks as described in Section 6.4.5. In this visualization, longer edges correspond to dissimilar neighbors. In red are objects a and c, which we queried to create these topic models. We observe that the connectivity between a and c from the published predicate is much higher than in the noisy example.

TopNetMod(a, c, T): Modularity of $\mathcal{N}$

6.4.6 Combination of Multiple Metrics

Each of the above methods are based on different assumptions regarding topic model or embedding space properties exhibited by published connections. To leverage each metric's strengths, we combined the top performing ones from each category into the following PolyMultiple method. We explored polynomial combinations in the form of $\sum_i \alpha_i x_i^{\beta_i}$ for ranges of $\alpha_i \in [-1, 1]$ and $\beta_i \in [1, 3]$ after scaling each x_i to the $[0, 1]$ interval. Through a blackbox optimization technique, we searched over one-million parameter combinations. In doing so we maximize for the AUC of our validation curve by sampling each α_i and β_i from their respective domains. We perform this search stochastically, sampling from parameter space and limiting our search space as we find stable local-minima. Our results represent the best parameter

values determined after one-million parameter samples.

$$\begin{aligned}\textsc{PolyMultiple}(a,c,T) = \alpha_1 \cdot L_2^{\beta_1} + \alpha_2 \cdot \textsc{BestCenterL}_2^{\beta_2} \\ + \alpha_3 \cdot \textsc{BestTopPerWord}(a,c,T)^{\beta_3} + \alpha_4 \cdot \textsc{TopCorr}(a,c,T)^{\beta_4} \\ \alpha_5 \cdot \textsc{TopWalkBtwn}(a,c,T)^{\beta_5} + \alpha_6 \cdot \textsc{TopNetCCoef}(a,c,T)^{\beta_6}\end{aligned}$$

6.5 Results and Lessons Learned

As described in Section 6.3, our goal is to distinguish publishable connections from noise. We run Moliere to generate topic models related to published, noise, and highly-cited pairs. Using this information, we plot ROC curves in Figures 6.5 and 6.6, and summarize the results in Table 6.2. These plots represent an analysis of 8,638 published vs. noise (PvN) pairs and 2,896 (HCvN) pairs (half of each set are noise). *Unfortunately, no alternative general-purpose query HG systems that perform in a reasonable time are freely available for the comparison with our ranking methods.* **Topic Model Correlation** metric (see Section 6.4.2) is a poorly performing metric with an ROC area of 0.609 (PvN) and 0.496 (HCvN). The core issue of this method is its sensitivity to the number of topics generated, and given that we generate 100 topics per pair, we likely drive down performance through topics which are unrelated to the query. In preliminary testing, we observe this intuition for queries with only 20 topics, but also find the network-biased metrics are less meaningful. In Section 6.8 we overview a potential way to combine multiple topic models in our analysis.

Surprisingly, this metric is less able to distinguish highly-cited pairs, which we suppose is because highly-cited connections often bridge very distant concepts [181] and likely results in more noisy topic models. Additionally, we may be able to limit

this noise by tuning the number of topics returned from a query, as described in Section 6.8.

L_2-based metrics exhibit even more surprising results. BestCentrL_2 performs poorly, with an ROC area of 0.578 (PvN) and 0.587 (HCvN), while the much simpler L_2 metric is exceptional, scoring a 0.783 (PvN) and 0.809 (HCvN). We note that if two words are related, they are more likely to be closer together in our vector space. We evaluate topic centroids based on their closeness to the midpoint between a and c, normalized by the distance between them, so if that distance is small, the radius from the midpoint is small as well. Therefore, it would seem that the distance between a and c is a better connection indication, and that the result of the centroid measurement is worse if this distance is small.

CSim-based metrics are more straightforward. The simple CSimmetric scores a 0.709 (PvN) and 0.703 (HCvN), which is interestingly consistent given that the L_2 metric increases in ROC area given highly-cited pairs. The BestTopicPerWord metric only scores a 0.686 (PvN), but increases substantially to 0.731 (HCvN). The topic centroid method BestCentroidCSim is the best cosine-based metric with an ROC area of 0.719 (PvN) and 0.742 (HCvN). This result is evidence that our initial hypothesis described in Figure 6.3 holds given cosine similarity, but as stated above, does not hold for Euclidean distance.

Topic network metrics are all outperformed by simple L_2, but we see interesting properties from their results that help users to interpret generated hypotheses. For instance, we see that TopicWalkBtwn is a negative indicator while TopicWalkEigen is positive. Looking at the example in Figure 6.4 we see that a and c are both far from the center of the network, connected to the rest of the topics through a very small number of high-betweenness nodes. In contrast, we see that in the network created from a published pair, the path from a to c is more central. We

Metric Name	PvN ROC	HCvN ROC
PolyMultiple	0.834	0.874
L_2*	0.783	0.809
CSim	0.709	0.703
BestCenterL_2	0.578	0.587
BestCenterCSim	0.719	0.742
BestTopicPerWord	0.686	0.731
TopicCorr	0.609	0.496
TopicWalkLength*	0.740	0.778
TopicWalkBtwn*	0.659	0.658
TopicWalkEigen	0.585	0.582
TopicNetCCoef*	0.651	0.638
TopicNetMod*	0.659	0.628

Table 6.2: The above summarizes all ROC area results for all considered metrics on the set of published vs. noise pairs (PvN) and highly-cited vs. noise pairs (HCvN). Metrics marked with a (*) have been sorted in reverse order for the ROC calculations.

also see a denser clustering for the noise pair network, which is echoed by the fact that TopicNetCCoef and TopicNetMod are both negative indicators. Lastly, we see that TopicWalkLength performs the best out of these network approaches, likely because it is most similar to the simple L_2 or CSim metrics.

Combination of metrics, PolyMultiple, significantly outperforms all others with ROC areas of 0.834 (PvN) and 0.874 (HCvN). This is unsurprising because each other metric makes a different assumption about what sort of topic or vector configuration best indicates a published pair. When each is combined, we see not only better performance, but their relative importances. By studying the coefficients of our polynomial we observe that the two L_2-based metrics are most important, followed by the topic network methods, and finally by TopicWalkCorr and BestTopicPerWord. Unsurprisingly, the coefficient signs correlate directly with whether each metric is a positive or negative indication as summarized in Table 6.2. Additionally, the ordering of importance roughly follows the same ordering as the ROC areas.

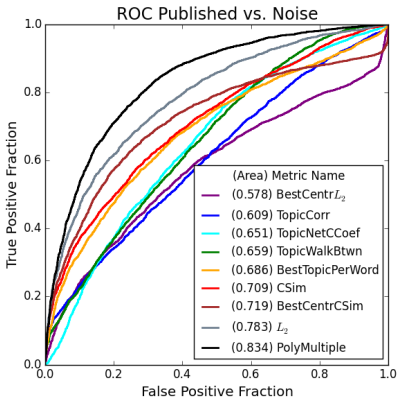


Figure 6.5: The above ROC curves show the ability for each of our proposed methods to distinguish the MOLIERE results of published pairs from noise. We use our system to generate hypotheses regarding 8,638 pairs, half from each set, on publicly available data released prior to 2,015. We only show the best performing metrics from Section 6.4.5 for clarity.

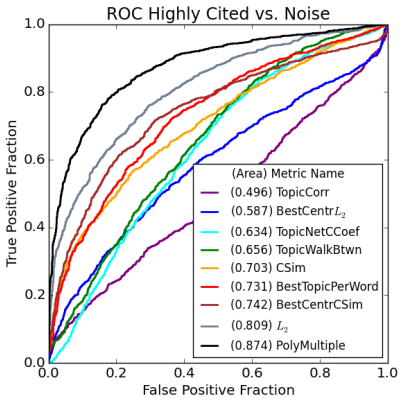


Figure 6.6: The above ROC curves show the ability for each of our proposed methods to distinguish the MOLIERE results of highly-cited pairs from noise. We identify 1,448 pairs who first occur in papers with over 100 citations published after our cut date. To plot the above ROC curve, we also select an random subset of equal size from the noise pairs.

6.6 Case-Study: HAND and DDX3 Candidate Selection

Our proposed validation method is rooted in the process of candidate selection. To demonstrate our method's applicability to real-world scenarios, we applied the above methods to a series of queries surrounding Human Immunodeficiency Virus - associated dementia (or HIV-associated neurodegenerative disease, HAND). HAND is one of the most common and clinically important complications of HIV infection [125]. The brain-specific effects of HIV are of great concern because the HIV-infected population is aging and unfortunately revealing new pathologies [210, 26]. About 50% of HIV-infected patients are at risk of developing HAND, which might be severely worsened by abusing drugs such as cocaine, opioids and amphetamines [24, 42].

We generated over 30,000 queries, each between HAND and a gene from the HUGO Gene Nomenclature Committee dataset [2]. The network that generated these results consisted of the 2017 MEDLINE dataset, the 2017AB UMLS release, and the 2016 SEMMEDDB release (latest at the time). We trained FASTTEXT using all of the available titles and abstracts, about 27 million in total, and selected a dimensionality of 500 for our word embeddings. Our results consist of each disease-gene query ranked by our POLYMULTIPLE metric.

Based on this ranking we select the first $\sim$1000 genes for further analysis. We observe that many of the top genes — such as APOE-4, T-TAU, and BASE1, which occur in our top five — are known to be linked to dementia. So to direct our search to yet-unknown connections, we select those genes that have no previous connection to HAND, but still ranked highly overall. This process limits our search to those proteins that have known selective compounds, which were often tested animal models or clinical trials.

From this candidate set we selected Dead Box RNA Helicase 3 (DDX3). We tested the activity of a DDX3 inhibitor on the tissue culture model of HAND, which is widely used for the analysis combine neurotoxicity of HIV proteins and drugs of abuse. Here we tested the effect of the DDX3 inhibition on combined toxicity of most toxic HIV protein, Trans-Activator of Transcription (Tat). The mouse cortical neurons had been treated with HIV Tat followed by the addition of cocaine. The combination of Tat and cocaine kills more than 70% of the neurons, while the inhibitor protects the neurons from Tat/cocaine toxicity (Figure 6.7).

Based on the analysis, we formulate following hypothesis: Exposing neurons with Tat protein causes internal stress and results in the formation of Stress-Granules (SGs) — the structures in cytoplasm formed by multiple RNAs and proteins. These gel-like structures sequester cellular RNA from translation, and the formation of SGs requires enzymatically active Dead Box RNA Helicase 3. The formation of SGs also allows the neurons to wait out the stress. However, prolonged stress associated with HIV-Tat treatment leads to the formation of pathological stress granules, which are denser and have a different composition relative to "normal" ones. Additional exposure to cocaine further exaggerates the "pathological" SGs and eventually causes neuronal death. The hypothesis, initially generated with MOLIERE, led to the following finding: *Treatment with a DDX3-specific inhibitor blocks the enzymatic activity of the DDX3. This lack of enzymatic activity, in turn, blocks Tat-dependent stress granules from formating and protects neurons from the combined toxicity of Tat and cocaine.* In Figure 6.7, we demonstrate the hypothesis scheme. Thus, the application of the automated HG system pointed to a new avenue for anti-HAND therapy and to the prototype of a small molecule for drug development.

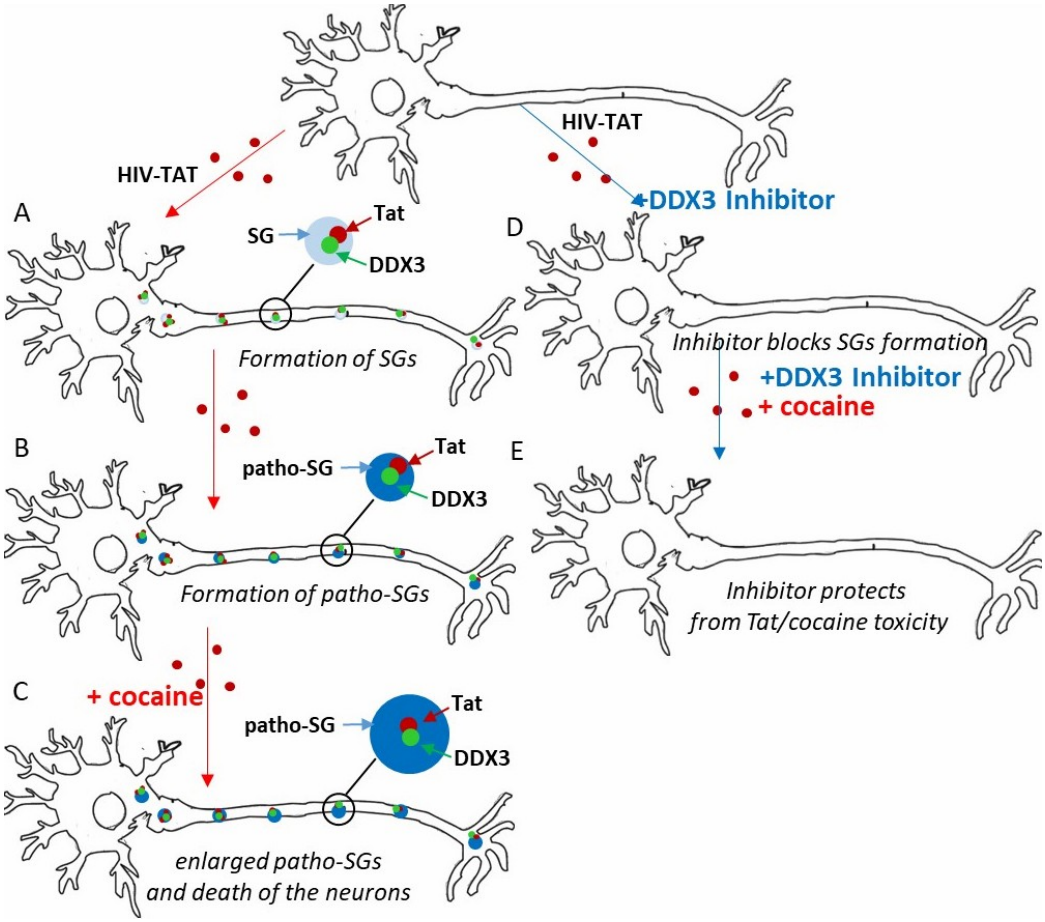


Figure 6.7: Scheme of the hypothesis of Stress-Granule dependent mechanism of neuroprotection by DDX3 inhibitor. Neurons are curved figures. Treatment with HIV-Tat leads to DDX3-dependent formation of SGs (A), which transform from "normal" to "pathological" (B). The addition of cocaine further enlarges the SGs and leads to the death of the neurons (C). Treatment with DDX3 specific inhibitor blocks DDX3 enzymatic activity and Tat-dependent SG formation (D) and protects the neurons from cocaine-induced death (E).

6.7 Related Work and Proposed Validation

The HG community struggles to validate its systems in a number of ways. Yetisgen-Yildiz and Pratt, in their chapter "Evaluation of Literature-Based Discovery Systems," outline four such methods (M1-M4) [41, 253].

M1: Replicate Swanson's Experiments. Swanson, during his development of ARROWSMITH [203], worked alongside medical researchers to uncover a number of new connections. These connections include the link between Raynaud's Disease and Fish Oil [219], the link between Alzheimer's Disease and Estrogen [202] and the link between Migraine and Magnesium [220]. As discussed in [253], a number of projects have centered their validation effort around Swanson's results [97, 28, 211, 104, 170]. These efforts always rediscover a number of findings using information before Swanson's discovery date, and occasionally apply additional metrics such as precision and recall in order to quantify their results [91].

While limiting discussion to Swanson's discoveries reduces the domain of discovery drastically, at its core this method builds confidence in a new system through its ability to find known connections. We expand on this idea by validating automatically and on a massive scale, freeing our discourse from a single researcher's findings.

M2: Statistical Evaluation. Hristovski et al. validate their system by studying a number of relationships and note their confidence and support with respect to the MEDLINE document set [103]. Then, they can generate potential relationships for the set of new connections added to UMLS [7] or OMIM [90]. By limiting their method to association rules, Hristovski et al. note that they can validate their system by predicting UMLS connections using data available prior to their publications. Therefore, this method is similar to our own, but we notice that restricting discussion to

only UMLS gene-disease connections results in a much smaller set than the predicate information present with SemMedDB.

Pratt et al. provide additional statistical validation for their system LitLinker [170]. This method also calculates precision and recall, but this time focusing on their B-set of returned results. Their system, like ARROWSMITH [203], returns a set of intermediate terms which may connect two queried entities. Pratt et al. run LitLinker for a number of diseases on which they establish a set of "gold standard" terms. Their method is validated based on its ability to list those gold-standard terms within its resulting B-sets. *This approach requires careful selection of a (typically small) set of gold-standard terms, and is limited to "ABC" systems like ARROWSMITH, which are designed to identify term lists* [201].

M3: Incorporating Expert Opinion. This ranges from comparisons between system output and expert output, such as the analysis done on the Manjal system [211], to incorporating expert opinion into gold-standard terms for LitLinker [170], to running actual experiments on potential results by Wren et al. [251]. Expert opinion is at the heart of many recent systems [243, 225, 145, 221], including the previous version of our own. This process is both time consuming and risks introducing significant bias into the validation.

Spangler incorporates expert knowledge in a more sophisticated manner through the use of visualizations [208, 209]. This approach centers around visual networks and ontologies produced automatically, which allows experts to see potential new connections as they relate to previously established information. This view is shared by systems such as DiseaseConnect [145] which generates sub-networks of ONIM and GWAS related to specific queries. Although these visualizations allow users to quickly understand query results, they do not lend themselves to a numeric and massive evaluation of system performance.

BioCreative, a set of challenges focused on assessing biomedical text mining, is the largest endeavor of its kind, to the best of our knowledge [101]. Each challenge centers around a specific task, such as mining chemical-protein interactions, algorithmically identifying medical terms, and constructing causal networks from raw text. Although these challenges are both useful and important, their tasks fall under the umbrella of *information retrieval* (and not HG) because their tasks compare expert analysis with software results given the same text.

M4: Publishing in the Medical Domain. This method is exceptionally rare and expensive. The idea is to take prevalent potential findings and pose them to the medical research community for another group to attempt. Swanson and Smalheiser rely on this technique to solidify many of their early results, such as that between magnesium deficiency and neurologic disease [205].

Bakkar et al. take a similar approach in order to demonstrate the efficacy of Watson for Drug Discovery [18, 209] To do so, this work begins by identifying 11 RNA-binding proteins (RBPs) known to be connected to Amyotrophic Lateral Aclerosis (ALS). Then, the automated system uses a recommender system to select RPBs that exhibit similar connection patterns within a large document co-occurrence network. Domain scientists then explore a set of candidates selected by the computer system, and uncover five RPBs that were previously unrelated to ALS.

An alternative to the domain-scientist approach is taken by Soldatova and Rzhetsky wherein a "robot scientist" automatically runs experiments posed by their HG system [206, 180]. This system uses logical statements to represent their hypotheses, so new ideas can be posed through a series of implications. Going further, their system even identifies statements that would be the most valuable if proven true [181]. *However, the scope of experiments that a robot scientist can undertake is limited; in their initial paper, the robot researcher is limited to small-scale yeast*

experiments. Additionally, many groups cannot afford the space and expense that an automated lab requires.

6.8 Deployment Challenges and Open Problems

Validation Size. Our proposed validation challenge involves ranking millions of published and noise query pairs. However, in Section 6.5 we show our results on a randomly sampled subset of our overall challenge set. This was necessary due to performance limitations of Moliere, a system which initially required a substantial amount of time and memory to process even a single hypothesis. To compute these results, we ran 100 instances of Moliere, each on a 16 core, 64 GB RAM machine connected to a ZFS storage system. Unfortunately, performance limitations within ZFS created a bottleneck that both limited our results and drastically reduced cluster performance overall. Thus, our results represent a set of predicates that we evaluated in a limited time period.

System Optimizations. While performing a keyword search, most network-centered systems are either I/O or memory bound simply because they must load and traverse large networks. In the case of Moliere, we initially spent hours trying to find shortest paths or nearby abstracts. But, we found a way to leverage our embedding space and our parallel file system in order to drastically improve query performance. In brief, one can discover a relevant knowledge-network region by inducing a subnetwork on a and c and expanding that selection by adding i^{th} order neighbors until a and c are connected. From our experiments, i rarely exceeds 4. This increases performance because, given a parallel file system and p processors, identifying the subnetwork from an edge list file is in order $\mathcal{O}(ni/p)$. The overall effect reduced the wall-clock runtime of a single query from about 12 hours to about 5-7 minutes. Ad-

ditionally, we reduced the memory requirement for a single query from over 400GB to under 16GB.

Highly-Cited Predicates. Identifying highly-cited predicates requires that we synthesize information across multiple data sources. Although SemMedDB contains Medline references for each predicate, neither contains citation information. For this, we turn to Semantic Scholar because not only do they track citations of medical papers, but they allow a free bulk download of metadata information (many other potential sources either provide a very limited API or none at all). In order to match Semantic Scholar data to Medline citation, it is enough to match titles. This process allows us to get citation information for many Medline documents, which in turn allow us to select predicates whose first occurrence was in highly-cited papers. We explored a number of thresholds for what constitutes "highly cited" and selected 100 because it was a round number and selected a sizable predicate set. Because paper citations follow a power-law distribution, any change drastically effects the size of this set. We note that the set of selected predicates was also limited by the quality of data in Semantic Scholar, and that the number of citations identified this was appeared to be substantially lower than that reported by other methods.

Quality of Predicates. Through our above methods we learned that careful ranking methods can distinguish between published and noise predicates, but there is a potential inadequacy in this method. Potentially, some predicates that occur within our published say may be untrue. Additionally, it is possible that a noise predicate may be discovered to be true in the future. If Moliere ranks the published predicate which is untrue below the noise predicate which is, the result would be a lower ROC area. This same phenomena is addressed by Yetisgen-Yildiz and Pratt when they discuss the challenges present in validating literature-based discovery systems [253] — if a HG systems goal is to identify novel findings, then it *should* find different

connections than human researchers.

We show through our results that despite an uncertain validation set, there are clearly core differences between publishable results and noise, which are evident at scale. Although there may be some false positives or negatives, we see through our meaningful ROC curves that they are far outnumbered by more standard predicates.

Automatic Question Posting. Going forward we wish to study highly ranked noise predicates for the purpose of automatic question posing. This would mean that a system would search through its set of entities, run queries, and report the most promising potential new connections. In order to do this effectively we need to gain an understanding of how we can intelligently search local regions of our knowledge network and how to define locality for this task.

Comparison with ABC Systems. Additionally, we would like to explore how our ranking methods apply to traditional ABC systems. Although there are clear limitations to these systems [201], many of the original systems such as ARROWSMITH follow the ABC pattern. These systems typically output a list of target terms and linking terms, which could be thought of as a topic. If we were to take a pre-trained embedding space, and treated a set of target terms like a topic, we could likely use our methods from Section 6.4 to validate any ABC system.

Verb Prediction. We noticed, while processing SemMedDB predicates, that we can improve Moliere if we utilize verbs. SemMedDB provides a handful of verb types, such as "TREATS," "CAUSES," or "INTERACTS_WITH," that suggest a concrete relationship between the subject and object of a sentence. Moliere currently outputs a topic model that can be interpreted using our new metrics, but does not directly state what sort of connection may exist between a and c. Thus we would like to explore accurately predicting these verb types given only topic model information.

Interpretability of Hypotheses remains one of the major problems in HG systems. Although topic-driven HG partially resolve this issue by producing readable output, we still observe many topic models T (i.e., hypotheses) whose $T_i \in T$ are not intuitively connected with each other. While the proposed ranking is definitely helpful for understanding T, it still does not fully resolve the interpretability problem. One of our current research directions is to tackle it using text summarization techniques.

Scope. While we focus on biomedical science, any field that is accurately described by *entities* that *act* on one another benefits from our network and text mining methods. For instance, economic entities, such as governments or the upper/lower class, interact via actions such as regulation or boycott. Similarly, patent law consists of inventions and the components that comprise them. Mathematics, in contrast, is not served by this representation — the algebra does not *act* on other math entities. Here automatic theorem proving is better equipped to generate hypotheses. We are presently unsure if the same is true for computer science.

Chapter 7

Are Abstracts Enough for Hypothesis Generation?

Abstract

The potential for automatic hypothesis generation (HG) systems to improve research productivity keeps pace with the growing set of publicly available scientific information. But as data becomes easier to acquire, we must understand the effect different textual data sources have on our resulting hypotheses. Are abstracts enough for HG, or does it need full-text papers? How many papers does an HG system need to make valuable predictions? How sensitive is a general-purpose HG system to hyper-parameter values or input quality? What effect does corpus size and document length have on HG results? To answer these questions we train multiple versions of knowledge network-based HG system, Moliere, on varying corpora in order to compare challenges and trade offs in terms of result quality and computational requirements. Moliere generalizes main principles of similar knowledge network-based HG systems and reinforces them with topic modeling components. The corpora include the

abstract and full-text versions of PubMed Central, as well as iterative halves of MEDLINE, which allows us to compare the effect document length and count has on the results. We find that, quantitatively, corpora with a higher median document length result in marginally higher quality results, yet require substantially longer to process. However, qualitatively, full-length papers introduce a significant number of intruder terms to the resulting topics, which decreases human interpretability. Additionally, we find that the effect of document length is greater than that of document count, even if both sets contain only paper abstracts.

7.1 Introduction

While the driving pace of research accelerates [130, 233], computer-aided methods become increasingly more important for improving scientific productivity. This is especially apparent in medicine and life sciences — the National Institute of Health introduced 1.1 million papers to MEDLINE in 2017 alone. Hypothesis Generation (HG) [208] is the process of finding unknown-yet-useful connections from the set of publicly available information. Usually, this involves a combination of text processing, data mining, and graph-based approaches.

When scientists miss cross-cutting connections, they leave behind *undiscovered public knowledge* [221], which many aim to detect through Hypothesis Generation (also called Literature-Based Discovery) Systems [41, 208]. Early attempts find important connections from the co-occurrences of keywords across paper titles [202], while more advanced methods, such as recommender systems [209] and topic modeling [225], rely on abstracts and preprocessed longer documents such as full-text papers in IBM Watson Drug Discovery system. No matter the method, every system is primarily dependent on text, yet to the best of our knowledge no one has directly

and systematically addressed the effect corpus quality, size, and document length has on the quality of knowledge network-based HG systems.

Because of huge practical importance of HG systems for accelerating biomedical discovery, there are many controversial arguments on the need of full-text papers in the scientific community [27, 190, 192, 248]. However, in the vast majority of studies, this issue is raised with respect to traditional *information retrieval* (IR) and *data mining* tasks and systems, which usually do not substitute HG. Clearly, full-texts are more beneficial for IR as they contain more information, but does the same hold for HG?

Our Contribution: We explore the effect corpus size and document length have on knowledge network-based HG systems, primarily by comparing their performance with full-text papers against abstracts. Our experimental studies are based on the HG system MOLIERE [225] that extends the basic principals of knowledge discovery networks introduced in earlier works [203, 208, 243]. This centers around two major studies: the first comparing the performance of our system trained on abstract and full-text versions of the same document set, the second comparing the performance of iterative halves of a large abstract set. Our results, while experimentally focused on MOLIERE, have important implications to other similar systems [209, 239, 243].

We evaluate our results in terms of quality, using the hypothesis ranking techniques developed in [226], and discuss practical challenges in terms of memory consumption, runtime, and interpretability. We find that corpora with a higher median document length perform better than those with shorter documents and that this effect can be more substantial than simply adding more documents. Most importantly, when comparing a corpus of full-text documents against a corpus of the abstracts of those same documents, we notice a *marginal* improvement in quality (if at all), yet a **45**$\times$ increase in runtime from 100 seconds to 75 minutes.

To perform our evaluation, we create multiple *instances* of our HG system. By this we mean that we perform our entire knowledge network construction process, starting from raw documents and ending at a large knowledge network [225], independently for each corpus version. We start our study with data from MEDLINE as well as PubMed Central (PMC). The former contains over 24 million abstracts dating back to the late 1800's, while the latter contains 4 million full-text documents (only 1.7M in XML) and started in the year 2000. Using PMC we explore the effect document length has on HG systems by training two instances of MOLIERE on the abstract and full-text versions of the same corpus. With MEDLINE, we evaluate the effect of corpus size using five instances of MOLIERE trained on repeated halves of the data set.

Our validation compares instances by their ability to distinguish published from noise connections based on their resulting hypotheses, given that the instance has no available information regarding either. This begins by selecting a *cut-year* — we choose 2015 — and filtering our data sources to only include information that was available prior to it. We extract recently published connections from SEMMEDDB, a database of MEDLINE *predicates* (subject-verb-object structures), by identifying those predicates that first occur after the cut year [120]. We additionally create an equal number of randomly sampled connections that do not exist within SEMMEDDB. By generating hypotheses for all of these connections, and ranking their results with regard to a number of metrics, we plot ROC curves that describe the quality of our system.

7.2 Background: Literature-Based HG

Swanson first introduced Hypothesis Generation (HG) and his ABC model for knowledge discovery [219]. He found a connection between Raynaud's syndrome (A) and fish oil (C) through their connection with blood viscosity (B). Although Swanson's early work managed to extract these ideas using only the titles of MEDLINE articles, recent systems, such as BRAINSCANR [239], DISEASECONNECT [145], and MOLIERE [225], use modern text-mining technologies to identify latent features from abstracts in order to better extract semantic information. These systems use abstracts for two reasons. First, abstracts are more easily available than full-text data. For example, MEDLINE contains 24 million abstracts, while only 4 million full-text documents are available through PubMed Central (and most are not available in XML). Second, there is conventional understanding that abstracts contain effective summaries of key findings [67], which means they have a better signal-to-noise ratio than full-text documents, which often contain textual information that is less relevant for the HG-task (e.g., references to figures, a detailed description of experimental conditions, inappropriate background). However, the latter has not been systematically tested in the literature.

We do, however, observe at least one commercial system that uses full-text documents. Watson for Drug Discovery [209] includes a sophisticated entity extraction and ontology creation pipeline that allows it to overcome the typical signal-to-noise challenges present in these longer texts. Additionally, the Watson discovery methods, such as co-occurrence networks and recommender systems, function on top of these pre-processed results, which means that Watson does not need to process full papers while performing individual queries. However, we are limited in our comparison because Watson, as well as most other HG systems, are proprietary or closed-source

and not available for a systematic comparison. In Section 7.5 we explore the tradeoffs present between these choices of methods.

7.2.1 Abstract versus Full-Text Comparisons

Previous studies that compare abstracts and full-text papers have done so for the purpose of information retrieval and pattern discovery in data mining (IR/DM). While IR/DM's goal is to extract known information (including finding patterns) in (un)structured data [152], HG's goal is to propose novel hypotheses and discover unknown information (not necessarily represented as a pattern). With this distinction in mind, it is clear that full-text documents, by nature of their length, contain more retrievable information than abstracts.

Shah et al. [192] perform keyword extraction from 104 articles published in Nature Genetics, showing that the full text of an article can contain as many as four times more relevant keywords than its abstract. Schuemie et al. [190] extract keywords from around 4,000 biomedical articles. They similarly find that full-texts include substantially more information than abstracts, leading to a greater number of identified keywords. Westergaard et al. [248] confirm this finding in the context of named entity recognition (protein–protein, disease–gene, and protein subcellular associations) from 15 million biomedical full-text articles.

While the above studies show that more text is better for information extraction, they also show that there is significant heterogeneity in information density between different sections of an article. Both Shah et al. [192] and Schuemie et al. [190] find that the information density (i.e., the ratio of relevant to irrelevant keywords) is highest in the abstract. Given that full-text articles are more difficult to obtain, restricting the analysis to abstracts can be a sensible choice (given 24M abstracts

are available through MEDLINE, but only about 4M full-text articles are available through PMC). Further, using full-text articles always requires significant efforts in additional text preprocessing, such as parsing parenthesized sentences or extracting text in footnotes.

Blair et al. [27] note the limitations in comprehending full-texts — longer documents typically mention many different concepts. For instance, in our later results, we notice that many full-text documents contain significant information related to experimental procedure, which may obfuscate more relevant information regarding conclusions of new findings. This added "noise" can decrease the quality of an analysis, depending on which metric is deemed most important. Sinclair and Webber [200], for example, perform Gene Ontology (GO) code classification on 1,000 articles. Their results show that classification on full-text articles has the highest recall but lowest precision, while the opposite was true when only titles and abstracts were used.

Outside the domain of biomedical literature research, there are similarly mixed results on the question whether more text is necessarily better. In an analysis of data from the online social network Twitter, Conover et al. [55] find that a classifier trained on hashtags (i.e., user-selected keywords attached to a message) outperforms a classifier trained on the full text of tweets for the purpose of predicting users' political alignment. They argue that this result is due to a better signal-to-noise ratio of keywords compared to full-text messages.

Syed and Spruit [227] apply LDA topic modeling [31] to full-text articles and abstracts from the domain of fisheries and aquatic sciences. Comparing the quality of estimated topics (both statistically and through human expert coders), they find that full text produces more high-quality topics than abstracts, but only when estimated on a small data set with 4,417 articles from a single journal. On a larger data set with around 15,000 articles from 12 journals, both full text and abstracts produce

similarly good results.

To summarize, previous work has found that more text is generally better for IR/DM tasks, but many applications suffer when trained with full texts because a longer length comes with a reduced signal-to-noise ratio, even for IR/DM [54]. Given that full-text documents are much harder to acquire and require more computational resources to process, it is important to quantify these trade-offs in the context of prediction in HG.

7.3 Methodology

In order to understand the effect corpus size and document length has on knowledge-network-based HG systems, we train multiple instances of Moliere using data from both PubMed Central (PMC) as well as Medline. For practical purposes, we limit our discussion to this system, but note our results have further-reaching implications. In this section, we provide an overview of these data sources, outline our training and validation procedure, and explain the quantitative and qualitative results we collect.

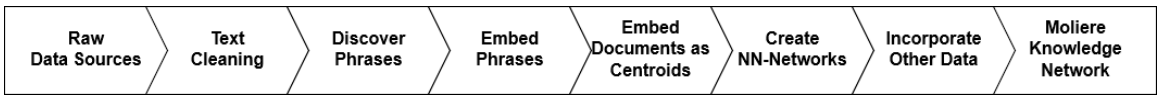
7.3.1 Moliere Pipeline Background

The process of generating fruitful hypotheses via Moliere begins with textual data sources. In this work, we will focus on the titles and abstracts provided by Medline, or the plain-text releases of full-text papers provided by PubMed Central as our input data, but it is useful to keep in mind that Moliere is intended to work well given various input sources. From there, we leverage recent phrase mining tools, such as ToPMine [69] or AutoPhrase [193], to segment our raw text into more easily interpretable n-grams. We find that this step is crucial to making our downstream

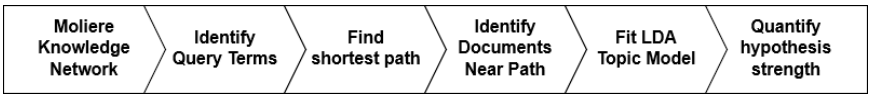
model output human understandable. From there we run FastText [114], a recent advance in the save vein as Word2Vec [153], to embed each n-gram into a 500-dimensional vector space. This process allows us to mathematically describe the semantic similarities between our terms through simple metrics such as L_2 norm or cosine similarity. We then project each document into the space by taking a weighted average of each n-grams embedding with respect to that terms TF-IDF score. Finally, we create a nearest-neighbors network within the abstract set, and separately within the n-gram set. Links between these sets derive from the TF-IDF scores between abstracts and n-grams. In addition, we introduce UMLS terms, codified medical entities with known links between them, as a "backbone" to the overall network. A diagram describing this process is shown in Figure 7.1(a).

To query this network for hypotheses, we begin with two nodes of interest, a and c. Typically, these are either keywords or UMLS terms. We identify a region of the overall network containing both keywords, and run Dijkstra algorithm within that region to quickly find a shortest-path connecting both terms. This path, at a high level, represents a series of terms and documents that ought to outline the relationship between a and c, but in practical cases, this path alone is not sufficient for a human scientist to form a useful hypothesis. Therefore, we increase the amount of relevant information by taking a large set of nearby documents, typically on the order of 5-15 thousand, that are first or second-degree neighbors to the path. This collection of nearby papers represents a sizable portion of related research, which more likely describes the nature of an a-c relationship. We use LDA topic models to uncover the structure of this document subset, which offers some initial insights though the clustering of interesting terms. The process of creating these topics from relevant document sets is diagrammed in Figure 7.1(b).

More recently, we proposed a number of metrics to evaluate a-c relevance



(a) Network Construction Pipeline



(b) Query Processing Pipeline

Figure 7.1: Above depicts the network construction and query pipeline. First, input from raw data sources is tokenized into meaningful n-grams, then embedded, and used with other features and sources to create a nearest-neighbors network. Once the network is constructed, the query process details how we use shortest paths to identify relevant abstracts on which we generate LDA topic models.

in [226]. This work describes how a number of embedding-based relationships, further summarized in the following section, quantify the fruitfulness of an individual query. While we use these metrics to validate our approach in the previously mentioned work, we leverage them here for the purpose of a numerical comparison between different data sources.

7.3.2 Metrics for Hypothesis Ranking

Many have noted key challenges that surround evaluating hypothesis generation systems [41]. Because these systems attempt to locate novel research directions, unknown to even those constructing the system itself, it is difficult to distinguish a proposed hypothesis that is incorrect verses one that is true yet unintuitive. Due to this conceptual limitation, many projects validate their system by simply rediscovering a handful of "gold-standard" connections [246, 211, 104, 170]. Some few projects show their utility beyond the gold-standard by incorporating expert analysis and experiments [219, 18, 226]. While these results are important to show real-world application areas for hypothesis generation, lab work is time consuming, expensive,

and clearly does not scale for large validation sets. For these reasons, in [226], we present a number of metrics that estimate the potential of an automatically generated hypothesis. In that work we demonstrate the usefulness of these metrics to identify recent fruitful hypotheses given historical training data. Additionally, we follow the recommendations of our metrics to identify new gene treatment targets for HIV associated neurocognitive disorder. In this work we use these same metrics to numerically compare the performance of a hypothesis generation system trained on abstracts against the same system trained on full text versions of the same papers.

Our metrics, which are summarized below, are predicated upon known properties of word embeddings. Mikolov et al. in [155] demonstrate that their word embeddings, which were trained on 6 billion news articles from the Google News corpus, capture a latent space with meaningful distances. For instance, the distance between the vectors for "man" and "woman" is similar to that between "king" and "queen". This gender-encoding distance is similarly seen for other male-female relationships across the English language, which is also observed in country-capital relationships as well as that of verb tense. Furthermore, similar words are grouped by their semantic meaning. We observe this property in our own embeddings trained in our previous work on over 25 million MEDLINE abstracts.

From these observations we derive the following metrics. Here, a and c are two terms of a proposed hypothesis, the plausibility of which we would like to estimate. T is an LDA topic model generated from a subset of papers relevant to a and c, and $T_i \in T$ is a single topic. Additionally, $\epsilon(x)$ is an embedding function that maps a term or a topic into an embedding space with the previously described properties. In the case of $\epsilon(T_i)$ we simply calculate a weighted centroid for topic T_i.

The simplest metric, L_2, is simply the norm of $\epsilon(a) - \epsilon(c)$. In our previous work we also explored the cosine similarity of our term vectors, but L_2 was our

higher performer. Next in complexity is CentrL$_2$ that captures the distance between the $\epsilon(a), \epsilon(c)$ midpoint from the topic model. We observe that for a hypothesis to be supported, at least one LDA cluster ought to center between the search terms. Then, TopicPerWord relaxes the assumption that topics are best represented as a centroid, and instead treats them as a weighed point cloud. Therefore, we average the distance between each $\epsilon(x) \forall x \in T_i$, and $(\epsilon(a) + \epsilon(c))/2$, weighting them by $P(x|T_i)$. TopicCorr calculates the correlation between all topics in a topic model with respect to both a and c. Put plainly, if a topic is close to a, is it also likely to be close to c? Next, to calculate TopicWalkBetweenness we generate a nearest-neighbors network containing $\epsilon(T_i) \forall T_i \in T$ as well as $\epsilon(a)$ and $\epsilon(c)$. We observe that plausible hypotheses have a higher connectivity within this network, which we calculate by first finding a short path from a to c across T_i, and then calculating the average betweenness of the nodes appearing along this path. Finally, in order to weight the heuristics present in each of the previously described metrics, we fit a polynomial based on our set of proposed hypothesis. This results in the best-performing metric of PolyMulti.

7.3.3 Training Corpora

In order to understand the effect of different dataset features on an HG system, we identify corpora that differ in terms size and document length. These data sets, outlined in Table 7.1, include the PMC set of abstracts, PMC full-text, and five iterative halves of Medline. We download each data source as XML from PMC, and apply a series of preprocessing steps, described below. We note that while PMC contains 4 million full-text articles, a substantial number either do not supply an abstract, or are not available as XML. While other groups have found success parsing

PDF documents [248], we note that future journals contributing to PMC must supply XML, and that parsing PDFs introduces a level of complexity that extends beyond the scope of this work.

We apply AutoPhrase [193], porter stemming, and then stopword removal to clean our text. Our stopword list comes from ARROWSMITH's list.[1] As a result, we can identify meaningful n-grams within our text that make our results more interpretable and robust.

Because we cannot experimentally increase the size of our data sets, we instead take iterative halves of MEDLINE until it falls below one million abstracts. We do so with random sampling without replacement, and we note that the smaller samples are contained within the larger corpora. This sampling fills two requirements: firstly it ensures that each is representative of the entire MEDLINE data set, secondly it preserves our ability to perform validation using the cut-year of 2014. This allows us to identify connections that first occur in 2015 or later, which we will use to evaluate our network's performance.

We observe in our test corpora that MEDLINE contains a significant number of single-sentence abstracts, typically just a title, that represent old documents that have not been entirely added. For instance, the document with PMID 711285 consists of the single word "hypertension." Additionally, MEDLINE contains a number of non-English documents, such as PMID 21014169, which is in Spanish. PMC, in contrast, contains a smaller set of more recent documents, which consist of fewer short or non-English abstracts.

[1] `http://arrowsmith.psych.uic.edu/arrowsmith_uic/data/stopwords_pubmed`

Corpus	Total Words	Unique Words	Corpus Size	Median Words per Document
PMC Abstracts	109,987,863	673,389	1,086,704	102
PMC Full-Text	1,860,907,606	6,548,236	1,086,704	1594
MEDLINE	1,852,059,044	2,410,130	24,284,910	71
1/2 MEDLINE	923,679,660	1,505,672	12,142,455	71
1/4 MEDLINE	460,384,928	920,734	6,071,227	71
1/8 MEDLINE	229,452,214	565,270	3,035,613	71
1/16 MEDLINE	114,385,607	349,174	1,517,806	71

Table 7.1: The above table displays the corpus size for each experimental corpus we evaluated. Note, each corpus has been filtered to only include documents available in XML and published before 2014. Additionally, the above numbers represent each corpus after our initial text-cleaning process.

7.3.4 System Training and Query Process

After selecting our corpora, we run the *entire* MOLIERE network construction process, described in detail in [225], to create our knowledge network. This process begins with phrase mining and FASTTEXT [113], a word embedding tool that allows us to numerically represent each n-gram in a dense, continuous, real-valued vector space. For this chapter, we chose an embedding dimensionality of 100. These n-gram embeddings allow us to project each document into the space as a centroid of its components. From there, we create an approximate nearest-neighbors network for abstract centroids and n-grams (separately) using FLANN [158]. We join these layers with cross-cutting edges through TF-IDF. Lastly, we introduce data from the Unified Medical Language System (UMLS) [144]. This human-curated network represents ground-truth entities and connections that improve our network performance. Then all link weights are renormalized. This entire process is automated by the source code available on-line[2].

We note that for validation purposes, we only include data published before

[2] `http://github.com/jsybran/moliere`

2015. This means not only that we filter each corpora by publication date, but we also use the 2014 archival release of the UMLS metathesaurus. Additionally, by including the UMLS release to each corpus, we ensure that we are able to identify the needed entities for the later validation process.

To generate a hypothesis using our system, one supplies queries in terms of target words a and c (when performing a 1-to-1 query). From there, the system identifies each in its internal knowledge network, and finds the shortest-path between the two. We then extend shortest path to include the *cloud* of nearby documents by first finding the set of p closest papers for each node along the shortest-path, and then taking the union of each set. We then use PLDA+ [148] to identify k LDA topics within the extracted cloud, which we interpret as our hypothesis result. For our tests here, we select $p = 5000$ and $k = 20$.

7.3.5 Validation

We evaluate each instance of MOLIERE using the technique established in [226]. This process begins with a cut-year, which we chose to be 2014. From there, we extract all SEMMEDDB predicates that were first published in 2015 or later [120], and create a set of noise predicates through random sampling that have never been published. This provides a set of positive (published) and negative (noise) hypotheses that our networks have not seen.

To evaluate the performance of MOLIERE, we generate both positive and negative hypotheses in order to evaluate the resulting topic models of each using a number of metrics. Each metric captures a different relationship between a, c, and the resulting topic model. These often include distances using the trained vector space, which makes the underling FASTTEXT results incredibly important. One of the met-

rics, PolyMultiple, is a polynomial combination of the others, with coefficients obtained through black-box optimization. For the purposes of our tests here, we refit this metric for each system provided a set of 1 million training iterations.

We then rank published and noise connections together with respect to each metric in order to create ROC plots. We choose ROC curves as they have a direct relationship to ranking and because our validation method includes an equal number of positive and negative samples. The area under each curve indicates an instance's ability to distinguish published connections from noise. For a fair comparison between systems, we select a validation set of 2,000, equal parts published and noise, and use the same predicates on every system. In addition to the qualitative result, we also measure memory, storage, and run-time requirements for each system. All of our runtime measurements are run on 24 core machines with 126 Gb of memory, connected to a ZFS parallel file system.

There is a potential issue applying our validation scheme to full-text papers. We get our predicates from SemMedDB, a data source that only extracts information from abstracts, and our validation makes the assumption that the published and noise sets are both unknown to the system under examination. This implies that it could be possible for validation predicates to appear in full-text data that we do not intend, and there does not exist a reliable source of full-text predicates. This stated, we note that authors typically attempt to highlight their key findings in their abstracts, and for a predicate to appear in our published validation set, its first occurrence must date after 2014. We find it unlikely that these new findings occur in any significant manner within the details of full-text papers, and by using SemMedDB as a standard, we are able to make better comparisons.

We additionally generate hypotheses regarding a recent highly-cited finding on every system in order to quantitatively evaluate each in a real-world use case. The

paper "Mitochondrial Dynamics Controls T Cell Fate through Metabolic Programming" (cited 131 times at time of writing) found in 2016 that the protein OPA1 is required for effector T-cells and not for memory T-cells. We run two queries on our systems to relate OPA1 to immune effector cells and OPA1 to memory cells.

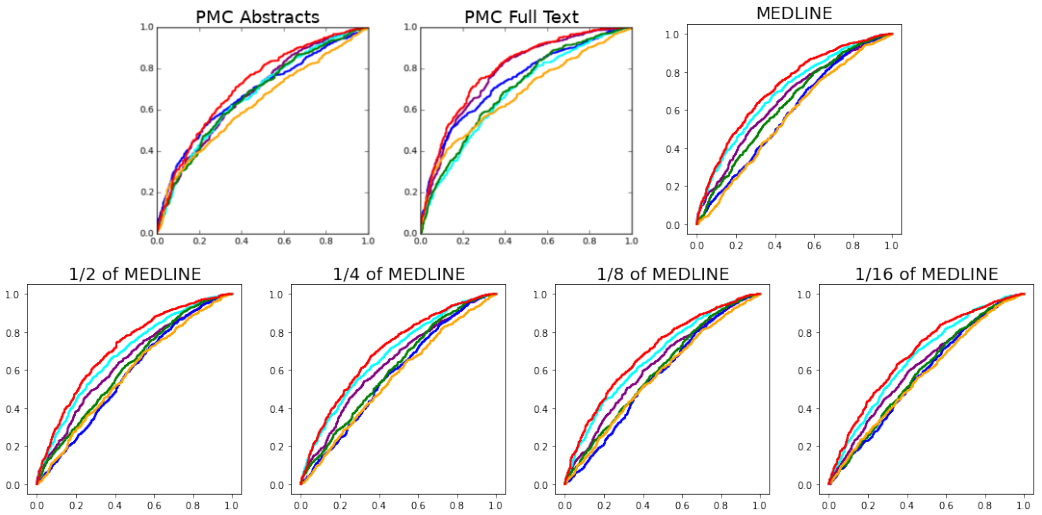
7.4 Results

After training instances of MOLIERE on each corpus and performing our validation task on each, we plot ROC curves for each across a number of metrics. We summarize these results in Figure 7.2 and discuss specific comparisons in the following sections.

7.4.1 PMC Abstracts vs Full-Text

We see from Table 7.1 that the median PMC full-text contains almost 16× as many words as the median PMC abstract. For this reason, we expect that the resulting embedding space is of higher quality — there is simply more training data. We observe this when comparing the L_2 metric because this metric only evaluates hypothesis quality by taking the distance between a and c, rather than incorporating topic model information. The full-text L_2 area is 0.777 while the abstract L_2 result is 0.678. This improvement is seen across many metrics, especially POLYMULTIPLE, the trained polynomial combination of other metrics. This is unsurprising because most metrics rely on the embedding space.

Looking practically we observe that constructing our full-text network takes 7× the runtime, and twice as much storage. Running each query takes 45× longer (1h 15m for full-text vs. 1m 40s for abstracts), and substantially more memory (1.4Gb vs. 0.41Gb). This is primarily due to the runtime of PLDA+, as it must read whole



	L_2	CentrL2	TopicPerWord	TopicCorr	TopicWalkBtwn	PolyMulti
PMC Ab.	0.678	0.681	0.671	0.670	0.629	0.718
PMC F.T.	0.777	0.738	0.680	0.696	0.674	0.795
Medline	0.651	0.584	0.691	0.628	0.565	0.718
1/2 Medline	0.643	0.578	0.684	0.615	0.580	0.717
1/4 Medline	0.634	0.576	0.677	0.603	0.556	0.700
1/8 Medline	0.621	0.566	0.666	0.593	0.570	0.691
1/16 Medline	0.612	0.572	0.658	0.585	0.569	0.684

Figure 7.2: Above are the ROC curves for each experiment, accompanied by the AUC for key metrics, as described in [226]. We evaluate a set of 2,000 predicates across each network to calculate each curve. Note that the L_2 metric, which relies entirely on simple vector embeddings, is the best indication of embedding quality, while the PolyMulti metric combines others for peak performance.

documents multiple times in order to fit a topic model. Other differences in the query process come from network topology differences that result from the drastic change in document length. Because we use TF-IDF to make cross-cutting connections between documents and keywords, we see that each document node has a substantially higher degree.

These network differences also account for qualitative differences in result quality between the PMC abstract and full-text systems. The full-text system contains many more keywords that occur in practically every paper, such as gene, mouse, and cell. While these words are certainly present in abstracts as well, their prevalence in methods and experimental sections *biases them heavily in full text*. Yet, we find their

removal would substantially detract from our ability to interpret topics in the case of abstracts.

Looking at the best topics for the query between OPA1 and effector T-Cells, we see that the best topic for the abstract system has the leading words "MGM1," "mitochondrial," "cell" and "GTPase" while the full-text topic has the leading words "cell," "mitochondrion," "mitochondrial," and "protein." While both seem to capture the same content on a broad level, the abstract topic is much more focused on a single entity, MGM1 (which is a mitochondrial GTPase related to OPA1). Still, neither network properly ranks effector T-Cells above memory T-Cells in relation to OPA1.

Overall, we see that abstracts give better qualitative and interpretable results using less time and memory, but full-text delivers a better vector space, and in turn allows for better evaluation via our metrics. We anticipate that a hybrid approach that constructs the system with an embedding space trained on full-text but only abstract text for the purposes of running queries would be optimal.

7.4.2 Medline Scaling Study

Observing the results in Figure 7.2, we see the effect of adding additional papers of similar quality to our HG system. Starting with the 1/16 sample of MEDLINE and working up to the whole data set, we see a consistent improvement across all results. In a similar manner to the above, our metric increase seems to come primarily from an increase in embedding space quality. We find that increasing our corpus size across the MEDLINE experiments only has a marginal increase in L_2 performance, ranging from an ROC area of 0.604 in the 1/16 sample to 0.638 in the 1/2 sample. Surprisingly, increasing the number of abstracts from the 1/2 sample to the entirety of MEDLINE has practically no effect on the resulting ROC curves. We believe this is a

side effect pertaining to the prevalence of very short MEDLINE articles, as mentioned above.

Additionally, there is a discrepancy between our results here and those found in our previous work [226] wherein we achieved an ROC score of 0.833 on the same corpus. In that case we created our network using an embedding space of dimensionality 500, as opposed to here where we use 100. In that case, or L_2 metric was 0.783, which indicates that the higher dimensionality results in a significantly better embedding space. In this study, we chose the smaller dimensionality to match the (typically) smaller corpus size. Although further study is needed, we anticipate that given a higher vector dimensionality for these studies, we would see a greater difference between the MEDLINE subsets as the higher dimensionality also implies it would be hard to train on smaller data sets.

In terms of performance, there is a bit of a difference in runtime between the scaled networks. The 1/16 system is able to run queries in about a minute and a half, using about 0.6 Gb of memory. Meanwhile the 1/2 system requires about 3 minutes and 15 seconds, and 3 Gb of memory. We note that the difference in runtime and memory usage primarily relates to the size of our overall network file (21 Gb for the 1/2 and 3.2 Gb for the 1/16 sample). Our query algorithms rely on our parallel file system to help subset, load, and process this network in parallel, which helps keep our runtime down. It is also worth noting that that the runtime of PLDA+ as well as our evaluation metrics is unchanged by the growing size of our network. Each query still results in a similar number sized abstract cloud, which takes just as long to produce a topic model for.

Qualitatively, we also see a slight increase in result specificity as the corpus size increases. This is not surprising as the 1/16 sample likely excludes many important papers that would help to explain important connection. In the case of OPA1 and

effector T-Cells, as previously discussed, we see the 1/16 best topic contains key words such as "mitochondri," "express," "regul," "activ," while the best topic for the 1/2 system produces a best topic of "protein," "mitochondri," "import," "mitochondria." Although the 1/2 sample is distinctly less informative from the PMC results above, its focus is much narrower than that of the 1/16 sample.

7.4.3 Cross-Comparison of Hypotheses using PMC and Medline

We observe in Table 7.1 that all of MEDLINE contains approximately the same number of words as the PMC full-text dataset. Additionally, we observe that the 1/16 sample of MEDLINE contains approximately the same number of documents as our PMC datasets. Furthermore we note that the PMC abstract set is a subset of MEDLINE. This allows us to compare the two sources, and in doing so understand the effect document length, count, and quality have on our results.

We see that the PMC abstract set has approximately the same ROC area as the 1/2 MEDLINE set, yet has a similar number of words as the 1/16 MEDLINE set. This shows us that a higher number of words per document is a big contributor to HG success. Additionally, this increase in performance only increases the average runtime by ten seconds — the PMC abstract system performs queries at about the same rate as the smallest MEDLINE sample due to their similar network size.

We hypothesize that a future study wherein we only include MEDLINE articles of sufficient length (at least two sentences) would be our best performer, or at least compete with the full-text results. From our scaling tests we see that additional papers certainly improve results, and our cross comparison shows that a lower median document length is detrimental to HG results. For these reasons a pruned set of

MEDLINE articles would have a larger median document length, but also contain an order of magnitude more documents than the PMC set alone. A benefit of this strategy would likely be a substantially smaller runtime when compared to the full-text data set, as it would likely resemble the full MEDLINE set in this aspect.

Looking quantitatively, comparing the OPA1—T-cell examples above, we note that sufficiently many abstracts provides better topics overall when compared to the full-text network. This is because we see many frequent terms in full-text, such as "fig," (meaning figure) that convey no additional content for our purposes. We are reluctant, though, to expand our stopword list to include words like "fig," or "ref" as their meanings in different contexts can be important, such as a fig-tree, or refer to the gene "ALYREF." While additional development into entity extraction efforts could address these concerns directly, we note that the difference in author intent between abstract and full-text documents will still imply a difference in the sort of topics we will uncover.

7.5 Tradeoffs

Interpreting our results from above, we can see a number of clear tradeoffs for HG when it comes to corpus quality. The key issue is runtime vs. result quality, but other challenges such as data availability can also lead to tradeoffs beyond those captured in ROC curves.

We observe that longer documents require more processing — they often have figures, extraneous text, and formulas, additionally, they often are distributed via PDF or were scanned through OCR. These concerns do not even address legal challenges associated with collecting substantial collections of published papers. As a result, uncovering their content for the purpose of HG is a non-trivial technical chal-

lenge that is likely to introduce noise. We circumvented these issues by restricting our discussion to only papers available via XML, and we still faced many of these noise-related challenges.

Yet, our results indicate longer documents produce higher quality systems, assuming one can handle the drastically increased runtime. This runtime limit becomes infeasible for many when considering large batch queries. For instance, to find candidate genes related to a specific disease, we must run over 40,000 query pairs (one for each gene), and while these tasks are independent, we run into logistical challenges reserving the computational resources necessary to run 40,000 queries if each takes over an hour.

A more subtle tradeoff comes from our choice of topic models over other discovery methods. Watson for Drug Discovery [209] uses many preprocessing techniques to make the runtime of individual queries much faster, which allow it to support full-text documents. A potential downside to this is that users may want to run queries for entities that have not been identified. While Watson is capable of handling these cases for many of its sub-systems, there are a number of query types that require preprocessed input. MOLIERE's preprocessing involves identifying n-grams, which can pose similar problems, but then our approach compensates by facilitating queries between any pair of nodes, even papers and UMLS terms. But, as described in our results, the method introduces document length as a significant factor in our algorithmic complexity.

7.6 Lessons Learned and Open Problems

Effect of Hyperparameters. Our network construction and validation method have a few hyperparameters that further study could help us inform their values. In

the network construction process, these include the dimensionality of our embedding space, the number of nearest-neighbors, and the weighting between layers of our network. Following the example of Mikolov et al. [153, 113], we have previously used an embedding space of 500, but in this work we reduce that to 100. We do so to see consistent performance across a number of corpus sizes, but we see reduced performance in our full-MEDLINE experiment (when compared to our previous results in [226]). Therefore, we note that larger corpus sizes ought to correlate with higher dimensionality vector spaces. However, further study is necessary to understand the effect of our other network construction parameters.

In the query process, the two main hyperparameters are cloud size and number of topics. We select a cloud of 5,000 documents per node along our shortest-path so that we have a sufficiently large sub-corpus even on paths with few hops. Clearly there is a balance between quantity and quality — the more documents we select, more of the documents will be unrelated to our query. With regard to the number of topics, we find that fewer topics lead to more interpretable results, while more topics tend to aid in our automatic validation. Yet, we have only experimented with topic values between 10 and 100 (20 for shown results).

Preprocessing. Automatic summarization and preprocessed topic models could each improve the runtime of full-text systems. Automatic summarization should improve our signal-to-noise ratio and reduce the number of edges incident to each paper. We could also generate a topic model for the paragraphs within each paper. This may allow us to assemble a topic model for our abstract cloud by combining topics that are similar across multiple full-text documents. This would be much faster than processing these documents directly for each query.

Domain Relevance. While we explore the effect of corpus size and document length in this work, we focus only on biomedical texts. We hypothesize that we

could improve MOLIERE results through the inclusion of additional documents from other fields such as physics, or more general sources such as Wikipedia. These other sources would provide additional examples of technical writing to inform our embedding space, but may also implicitly diminish the effect non-medical terminology has in our network. Because our cross-cutting edges between documents and keywords are weighted by TF-IDF, when we include documents from other fields, we expect that many methodology-related keywords that are shared between disciplines (such as "study," and "experiment") will decrease in relevance as they are shared in practically every paper. This method may allow us to bias against phrases that many may consider stopwords, without explicit creating a stopword list, which as we mention above poses its own challenges.

7.7 Deployment Challenges

Parsing full-text papers is a major challenge for HG systems. We found that only a subset of 1.7 million out of the 4.5 million documents in PubMed Central (PMC) was in XML format. Even then, some of the documents did not have an abstract (sometimes not the XML tag and sometimes not even the string "abstract" in the file). Our strategy was chosen to remove poor quality XML parsings. We first parsed the XML of papers according to the listed specifications, but only successfully parsed 0.6 million documents. By looking at the patterns that were missed (e.g. tags nested in others like: [article meta] $\rightarrow$ [title group] $\rightarrow$ [article title]). We made our patterns less constraining to allow for more documents to be passed into the dataset. We ended up with 1.3 million documents with full-text and abstracts being parsed.

When looking at some of the documents that failed, we found that they could have simply not tagged or failed XML parsers (e.g., with chains of emails or unrec-

ognized symbols). A similar parsing task for abstracts was much simpler. As long as a paper had the abstract tagged, it was easy to add it to our dataset. However, we added abstracts if and only if the full-text was properly parsed.

After stemming we were still left with the task of reducing the number of unique tokens during parsing. There was about 10 times the number of unique words within full-text documents than there were within the abstracts only. Even if we were comparing the larger set of abstracts to MEDLINE, there were many fewer unique words (2.4M in all of MEDLINE vs. 6.5M in PMC full-text). The number of unique words increased dramatically with every new addition of a particular document.

The data storage requirements were highest, not at the end, but in the middle of the experiment when running AutoPhrase. We started with 129 GB of textual data in XML format for 1.7 million documents from PMC (Downloaded in November 2017). After text parsing and cleaning, we were left with 24 GB for full text and only 1.4 GB for abstracts. Initially, we relied on TOPMINE for our phrase mining [69] but had to switch after encountering memory and runtime problems. Although we were using computers with 500 GB of RAM, it was still not enough. We had to switch to a computer with 2 TB of RAM just to avoid crashing. When finally getting TOPMINE to run properly, it was still too slow. Since our cluster only allows for jobs to run for 72 hours at a time, it was simply not able to finish running on the full-text dataset.

In order to mine phrases on full-texts we switched to AutoPhrase [193]. It was not only much faster, but also did not have the strict memory requirements that TOPMINE had. The information that was saved for the model was large (about 100 GB for the full text), but it was irrelevant since it was already using more than that in RAM. The process of topic modeling was simply memory intensive and unavoidable. Although we created the phrase model for the abstract dataset on a 24 core computer, it was still 4× faster than the full-text phrase modeling on a 64 core computer. This

amounted to an almost linear trend assuming processing scaled linearly. The ratio of runtime accounted for the number of cores, but not any other computer architecture specifics, ($4 * 64/24 \approx 10.7$) was almost equivalent to the difference in the data size (24Gb/1.4Gb ≈ 17).

7.8 Conclusion

In this chapter, we systematically study the effects different types and sizes of data have on knowledge network based hypothesis generation systems. The experimental work is performed using MOLIERE [225] which extends traditional network-based HG systems using topic modeling and various hypothesis ranking techniques. The computational evaluation is demonstrated using seven different corpora in order to answer four key questions.

What effect does corpus size and document length have on results? To answer this we compare the performance using ROC curves derived from our system when trained on PMC and MEDLINE data sets. We find that while increased corpus size does increase performance, document length is a better indication. Our results show that selecting a corpus with a greater median document length of 30 words can have the same effect as selecting a corpus that is eight times larger. However, we must emphasize that much longer documents typically result in less interpretable (or too general) topic models of the hypotheses.

How sensitive is a general-purpose HG system to hyperparameter value or input quality? MOLIERE is a general-purpose HG system that accepts queries for any terms covered in its input literature. Because of its scale, we anticipate that proper parameter tuning would require an analysis of more term pairs than is feasible. This challenge is not present in more specialized systems, such as those targeting

specific types of connections (gene-disease) or further specialized systems designed to explore a specific gene. Still, we can explore our system's performance across a number of datasets given a fixed parameter setting in order to understand the hyperparamter's stability. Our experiments show that a system trained on PMC abstracts outperforms one trained on a similar set of abstracts from MEDLINE. Looking at simple metrics, such as median document length and the length distributions, we observe a significant quality difference. For this reason we conclude that quality is more important than quantity for HG.

How many papers does a HG system need? We found that a set of at least 1-million papers is sufficient to achieve reasonable results, but more papers seem to improve result quality provided the underlying models are complex enough to capture the additional features. These additional papers improve the performance of our embedding space, which underpins much of the MOLIERE query process, but this effect wanes if the dimensionality is too low.

Are abstracts enough? We show that the tradeoff between quality and runtime is drastic when evaluating MOLIERE queries on full-text documents. We compare our system trained on PMC abstracts against our system trained on the full-text versions of the same papers. The longer documents cause longer topic modeling runtimes and result in a ROC area increase of 0.077 and a runtime increase from 100 seconds to 75 minutes. While this tradeoff may be acceptable for some, we note that many batch query applications may not be able to afford this marginal improvement.

Chapter 8

AGATHA: Automatic Graph-mining And Transformer based Hypothesis generation Approach

Abstract

Medical research is risky and expensive. Drug discovery, as an example, requires that researchers efficiently winnow thousands of potential targets to a small candidate set for more thorough evaluation. However, research groups spend significant time and money to perform the experiments necessary to determine this candidate set long before seeing intermediate results. Hypothesis generation systems address this challenge by mining the wealth of publicly available scientific information to predict plausible research directions. We present AGATHA, a deep-learning hypothesis generation system that can introduce data-driven insights earlier in the

discovery process. Through a learned ranking criteria, this system quickly prioritizes plausible term-pairs among entity sets, allowing us to recommend new research directions. We massively validate our system with a temporal holdout wherein we predict connections first introduced after 2015 using data published beforehand. We additionally explore biomedical sub-domains, and demonstrate AGATHA's predictive capacity across the twenty most popular relationship types. This system achieves best-in-class performance on an established benchmark, and demonstrates high recommendation scores across subdomains. Additionally, AGATHA is fast and scalable, is constructed using distributed data preparation and training, and can analyze thousands of hypotheses per-minute. **Reproducibility:** All code, experimental data, and pre-trained models are available online: sybrandt.com/2020/agatha .

8.1 Introduction

As the rate of global scientific output continues to climb [233], an increasing portion of the biomedical discovery process is becoming a "big data" problem. For instance, the US National Library of Medicine's (NLM) database of biomedical abstracts, *MEDLINE*, has steadily increased the number of papers added per-year, and has added significantly over 800,000 papers every year since 2015 [1]. This wealth of scientific knowledge comes with the overhead cost payed by practitioners who often struggle to keep up with the state-of-the-art. In 2018 alone, there was an average of 103 papers added to MEDLINE per *hour*, or one new paper every *35 seconds*.

Buried within the large and growing MEDLINE database are many undiscovered implicit connections — those relationships that are implicitly discoverable, yet have not been identified by the research community. One connection of this type was first proposed and subsequently discovered by Swanson and Smalheiser in the mid-to-

late 1980's [221]. Their landmark finding, using only the co-occurrences of keywords across MEDLINE titles, was to establish a connection between fish oil and Raynaud's Syndrome [219]. At that time, it was known that fish oil modified various bodily properties, such as blood viscosity, which were key factors pertaining to Raynaud's syndrome. However, while each explicit relationship was known, the *implicit* relationship was not discovered before Swanson's ARROWSMITH hypothesis generation system identified the connection algorithmically.

Modern advances in machine learning, specifically in the realms of text and graph mining, enable contemporary hypothesis generation systems to identify fruitful new research directions while taking far more than title co-occurrence rates into account. Modern systems predict missing links on domain specific graphs, such as BioGraph on gene-disease network [142] or MeTeOR on the term-co-occurrence graph [249]. Other systems focus on identifying relevant key terms, similar to Swanson's work, but using modern techniques. For instance, Jha et al. study the evolution of word embedding spaces over time to learn contemporary trends relevant to particular queries [109]. Further work by Jha et al. continues to study the joint evolution of corpora and ontologies within biomedical research [110]. Another approach is to produce visualizations for interpretation by domain scientists [208], such as the closed-source Watson for Drug Discovery [51]. Moliere, our prior hypothesis generation system [225], produces data for scientific interpretation in the form of LDA topic models [29]. Additional work produced heuristically-backed ranking criteria to help automate the analysis process [226].

While prior hypothesis generation systems have been valuable in real-world explorations, such as Swanson's fish-old and Raynaud's syndom finding [219], Watson's discovery of ALS treatments [51], or Moliere's discovery of DDX3 inhibition as a treatment for HIV-associated neurodegenerative disease [18], there remains sig-

nificant drawbacks to the state of the art. Most systems require significant human oversight to produce useful results [209, 51, 225], or are only tested on very small evaluation sets [109, 110, 82, 188]. Systems still using the "ABC" model of discovery [109, 110, 121], posed by Swanson in 1986 [219], face many known limitations such as reduced scalabiltiy and a bias towards incremental discoveries [201].

To overcome these limitations, we present a new hypothesis generation system that scales to the entirety of biomedical literature, and is backed by efficient deep-learning techniques to enable thousands of queries a minute, enabling new types of queries. This system constructs a new semantic multi-layered graph, and places its millions of nodes into a shared embedding. From there, we use a *transformer encoder* architecture [236] to learn a ranking criteria between regions of our semantic graph and the plausibility of new research connections. Because our graph spans all of MEDLINE, we are able to generate hypotheses from a large range of biomedical subdomains. *Other than our prior work [225], we are unaware of any system that is capable of the same breadth of cross-domain discovery that is also open source, or even just publicly available for comparison.* Because we efficiently pre-process our graph and its embeddings, we can perform hundreds of queries per-second on GPU, which enables new many-to-many recommendation queries that were not previously feasible. Because we replace our heuristically determined ranking criteria from our prior work [226] with a learned ranking criteria, we achieve significantly improved performance, as demonstrated by an increase in benchmark performance using the same training and validation from and ROC AUC of 0.718 [226] to 0.901.

To expand the interpretabiliy of our proposed system's output, we additionally provide an updated version of our prior topic-modeling approach to operate efficiently on the much larger sentence-focused semantic graph. While these topic-model queries do take longer (a few minutes, compared to a fraction of a second), they can provide

descriptive results for biomedical scientists. Because our topic-model query process selects relevant sentences, as opposed to the prior model that selected whole abstracts, we observe that our resulting topic models are more descriptive regarding the query at hand. We envision that these descriptive queries can supplement the automated discovery enabled by the deep-learning model.

Our contribution:

(1) We introduce a novel approach to construct large semantic graphs that use the granularity of sentences to represent documents. These graphs are constructed using a pipeline of state of the art NLP techniques that have been customized for understanding scientific text, including SciBERT [22] and ScispaCy [163].

(2) We deploy our deep-learning transformer-based model that trained to predict likely connections between term-pairs at scale. This is done by embedding our proposed semantic graph to encode all sentences, entities, n-grams, lemmas, UMLS terms, MeSH terms, chemical identifiers, and SemRep predicates [17] in a common space using the PyTorch-BigGraph embedding [139].

(3) We validate our system using the massive validation techniques presented in [226], and also demonstrate the ability of AGATHA to generalize across biomedical subdomains. For instance, in the scope of "Gene - Cell Function" relationships, our system has a top-10 average precision of 0.83, and a mean-reciprocal-rank of 0.61.

This system is open-source, easily installed, and all prepared data and trained models are available to perform hypothesis queries at sybrandt.com/2020/agatha .

8.2 Background

Our proposed system, AGATHA, is a novel deep-learning approach to hypothesis generation, enabled by recent advances in text- and graph-mining techniques.

This section expounds our conceptual influences and incorporated technologies.

Hypothesis Generation Systems. Swanson posited that *undiscovered public knowledge*, those facts that are implicitly available but not explicitly known, would accelerate scientific discovery if an automated system were capable of returning them [221]. His work established what is now known as the "A-B-C" model of literature-based discovery [4]. This formulation follow that a hypothesis generation system, given two terms A and C, should uncover some likely B-terms that explain the quality of a potential $A - B - C$ connection. This technique fueled Swanson's own system, ARROWSMITH [219], and still forms the backbone of some contemporary successors [121].

The ABC model has significant limitations. Firstly, many real-world scientific hypotheses cannot be so easily distilled into a single set of "first-order" interactions. Instead, many connections may be better described as longer $A - B - C - \ldots$ connection paths or more complicated structures. Secondly, any system that returns only a set of B-terms will be limited to small-scale searches unless it also provides an automatic way to quantify connection plausibility. Otherwise, biomedical researchers will spend a significant amount of valuable time studying query results, rather than performing necessary experiments.

Our former approach to address these challenges is posed by the Moliere system [225], and its accompanying plausibility ranking criteria [226]. This system expands on the $A - B - C$ model by describing a range of connection patterns, as represented by an LDA topic model [29], when receiving an A, C query. To do so, the Moliere system first finds a short-path of interactions bridging the $A - C$ connection from within a large semantic graph. This structure includes nodes that correspond to different entity types that are both textual and biomedical, such as abstracts, predicate statements, genes, diseases, proteins, etc. Edges between entities indicate

similarity. For instance, an edge may exist between an abstract and all genes discussed within it, or between two proteins that are discussed in similar contexts. Using the short-path discovered within the semantic network between A and C, the Moliere system also reports an LDA topic model [29]. This model summarizes popular areas of conversation pertaining to abstracts identified near to the returned path. As a result, the user can view various fuzzy clusters of entities and the importance of interesting concepts across documents.

To reduce the burden of topic-model analysis on biomedical researchers, the Moliere system is augmented by a range of techniques that automatically quantify the plausibility of the query based on its resulting topic models. Our measures, such as the embedding-based similarity between keywords and topics, as well as network analytic measures based on the topic-nearest-neighbors network, were heuristically backed, and were combined into a meta-measure to best understand potential hypotheses. Using this technique, we both validated the overall performance of the Moliere system, and used it to identify a new gene-treatment target for HIV-associated neurodegenerative disease through the inhibition of DDX3X [10].

The work presented here departs from heuristically-backed prior necessities. Our AGATHA semantic graph is built using sentences, not abstracts, as the primary node type, and uses ScispaCy [163] in order to produce higher-quality graph content pertaining to key terms and entities. The resulting graph, with a different overall schema, is then embedded by the PyTorch-BigGraph heterogeneous graph embedding technique [139]. Now, instead of performing expensive short-path queries, generating topic models, or applying heuristically-backed measures, we formulate knowledge discovery as a deep-learning problem, and learn to rank fruitful new research directions directly from the data using a combination of graph embeddings and a transformer-encoder network [236].

Related and Incorporated Technologies. In order to prepare the wealth of biomedical information stored in Medline abstracts for deep-learning queries, we leverage a range of software tools and machine learning techniques.

SemRep [17] is a utility that extracts *predicate statements* in the form of "subject-verb-object" from the entirety of Medline. This utility further classifies its predicate components into the set coded keywords provided by the Unified Medical Language System (UMLS), and a small set of coded verb-types. These UMLS terms provide a way to unify synonyms and acronyms from across medicine. Additionally, all content extracted by SemRep is provided in the Semantic Medical Database (SemMedDB) for direct use.

Dask [178] is a library for writing distributed data-processing pipelines in Python. As a result of using this library, AGATHA data preparation is efficiently completed across a large cluster of workers. Furthermore, this library allows us to implement modular functional components of the processing pipeline, enabling easier extensions.

ScispaCy [163], a version of the popular spaCy text processing library provided by AllenNLP, is designed to properly handle scientific text. Using a deep-learning approach for its part-of-speech tagging, dependency parsing, and entity recognition, this tool achieves state-of-the-art performance on a range of scientific and biomedical linguistic benchmarks. Additionally, this software is optimized sufficiently to operate on each sentence of MEDLINE, which numbers over 188 million as of 2020.

SciBert [22] is a version of the BERT transformer model for scientific language. This model learns representations for each word part in a given sentence. Word parts are derived from the WordPiece algorithm [252] when trained on a sample of scientific full-text papers. The resulting embeddings for each word part are determined by its relationship to all other word parts. As a result, the output word-part embeddings are highly content-dependent, and homographs, words with the same spelling but

different meanings, receive significantly different representations. We use this model to learn embeddings per-sentence that capture scientific content. These embeddings inform the sentence-nearest-neighbors network component of our semantic graph.

FAISS [112], the open-source similarity-search utility, is capable of computing an approximate nearest-neighbors network for huge point clouds. We adapt this tool for use within Dask in order to compute the nearest neighbors edges between the SciBert embeddings for all sentences in MEDLINE. This technique scales to various graph sizes by its modular component set, and we choose PQ-quantization and k-means bucketing to reduce the dimensionality of our sentences, and reduce the search space per-query.

PyTorch-BigGraph (PTBG) [139] is an open-source, large-scale, distributed graph-embedding technique aimed at heterogeneous information networks [198]. These graphs consist of nodes of various types, connected by typed edges. We define each node and relationship type contained in our semantic graph as input to this embedding technique. PTBG distributes edges such that all machines compute on disjoint node-sets. We choose to encode edges through the dot product of transformed embeddings, which we explain in more detail in Section 8.3. Using the Hogwild! [176] optimization technique, distributed workers are unrestricted by locking while performing this optimization, which has an added regularization effect.

The Transformer [236] model is built with multi-headed attention. Conceptually, this mechanism works by learning weighted averages per-element of the input sequence, over the entire input sequence. Specifically, this includes three projections of each element's embedding, represented as packed matrices: Q, K, and V. Each projection functions differently, with Q acting as a "query" that is compared against "keys" K and "values" V. The specific mechanism is defined as follows, with d_k

representing the dimensionality of each Q and K embedding:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\intercal}{\sqrt{d_k}}\right)V \tag{8.1}$$

The "multi-headed" aspect of the transformer indicates that the attention mechanism is applied multiple times per-layer, and recombined to form a joint representation. If $W^{(x)}$ indicates a matrix of learned weights, then this operation is defined as:

$$\begin{aligned} \text{MultiHead}(X) &= [h_1; \ldots; h_k]W^{(4)} \\ \text{where } h_i &= \text{Attention}\left(XW_i^{(1)}, XW_i^{(2)}, XW_i^{(3)}\right) \end{aligned} \tag{8.2}$$

While the transformer model was initial proposed for sequence-to-sequence modeling, and includes both an "encoder" and "decoder" stack of attention layers, we note that the self-attention layer fundamentally performs a *set* operation. In fact, text models such as BERT [63] require the addition of a positional encoding to each input token to ensure that positional information is not erased by self-attention. By using only the encoder half of the transformer model, and by omitting any positional mask or encoding, we apply the self-attention mechanism to understand input sets while reducing the effect of the arbitrary ordering imposed by a sequence model. One encoder layer is defined as:

$$\begin{aligned} \mathcal{E}(X) &= \text{LayerNorm}(FF(\alpha) + \alpha) \\ \text{where } FF(\alpha) &= \max\left(0, \alpha W^{(5)}\right) W^{(6)} \\ \text{and } \alpha &= \text{LayerNorm}(\text{MultiHead}(X) + X) \end{aligned} \tag{8.3}$$

By composing multiple $\mathcal{E}$ encoders, we create the full encoder stack. While some order-sensitive operations do exist within the encoder stack, such as the opera-

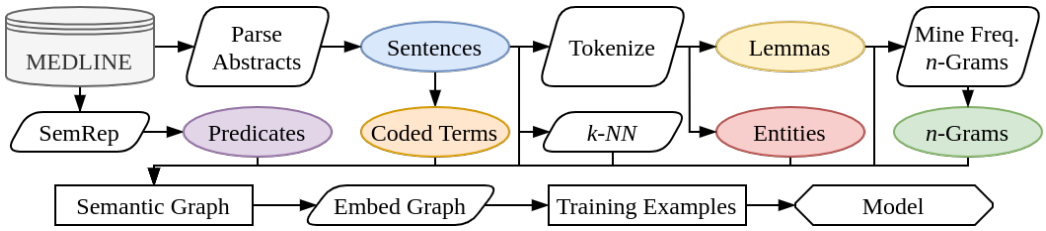


Figure 8.1: System Diagram of the AGATHA process.

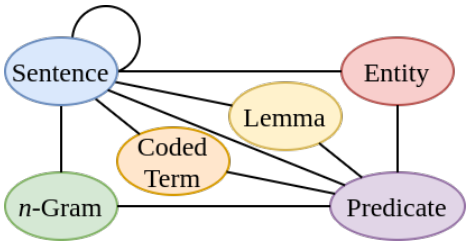


Figure 8.2: AGATHA multi-layered graph schema.

tion that merges multiple attention heads into a joint feed-forward layer, we observe these artifacts are overcome during the training process by randomizing the order of input elements.

8.3 Data Preparation

In order to convert the MEDLINE raw text into a form that enables the deep-learning of scientific hypotheses, we propose a significant pre-processing pipeline. In short, we begin by downloading relevant data from Medline and SemMedDB, then extract all relevant information per-sentence to formulate a semantic graph, following the schema in Figure 8.2. From there, we embed the entire network using PTBG [139]. We then formulate training-set SemRep predicates [17] as sets of node embeddings for our proposed *transformer encoder* neural-network model, depicted in Figure 8.3. This pipeline overall is depicted in Figure 8.1, and is expound below.

Text Pre-Processing. We begin with raw MEDLINE XML files [1]. Each can be independently processed for a majority of the AGATHA distributed data processing operations. We attempt to extract the paper id (PMID), version, title, abstract text, date of first occurrence, keywords, and publication language. Next, we filter out non-English documents. About 15% of MEDLINE documents were not originally published in English, and of those translated a vast majority contain only a title. *In order to validate our system, we additionally discard any document that is dated after January 1st, 2015.*

We split the text of each abstract into sentences. For each sentence, we identify parts-of-speech, dependency tags, and named entities using ScispaCy [163]. We ran into performance challenges when parsing longer texts. Therefore, we first perform sentence splitting through a rules-based system provided by the Natural Language Toolkit (NLTK) [150]. The result of this process is a record per-sentence, including the title, that contains all metadata associated with the original abstract, as well as all algorithmically identified annotations.

Using the lemma information of each sentence, we perform n-gram mining in order to identify common phrases that may not have been picked up by entity detection. In our prior work [225], we leveraged a then-contemporary phrase mining tool ToPMine [69] to extract similar phrases. However, in our new distributed paradigm, we found it to be more efficient and produce more useful results to devise a simple rules-based system built on top of ScispaCy lemma information. First, we provide a set of part-of-speech tags we mark as "interesting" from the perspective of n-gram mining. These are: nouns, verbs, adjectives, proper nouns, adverbs, interjections, and "other." We additionally supply a short stopword list, and assert that stop words

[1] At the time of writing, the bulk release at the end of 2019 contained 1,014 files, containing nearly 30-million documents

are uninteresting. Then, for each sentence, we produce the set of n-grams of length two-to-four that both start and end with an interesting lemma. We record any n-gram that achieves an overall support of at least 100. However, we find it necessary to introduce an approximation factor, that an n-gram must have a minimum support of five within a datafile for those occurrences to count.

Semantic Graph Construction. After splitting sentences, while simultaneously identifying lemmas, entities and n-grams, we can begin constructing the semantic graph. The semantic graph, as a whole, contains all textual and biomedical entities within MEDLINE, and follows the schema depicted in Figure 8.2. We begin this process by creating edges between similar sentences. The simplest edge we add is that between two adjacent sentences from the same abstract. For instance, sentence i in abstract A will produce edges to A_{i-1} and A_{i+1}, with the paper title serving as A_0.

To capture edges between similar sentences in different abstracts, we compute an approximate-nearest-neighbors network on the set of sentence embeddings. We derive these embeddings from the average of the final hidden layer of the SciBert [2] NLP model for scientific text [22]. This 768-dimensional embedding captures context-sensitive content regarding each word in each sentence.

However, we have over 155-million sentences in the 2015 validation instance of AGATHA, which makes performing a nearest-neighbors search per-sentence (typically $\mathcal{O}(n^2 d)$) computationally difficult. Therefore, we leverage FAISS to perform dimensionality reduction, as well as approximate-nearest neighbors, in a distributed setting. First, we collect a one-percent sample of all embeddings on a single machine, wherein we perform product quantization (PQ) [106]. This technique learns

[2] We specifically use the pre-trained "scibert-scivocab-uncased" model, which was trained on over 1.14-million full-text papers.

an efficient bit representation of each embedding. We select parameters for PQ such that each dimension of the input embedding receives a unique bit in the output code. We use 96-quantizers, and each considers a disjoint an 8-dimensional chunk of the 768-dimensional SciBert embeddings. Each quantizer then learns to map its input real-valued chunk into output 8-bit codes, such that similar input chunks receive output codes with low hamming distance. This technique reduces size of SciBert embeddings by a factor of 32.

Still using the 1% sample on one machine, FAISS performs k-Means over PQ codes in order to partition the reduced space into self-similar buckets. By storing the centroid of each bucket, we can later select a relevant sub-space pertaining to each input query, dramatically reducing the search space. We select 2048 partitions to divide the space, and when performing a query, each input embedding is compared to all embeddings residing in the 16 most-similar buckets.

Once the PQ quantizers and k-means buckets are determined, the initial parameters are distributed to each machine in the cluster. Every sentence can be added to the FAISS nearest-neighbors index structure in parallel, and then the reduced codes and buckets can be merged in-memory on one machine. We again distributed the nearest-neighbors index, now containing all 155-million sentence codes, to each machine in the cluster. In parallel, these machines can identify relevant buckets per-point, and record their 25 approximate nearest-neighbors. If we have m machines, each with p cores, and search $q = 16$ of the $b = 2048$ buckets-per-query, we reduce complexity for identifying all nearest-neighbors from $\mathcal{O}(dn^2)$ to $\mathcal{O}\left({}^{qdn^2}/_{32bpm}\right)$.

We additionally add simpler sentence-occurrence edges for lemmas, n-grams and entities. In each case, we produce an edge between s and x provided that lemma, entity, n-gram, or metadata-keyword x occurs in sentence s. The last node type is SemRep predicates [17]. Each has associated metadata, such as the sentence in

which it occurred, its raw text, and its relevant UMLS coded terms. For each unique subject-verb-object triple, we create a node in the semantic graph. We then create edges from that node to each relevant sentence, keyword, lemma, entity, and n-gram. Our overall graph consists of *184-million nodes and 12.3-billion edges*. We store this representation of our network first in a series of Tab-Separated-Value (TSV) files, and provide export utilities to compile this information into both MongoDB and Sqlite3 databases.

Graph Embedding. We utilize the PyTorch-BigGraph (PTBG) embedding utility to perform a distributed embedding of the entire network [139]. This requires an expensive index operation from our distributed edge-list structure to partitioned nodes and bucketed edges expected by PTBG. For this, we provide an efficient and optimized C++ utility to perform this index in parallel. PTBG learns typed embeddings, and we define node types corresponding to each presented in our semantic graph schema. Each undirected edge in our graph schema is also coded as two directional edges of types $x \rightarrow y$ and $y \rightarrow x$.

While there are many different configurations possible for PTBG, we explored a subset to settle on a balance between computational efficiency and embedding quality. We partition the semantic graph into 100 roughly even sized partitions per-type by hashing each node's id string. This partitioning results in 10,000 edge buckets, one for each ordered pair of partitions, which easily fit in one machine's memory. We explore two different embedding dimensionalities: 256 and 512. When computing both embeddings, we specify for edges to be encoded via the dot-product of nodes, and for relationship types to be encoded using a learned translation per-type. We generate a total of 100 negative samples per edge, 50 chosen from nodes within each batch, and 50 chosen from nodes within the corresponding partitions. Dot products between embeddings are learned using the supplied softmax loss, with the first dimension of

every embedding acting as a bias unit.

Formally, if an edge ij exists between nodes i and j of types t_i and t_j respectively, then we learn an embedding function $e(\cdot)$ that is used to create a score for ij by projecting each node into $\mathbb{R}^N$ where N is a predetermined embedding dimensionality. In our experiments we consider $N = 256$ and 512. This embedding function uses the typed translation vector $T^{(t_i t_j)} \in \mathbb{R}^N$ that is shared for all edges of the same type as ij. This score is defined as:

$$s(ij) = e(i)_1 + e(j)_1 + T_1^{(t_i t_j)} + \sum_{k=2}^{N} e(i)_k \left(e(j)_k + T_k^{(t_i t_j)} \right) \tag{8.4}$$

Then, for each edge ij, we generate 100 negative samples in the form $x_n^{(ij)} y_n^{(ij)}$. Their scores are compared to that of the positive sample using the following loss function, which indicates the component of overall loss corresponding to edge ij:

$$\text{GraphLoss}_{ij} = -s(ij) + \log \sum_{n=0}^{100} \exp\left(s\left(x_n^{(ij)} y_n^{(ij)} \right) \right) \tag{8.5}$$

Deployment technical note: When optimizing our semantic graph embedding, we find that maximal performance is achieved using a compute cluster of twenty twenty-four-core machines. Within the 72h time restriction of the Palmetto super computing cluster, we have enough time to see every edge in the graph 10 times, in the case of the 256-dim embedding, and 5 times in the case of the 512-dim embedding. Once complete, we are ready to begin training the AGATHA deep learning hypothesis generation model.

Training Data. In order to learn what makes a plausible biomedical connection, we collect the set of published connections present in our pre-2015 training set. For this, we turn to the Semantic Medical Database (SemMedDB), which contains over

AGATHA-512 Parameters	1,313,280
AGATHA-512 Embeddings	10,115,707,904
AGATHA-256 Parameters	328, 960
AGATHA-256 Embeddings	5,057,853,952

Table 8.1: Model Size. Because embeddings are trained separately from the hypothesis prediction model, both numbers are listed. Embedding numbers correspond to the amount of floating-point values associated with predicate and coded-term embeddings needed to use the model.

19-million pre-2015 SemRep [17] predicates parsed from all of MEDLINE. A SemRep predicate is a published subject-verb-object triple that is identified algorithmically. In lieu of a true data set of attempted hypotheses, we can train our model on these published connections. However, this approach comes with some drawbacks. Firstly, SemRep predicates are defined on the set of UMLS terms, which will restrict our system to only those entities that have been coded. This limitation is acceptable given size size of UMLS, and presence of existing benchmarks defined among UMLS terms [226]. Secondly, the predicate set is noisy, and may contain entries that are incorrect or obsolete, as well as algorithmically introduced inaccuracies. However, we find at scale that these sources of noise do not overwhelm the useful signal present within SemMedDB.

8.4 Ranking Plausible Connections

We train a model to rank published SemRep [17] predicates above noisy negative samples using the transformer architecture [236]. To do so we first formulate a predicate with subject α and object β for input into the model. Those predicates that are collected from SemRep are "positive samples" (PS). The function $\Gamma(\cdot)$ indicates the set of neighbor predicates that include a term as either a subject or object. We represent the $\alpha\beta$ predicate as a set with elements that include both terms, as

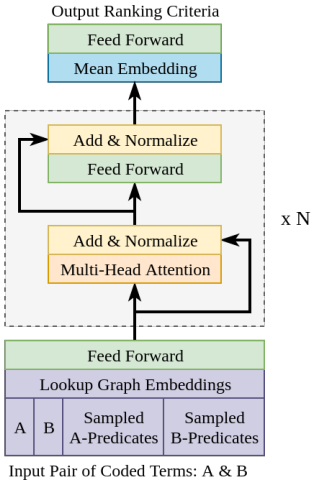


Figure 8.3: AGATHA ranking transformer encoder. Given entity-pair and neighborhoods, looks up graph embeddings and produce ranking criteria.

well as a fixed-size sample with-replacement of size $s = 15$ of each node's non-shared predicates:

$$\begin{gathered} \mathrm{PS}_{\alpha\beta} = \left\{\alpha, \beta, \gamma_1^{(\alpha)}, \ldots, \gamma_s^{(\alpha)}, \gamma_1^{(\beta)}, \ldots, \gamma_s^{(\beta)}\right\} \\ \text{where } \gamma_i^{(\alpha)} \sim \{\Gamma(\alpha) - \Gamma(\beta)\}, \text{ and } \gamma_i^{(\beta)} \sim \{\Gamma(\beta) - \Gamma(\alpha)\} \end{gathered} \tag{8.6}$$

Negative Samples We cannot learn to rank positive training examples in isolation. Instead, we first generate negative samples to accompany each published predicate. This include two types of samples: scrambles and swaps. Both are necessary, as we find during training that the easier-to-distinguish scrambles aid early convergence, while the swaps require the model to understand the biomedical concepts encoded by the semantic graph embedding.

The negative scramble (NScr) selects two arbitrary terms x and y, as well as $2s$ arbitrary predicates from the set of training data. While we enforce that x and y do not share a predicate, we do not enforce any relationship between the sampled predicates and these terms. Therefore these samples are easy to distinguish from

positive examples. If T denote all positive-set terms, and P denotes all predicates, then a negative scramble associated with positive sample $\alpha\beta$ is notated as:

$$\begin{aligned} \text{NScr}_{\alpha\beta} &= \{x, y, \gamma_1, \ldots, \gamma_{2s}\} \\ \text{where } x, y &\sim T, \text{ and } \gamma_i \sim P \\ \text{s.t. } \Gamma(x) &\cap \Gamma(y) = \emptyset \end{aligned} \tag{8.7}$$

The negative swap (NSwp) selects two arbitrary terms, but samples the associated predicates in the same manner as the positive sample. Therefore, the observed term-predicate relationship will be the same for each half of this negative sample (α and $\gamma_i^{(\alpha)}$). This sample requires the model to learn that some $\alpha\beta$ pairs should not go together, and this will require an understanding of the relationships between biomedical terms. A negative scramble associated with $\alpha\beta$ is notated as:

$$\begin{aligned} \text{NSwp}_{\alpha\beta} &= \left\{x, y, \gamma_1^{(x)}, \ldots, \gamma_s^{(x)}, \gamma_1^{(y)}, \ldots, \gamma_s^{(y)}\right\} \\ \text{where } \gamma_i^{(x)} &\sim \{\Gamma(x) - \Gamma(y)\}, \text{, and } \gamma_i^{(y)} \sim \{\Gamma(y) - \Gamma(x)\} \\ &\text{s.t. } \Gamma(x) \cap \Gamma(y) = \emptyset \end{aligned} \tag{8.8}$$

Objective. We minimize the margin ranking loss between each positive sample and all associated negative samples. The contribution of positive sample $\alpha\beta$ to the overall loss is defined as:

$$\begin{aligned} \mathcal{L}(\alpha, \beta) = \sum_{i=0}^{n} L\left(\text{PS}_{\alpha\beta}, \text{Nscr}_{\alpha\beta}^{(i)}\right) + \sum_{j=0}^{n'} L\left(\text{PS}_{\alpha\beta}, \text{Nswp}_{\alpha\beta}^{(j)}\right) \\ \text{where } L(p, n) = \max\left(0, m - \mathcal{H}(p) + \mathcal{H}(n)\right) \end{aligned} \tag{8.9}$$

Here $n = 10$ denotes the number of negative scrambles, $n' = 30$ is the number of negative swaps, $m = 0.1$ is the desired margin between positive and negative

samples, and $\mathcal{H}$ is the learned function that produces a ranking criteria given two terms and a sample of predicates.

Model. Using the transformer encoder summarized in Section 8.2, as well as the semantic graph embedding, we construct our model. If $e(x)$ represents the semantic graph embedding of x, FF represents a feed-forward layer, and $\mathcal{E}$ represents an encoder layer, then our model $\mathcal{H}$ is defined as:

$$\begin{aligned} \mathcal{H}(X) &= \text{sigmoid}(\mathcal{M}W) \\ \mathcal{M} &= \frac{1}{|X|}\sum_{x_i \in X} E_N(FF(e(x_i))) \\ E_{i+1}(x) &= \mathcal{E}(E_i(x)), \text{ and } E_0(x) = x \end{aligned} \tag{8.10}$$

Here $N = 4$ represents the number of encoder layers, and W indicates the learned weights associated with the final ranking projection. By averaging the transformer output over the input sequence X, then projecting that result down to a single real value with W, and applying the sigmoid function, we produce an output per-predicate in the unit interval. This function is depicted in Figure 8.4. *Deployment technical note*: We minimize the ranking loss over all published predicates using the LAMB optimizer [257]. This allows us to efficiently train using very large batch sizes, which is necessary as we leverage 10 NVIDIA V100 GPUs to effectively process 600 positive samples (and therefore 2,400 total samples) per batch. In terms of hyperparameters, we select a learning rate of $\eta = 0.01$ with a linear warm up of 1,000 batches, a margin of $m = 0.1$, a neighborhood sub-sampling rate of $s = 15$, and we perform cross-validation on a 1% random holdout to provide early stopping and to select the best model with respect to validation loss. Due to the large size of training data, one epoch consists of only 10% of the overall training data. This process is made easier through the helpful Pytorch-Lightning library [73].

8.5 Validation

Testing hypothesis generation, in contrast to information retrieval, is difficult as ultimately these systems are intended to discover information that is unknown to even those designing them [253]. A thorough evaluation would require a costly process wherein scientists explore automatically posed hypotheses. Instead, we perform a historical validation, in a manner similar to that performed in [226, 226]. This method enables large-scale evaluation of many biomedical subdomains almost instantly, but cannot truly tell us how our system will perform in a laboratory environment. To attempt to incorporate some expert oversight into the validation process, we supplement automatic validation with qualitative analysis from a domains scientist, which follows the older validation process found in [225]. After ensuring the system is capable of uncovering recent connections from historical data, we begin the much longer process of testing contemporary ideas system in real-world scenarios, as was pursued by Moliere [10], Watson [18], and ARROWSMITH [219].

Comparison with Heuristic-Based Ranking. We begin by comparing the performance numbers obtained through our proposed learned ranking criteria with other ranking methods posed in [226]. Specifically, the Moliere system presents experimental numbers for various training-data scenarios for the same 2015 temporal holdout as used in this work [226]. For a direct comparison, we use our proposed method to rank the same set of positive and negative validation examples.

Comparison by Subdomain Recommendation. As mentioned in [96], the Moliere validation set has limitations. We improve this set by expanding both the quantity and diversity of considered term pairs, as well as evaluating AGATHA through the use of all-pairs recommendation queries within popular biomedical subdomains. As a result, this comparison effectively uses subdomain-specific negative examples, which

makes for a harder benchmark than that presented in the Moliere work. It is worth nothing that these all-pairs searches are made possible by the very efficient neural-network inference within AGATHA, and would not be as computationally efficient in the Moliere shortest-path and topic-modeling approach.

This analysis begins by extracting *semantic types* [5], which categorize each UMLS term per-predicate into one of 134 categories, including "Lipid," "Plant," or "Enzyme." From there, we can group $\alpha\beta$ predicate-term pairs by types t_α and t_β. We select the twenty predicate type pairs with the most popularity in the post-2015 dataset, and within each type we identify the top-100 predicates with the most rapid non-decreasing growth of popularity determined by the number of abstracts containing each term-pair per year. These predicates form the positive class of the validation set. We form the rest of the subdomain's validation set by recording all possible undiscovered pairs of type $t_\alpha t_\beta$ from among the UMLS terms in the top-100 predicates. We then rank the resulting set by the learned ranking criteria, and evaluate these results using a range of metrics.

Metrics. The first metrics we consider are typical for determining a classification threshold: the area under the receiver-operating-characteristic curve (AUC ROC) and the area under the precision-recall curve (AUC PR). An AUC ROC of 0.5 indicates that the ranking criteria randomly orders the published term pairs relative to the undiscovered, while an area of 1 indicates that all published term pairs occur first in the ranking. Similarly, the PR curve determines how varying levels of precision could be achieved while still retrieving a certain amount of published connections. An AUC PR closer to 1 again indicates that all published term pairs occur at the start of the list, and a PR closer to 0 indicates that they occur towards the end. However unpublished predicates occurring to the start of the list typically have a larger negative impact in the score. We additionally provide recommendation system metrics, such as

top-k precision (P.@k), average precision (AP.@k), and overall reciprocal rank (RR). Top-k precision is simply the number of published term-pairs appearing in the first k elements of the ranked list, divided by k. Top-k *average* precision weights each published result by its location in the front of the ranked list. The reciprocal rank is the inverse of the rank of the first published term pair.

The above recommender system metrics all consider the single many-to-many query within a biomedical subdomain. However, this same result can be interpreted as a set of one-to-many recommendation queries. Doing so enables us to compute the mean average precision(MAP.@k), and mean-reciprocal rank(MRR.@k) for the set of recommendations. A high MRR within a domain indicates that the researcher should expect to see a useful result within the first few results. A high MAP indicates that out of the top k results, more of them are useful. These metrics, taken together, should influence biomedical researchers when exploring the results of a one-to-many query.

While all of the above metrics quantify the performance of the AGATHA learned ranking criteria, it is also important to provide interpretable results to biomedical researchers. For this reason we also perform Moliere-style shortest-path and topic-model queries on the AGATHA semantic graph. Our newly optimized framework enables us to perform one query on a single thread in a few minutes, which can allow a medical researcher to explore a subset of recommendations output by the deep learning model. We visualize these topic model outputs and provide them to domain scientists in order to give feedback on their predictive power for recent findings. One such finding, the relationship between HIV-associated Neurodegenerative Disease, which was found by Moliere in 2019 [10], is among this qualitative study.

8.6 Results

We compare the performance of AGATHA against Moliere, as presented in [226]. In that work, multiple trained instances of Moliere rank a benchmark set of positive and negative potential connections using a range of criteria defined in [226]. These Moliere instances each use different datasets published prior to 2015 in order to perform hypothesis queries, of which we focus on two: all of MEDLINE (Moliere: MEDLINE), and all of PubMedCentral (Moliere: Full Text). The former instance represents a system trained on the same raw data as the AGATHA system presented here, while the latter represents a system trained on all publicly available full-text papers provided by the NLM released in the same date range.

The prior work establishes that the Moliere topic-modeling approach is improved by the additional information made available by full-text papers, but at a overwhelming 45x runtime penalty. These quality results are reproduced in Table 8.3, and we include additional results for the AGATHA system when evaluated on only abstracts, and exactly the same set of predicates. We observe that the AGATHA system, when trained with 512-dimensional graph embeddings, improves upon Moliere: Medline by 25% and Moliere: Full Text by 13%. Importantly, this increase in quality comes at an overwhelming *decrease* in runtime, with the wall time per-query dropping from minutes to milliseconds, due to the introduction of the deep-learning approach.

Figures 8.4(a) and 8.4(b) depict ROC and PR curves for the top-performing AGATHA model.

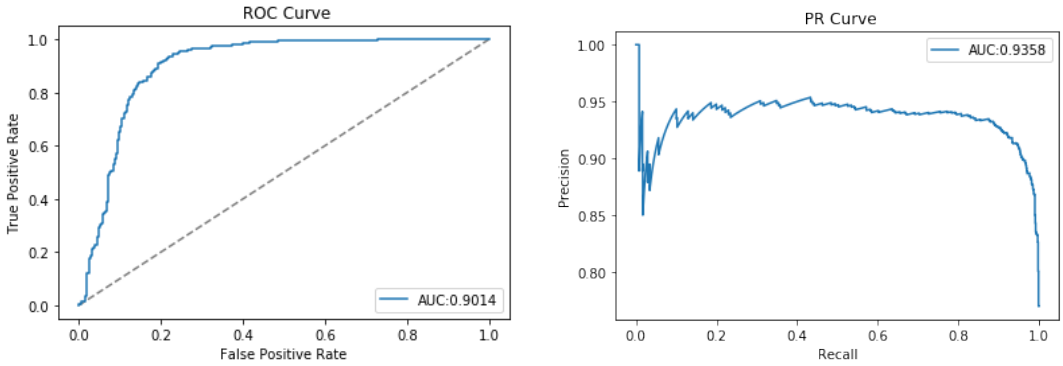
To further study the performance differences between the Moliere and AGATHA systems, we quantify the correlations between their different ranking criteria. We depict these correlations with Moliere: Medline and Moliere: Full Text in Figures 8.5(a)

Type	Training %	Training Rank	AUC PR	AUC ROC	RR	P.@ 10	P.@ 100	AP.@ 10	AP.@ 100	MAP.@ 10	MAP.@ 100	MRR.@ 10	MRR.@ 100
gngm, celf	0.29	74	0.44	0.62	1.00	0.50	0.47	0.83	0.54	0.57	0.56	0.61	0.61
gngm, neop	0.35	61	0.34	0.65	0.50	0.50	0.43	0.54	0.47	0.46	0.41	0.52	0.52
aapp, neop	0.35	62	0.20	0.62	0.33	0.30	0.26	0.34	0.28	0.40	0.35	0.46	0.47
gngm, cell	0.43	42	0.19	0.72	0.25	0.30	0.17	0.27	0.21	0.35	0.32	0.38	0.38
aapp, cell	0.67	26	0.19	0.63	0.50	0.20	0.17	0.36	0.21	0.34	0.33	0.37	0.38
aapp, gngm	1.05	13	0.17	0.68	1.00	0.50	0.22	0.61	0.31	0.36	0.27	0.39	0.40
cell, aapp	1.59	4	0.17	0.67	0.14	0.10	0.19	0.14	0.18	0.35	0.32	0.40	0.41
gngm, gngm	0.50	37	0.17	0.66	1.00	0.40	0.20	0.77	0.37	0.31	0.26	0.33	0.34
orch, gngm	0.41	49	0.16	0.69	0.05	0.00	0.22	0.00	0.21	0.33	0.27	0.34	0.36
aapp, dsyn	0.67	25	0.15	0.69	0.33	0.20	0.24	0.28	0.22	0.34	0.27	0.37	0.38
gngm, dsyn	0.21	97	0.15	0.71	0.50	0.40	0.24	0.59	0.32	0.29	0.23	0.30	0.31
bpoc, aapp	1.06	12	0.14	0.67	1.00	0.20	0.18	0.70	0.28	0.35	0.30	0.38	0.39
bacs, gngm	0.29	73	0.12	0.67	0.33	0.10	0.14	0.33	0.19	0.26	0.24	0.29	0.30
bacs, aapp	0.73	22	0.12	0.68	0.17	0.30	0.14	0.28	0.18	0.27	0.24	0.30	0.32
dsyn, humn	7.02	1	0.11	0.64	0.05	0.00	0.10	0.00	0.10	0.27	0.25	0.29	0.31
aapp, aapp	1.57	5	0.11	0.69	1.00	0.20	0.11	0.67	0.25	0.28	0.24	0.32	0.33
gngm, aapp	0.40	52	0.11	0.71	1.00	0.20	0.11	0.61	0.22	0.23	0.21	0.25	0.26
phsu, dsyn	0.76	20	0.10	0.61	0.04	0.00	0.14	0.00	0.11	0.27	0.20	0.30	0.31
dsyn, dsyn	1.35	6	0.09	0.62	0.17	0.20	0.12	0.19	0.15	0.22	0.18	0.25	0.27
topp, dsyn	1.19	9	0.09	0.64	0.10	0.10	0.17	0.10	0.12	0.28	0.22	0.30	0.31

Table 8.2: AGATHA-512. Above are hypothesis prediction results on biomedical sub-domains. Indicated along with performance numbers are the percentage of training data (pre-2015 predates) as well as the training-data popularity rank out of 6396, with 1 being most popular. Metrics described in detail in Section 8.5.

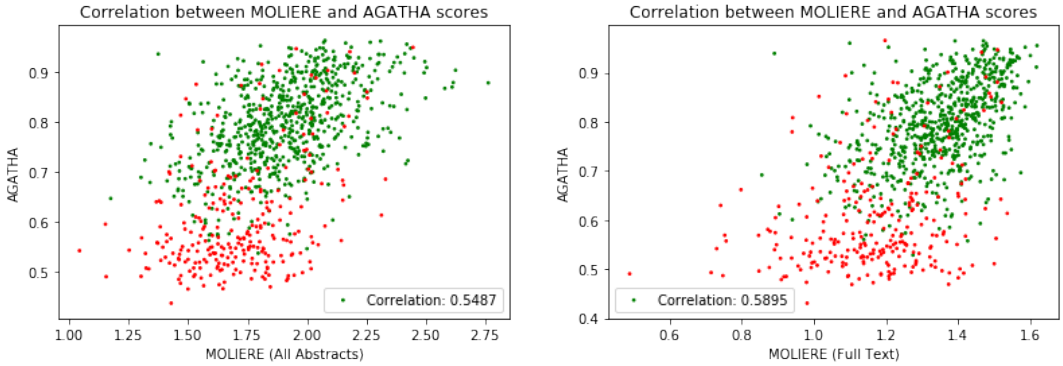
System Instance	ROC AUC	PR AUC
Moliere: Medline	0.718	0.820
Moliere: Full Text	0.795	0.778
AGATHA-256	0.826	0.895
AGATHA-512	**0.901**	**0.936**

Table 8.3: Benchmark comparison between Moliere and AGATHA on the same benchmark.



(a) Receiver-Operating Characteristic (b) Precision-Recall

Figure 8.4: Validation Benchmark 2015



(a) AGATHA vs. Moliere (100% Medline) (b) AGATHA vs. Moliere (PMC Full text)

Figure 8.5: Correlations between Moliere and AGATHA-512 scores on the 2015 benchmark. Green and red dots indicate positive and negative hypotheses.

and 8.5(b) respectively. Each point in these scatter plots indicate a hypothesis, which is colored green if it was published following 2015, and red if it was negatively sampled. The scale for both scatter plots is determined by the intervals spanned by each system's ranking criteria. We observe that there is very little correlation between these scores, and that the separation between positive and negative samples is clearly seen in the AGATHA ranking, and muddied in the Moliere rankings. As a result, we do not believe there would be a substantial benefit in creating an ensemble method to combine the deep-learning and classical ranking methods.

To extend the validation beyond the above results, provided that we can now generate thousands of hypothesis per-minute, we explore the capacity of our deep-learning ranking criteria to perform hypothesis recommendation within various many-to-many queries across different biomedical sub-domains. These results, displayed in Table 8.2, list the 20 predicate types with the most popularity following 2015. Due to space limitations, we present predicate types using NLM semantic type codes [5]. All numbers are reported from the AGATHA-512 model.

We observe that the (Gene)→(Cell Function)(gngm, celf) predicate type, is the

easiest predicate type for AGATHA-512 to recommend, even though connections of this type only account for 0.29% of the training data. Of the top-10 recommendations the highest ranked is a valid connection and half are valuable. When performing a one-to-many query within this type of connection, we observe 85% of all top-10 suggestions to be useful on average, and that a useful result occurs typically within the first two recommendations. We see similar performance in the (Gene)→(Neoplastic Process) (gngm, neop) and (Amino Acid, Peptide, or Protein)→(Neoplastic Process) (aapp, neop) sub-domains. Interestingly, there appears to be little correlation between the popularity of a predicate type in the training data and the quality of the resulting recommendations. This result enforces the idea of AGATHA as a *general-purpose* biomedical hypothesis generation system.

Of the 20-most-popular predicate subdomains considered, AGATHA-512 has the most difficulty with the (Therapeutic or Preventive Procedure)→(Disease or Syndrome)(topp, dsyn). In this subdomain, the the highest ranked positive predicate is ranked tenth, and only twelve of the top-100 suggestions are useful. Still, in a one-to-many query, we expect about one-in-ten recommended predicates to be useful, and for the top-3 predicates to contain a useful result. While the lower-performing subdomains are significantly harder for AGATHA-512 than the top few, we note that even a low-precision tool can be useful for aiding the biomedical discovery process. Furthermore, these difficult subdomains are still ranked significantly better than random chance, and even better than many of the classical ranking measures presented in [226]. Using this information, future work may wish to fine-tune the AGATHA method to a specific subdomain for improved performance.

8.7 Lessons Learned and Open Problems

Result Interpretability. While deep-learning models are notoriously hard for human decision makers to interpret, we find that biomedical researchers still need to understand how a result was produced in order to act on model predictions. However, we cannot leave the entire analysis up for human judgement, as this drastically reduces the benefits of "automatic" hypothesis generation. To walk the narrow edge between these conflicting objectives, we implement both an automatic ranking component, as well as a more interpretable topic-model query system. We find that these tools serve different functions during different times of the discovery process.

At first, a researcher may be considering a wide range of potential research directions, such as during the candidate selection phase of the drug-discovery process. This often requires assembling hundreds (or thousands) of target ingredients, compounds, genes, or deceases, and determining whether elements of this large set have a relationship to an item of interest. For instance, when we evaluated HIV-associated Neurodegenerative Disease, we explored over 40,000 potential human genes [10]. This component of the discovery process fits nicely into the deep-learning ranking and recommendation system proposed here, especially when the target set is so large that a manual literature review may prove costly.

Once a candidate set of targets has been winnowed from the large target set, the researcher will prioritize interpretability. However, the candidate set is typically orders of magnitude smaller than the target set. Therefore, we can afford to run more costly-yet-interpretable routines, even if these routines do not provide any form of "automatic" analysis. At this stage, we switch from our deep-learning ranking method to the topic-modeling approach similar to that presented in Moliere [225]. This process finds a path within our semantic network containing the textual infor-

mation necessary to describe a potential connection. We present that path along with the set of relevant sentences, as well as a visualization of the topic model built from those sentences. Researchers can explore the sets of entities that are frequently mentioned together in order to expand their mental models of each hypothesis's quality.

Datasets and Expandability. When discussing hypothesis generation systems with prospective adopters in the biomedical community, we often are asked to include specific datasets that has domain-relevance to an individual's research direction. For instance, the set of clinical trials, internal experimental findings, or a database of chemicals.

Currently, new network-based datasets can be introduced trivially. After processing Medline, the network lives as a collection of simple TSV files. This set can receive new datasets, provided that the new entity types are included in the following PyTorch-BigGraph configuration. We use this graph-addition technique to merge SemRep predicate data into the Medline dataset.

New textual datasets can similarly be introduced earlier in the process. Text records, once formulated into python dictionaries with a particular set of fields, may be added to the pipeline, participate in the tokenization and network-construction process, and will eventually be included in topic-model queries. however, this process requires a minor modification to the existing data pipeline code. We are working currently to make this import operation as simply as the network-addition process described above.

In contrast to the graph and text sources, it is not currently clear how to incorporate experimental data into the AGATHA system. This challenge arises from the many forms experimental data can take. In the case where an experiment can be reformulated as a network, such as converting the gene-expression matrix into a gene-to-gene network, these results can trivially be introduced as new edges. Other

experimental results, such as many clinical trials, include a thorough summary of that trial's findings. These may be introduced as a combination of textual and graph-based sources, including both the description text, as well as any links to known publications that reference the trial. Importantly, we do not find a "one size fits all" solution for experimental data, and more work should explore the costs and benefit associated with various datasets.

8.8 Related Work

Foster et al. [78] identify a series of common successful research strategies often used by scientists. In doing so they demonstrate that high-risk and innovative strategies are uncommon among the scientific community in general. It follows that the field of hypotheses generation obeys similar rules. Many systems have found success using algorithmic techniques that approximate these common research strategies by studying term co-occurrences [108, 102, 245], or predicting links with a graph of biomedical entities [171, 71]. While the Foster's model of research strategies has proven to be useful, the mechanisms involved in complex scientific discoveries remain unexplored.

Unsurprisingly, we find that hypothesis generation systems utilize algorithmic techniques in a range of complexity that is analogous to these human research strategies. The first hypothesis generation system, ARROWSMITH, presents the ABC model of automatic discovery [219]. This technique identifies a list of terms that are anticipated to help explain a connection between two terms of interest. This basic algorithm remains in some modern systems, such as [121]. However, ABC-based techniques have significant limitations [201], including their similarity metrics defined on heuristically determined term lists, as well as their reliance on manual validation pro-

cesses. As a result, ABC systems are know to be biased towards finding incremental discoveries [124].

A completely different strategy of performing LBD is proposed by Spangler et al. in [209]. To explore the p53 kinase, the authors use neighborhood graphs constructed from entity co-occurrence rates. The approach relies on domain experts and requires manual oversight to provide MEDLINE search queries, and to prune redundant terms, but produces promising results. In [52] the authors demonstrate that this technique can identify kinase NEK2 as an inhibitor of p53, and in [18] a similar scientist-in-the-loop technique identifies a number of RNA-binding proteins associated with ALS.

A significant step beyond ABC and human-assisted techniques is to incorporate a domain specific datasets. Bipartite graphs, such as the gene-disease [145] or the term-document [81] networks, are frequent choices. These systems usually aim to perform a number of graph traversals between node-pairs in order to rank the most viable options. However, the number of generated paths may be prohibitively large, which reduces ranking quality [82] To address this problem, Gopalakrishnan proposes two-stage filtering through a "single-class classifier" which is able to prune up to 90% hypotheses prior to the ranking scheme [81]

One recent approach is to use deep learning models to help extract viable biomedical hypotheses. Sang et al. [188] describe GrEDeL, a way to generate new hypotheses using knowledge graphs obtained from predicate triples in the form of "subject, verb, object". This approach finds all possible paths between a given drug and decease, provided those paths include a particular target entity. Then these paths are evaluated using a LSTM model that captures features related to drug-disease associations. While the GrEDeL system is successful at identifying some novel drug-disease relationships, this approach has some important trade-offs: (1) Their proposed

model is trained using SemRep graph traversals as a sequence, which the authors note is a highly noisy dataset. Furthermore, multiple redundant and similar paths exist within their dataset, which decrease the quality of their validation holdout set. The AGATHA system overcomes this limitation by leveraging node neighborhoods in place of paths. (2) Their knowledge graph is constructed exclusively from predicates mined from MEDLINE abstracts using SemRep. This process affects the model quality significantly and, being the only resource of knowledge, it requires careful manual filtering of false positive and isolated predicates. (3) The GrEDeL LSTM model is trained to only discover drug-disease associations, and does not generalize to other biomedical subdomains. (4) This approach embeds their predicate knowledge graph using the TransE method [38], which supposes that relationships can be modeled as direct linear transformations. When using the large number of relationship types present in SemRep, this assumption greatly reduces the useful variance in the resulting node embeddings.

8.9 Conclusions

This work presents AGATHA, a deep-learning biomedical hypothesis generation system, which can accelerate discovery by learning to detect useful new research ideas from existing literature. This technique enables domain scientists to keep pace with the accelerating rate of publications, and to efficiently extract implicit connections from the breadth of biomedical research. By constructing a large semantic network, embedding that network, and then training a transformer-encoder deep-learning model, we can learn a ranking criteria that prioritizes plausible connections. We validate this ranking technique by constructing an instance of the AGATHA system using only data published prior to January 1st 2015. This system then evaluates

both a benchmark of predicates established from prior work [226], and performs recommendation in twenty popular biomedical subdomains. The result is state-of-the art prediction quality on the 2015 benchmark, as well as strong performance across a range of subdomains. In the case where a simple recommendation is not sufficient, we also implement topic-model-based interpretability queries, that enable researchers to learn more about particular connections of interest, after the initial ranking has limited their field of consideration. The AGATHA system is open-source and written entirely in Python and PyTorch, which enable to be easily used or adapted anywhere. We release both the 2015 validation system, as well as an up-to-date 2019 system to accelerate the broader community of biomedical sciences.

Chapter 9

CBAG: Conditional Biomedical Abstract Generation

Abstract

Biomedical research papers use significantly different language and jargon when compared to typical English text, which reduces the utility of pre-trained NLP models in this domain. Meanwhile Medline, a database of biomedical abstracts, introduces nearly a million new documents per-year. Applications that could benefit from understanding this wealth of publicly available information, such as scientific writing assistants, chat-bots, or descriptive hypothesis generation systems, require new domain-centered approaches. A conditional language model, one that learns the probability of words given some a priori criteria, is a fundamental building block in many such applications. We propose a transformer-based conditional language model with a shallow encoder "condition" stack, and a deep "language model" stack of multi-headed attention blocks. The condition stack encodes metadata used to alter the output probability distribution of the language model stack. We sample this

distribution in order to generate biomedical abstracts given only a proposed title, an intended publication year, and a set of keywords. Using typical natural language generation metrics, we demonstrate that this proposed approach is more capable of producing non-trivial relevant entities within the abstract body than the 1.5B parameter GPT-2 language model. **Reproducability:** All code, data, pre-trained models, and experimental parameters are available online: sybrandt.com/2020/cbag

9.1 Introduction

The biomedical sciences are becoming more data driven due to the increased availability of experimental data and the democratization of machine learning algorithms. One subfield of biomedical data science, literature-based discovery [225], produces algorithms to automatically identify plausible research directions from the growing body of scientific literature [41]. While these systems have seen early successes aiding biomedical science [10, 18], these techniques often lack the interpretability necessary to persuade domain scientists to pursue algorithmically generated leads. While customizable visualizations aid significantly [208], many researchers would prefer a textual description to accompany generated hypotheses. Thinking much further into the future, if hypothesis generation systems are ever going to function as automated scientists in their own right, they will require the ability to generate textual arguments supporting their own ideas.

Today, modern deep-learning language generation models can produce text in a range of contexts. Building off of the transformer architecture [236], models like BERT [63] and GPT/GPT-2 [172, 173] have set a new standard in a range of natural language benchmarks [241]. Adaptations of these models, such as SciBert [22] and BioBert [135], have retrained the baseline models for domain-specific tasks in order

to advance the state of domain-specific benchmarks as well [101, 167]. However, these domain-specific models are not designed for natural language generation (NLG). Models like GPT are trained to generated text in the language of typical English writing, and we demonstrate below that these generations are ill-suited for the jargon-filled particular language used by biomedical scientists.

Compounding these challenges around generating biomedical text, much less work has focused on *conditional* generation, wherein the output language distribution is affected by a priori knowledge. Older text captioning systems [256] follow a similar approach using sequence models informed by image encodings. However, more control over generated text is necessary for applications like hypothesis generation systems, where semantic information detected by the system should be leveraged in an automatically produced argument. Modern systems trained outside the biomedical domain, such as Ctrl [119] allow for some conditions, but lack the flexibility needed to capture sets of semantic information. More generalizable methods, such as those produced by variational auto-encoders [105], can capture rich latent language semantics, but cannot straightforwardly encode domain-based information, such as a set of keywords one wishes to include in the output text.

A language model that can enable complex domain-specific applications, such as hypothesis generation, therefore requires a new approach. This technique should accept an arbitrary set of semantic criteria as a condition, should be aware of domain-specific entities and jargon, and should produce text that would be expected by biomedical scientists.

In this work we propose CBAG, a conditional biomedical abstract generation model that seeks to address the above requirements. This transformer model includes a shallow encoder stack to encode qualities of the condition, and an deep decoder stack to produce a high quality language model. We train this model us-

ing semi-supervised multi-task generative pre-training, wherein to minimize our proposed objective function, the model must predict successive tokens, parts of speech, dependency tags, as well as entity labels. We train this model using over 20-million biomedical records provided by the National Library of Medicine (NLM) through the Medline database. Each record consists of a title, abstract, publication year, and an optional set of author-provided keywords. Text processing and annotations are provided by a biomedical NLP model trained on the "BIONLP13CG" BioCreative training set [101]. This pre-trained domain-specific model allows the CBAG model to apply the knowledge gain from the relatively small human-annotated dataset to the larger set of unstructured text present in Medline.

We train the proposed model by sampling textual windows from within MEDLINE abstracts. The publication date, and any author-supplied Medical Subject Headings (MeSH terms, a set of biomedical keywords and phrases) form the condition. The sampled window serves as input to the decoder stack. Windows are split into subword units using the unigram subword-regularization algorithm [127]. Using masked-self attention, we train the model to predict each subword $i + 1$ using only the condition and tokens $1, \ldots, i$.

To the best of our knowledge, this work is the first attempt to design a biomedical abstract generator. Therefore, without a direct point of comparison, we leverage the 1.5-billion parameter "huge" version of GPT-2 to compare against. As this language model was trained on a range of online data sources, such as the BooksCorpus and English Wikipedia, it is a disadvantage in our domain-specific task. However, the authors find that this model is capable of a range of specific tasks across domains, such as language translation, question answering, and commonsense reasoning [173]. Furthermore, other work has even found that the GPT-2 language model can function as a general purpose knowledge base [169]. For these reasons, we can expect GPT-2

to be a relevant, albeit disadvantaged, point of comparison.

When generating an abstract during evaluation, we formulate a human written title, as well as relevant condition information where applicable, for model input. We then sample each model's subword probability distribution for each generated result until the new abstract is written. We evaluate computer-generated abstracts based on their ability to produce relevant n-grams that occur in the human-written abstract associated with the input title. We leverage a range of NLG metrics [194], such as Bleu, METEOR, ROUGE-L and CIDEr, including a version of CIDEr that omits input n-grams from consideration. Through all considered metrics we quantitatively demonstrate increased performance through the use of CBAG. Qualitatively, we present full-abstracts, as well as a handful of sentences for assorted generations, which show the ability of our proposed model to capture the overarching flow of scientific summaries. We additionally demonstrate the ability for condition keywords to influence model generations by producing a varied set of completions for the seed-phrase, "*In this study, we found...*"

Our contribution: We present CBAG, a transformer-based language model for conditional biomedical abstract generation. Trained using Medline records and informed by semi-supervised domain-specific annotations, this model captures biomedical jargon, entities, and pattern of scientific discussion. We compare generated abstracts against the 1.5B parameter GPT-2 language model, and demonstrate a superior ability to produce relevant n-grams across a range of NLG metrics.

All code, data, pre-trained models, preprocessing pipelines, and experimental parameters are available online[1]. We additionally supply a set of over 13,000 automatically generated abstracts for a wide range of test-set titles. Using the generalizable precondition approach presented here, we hope to enable future applications, such as

[1]sybrandt.com/2020/cbag

descriptive hypothesis generation. However, we are also cognisant of the potential for abuse surrounding high quality domain-specific language models. We discuss these concerns further in Section 9.7.

9.2 Background

While recent **language models** receive a newfound popularity in proportion to their surprising capacity across a range of tasks [173], their study predates modern machine learning techniques [23]. Formally, a language model is a probabilistic model that captures the conditional probability of each next element in a sequence given all prior elements. Specifically, this is described by the function:

$$\mathsf{Pr}(s) = \prod_{i=1}^{n} \mathsf{Pr}(s_i | s_1, \ldots, s_{i-1}) \tag{9.1}$$

Here, s is a sequence of n elements. The probability of observing sequence s is determined by the product of the conditional probabilities of observing each token s_i given all prior tokens. These models can generate new text by iteratively sampling new elements from the probability distribution $\mathsf{Pr}(s_{i+1} | s_1, \ldots, s_i)$.

The conditional language model introduces a new term c into the above equation. The condition can allow applications to alter the resulting sequence based on a priori knowledge [105]. Formally, the conditional language model is defined as:

$$\mathsf{Pr}(s|c) = \prod_{i=1}^{n} \mathsf{Pr}(s_i | s_1, \ldots, s_{i-1}, c) \tag{9.2}$$

Modern neural network language models [173, 119], model these probability distributions by minimizing the negative log-likelihood of these distributions over a

large training set of sequences:

$$\begin{aligned} &\mathcal{L}\left(\left(s^{(1)}, c^{(1)}\right), \ldots, \left(s^{(m)}, c^{(m)}\right)\right) \\ &= -\sum_{j=1}^{m}\sum_{i=1}^{n} \log \mathsf{Pr}_\theta\left(s_i^{(j)} | s_1^{(j)}, \ldots, s_{i-1}^{(j)}, c^{(j)}\right) \end{aligned} \tag{9.3}$$

Here, Pr_θ indicates the parameterized model that approximates the language model distribution. Modern systems often use the transformer architecture [173, 119, 240] for state-of-the-art quality estimating Pr_θ.

The transformer [236], a sequence-to-sequence model built through multi-headed attention layers, has been customized for a number of NLP tasks, as best demonstrated by BERT [63], GPT-2 [173], and a range of notable follow-ups [174, 216, 147]. Conceptually, the attention mechanism works by learning multiple weighted averages per-element of the input sequence. Specifically, this includes three projections of each element's embedding, represented as packed matrices: Q, K, and V. Each projection functions differently, with Q acting as a "query" that is compared against "keys" K and "values" V. The specific mechanism is defined as follows, with d_k representing the dimensionality of each Q and K embedding:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\intercal}{\sqrt{d_k}}\right)V \tag{9.4}$$

The "multi-headed" aspect of the transformer indicates that the self-attention mechanism is applied multiple times per-layer, per-element of the sequence. These multiple heads are then recombined through a feed-forward layer:

$$\begin{aligned} \text{MultiHead}(X, Y) &= [h_1; \ldots; h_k]W^{(4)} \\ \text{where } h_i &= \text{Attention}\left(XW_i^{(1)}, YW_i^{(2)}, YW_i^{(3)}\right) \end{aligned} \tag{9.5}$$

The transformer model presented by Vaswani et al. [236] use the attention attention mechanism in three different ways. Within the encoder stack, which processes the input sequence in their proposed sequence-to-sequence model, the K, Q, and V embeddings all come from the same sequence of tokens. This is referred to all "self attention." In the decoder stack, the part of the model that uses the encoder output to generate a new sequence, these embedding matrices are masked during the attention function such that the output embedding for position i can only depend on prior elements. This is called "masked self attention". Following this operation, each decoder embedding is attended with all of the encoder embeddings. Specifically, Q values are derived from the decoder, while K and V values depend on the encoder. We refer to this operation as "Encoder-Decoder Attention." Note that BERT [240] uses only the encoder self-attention layers, while GPT-2 [173] uses the decoder's masked self-attention layers. The work presented here uses all three.

The multi-head components are combined with a feed-forward operation, denoted FF, that projects the concatenated embedding into a larger dimensionality, applies the ReLU activation function, and then reduces back to the set embedding rank:

$$\mathrm{FF}(X) = \max(0, XW)W' \tag{9.6}$$

Then, combined with a learned layer-wise normalization, these components combine to form encoder and decoder blocks. Omitting the standard dropout between each operation, the encoder block is defined as:

$$\begin{aligned} \mathcal{E}(X) &= \mathrm{LayerNorm}(\mathrm{FF}(\alpha) + \alpha) \\ \alpha &= \mathrm{LayerNorm}(\mathrm{MultiHead}(X, X) + X) \end{aligned} \tag{9.7}$$

while the decoder block is defined as:

$$\begin{aligned} \mathcal{D}(X, Y) &= \text{LayerNorm}(\text{FF}(\alpha) + \alpha) \\ \alpha &= \text{LayerNorm}(\text{MultiHead}(\beta, Y) + \beta) \\ \beta &= \text{LayerNorm}(\text{MultiHead}(X, X) + X) \end{aligned} \tag{9.8}$$

Tokenization chunks an input sequence of characters into input for a transformer-based model. BERT leverages the WordPiece algorithm [252], which first learns to identify a predetermined number of character-groups from a sample of text in order to minimize the expected number of character groups per sentence. The fact that practitioners can tune the number of tokens in a WordPiece tokenization of critical for lowering the overall vocabulary words, and ultimately the size of the model. This approach also allows the model to more easily adapt to out-of-vocabulary words, as infrequent words can simply be constructed by assembling smaller word-chunks (often the chunks containing a single character) [191]. While the WordPiece algorithm itself is proprietary, SentencePiece is an official open-source implementation.

Many groups have worked to endow transformer-based language models with domain-based information. In the field of scientific language, two major models have been proposed: SciBERT from AllenNLP [22], and BioBERT from Korea University in Seoul [135]. SciBERT is trained on over one-million papers from SemanticScholar.org, and constructed to completed named entity recognition, PICO Extraction, Text Classification, Relation Classification, and Dependency Parsing. For each of these tasks, training data is provided by relatively small human annotated datasets. Improved performance comes from initial pretraining done on the base of the model, in the same manner as was performed for the original BERT. From there, the base model can be used to instantiate fine-tuned version of SciBERT, each with differ-

ent "task-heads," which learn to associate the fundamental semantic content of the base SciBERT model with the particular task at hand. BioBERT performs a similar procedure, focusing on texts available from Medline and PubMedCentral, as well as English Wikipedia and the Books Corpus. Then, after being pretrained on all four datasets, BioBERT fine-tunes for named entity recognition, relation extraction, and question answering. Again, the datasets used for fine-tuning are significantly smaller than the datasets used for the BioBERT pretraining phase. In both cases, SciBERT and BioBERT demonstrate superior performance in their respective tasks.

9.3 Multi-Conditional Language Model

The CBAG model follows the transformer architecture [236] with a shallow "condition" encoder, and a deep "language model" decoder. This model is depicted in Figure 9.1. The condition is specified as a set of embeddings that enable a high degree of control. To capture information that is particular to language within biomedical domain, we add terms in our objective representing not only elements of the textual sequence, but also the part-of-speech, dependency tags, and entity class labels associated with each textual element. For each class of prediction, we minimize the sum of negative log likelihood:

$$\mathcal{L}(t, p, d, e, c) = L_T(t, t, c) + L_P(p, t, c) + L_D(d, t, c) + L_E(e, t, c) \tag{9.9}$$

where $t = t_1, \ldots, t_n$ are the set of ground-truth textual elements, each with associated $p_i \in p$ part-of-speech tags, $d_i \in d$ dependency labels, $e_i \in e$ entity labels. The term $c = c_1, \ldots, c_m$ indicates the set of conditions associated with t, and captures

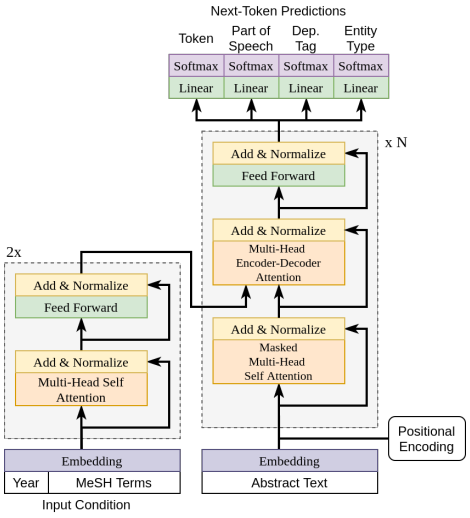


Figure 9.1: Abstract Generator Model.

information such as metadata keywords and the publication year of the ground truth elements. Each term of (9.9) follows the form of:

$$
\begin{aligned}
L_{[\cdot]}(\ell, t, c) &= \sum_{i=1}^{n} -p_{\ell_i}^{(i)} + \log\left(\sum_{j \neq i} \exp\left(p_j^{(i)}\right)\right) \\
\text{where } p^{(i)} &= \text{softmax}\left(\mathcal{H}\left(\{t_1, \ldots, t_{i-1}\}, c\right) W_{[\cdot]}\right)
\end{aligned}
\tag{9.10}
$$

where the symbol $[\cdot]$ is replaced by T, P, D, or E for each classification objective. The sequence ℓ indicate the ground-truth labels associated with each element of t with respect to the particular classification task. Additionally, $\mathcal{H}(t, c)$ is the proposed transformer model, which accepts all text elements $\{t_1, \ldots, t_{i-1}\}$ and c in order to produce an encoding for t_i. This model is defined as:

$$
\begin{aligned}
\mathcal{H}(t, c) &= D_d \\
D_{i+1} &= \mathcal{D}(D_i, E_e) \text{ and } D_0 = t + \text{PE} \\
E_{i+1} &= \mathcal{E}(E_i) \text{ and } E_0 = c
\end{aligned}
\tag{9.11}
$$

Here, PE references the positional encoding defined by the sinusoidal func-

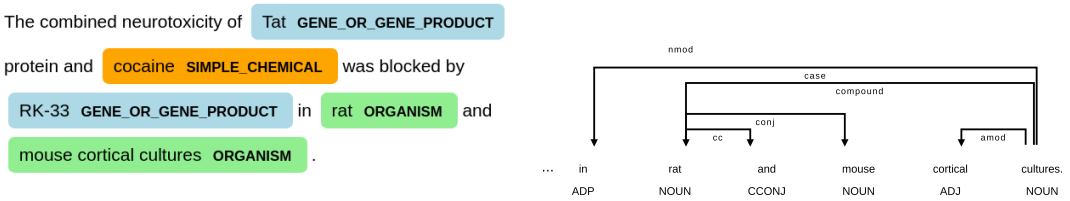
tion presented in [236]. Each input element of t and c is first assigned an input encoding and put through their respective stacks of encoder and decoder layers. Input encodings are provided by an embedding table that begins randomly initialized. We determine textual elements through the unigram word-part tokenizer [127], and contextual elements consist of a learned embedding per-publication year, as well as embeddings for each Medical Subject Heading (MeSH term). These input factors are described in father detail in Section 9.4.

Hyperparameters. We selected hyperparameters similar to the GPT-2 "medium" model. This includes an embedding dimensionality of $d_k = 1,024$, $k = 16$ attention heads per multi-headed attention layer, $e = 2$ encoder blocks, $d = 16$ decoder blocks, a fully-connected size of 3,072, and an inner-block dropout rate of 0.1. We additionally use a max sequence length of $n = 128$. Our set of initial embeddings contains 16,000 text tokens, 48,133 MeSH headings, and 230 year embeddings.

Optimization. We minimize $\mathcal{L}$ using the large-batch optimizer LAMB [257] across 40 Nvidia V100 GPUs using an effective batch size of 480. We selected a learning rate of 0.001, with a 500-batch linear warm up. We check pointed the model each epoch after viewing 5% of the training data (about 700,000 abstracts). Note that each time an abstract is viewed, we select from it a different training window. We trained this model for 72 hours using PyTorch Lightning [73] to aid in the distribution and check pointing.

9.4 Data Preparation

In order to train the model described above, we collect training samples (t, c) from the set of publicly available biomedical abstracts provided in the MEDLINE database. This dataset contains publication dates, author-supplied MeSH terms,



(a) Typed entity recognition. (b) Dependency tags and parts of speech.

Figure 9.2: Annotations provided by ScispaCy "BIONLP13CG."

titles, and abstracts for mote than 30-million citations. We filter for documents that were originally published in English, as well as documents that contain at least one non-title sentence. Documents without metadata keywords are allowed. We split the remaining 20-million abstracts into a training and test set following a 70-30 split.

Within the domain of biomedical text mining, there are relatively few annotated training sources [101, 167]. To endow the CBAG model with biomedical-domain knowledge, we annotate the entire MEDLINE training set using an NLP model trained on a smaller annotated training set. Because we leverage patterns mined from a small human-annotated dataset to gain broader insights across a vast unstructured dataset, we refer to our overall approach as semi-supervised. The ScispaCy model [163] trained on the "BIONLP13CG" BioCreative dataset [101] provides our biomedical NLP model. This model was selected because it produces the widest range of entity labels when performing named entity recognition, which consist of: cancer, organ, tissue, organism, cell, amino acid, gene or gene product, simple chemical, anatomical system, immaterial anatomical entity, multi-tissue structure, developing anatomical structure, organism subdivision, and cellular component. We add a class corresponding to "not an entity" as well.

Using the ScispaCy model and a cluster of 100 machines, we quickly identify every token, part-of-speech, dependency tag, and entity label for all 14-million training-set MEDLINE documents. We depict examples of these automatic annota-

tions in Figure 9.2. However, in order to formulate these textual features for input into the CBAG model, we also leverage the unigram subword regularization method from Kudo et al. [127]. This method learns an efficient tokenization sentences. Each token corresponds to a "chunk" of characters, many of which correspond to subword components. The unigram approach adds a normalization factor wherein the specific tokenization for each word is probabilistic determined from the set of ambiguous subword sequences. These subword sequences, along with special "start of abstract" and "end of abstract" tokens, create input t.

We train the unigram tokenization method on one-million randomly sampled sentences from the training set, specifying a fixed-size vocabulary of 16,000 subword tokens. We additionally lowercase the entire training corpus, and enforce that every character within the sampled training set receive its own token. Using the resulting model, we tokenize the entire training set, and cross reference the subwords with the multi-task labels provided by ScispaCy. This way, each subword token t_i in the training set is associated with a part-of-speech p_i, dependency tag d_i, and entity label e_i.

Next we index each training-set publication years and author-supplied MeSH keywords, which form the condition c. For publication years, we simply identify the earliest year within the training set, 1790, and add an index for each year between then and 2020. We identify over 4-million author-supplied keywords within MEDLINE, which is prohibitively large for our model to capture. We prune any keyword that occurs fewer than ten times, reducing that set to a manageable 48,133. We add each to our excising embedding index, which contains nearly 50,000 total embeddings.

When training, we select a batch of abstracts, and for each abstract we select a window of 128 subword tokens to form t, restricted such that the first token of each window corresponds to the first token of a sentence. In addition, we supply the

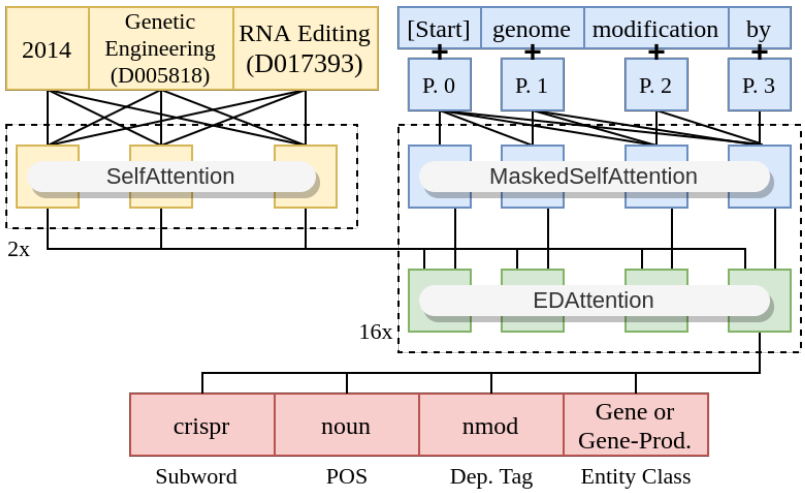


Figure 9.3: Abstract Generator Example Input.

condition indices c. The sequence of labels ℓ is formulated by shifting the subword token window by one token, such that t_{i-1} is used to predict t_i, p_i, d_i, and e_i. An example of model input and output is depicted in Figure 9.3.

9.5 Results

While NLP benchmarks such as GLUE [241] and its biomedical counterpart BLUE [167] help researchers compare performance across a range tasks, we are unaware of a benchmark for the generation of biomedical abstracts. In lieu of such a dataset, we leverage our held-out test-set of Medline abstracts, and a set of traditional NLG metrics [194]. We generate abstracts by providing a title t and condition c from a test-set abstract. We extend t by sampling from the resulting probability distribution over subword tokens $p^{(i)}$ until observing the "end of abstract" special token. The quality of the resulting abstract is quantified for each metric, Bleu [166], METEOR [131], ROUGE-L [143], and CIDEr [237], by comparing each generated sentence against the set of "reference" sentences comprising the corresponding human-written abstract.

To add context to our reported performance numbers, we also generate text using OpenAI's recently released 1.5-billion parameter "huge" GPT-2 model [173]. This model has been shown to excel on a number of tasks without modification, inducing as a replacement to traditional knowledge bases [169]. However, as this model was trained to generate language found online, such as in the BooksCorpus and English Wikipedia, it is at a disadvantage when generating domain-specific text. Because GPT-2 does not produce any "end of document" indicator, we generate the same number of subword tokens as present in the human-written counterpart, and truncate the potential partial sentence at the end of the abstract.

We present a full abstract from both CBAG and GPT-2 in Table 9.1. Note, newline characters produced by GPT-2 are replaced with "[\n]" due to space limitations. In this example, we observe that the CBAG model recovers a set of relevant biomedical entities. Unsurprisingly, the model parrots some entities that appear in the title, such as, "micro- and nanopatterned topographical cues," as well as "cellular functions" in this example. However, it is also able to produce more advanced concepts including "multiple imaging modalities," and "multiscale substrates" that do not appear in the title but do appear in the corresponding human-written abstract (not reproduced here for space concerns, but is publicly available). The GPT-2 model does recover some biomedical entities, such as "damaged retinal pathway" and "retinal pigment epithelium," however these keywords are unrelated to the considered document. Other out-of-context entities such as "artificial neural network," "computer screen," and reference to a blog reduce the ability of a human reader to extract any meaningful biomedical information from this text. We find that these example abstracts help motivate the need for domain-specific language models.

Because CBAG is a conditional language model, we explore the range of responses the model can produce given different conditions. In Table 9.2 we present

(PMID: 28026780) Hierarchically Micro- and Nanopatterned Topographical Cues for Modulation of Cellular Structure and Function.
CBAG
the ability to integrate multiple physiological cues and thereby mediate many cellular functions is critical for many complex life history processes. despite recent advances in high-throughput imaging of biomolecules and their spatiotemporal integration into dynamic structure and function, the precise structural organization and temporal structure of tissue architecture remains poorly understood. here, we present an efficient system for temporally and spatially mapping micro- and nanopatterned topographical cues in organ-specific spatial and temporal properties using multiple imaging modalities. the micro-/nanopatterned geometrical cues can be localized to cell membranes, cells, and proteins. the spatial and temporal dynamics of these local signals are precisely represented by the cross-correlation function, which forms the basis of a geometrical model that accurately provides spatiotemporal information about the spatial location and spatial coordinate of the labels and their functionalities. the model is also capable of correlating the properties of neural cells within their network without affecting the spatial and temporal organization of their spatial features, as well as those of their surrounding tissue. as an example of this model, cell types grown as multilayers are described.
GPT-2 Huge
[\n] [\n] The first example of what happens when an artificial neural network'see' complex features on a computer screen is provided by the vision of people with sight loss. The most recent results (e.g., see Krueger's blog) suggest that a person who has a damaged retinal pathway for visual light-scattering called LID is able to'see' objects using low power laser sources. The finding is unexpected because it suggests an adaptive mechanism for recovering limited visual function. The finding has profound implications for understanding how the retina and brain develop, how vision is learned, and how people can potentially regain impaired vision with new retinal implants. [\n] The retinotopic map shows which parts of the retina belong to what parts of the visual field (more details in the video below). [\n] The following images are from a retinotopic map showing a person who has lost some peripheral vision due to an injury in his optic nerve and retinal pigment epithelium. Note that he can make out details when looking toward the left, and when looking toward the right, details become distorted or are obscured.

Table 9.1: Full abstracts generated with respect to the same title.

Condition	Response
D003270: Contraceptive Agents	...that, during a prospective observational period, the patients were aware of the possibility of adverse cardiac events.
D003634: DDT	...that the aromatic (g)-tse, which is often produced in fruit, is potentially useful to suppress green algae as well as pesticide toxicity.
D004042: Unsaturated Dietary Fats	...that vitamin e levels are associated with early childhood health consequences.
D006046: Gold	...that the nanoparticles provide improved sensitivity to gold nanoparticles, and they are sensitive to ag-b interaction rather than ca-a interaction.
D005395: Fish Oils	...that the combination of pinkland and fish oil intakes (ca-like and ca-like) improves the antioxidant effect of yinneria (tricapsa vul) and that can significantly decrease food intake.

Table 9.2: Differing generations of the same prompt given various MeSH preconditions. We record the first sentence completing the prompt *"In this study, we found..."*

the first sentence produced by the model for the input "*In this study, we found...*" given different conditions. The results indicate that the condition has a significant impact in the resulting text. When conditioned with the MeSH term for contraceptive agents, the model discusses a patient study on cardiac side-effects. The output conditioned on the pesticide DDT describes fruit and toxicity. The output on gold describes describes gold-nanoparticle sensitivity. These results demonstrate the ability for the CBAG model to learn domain-specific research content provided by various keyword preconditions.

To provide further qualitative comparison between the considered models, we additionally provide a few first-sentences produced given various test-set titles in Table 9.3. In these sentences, and across the test set, we observe that CBAG produces a number of scientific cliqués. Most clearly, the model captures biomedical turns of phrase such as "in clinical practice." Additionally we observe that it is common for

(PMID: 28029317) Laparoscopy to Predict the Result of Primary Cytoreductive Surgery in Patients With Advanced Ovarian Cancer: A Randomized Controlled Trial.	
laparoscopic surgery is the standard treatment for patients with advanced ovarian cancer; however, these patients do not receive a standard palliative regimen.	J Natl Cancer Inst 2008;100:1567–1572. 24. The focus of this review is the effect of apoE4 levels on the risk of poor surgical outcome in patients with advanced ovarian cancer.
(PMID: 27993387) Low vitamin D does not predict statin associated muscle symptoms but is associated with transient increases in muscle damage and pain.	
in clinical practice, patients with moderate-to-severe hypervitaminosis d present with debilitating side effects related to statin use.	ow vitamin d does not predict statin associated muscle symptoms but is associated with transient increases in muscle damage and pain.
(PMID: 28012718) Skin-Resident Effector Memory CD8+CD284− T Cells Exhibit a Profibrotic Phenotype in Patients with Systemic Sclerosis.	
systemic sclerosis (ssc) is an inflammatory disease characterized by the infiltration of t cells into skin and skin surfaces. the presence of autoantibodies can lead to the development of cutaneous t-cell hyperactivity.	J. Clin. Invest. 117 : 2748-2759; Dilating collagen in chronic neuropathic pain. Arch. Neurol. 63 : 983-989
(PMID: 27999935) Laparoscopic sentinel node navigation surgery for early gastric cancer: a prospective multicenter trial.	
to compare the feasibility and safety of laparoscopic sentinel node navigation surgery with that of conventional in-field navigation (oif) surgery in the treatment of early gastric cancer (egc).	Patel S et al. (2003) Age associated factors associated with false-positive result of prognostic biomarkers in prostate and breast cancer.

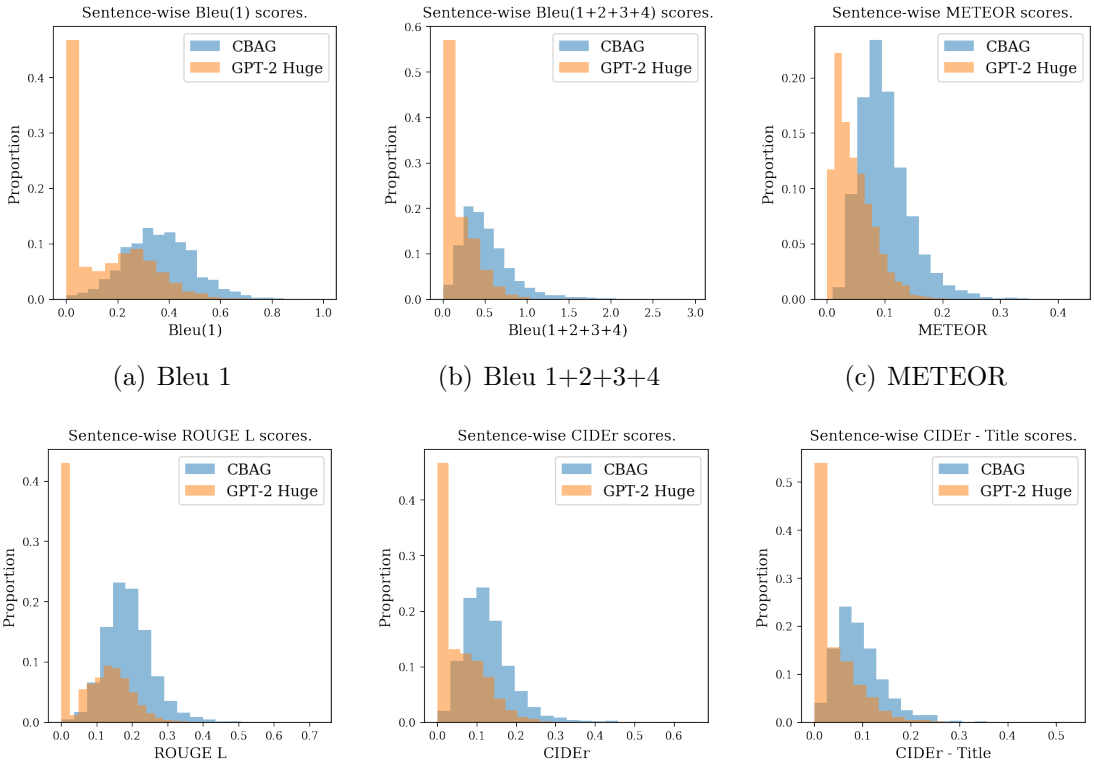
Table 9.3: CBAG (left) compared to GPT-2 "huge" with 1.5B parameters (right). Both systems are given the same title as a prompt. CBAG receives metadata. Results truncated for space.

CBAG to produce an entity followed by an abbreviation that it will repeat throughout the text. However, we observe that some abbreviations are not sensible from a human perspective, such as "in-field navigation (oif)." In these cases, the incorrect abbreviation will still be repeated by the model.

Not seen in these first-sentences is a trend for the model to follow major abstract claims with a fictional p-value or sample-size. We find p-values in approximately 10% of abstracts, with a median value of 0.02, and when plotting this distribution of generated p-values we find it matches the expected (and troubling) trend of p-values in real-world science [93].

To provide a more rigorous and scalable analysis of CBAG generations, we turn to a collect of NLP metrics, mentioned above. We use two version of Bleu, one that includes only 1-grams, and one that sums Bleu scores for 1-through-4-grams. We do not apply smoothing or any additional normalization to Bleu scores in an effort to reduce unnecessary hyperparameters. Furthermore, we present two versions of CIDEr. While both use a sub-sample of training-set abstracts to approximate n-gram document frequency, we also want to determine whether the generated text can produce uncommon n-grams that were not supplied in the title. Our "CIDER-Title" metric sets the weight of any n-gram that appeared in the title to zero. The sentence-wise score distribution for all metrics for a sample of test-set abstracts are depicted in Figure 9.4, including both scores for CBAG and GPT-2 generations. Note, these histograms are scaled such that all bars for a particular model sum to one.

We observe that about half of the sentences produced by GPT-2 contain very little content. As seen in Table 9.3, we see many of these sentences appear to be in the style of citations, including page numbers and titles. Therefore, sentences such as "J Natl Cancer Inst 2008;100:1567–1572. 24." are unlikely to recall many relevant n-grams. Other examples, such as the full GPT-generated abstract shown above, seem



(a) Bleu 1

(b) Bleu 1+2+3+4

(c) METEOR

(d) ROUGE_L

(e) CIDEr

(f) CIDEr without Title n-grams

Figure 9.4: Score distributions per-sentence comparing GPT-2 Huge with CBAG.

to discuss scientific findings from the perspective of an online news outlet covering the new research. While the CBAG generations are imperfect, they do score higher, on average, across all considered metrics. In the case of ROUGE-L, which measures the ability for generated sentences to recall long sub sequences of text, that many biomedical cliqués are likely easy for CBAG to predict, such as "the study examined the" or "we conclude that the." Our higher METEOR scores, which indicates the ability to recall n-grams in the same order as found in a reference sentence, are also effected by these common sequences. However, the "CIDEr-Title" metric explicitly decreases the weight of these common n-grams, while only considering text that could not be identified trivially. Our improved performance in this measure, when seen in the context of our overall improvement, demonstrates the ability for CBAG to produce more relevant and nontrivial biomedical text than the baseline.

9.6 Related Work

BERT [63] is a transformer-based model that consists a stack of unmasked multi-headed self-attention, which means that every output embedding depends on all input embeddings. This all-to-all dependency is what the authors mean when describing the model as "bidirectional," which departs from the more traditional left-to-right, right-to-left LSTM model.

When training BERT, input text is tokenized by the WordPiece algorithm [252], and two different types of training examples are input. In the first, some tokens are randomly replaced with a masked reserve token. The objective of the model during the unsupervised pre-training phase is to predict the original token, using the rest of the input. In the second, two sentences are supplied and, using the output embedding of the "start-of-input" character, the model must determine

whether the second sentence followed the first in the training data.

GPT [172] and **GPT-2** [173] both use a transformer-decoder stack of *masked* multi-headed self-attention. The mask, in this case, enforces that the output embedding of token i may only depend on inputs $1, \ldots, i$). This masking formulation, which we adopt in this work, restricts the GPT-models to function as pure language models. These models are pre-trained through a generative objective. For each input sequence $1, \cdots, n$, the model is input $1, \cdots, (n-1)$ and required to generate the sequence $2, \cdots, n$. Due to the masked-self-attention layers, this means that each prefix sequence of the input is simultaneously predicted each follow-up word.

The major difference between the GPT and GPT-2 models is the larger training corpus, which leads to state-of-the-art text generation. In [173], this model is even shown to improve the state-of-the-art of other objectives such as question answering and translation, even without a fine-tuning phase. Follow-up work [169] identifies that high-performance language models like GPT-2 can even replace specialty knowledge-bases.

SciBert [22] achieves state-of-the-art performance across a range of scientific NLP benchmarks by retraining the WordPeice tokenizer [252], and a BERT model [63] on 1.14-million papers collected by semantic scholar. Beltagy et al. demonstrate that by performing unsupervised pre-training on this scientific dataset, they are able to improve performance over the standard BERT-pre-trained weights on their ultimate fine-tuned models for entity recognition, PICO extraction, text classification, relation classification, and dependency parsing. These finding make the case that scientific text is sufficiently dissimilar from that found in general language to require custom models.

BioBert [135] follows the same pattern as SciBert, but pre-trains on the biomedical texts supplied by MEDLINE and PubMedCentral. As opposed to SciBert, this

method does not replace the general-language training data supplied by English Wikipedia and BooksCorpus, and instead appends both biomedical text databases. Lee et al. explore the resulting fine-tuned performance across a large range of small biomedical NLP tasks, and find mixed results. We interpret these results to indicate the importance of finding training data that is not only sufficiently large, but also relevant to the task at hand.

Wang et al. [240] explore the capacity for a BERT model to effectively function as a Markov random field language model. This technique takes advantage of the masked pre-training used in the base BERT model to predict unknown tokens. This approach also departs from the traditional language model described here as every sequence element determines the probability of every other element. Generation is performed by iterative freezing highest-probability elements from within a fixed-length sequence of initially free variables.

Ctrl [119] is a conditional language generation method that extends GPT by including "control codes" that prefix the sequence of text elements. For instance, each website represented in the training data is represented by a code, and as a result generated text can switch styles based on these prefixes. Additionally, various model functions, such as question answering, are learned via generation with various codes. As a result, prefixing questions with the respective code results in a higher probability assigned to relevant answers. Furthermore, this work includes some multi-code prefixes, such as "Rating 5.0" or "Sentence Title" to further condition the generated result. While the CTRL model is the most similar to the method presented here, it has some key differences. Firstly, the CTRL model uses prefix tokens to condition generated text, while we apply a shallow transformer-encoder stack. As a result, the CTRL approach is limited in that training requires a strict set of codes, or a small set of enumerable code-pairs. In contrast, the CBAG approach allows the method proposed here to

accept arbitrary-length sequences of keywords as a condition.

9.7 Future Challenges and Ethical Considerations

Many readers have likely heard of "hoax" paper generators similar to Scigen [214]. This particular project generates computer science full-text articles by randomly sampling from a context-free grammar, and has produced publications actually accepted by some venues. This 15-year-old system, however, is incapable of fooling but the least-observant of gate keepers. However, high quality text generation introduces NLP to a range of challenges currently posed by "deepfake" images. These problematic pictures permeate the zeitgeist and stir a response reaching further than computer science [65], extending into law [32], culture [13], and philosophy [76]. Meanwhile, misinformation spread by human actors online already cascades throughout social network echo chambers at an alarming rate [59]. One needs very little imagination to conceive of ways that the automatic generation of "pseudo-science" online could lead to public distrust of the scientific community.

OpenAI is forming partnerships between computer-science and the social sciences in order to understand these implications in society [137]. One major challenge they note is a distinct lack of "correctness" measures for text generation. In completing this work, we find that some correctness measures do exist, such as the SPICE metric to judge image caption correctness [14]. Unfortunately, this technique does not scale well to large knowledge bases as it requires the graph of predicate arguments induced by reference sentences. Not only are there a lack of methods to extract arguments from text, but we need to find new algorithms for quantifying correctness for large graphs induced by all of biomedical science.

Despite the potential for abuse, we designed CBAG with our own vision to-

ward enabling human-understandable hypothesis generation systems. For instance, our model architecture could be conditioned on more generalized forms of existing biomedical knowledge, such as semantic graph embeddings, in order to produce textual descriptions of plausible future research directions. These explanations could potentially persuade domain scientists to pursue new research directions, as similar systems have already done [10, 18]. However, these systems require specialized analysis and introduce new cognitive burdens for scientists to understand and act on their outputs. If similar hypothesis generation systems instead could produce human-readable arguments, then we could better utilize the wealth of publicly available information, improve the productivity of biomedical researchers, and ultimately find new treatments and cures for people worldwide.

9.8 Conclusions

We present the Conditioned Biomedical Abstract Generation (CBAG) model for understanding scientific abstracts. We train this model using publicly available biomedical data provide through MEDLINE to predict text that is conditioned on publication year and arbitrary sets of author-supplied keywords. This model leverages the transformer architecture [236], featuring a shallow condition encoder, as well as a deep language model decoder. Using CBAG, and a range of natural language generation metrics [194], we demonstrate the need for such a domain-specialized model, as opposed to a larger more general model like GPT-2.

We anticipate that conditioned language generation can be used to build new applications in the biomedical domain, such as a hypothesis generation system that produces textual descriptions of proposed new research directions. To do so, the conditional aspect of the CBAG model will likely be a necessity. However, we also

acknowledge the ethical considerations behind the proliferation of convincing scientific language generation models. We provide the pre-trained model, over 13,000 generated abstracts, and all necessary training and evaluation code to aid in exploration and reproduceability.

Appendices

Appendix A

Hypergraph Partitioning Details

Table A.1: Hypergraph Details.

H	$\|V\|$	$\|E\|$	V Deg.		E Size	
			Mean	Std.	Mean	Std.
as-caida	26475	16538	3.657	29.016	5.855	29.016
baxter	30722	24255	3.528	5.359	4.469	5.359
brainpc2	27606	27606	6.498	131.257	6.498	131.257
c-39	9271	9271	14.757	41.233	14.757	41.233
c-42	10471	10471	10.532	41.339	10.532	41.339
c-43	11125	11125	11.117	72.602	11.117	72.602
c-45	13206	13206	13.210	85.104	13.210	85.104
c-46	14913	14913	8.744	42.022	8.744	42.022
c-48	18354	18354	9.049	16.866	9.049	16.866
c-49	21132	21132	7.431	14.452	7.431	14.452
c-50	22401	22401	8.644	22.902	8.644	22.902
c/lp Mixture W/ Noise	107776	69568	5.379	12.552	8.333	12.552
c/lp Mixture	107776	69368	5.374	12.552	8.350	12.552
ca-AstroPh	18479	17490	21.369	30.683	22.577	30.683
ca-CondMat	22523	20760	8.194	10.671	8.890	10.671
ca-HepPh	11670	10514	20.181	47.173	22.400	47.173
cari	1200	400	127.333	178.662	382.000	178.662
case9	14453	14453	10.238	105.257	10.238	105.257
cit-HepPh	28093	29526	14.913	27.227	14.189	27.227
cit-HepTh	22908	22610	15.294	43.314	15.496	43.314
co9	22829	10694	4.799	5.264	10.245	5.264
com-dblp-cmty	260998	13477	2.758	4.340	53.411	4.340
coupled	11341	11317	8.685	30.083	8.704	30.083
cq9	21503	9247	4.493	4.673	10.449	4.673
cvxqp3	17500	17500	6.998	3.626	6.998	3.626
deter0 Mixture	21872	7845	2.061	0.900	5.746	0.900
e18	38601	24617	4.053	5.472	6.356	5.472
email-Enron	35153	25481	10.140	33.809	13.989	33.809
email-EuAll	60532	33292	3.765	24.650	6.846	24.650

H	$\|V\|$	$\|E\|$	V Deg.		E Size	
			Mean	Std.	Mean	Std.
fome12	48920	24284	2.913	1.303	5.869	1.303
hangGlider_4	15561	15561	9.609	110.835	9.609	110.835
hangGlider_5	16011	16011	9.696	112.427	9.696	112.427
hvdc1	24842	24842	6.440	2.936	6.440	2.936
jnlbrng1	40000	40000	4.980	0.141	4.980	0.141
lowThrust_4	13562	13562	11.867	64.031	11.867	64.031
lowThrust_5	16262	16262	12.198	70.124	12.198	70.124
lp_ken Mixture	32418	19219	10.513	10.796	17.733	10.796
lp_ken_13	42659	23393	2.157	0.542	3.933	0.542
lp_osa_07	25067	1118	5.777	1.032	129.528	1.032
lp_pds_10	49932	16239	2.149	0.424	6.607	0.424
lpi_bgindy Mixture	97920	30322	11.824	9.319	38.182	9.319
lpi_ceria3d Mixture	39600	35384	11.891	23.733	13.308	23.733
lpi_gosh	13356	3662	7.474	5.231	27.260	5.231
lpi_greenbea Mixture	50319	24711	12.288	9.328	25.023	9.328
lpl3	33686	10655	2.979	0.184	9.418	0.184
Graph Mixture W/ Noise	110703	55507	3.650	2.971	7.279	2.971
Graph Mixture	110703	55307	3.645	2.970	7.296	2.970
memplus	17758	17758	7.104	22.035	7.104	22.035
mod2	65990	34355	3.022	2.883	5.804	2.883
model10	16819	4398	8.940	4.645	34.191	4.645
mult_dcop_01	25019	24817	7.710	144.682	7.773	144.682
mult_dcop_02	25019	24817	7.710	144.682	7.773	144.682
mult_dcop_03	25019	24817	7.708	144.682	7.771	144.682
nemsemm2	48857	6922	3.725	2.568	26.292	2.568
nemswrld	28496	6512	6.743	5.108	29.507	5.108
nsir	10055	4450	15.409	25.894	34.817	25.894
nug08-3rd	29856	19728	4.971	3.505	7.523	3.505
obstclae	39996	39996	4.941	0.418	4.941	0.418
OPF_3754	15435	15435	10.254	5.531	10.254	5.531
p010	19081	10071	6.183	4.984	11.715	4.984
p2p-Gnutella30	36345	9205	2.416	2.594	9.539	2.594
p2p-Gnutella31	62023	15383	2.368	2.669	9.549	2.669
pds10	16558	16558	9.038	7.258	9.038	7.258
pltexpa	70364	26894	2.033	1.288	5.319	1.288
psse0	11028	26694	9.286	6.075	3.836	6.075
psse2	11028	28632	10.452	6.713	4.026	6.713
rajat09	24482	24391	4.309	1.117	4.325	1.117
rajat10	30202	30101	4.311	1.116	4.326	1.116
rajat22	39801	38431	4.919	24.574	5.095	24.574
rajat27	20540	19163	4.786	16.261	5.130	16.261
release-flickr-links_x0_10	18612	18612	15.842	38.179	15.842	38.179
release-youtube-links_x0_100	115782	115778	3.999	7.222	3.999	7.222
release-youtube-links_x0_25	28945	28938	3.996	7.784	3.997	7.784
release-youtube-links_x0_50	57891	57888	3.999	7.222	3.999	7.222
sc205-2r	62422	35212	1.974	18.129	3.500	18.129
scagr7-2r	46679	32846	2.574	35.334	3.658	35.334
scfxm1-2b	33047	18266	3.337	6.509	6.038	6.509
scsd8-2b	35910	5130	3.140	17.607	21.982	17.607
scsd8-2c	35910	5130	3.140	17.607	21.982	17.607
scsd8-2r	60550	8650	3.141	22.885	21.990	22.885
sctap1-2b	33858	15390	2.937	17.447	6.462	17.447

H	$\|V\|$	$\|E\|$	V Deg.		E Size	
			Mean	Std.	Mean	Std.
sctap1-2r	63426	28830	2.938	23.857	6.464	23.857
soc-Epinions1	50328	31149	9.530	39.646	15.398	39.646
soc-Slashdot0811	77355	70893	11.622	37.228	12.682	37.228
soc-Slashdot0902	82159	71882	11.464	37.486	13.103	37.486
soc-sign-Slashdot081106	64371	27753	7.783	32.163	18.051	32.163
soc-sign-Slashdot090216	68836	30554	7.734	32.971	17.423	32.971
soc-sign-Slashdot090221	69038	30670	7.761	33.115	17.471	33.115
soc-sign-epinions	76359	42470	10.327	43.547	18.567	43.547
south31	35885	17989	3.120	132.364	6.224	132.364
stormg2-27	37485	14306	2.513	2.000	6.584	2.000
torsion1	39996	39996	4.941	0.418	4.941	0.418
ulevimin	46754	6394	3.515	2.714	25.703	2.714
wiki-Vote	2355	3728	43.018	40.735	27.175	40.735
world	66747	34106	2.974	2.751	5.820	2.751
youtube	90581	18173	3.107	8.121	15.487	8.121

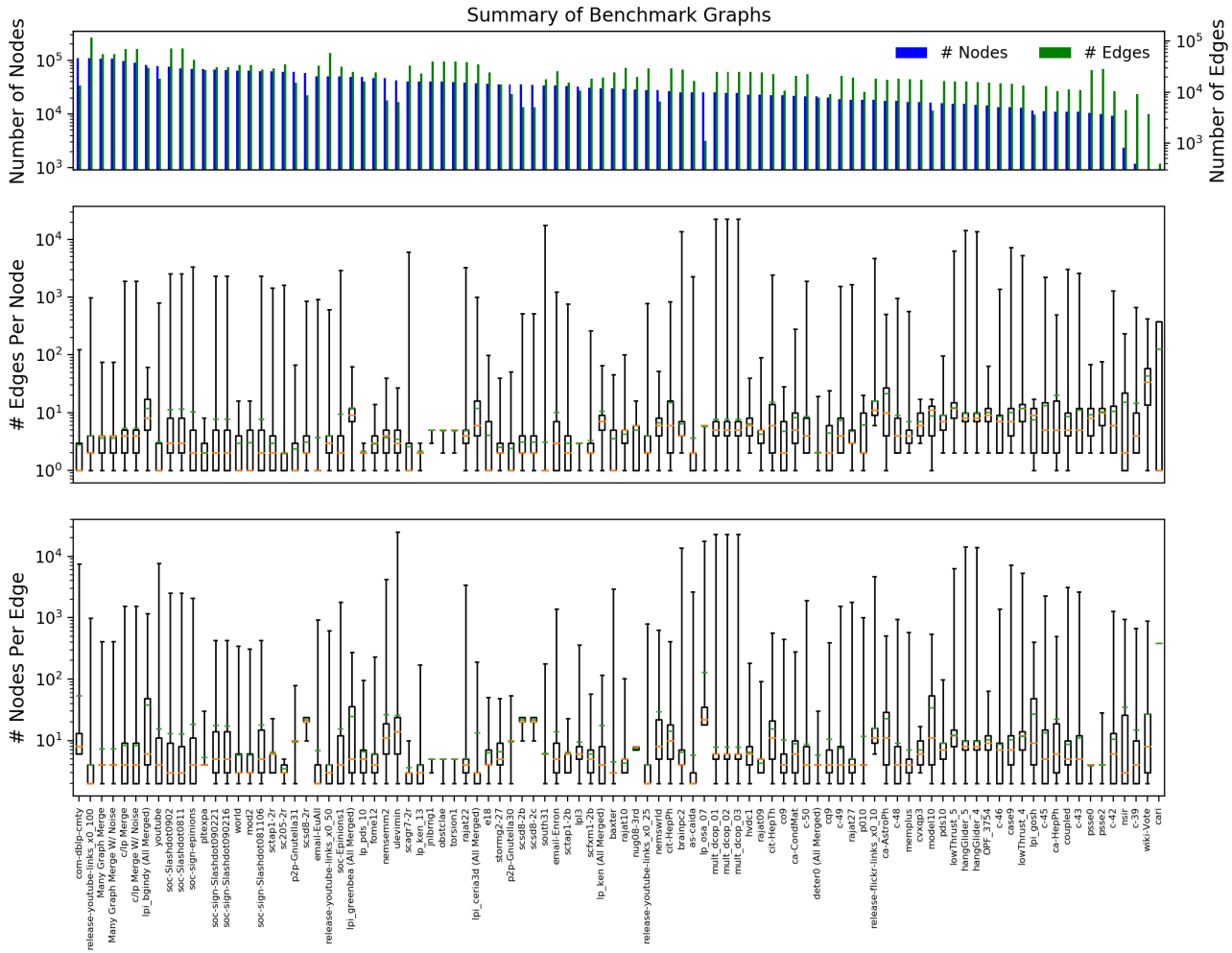


Figure A.1: Distribution of nodes and edges for each hypergraph present in our benchmark. Graphs are sorted by number of nodes.

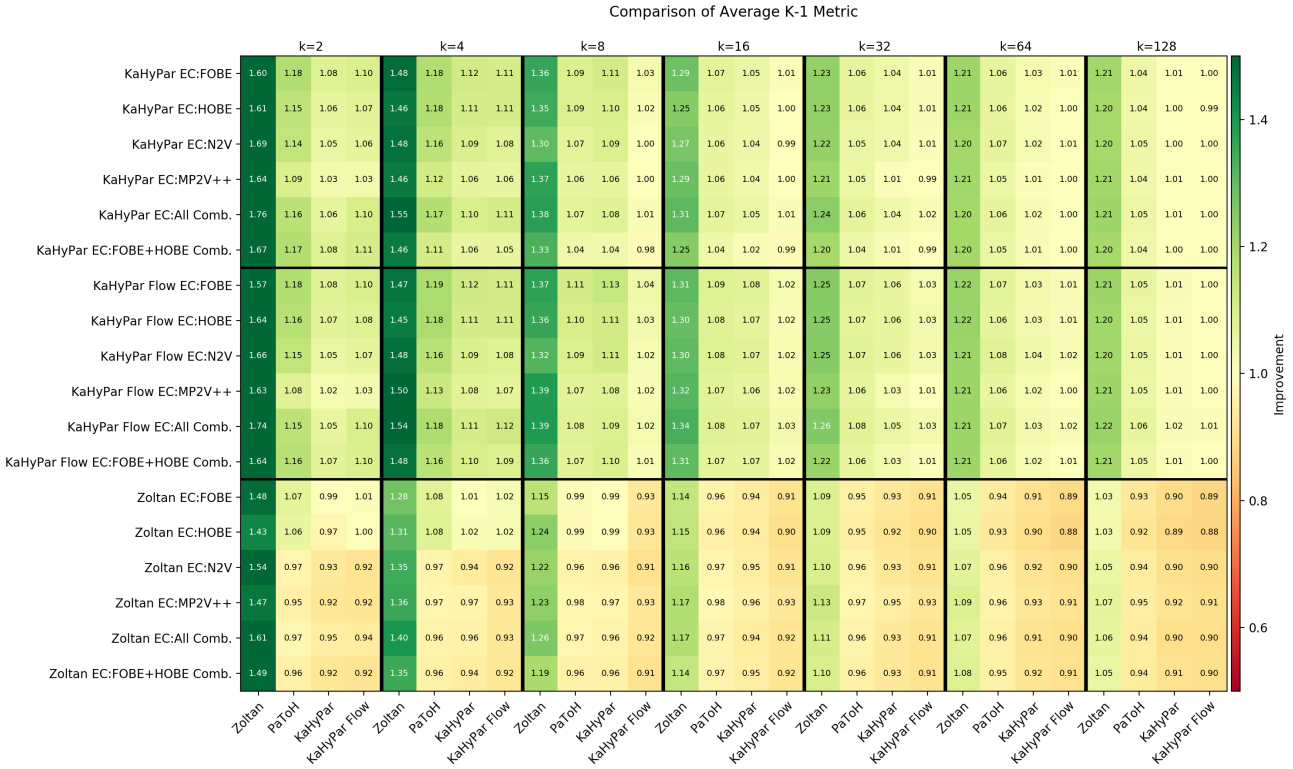


Figure A.2: The above depicts the average improvement of the connectivity objective for all considered partitioners against all baselines. The values in each cell correspond to the macro-summary $\mathcal{I}$ using $G = \text{mean}$ to summarize trials.

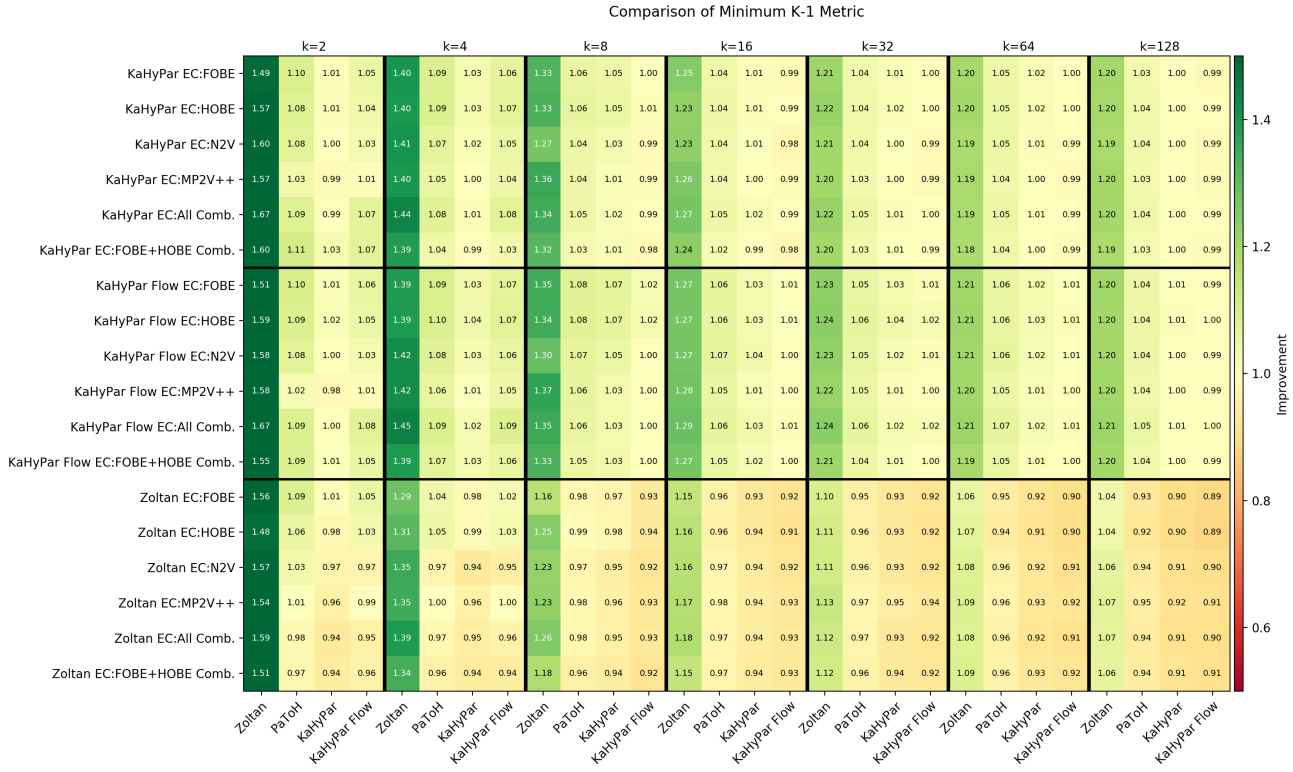


Figure A.3: The above depicts the average improvement of the connectivity objective for all considered partitioners against all baselines. The values in each cell correspond to the macro-summary $\mathcal{I}$ using $G = \min$ to summarize trials.

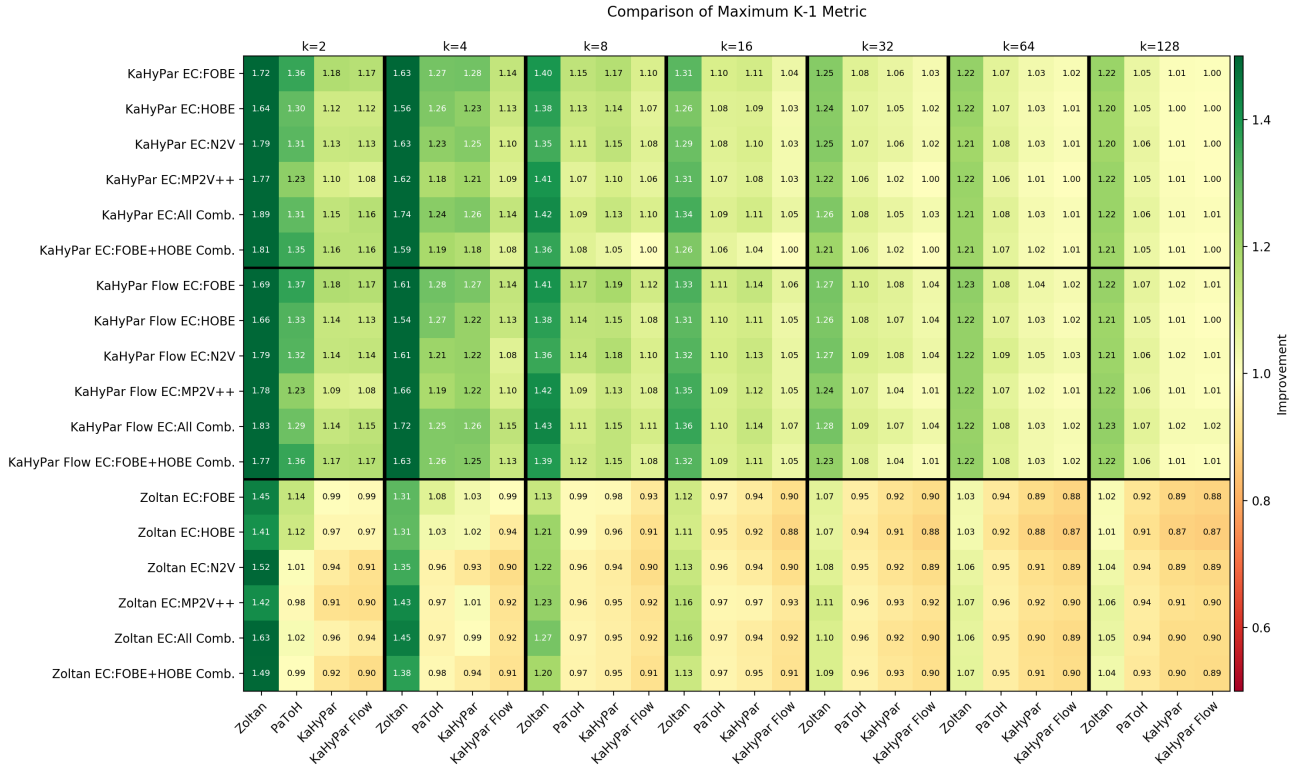


Figure A.4: The above depicts the average improvement of the connectivity objective for all considered partitioners against all baselines. The values in each cell correspond to the macro-summary $\mathcal{I}$ using $G = \max$ to summarize trials.

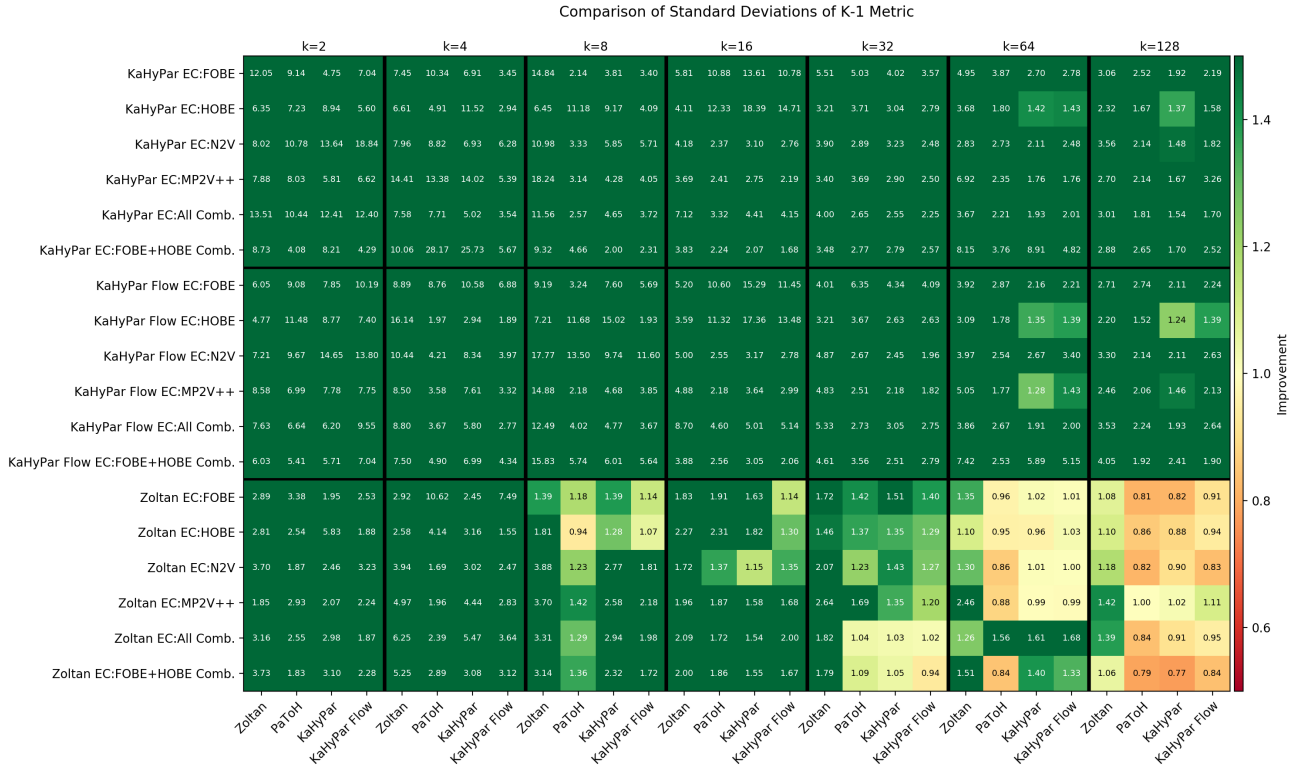


Figure A.5: The above depicts the average improvement of the connectivity objective for all considered partitioners against all baselines. The values in each cell correspond to the macro-summary $\mathcal{I}$ using $G = \text{std}$ to summarize trials.

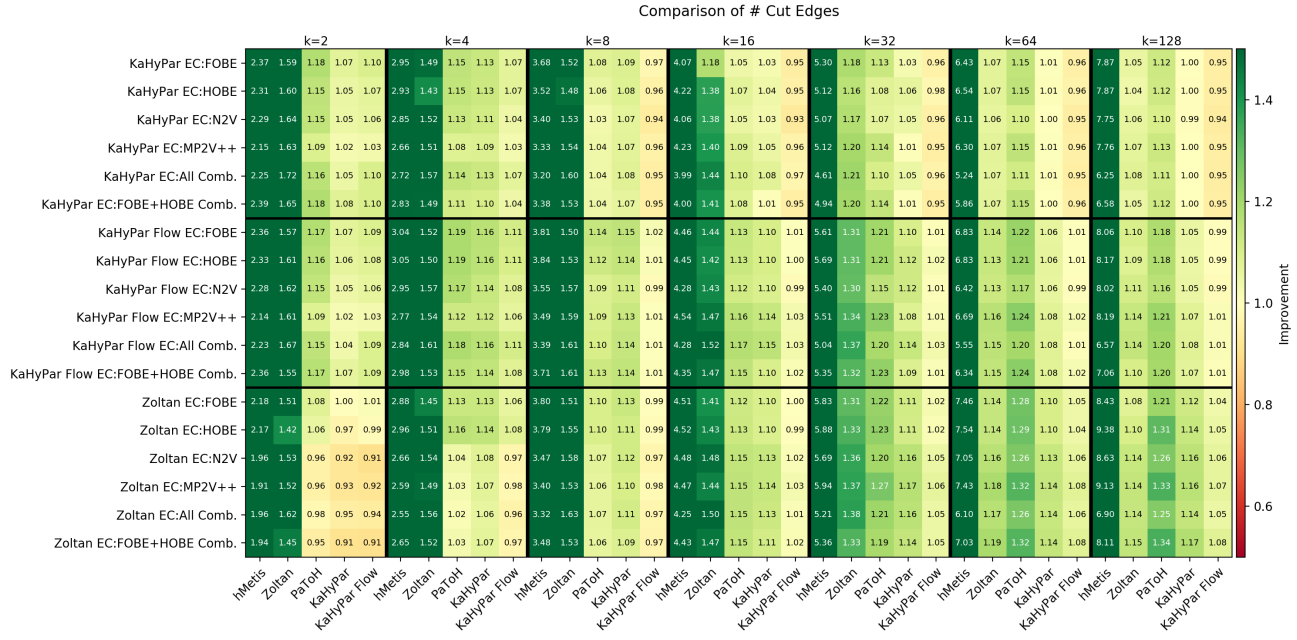


Figure A.6: The above depicts the average improvement of the cut objective for all considered partitioners against all baselines. The values in each cell correspond to the macro-summary $\mathcal{I}$ using $G = \text{mean}$ to summarize trials.

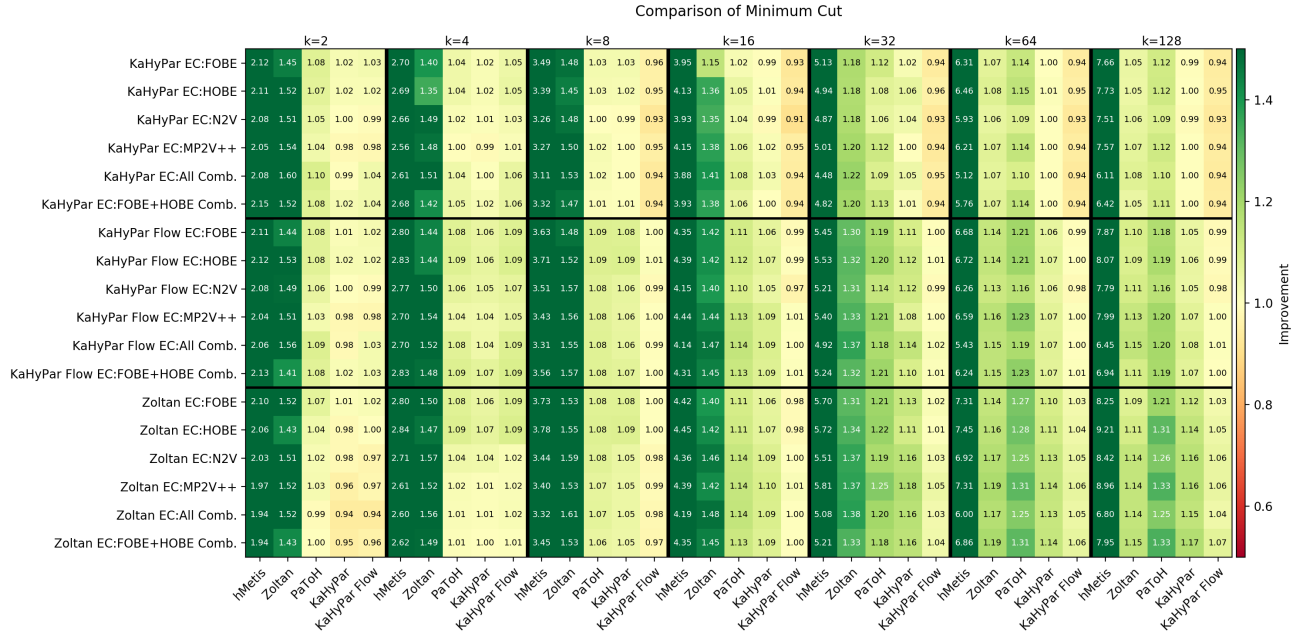


Figure A.7: The above depicts the average improvement of the cut objective for all considered partitioners against all baselines. The values in each cell correspond to the macro-summary $\mathcal{I}$ using $G = \min$ to summarize trials.

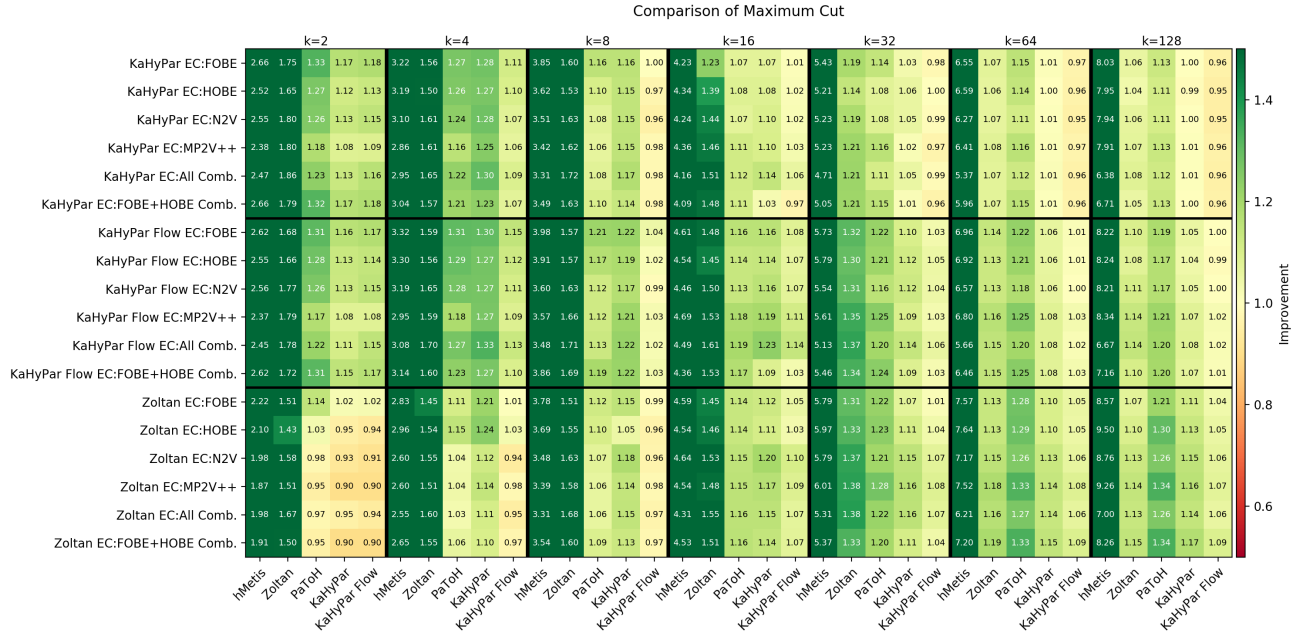


Figure A.8: The above depicts the average improvement of the cut objective for all considered partitioners against all baselines. The values in each cell correspond to the macro-summary $\mathcal{I}$ using $G = \max$ to summarize trials.

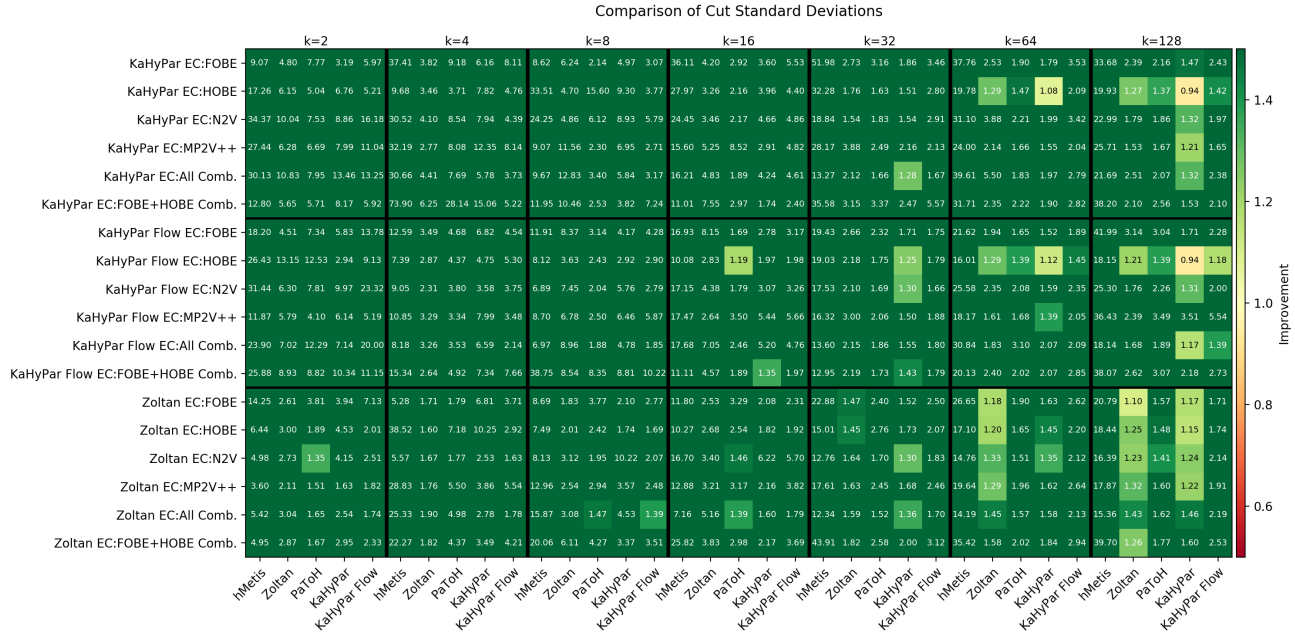


Figure A.9: The above depicts the average improvement of the cut objective for all considered partitioners against all baselines. The values in each cell correspond to the macro-summary $\mathcal{I}$ using $G = \text{std}$ to summarize trials.

Bibliography

[1] Citations added to medline by fiscal year. https://www.nlm.nih.gov/bsd/stats/cit_added.html.

[2] *HGNC Database*. European Molecular Biology Laboratory, European Bioinformatics Institute.

[3] MadGrades - UW Madison Grade Distributions. https://madgrades.com. Accessed: 2018-10-25.

[4] Rediscovering don swanson: The past, present and future of literature-based discovery.

[5] Semantic types. https://mmtx.nlm.nih.gov/MMTx/semanticTypes.shtml.

[6] Specialist nlp tools. https://lsg3.nlm.nih.gov/LexSysGroup/Projects/lvg/current/web/index.html, 2006.

[7] Umls reference manual. https://www.ncbi.nlm.nih.gov/books/NBK9684/, 2009.

[8] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[9] Sameer Agarwal, Kristin Branson, and Serge Belongie. Higher order learning with graphs. In *Proceedings of the 23rd international conference on Machine learning*, pages 17–24. ACM, 2006.

[10] Marina Aksenova, Justin Sybrandt, Biyun Cui, Vitali Sikirzhytski, Hao Ji, Diana Odhiambo, Matthew D Lucius, Jill R Turner, Eugenia Broude, Edsel Peña, et al. Inhibition of the dead box rna helicase 3 prevents hiv-1 tat and cocaine-induced neurotoxicity by targeting microglia activation. *Journal of Neuroimmune Pharmacology*, pages 1–15, 2019.

[11] Ludmil B Alexandrov, Serena Nik-Zainal, David C Wedge, Samuel AJR Aparicio, Sam Behjati, Andrew V Biankin, Graham R Bignell, Niccolo Bolli, Ake Borg, Anne-Lise Børresen-Dale, and others. Signatures of mutational processes in human cancer. 500(7463):415–421.

[12] Malorye Allison. *NCATS launches drug repurposing program.* Nature Research.

[13] Oxford Analytica. 'deepfakes' could irreparably damage public trust. *Emerald Expert Briefings*, (oxan-db).

[14] Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. Spice: Semantic propositional image caption evaluation. In *European Conference on Computer Vision*, pages 382–398. Springer, 2016.

[15] Robin Andre, Sebastian Schlag, and Christian Schulz. Memetic multilevel hypergraph partitioning. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '18, pages 347–354, 2018.

[16] Christos Andronis, Anuj Sharma, Vassilis Virvilis, Spyros Deftereos, and Aris Persidis. Literature mining, ontologies and information visualization for drug repurposing. 12(4):357–368.

[17] Patrick Arnold and Erhard Rahm. Semrep: A repository for semantic mapping. *Datenbanksysteme für Business, Technologie und Web (BTW 2015)*, 2015.

[18] Nadine Bakkar, Tina Kovalik, Ileana Lorenzini, Scott Spangler, Alix Lacoste, Kyle Sponaugle, Philip Ferrante, Elenee Argentinis, Rita Sattler, and Robert Bowser. Artificial intelligence in neurodegenerative disease research: use of IBM watson to identify additional RNA-binding proteins altered in amyotrophic lateral sclerosis. 135(2):227–247.

[19] Albert-László Barabási, Natali Gulbahce, and Joseph Loscalzo. Network medicine: a network-based approach to human disease. *Nature reviews genetics*, 12(1):56, 2011.

[20] Stephen T Barnard and Horst D Simon. Fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. *Concurrency and computation: Practice and Experience*, 6(2):101–117, 1994.

[21] Fredrik Barrenas, Sreenivas Chavali, Petter Holme, Reza Mobini, and Mikael Benson. Network properties of complex human disease genes identified through genome-wide association studies. *PloS one*, 4(11):e8090, 2009.

[22] Iz Beltagy, Arman Cohan, and Kyle Lo. Scibert: Pretrained contextualized embeddings for scientific text. *arXiv preprint arXiv:1903.10676*, 2019.

[23] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

[24] Chris Beyrer, Andrea L Wirtz, Stefan Baral, Alena Peryskina, and Frangiscos Sifakis. Epidemiologic links between drug use and HIV epidemics: an international perspective. 55:S10–S16.

[25] Charles-Edmond Bichot and Patrick Siarry. *Graph partitioning*. Wiley Online Library, 2011.

[26] Mohammed Bilgrami and Paul O'keefe. Neurologic diseases in HIV-infected patients. In *Handbook of clinical neurology*, volume 121, pages 1321–1344. Elsevier.

[27] David C. Blair and M. E. Maron. An evaluation of retrieval effectiveness for a full-text document-retrieval system. 28(3):289–299.

[28] Catherine Blake and Wanda Pratt. Automatically identifying candidate treatments from existing medical literature. In *AAAI Spring Symposium on Mining Answers from Texts and Knowledge Bases*, pages 9–13.

[29] David M Blei. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84, 2012.

[30] David M Blei and John D Lafferty. Dynamic topic models. In *Proceedings of the 23rd international conference on Machine learning*, pages 113–120. ACM.

[31] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. 3:993–1022.

[32] Marc Jonathan Blitz. Lies, line drawing, and deep fake news. *Okla. L. Rev.*, 71:59, 2018.

[33] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. Recommender systems survey. *Knowledge-based systems*, 46:109–132, 2013.

[34] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[35] Guus M Bol, Farhad Vesuna, Min Xie, Jing Zeng, Khaled Aziz, Nishant Gandhi, Anne Levine, Ashley Irving, Dorian Korz, Saritha Tantravedi, and others. Targeting DDX3 with a small molecule inhibitor for lung cancer therapy. 7(5):648–669.

[36] Guus Martinus Bol, Min Xie, and Venu Raman. DDX3, a potential target for cancer treatment. 14(1):188.

[37] Erik G Boman, Umit V Catalyurek, Cédric Chevalier, Karen D Devine, Ilya Safro, and Michael M Wolf. Advances in parallel partitioning, load balancing and matrix ordering for scientific computing. In *Journal of Physics: Conference Series*, volume 180, pages 12008–12013. Institute of Physics Publishing, 2009.

[38] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795, 2013.

[39] A. Brandt and D. Ron. Chapter 1 : Multigrid solvers and multilevel optimization strategies. In J. Cong and J. R. Shinnerl, editors, *Multilevel Optimization and VLSICAD*. Kluwer, 2003.

[40] Achi Brandt, James J. Brannick, Karsten Kahl, and Irene Livshits. Bootstrap AMG. *SIAM J. Scientific Computing*, 33(2):612–632, 2011.

[41] Peter Bruza and Marc Weeber. *Literature-based discovery*. Springer Science & Business Media.

[42] Shilpa Buch, Honghong Yao, Minglei Guo, Tomohisa Mori, Blaise Mathias-Costa, Vijeta Singh, Pankaj Seth, John Wang, and Tsung-Ping Su. Cocaine and HIV-1 interplay in CNS: cellular and molecular mechanisms. 10(5):425–428.

[43] Thang Nguyen Bui and Curt Jones. Finding good approximate vertex and edge partitions is np-hard. *Information Processing Letters*, 42(3):153–159, 1992.

[44] Aydin Buluc and Erik G. Boman. Towards scalable parallel hypergraph partitioning. In *CSRI Summer Proceedings 2008*, pages 109–119. Sandia National Labs, 2008.

[45] Aydın Buluç, Henning Meyerhenke, Ilya Safro, Peter Sanders, and Christian Schulz. Recent advances in graph partitioning. In *Algorithm Engineering*, pages 117–158. Springer, 2016.

[46] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Deep neural networks for learning graph representations. In *Thirtieth AAAI conference on artificial intelligence*, 2016.

[47] Ümit Çatalyürek and Cevdet Aykanat. Patoh (partitioning tool for hypergraphs). *Encyclopedia of Parallel Computing*, pages 1479–1487, 2011.

[48] Umit V Catalyurek and Cevdet Aykanat. Hypergraph-partitioning based decomposition for parallel sparse-matrix vector multiplication. 10(7):673–693.

[49] HH Chen, HI Yu, WC Cho, and WY Tarn. DDX3 modulates cell adhesion and motility and cancer cell metastasis via rac1-mediated signaling pathway. 34(21):2790–2800.

[50] Jie Chen and Ilya Safro. Algebraic distance on graphs. 33(6):3468–3490.

[51] Ying Chen, JD Elenee Argentinis, and Griff Weber. Ibm watson: how cognitive computing can be applied to big data challenges in life sciences research. *Clinical therapeutics*, 38(4):688–701, 2016.

[52] Byung-Kwon Choi, Tajhal Dayaram, Neha Parikh, Angela D Wilkins, Meena Nagarajan, Ilya B Novikov, Benjamin J Bachman, Sung Yun Jung, Peter J Haas, Jacques L Labrie, et al. Literature-based automated discovery of tumor suppressor p53 phosphorylation and inhibition by nek2. *Proceedings of the National Academy of Sciences*, 115(42):10666–10671, 2018.

[53] François Chollet et al. Keras. `https://keras.io`, 2015.

[54] K. Bretonnel Cohen, Helen L. Johnson, Karin Verspoor, Christophe Roeder, and Lawrence E. Hunter. The structural and content aspects of abstracts versus bodies of full text journal articles are different. 11(1):492.

[55] Michael D Conover, Bruno Gonçalves, Jacob Ratkiewicz, Alessandro Flammini, and Filippo Menczer. Predicting the political alignment of twitter users. In *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on*, pages 192–199. IEEE.

[56] Mark Craven, Johan Kumlien, et al. Constructing biological knowledge bases by extracting information from text sources. In *ISMB*, volume 1999, pages 77–86, 1999.

[57] Cristina-Maria Cruciat, Christine Dolde, Reinoud EA de Groot, Bisei Ohkawara, Carmen Reinhard, Hendrik C Korswagen, and Christof Niehrs. RNA helicase DDX3 is a regulatory subunit of casein kinase 1 in wnt–β-catenin signaling. 339(6126):1436–1441.

[58] Timothy A Davis and Yifan Hu. The university of florida sparse matrix collection. *ACM Transactions on Mathematical Software (TOMS)*, 38(1):1, 2011.

[59] Michela Del Vicario, Alessandro Bessi, Fabiana Zollo, Fabio Petroni, Antonio Scala, Guido Caldarelli, H Eugene Stanley, and Walter Quattrociocchi. The spreading of misinformation online. *Proceedings of the National Academy of Sciences*, 113(3):554–559, 2016.

[60] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[61] Neo4J Developers. Neo4j. `https://neo4j.com/`.

[62] Karen D Devine, Erik G Boman, Robert T Heaphy, Rob H Bisseling, and Umit V Catalyurek. Parallel hypergraph partitioning for scientific computing. In *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, pages 10–pp. IEEE, 2006.

[63] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

[64] Anna Divoli, Eneida A Mendonça, James A Evans, and Andrey Rzhetsky. Conflicting biomedical assumptions for mathematical modeling: the case of cancer metastasis. 7(10):e1002132.

[65] Brian Dolhansky, Russ Howes, Ben Pflaum, Nicole Baram, and Cristian Canton Ferrer. The deepfake detection challenge (dfdc) preview dataset. *arXiv preprint arXiv:1910.08854*, 2019.

[66] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 135–144. ACM, 2017.

[67] Mauro Bittencourt Dos Santos. The textual organization of research paper abstracts in applied linguistics. 16(4):481–500.

[68] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. 12:2121–2159.

[69] Ahmed El-Kishky, Yanglei Song, Chi Wang, Clare R Voss, and Jiawei Han. Scalable topical phrase mining from text corpora. 8(3):305–316.

[70] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[71] Lauri Eronen and Hannu Toivonen. Biomine: predicting links between biological entities using network models of heterogeneous databases. *BMC bioinformatics*, 13(1):119, 2012.

[72] James A Evans and Andrey Rzhetsky. Advancing science through mining libraries, ontologies, and communities. 286(27):23659–23666.

[73] W.A. et al. Falcon. Pytorch lightning. `https://github.com/PytorchLightning/pytorch-lightning`, 2019.

[74] Christos Faloutsos. Signature-based text retrieval methods: A survey. *IEEE Data Eng. Bull.*, 13(1):25–32, 1990.

[75] Charles M Fiduccia and Robert M Mattheyses. A linear-time heuristic for improving network partitions. In *Papers on Twenty-five years of electronic design automation*, pages 241–247. ACM, 1988.

[76] Luciano Floridi. Artificial intelligence, deepfakes and a future of ectypes. *Philosophy & Technology*, 31(3):317–321, 2018.

[77] Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3-5):75–174, 2010.

[78] Jacob G Foster, Andrey Rzhetsky, and James A Evans. Tradition and innovation in scientists' research strategies. 80(5):875–908.

[79] Ming Gao, Leihui Chen, Xiangnan He, and Aoying Zhou. BiNE: Bipartite Network Embedding. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, pages 715–724, New York, NY, USA, 2018. ACM.

[80] Kwang-Il Goh, Michael E Cusick, David Valle, Barton Childs, Marc Vidal, and Albert-László Barabási. The human disease network. 104(21):8685–8690.

[81] Vishrawas Gopalakrishnan, Kishlay Jha, Guangxu Xun, Hung Q Ngo, and Aidong Zhang. Towards self-learning based hypotheses generation in biomedical text domain. *Bioinformatics*, 34(12):2103–2115, 2018.

[82] Vishrawas Gopalakrishnan, Kishlay Jha, Aidong Zhang, and Wei Jin. Generating hypothesis: Using global and local features in graph to discover new knowledge from medical literature. In *Proceedings of the 8th International Conference on Bioinformatics and Computational Biology, BICOB*, pages 23–30, 2016.

[83] Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94, 2018.

[84] Derek Greene, Gerard Cagney, Nevan Krogan, and Pádraig Cunningham. Ensemble non-negative matrix factorization methods for clustering protein–protein interactions. 24(15):1722–1728.

[85] Thomas L Griffiths, Michael I Jordan, Joshua B Tenenbaum, and David M Blei. Hierarchical topic models and the nested chinese restaurant process. In *Advances in neural information processing systems*, pages 17–24.

[86] Thomas L Griffiths and Mark Steyvers. Finding scientific topics. 101:5228–5235.

[87] Chris Gropp, Alexander Herzog, Ilya Safro, Paul W Wilson, and Amy W Apon. Scalable dynamic topic modeling with clustered latent dirichlet allocation (CLDA).

[88] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864. ACM, 2016.

[89] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304, 2010.

[90] A. Hamosh, A. F. Scott, J. S. Amberger, C. A. Bocchini, and V. A. Mckusick. Online mendelian inheritance in man (OMIM), a knowledgebase of human genes and genetic disorders. 33.

[91] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier.

[92] Lei Hao, Zhongmin Zou, Hong Tian, Yubo Zhang, Huchuan Zhou, and Lei Liu. Stem cell-based therapies for ischemic stroke. 2014:468748.

[93] Megan L Head, Luke Holman, Rob Lanfear, Andrew T Kahn, and Michael D Jennions. The extent and consequences of p-hacking in science. *PLoS biology*, 13(3):e1002106, 2015.

[94] Robert Hecht-Nielsen. Theory of the backpropagation neural network. In *Neural networks for perception*, pages 65–93. Elsevier, 1992.

[95] Matthias Hein, Simon Setzer, Leonardo Jost, and Syama Sundar Rangapuram. The total variation on hypergraphs-learning on hypergraphs revisited. In *Advances in Neural Information Processing Systems*, pages 2427–2435, 2013.

[96] Sam Henry. Indirect relatedness, evaluation, and visualization for literature based discovery. 2019.

[97] Go Eun Heo, Keeheon Lee, and Min Song. Inferring undiscovered public knowledge by using text mining-driven graph model. In *Proceedings of the ACM 8th International Workshop on Data and Text Mining in Bioinformatics*, pages 37–37. ACM.

[98] William Hersh, Chris Buckley, TJ Leone, and David Hickam. OHSUMED: an interactive retrieval evaluation and new large test collection for research. In *SIGIR'94*, pages 192–201. Springer.

[99] Tobias Heuer, Peter Sanders, and Sebastian Schlag. Network Flow-Based Refinement for Multilevel Hypergraph Partitioning. In *17th International Symposium on Experimental Algorithms (SEA 2018)*, pages 1:1–1:19, 2018.

[100] Tobias Heuer and Sebastian Schlag. Improving coarsening schemes for hypergraph partitioning by exploiting community structure. In *16th International Symposium on Experimental Algorithms, (SEA 2017)*, pages 21:1–21:19, 2017.

[101] Lynette Hirschman, Alexander Yeh, Christian Blaschke, and Alfonso Valencia. Overview of BioCreAtIvE: critical assessment of information extraction for biology. 6(1):S1.

[102] Dimitar Hristovski, Carol Friedman, Thomas C Rindflesch, and Borut Peterlin. Exploiting semantic relations for literature-based discovery. In *AMIA annual symposium proceedings*, volume 2006, page 349. American Medical Informatics Association, 2006.

[103] Dimitar Hristovski, Borut Peterlin, Joyce A Mitchell, and Susanne M Humphrey. Using literature-based discovery to identify disease candidate genes. 74(2):289–298.

[104] Xiaohua Hu, Guangrong Li, Illhoi Yoo, Xiaodan Zhang, and Xuheng Xu. A semantic-based approach for mining undiscovered public knowledge from biomedical literature. In *Granular Computing, 2005 IEEE International Conference on*, volume 1, pages 22–27. IEEE.

[105] Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. Toward controlled generation of text. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1587–1596. JMLR. org, 2017.

[106] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128, 2010.

[107] Antti Jekunen. Decision-making in product portfolios of pharmaceutical research and development–managing streams of innovation in highly regulated markets. 8:2009.

[108] Rob Jelier, Martijn J Schuemie, Antoine Veldhoven, Lambert CJ Dorssers, Guido Jenster, and Jan A Kors. Anni 2.0: a multipurpose text-mining tool for the life sciences. *Genome biology*, 9(6):R96, 2008.

[109] Kishlay Jha, Guangxu Xun, Yaqing Wang, Vishrawas Gopalakrishnan, and Aidong Zhang. Concepts-bridges: Uncovering conceptual bridges based on biomedical concept evolution. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1599–1607, 2018.

[110] Kishlay Jha, Guangxu Xun, Yaqing Wang, and Aidong Zhang. Hypothesis generation from text based on co-evolution of biomedical concepts. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 843–851, 2019.

[111] Emmanuel John and Ilya Safro. Single-and multi-level network sparsification by algebraic distance. *Journal of Complex Networks*, 5(3):352–388, 2016.

[112] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*, 2017.

[113] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. FastText.zip: Compressing text classification models.

[114] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics, April 2017.

[115] George Karypis. hmetis 1.5: A hypergraph partitioning package. *http://www.cs. umn. edu/˜ metis*, 1998.

[116] George Karypis, Rajat Aggarwal, Vipin Kumar, and Shashi Shekhar. Multilevel hypergraph partitioning: applications in vlsi domain. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 7(1):69–79, 1999.

[117] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, 20(1):359–392, 1998.

[118] Brian W Kernighan and Shen Lin. An efficient heuristic procedure for partitioning graphs. *Bell Syst. Tech. J.*, 49(2):291–307, 1970.

[119] Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*, 2019.

[120] Halil Kilicoglu, Dongwook Shin, Marcelo Fiszman, Graciela Rosemblat, and Thomas C Rindflesch. SemMedDB: a PubMed-scale repository of biomedical semantic predications. 28(23):3158–3160.

[121] Yong Hwan Kim and Min Song. A context-based abc model for literature-based discovery. *PloS one*, 14(4), 2019.

[122] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[123] Jon M Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999.

[124] Ronald N Kostoff, Joel A Block, Jeffrey L Solka, Michael B Briggs, Robert L Rushenberg, Jesse A Stump, Dustin Johnson, Terence J Lyons, and Jeffrey R Wyatt. Literature-related discovery. *Annual review of information science and technology*, 43(1):1–71, 2009.

[125] Jane Kovalevich and Dianne Langford. Neuronal toxicity in HIV CNS disease. 7(7):687–698.

[126] Jacek Krol, Ilona Krol, Claudia Patricia Patino Alvarez, Michele Fiscella, Andreas Hierlemann, Botond Roska, and Witold Filipowicz. A network comprising short and long noncoding RNAs and RNA helicase controls mouse retina architecture. 6.

[127] Taku Kudo. Subword regularization: Improving neural network translation models with multiple subword candidates. *arXiv preprint arXiv:1804.10959*, 2018.

[128] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[129] Ann D Kwong, B Govinda Rao, and Kuan-Teh Jeang. Viral and cellular RNA helicases as antiviral targets. 4(10):845–853.

[130] Peder Olesen Larsen and Markus Von Ins. The rate of growth in scientific publication and the decline in coverage provided by science citation index. 84(3):575–603.

[131] Alon Lavie and Abhaya Agarwal. Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments. In *Proceedings of the second workshop on statistical machine translation*, pages 228–231, 2007.

[132] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196.

[133] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[134] Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pages 556–562, 2001.

[135] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: pre-trained biomedical language representation model for biomedical text mining. *arXiv preprint arXiv:1901.08746*, 2019.

[136] Sejoon Lee, Kwang H Lee, Min Song, and Doheon Lee. Building the process-drug–side effect network to discover the relationship between biological processes and side effects. In *BMC bioinformatics*, volume 12, page S2. BioMed Central, 2011.

[137] Claire Leibowicz, Steven Adler, and Peter Eckersley. When is it appropriate to publish high-stakes ai research. *Partnership on AI blog post*, 2019.

[138] Thomas Lengauer. *Combinatorial algorithms for integrated circuit layout*. Springer Science & Business Media, 2012.

[139] Adam Lerer, Ledell Wu, Jiajun Shen, Timothee Lacroix, Luca Wehrstedt, Abhijit Bose, and Alex Peysakhovich. PyTorch-BigGraph: A Large-scale Graph Embedding System. In *Proceedings of the 2nd SysML Conference*, Palo Alto, CA, USA, 2019.

[140] Jure Leskovec and Andrej Krevl. Snap datasets: Stanford large network dataset collection.

[141] Sven Leyffer and Ilya Safro. Fast response to infection spread and cyber attacks on large-scale networks. *Journal of Complex Networks*, 1(2):183–199, 2013.

[142] Anthony ML Liekens, Jeroen De Knijf, Walter Daelemans, Bart Goethals, Peter De Rijk, and Jurgen Del-Favero. Biograph: unsupervised biomedical knowledge discovery via automated hypothesis generation. *Genome biology*, 12(6):R57, 2011.

[143] Chin-Yew Lin, Guihong Cao, Jianfeng Gao, and Jian-Yun Nie. An information-theoretic approach to automatic evaluation of summaries. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 463–470. Association for Computational Linguistics, 2006.

[144] Donald AB Lindberg, Betsy L Humphreys, and Alexa T McCray. The unified medical language system. 32(4):281–291.

[145] Chun-Chi Liu, Yu-Ting Tseng, Wenyuan Li, Chia-Yu Wu, Ilya Mayzus, Andrey Rzhetsky, Fengzhu Sun, Michael Waterman, Jeremy JW Chen, Preet M Chaudhary, and others. DiseaseConnect: a comprehensive web server for mechanism-based disease–disease connections. 42:W137–W146.

[146] Weibo Liu, Zidong Wang, Xiaohui Liu, Nianyin Zeng, Yurong Liu, and Fuad E Alsaadi. A survey of deep neural network architectures and their applications. *Neurocomputing*, 234:11–26, 2017.

[147] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[148] Zhiyuan Liu, Yuzhou Zhang, Edward Y Chang, and Maosong Sun. Plda+: Parallel latent dirichlet allocation with data placement and pipeline processing. 2(3):26.

[149] Oren E Livne and Achi Brandt. Lean algebraic multigrid (LAMG): Fast graph Laplacian linear solver. *SIAM Journal on Scientific Computing*, 34(4):B499–B522, 2012.

[150] Edward Loper and Steven Bird. Nltk: the natural language toolkit. *arXiv preprint cs/0205028*, 2002.

[151] Giovanni Maga, Federico Falchi, Anna Garbelli, Amalia Belfiore, Myriam Witvrouw, Fabrizio Manetti, and Maurizio Botta. Pharmacophore modeling and molecular docking led to the discovery of inhibitors of human immunodeficiency virus-1 replication targeting the human cellular aspartic acid- glutamic acid- alanine- aspartic acid box polypeptide 3. 51(21):6635–6638.

[152] Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, and others. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge.

[153] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[154] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*, 2010.

[155] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger,

editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

[156] Dunja Mladenić. Feature subset selection in text-learning. In *European Conference on Machine Learning*, pages 95–100. Springer, 1998.

[157] Frederic Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In *Aistats*, volume 5, pages 246–252, 2005.

[158] Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application VISSAPP'09)*, pages 331–340. INSTICC Press.

[159] Marius Muja and David G Lowe. Scalable nearest neighbor algorithms for high dimensional data. 36(11):2227–2240.

[160] Ryan L. Murphy, Balasubramaniam Srinivasan, Vinayak Rao, and Bruno Ribeiro. Janossy pooling: Learning deep permutation-invariant functions for variable-size inputs. In *International Conference on Learning Representations*, 2019.

[161] Uwe Naumann and Olaf Schenk. *Combinatorial scientific computing*. CRC Press, 2012.

[162] NCBI Resource Coordinators. PubMed. `https://www.ncbi.nlm.nih.gov/pubmed/`.

[163] Mark Neumann, Daniel King, Iz Beltagy, and Waleed Ammar. Scispacy: Fast and robust models for biomedical natural language processing. *arXiv preprint arXiv:1902.07669*, 2019.

[164] Mark Newman. *Networks: an introduction*. Oxford university press.

[165] TI Oprea and J Mestres. Drug repurposing: far beyond new targets for old drugs. 14(4):759–763.

[166] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

[167] Yifan Peng, Shankai Yan, and Zhiyong Lu. Transfer learning in biomedical natural language processing: An evaluation of bert and elmo on ten benchmarking datasets. In *Proceedings of the 2019 Workshop on Biomedical Natural Language Processing (BioNLP 2019)*, 2019.

[168] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014.

[169] Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*, 2019.

[170] Wanda Pratt and Meliha Yetisgen-Yildiz. LitLinker: capturing connections across the biomedical literature. In *Proceedings of the 2nd international conference on Knowledge capture*, pages 105–112. ACM.

[171] Murali K Pusala, Ryan G Benton, Vijay V Raghavan, and Raju N Gottumukkala. Supervised approach to rank predicted links using interestingness measures. In *2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 1085–1092. IEEE, 2017.

[172] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*, 2018.

[173] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[174] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv e-prints*, 2019.

[175] Juan Ramos et al. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 133–142. Piscataway, NJ, 2003.

[176] Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in neural information processing systems*, pages 693–701, 2011.

[177] David Reinsel, John Gantz, and John Rydning. Data age 2025: The evolution of data to life-critical. *Don't Focus on Big Data*, pages 2–24, 2017.

[178] Matthew Rocklin. Dask: Parallel computation with blocked algorithms and task scheduling. In Kathryn Huff and James Bergstra, editors, *Proceedings of the 14th Python in Science Conference*, pages 130 – 136, 2015.

[179] Dorit Ron, Ilya Safro, and Achi Brandt. Relaxation-based coarsening and multiscale graph organization. 9(1):407–423.

[180] Andrey Rzhetsky. The big mechanism program: Changing how science is done.

[181] Andrey Rzhetsky, Jacob G Foster, Ian T Foster, and James A Evans. Choosing experiments to accelerate collective discovery. 112(47):14569–14574.

[182] Ilya Safro, Paul D Hovland, Jaewook Shin, and Michelle Mills Strout. Improving random walk performance. In *CSC*, pages 108–112, 2009.

[183] Ilya Safro, Dorit Ron, and Achi Brandt. Graph minimum linear arrangement by multilevel weighted edge contractions. *Journal of Algorithms*, 60(1):24 – 41, 2006.

[184] Ilya Safro, Peter Sanders, and Christian Schulz. Advanced coarsening schemes for graph partitioning. *Journal of Experimental Algorithmics (JEA)*, 19:2–2, 2015.

[185] Ilya Safro and Boris Temkin. Multiscale approach for the network compression-friendly ordering. *Journal of Discrete Algorithms*, 9(2):190–202, 2011.

[186] Sabindra K Samal, Samapika Routray, Ganesh Kumar Veeramachaneni, Rupesh Dash, and Mahendran Botlagunta. Ketorolac salt is a newly discovered DDX3 inhibitor to treat oral cancer. 5:9982.

[187] Peter Sanders and Christian Schulz. Engineering multilevel graph partitioning algorithms. In *European Symposium on Algorithms*, pages 469–480. Springer, 2011.

[188] Shengtian Sang, Zhihao Yang, Xiaoxia Liu, Lei Wang, Hongfei Lin, Jian Wang, and Michel Dumontier. Gredel: A knowledge graph embedding based method for drug discovery from biomedical literatures. *IEEE Access*, 7:8404–8415, 2018.

[189] Sebastian Schlag, Vitali Henne, Tobias Heuer, Henning Meyerhenke, Peter Sanders, and Christian Schulz. k-way hypergraph partitioning via n-level recursive bisection. In *18th Workshop on Algorithm Engineering and Experiments, (ALENEX 2016)*, pages 53–67, 2016.

[190] M. J. Schuemie, M. Weeber, B. J. A. Schijvenaars, E. M. van Mulligen, C. C. van der Eijk, R. Jelier, B. Mons, and J. A. Kors. Distribution of information in biomedical abstracts and full-text publications. 20(16).

[191] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

[192] Parantu K Shah, Carolina Perez-Iratxeta, Peer Bork, and Miguel A Andrade. Information extraction from full text scientific articles: where are the keywords? 4(1):20.

[193] Jingbo Shang, Jialu Liu, Meng Jiang, Xiang Ren, Clare R Voss, and Jiawei Han. Automated phrase mining from massive text corpora.

[194] Shikhar Sharma, Layla El Asri, Hannes Schulz, and Jeremie Zumer. Relevance of unsupervised metrics in task-oriented dialogue for evaluating natural language generation. *CoRR*, abs/1706.09799, 2017.

[195] Ruslan Shaydulin, Jie Chen, and Ilya Safro. Relaxation-based coarsening for multilevel hypergraph partitioning. *Multiscale Modeling & Simulation*, 17(1):482–506, 2019.

[196] Ruslan Shaydulin and Ilya Safro. Aggregative coarsening for multilevel hypergraph partitioning. *17th International Symposium on Experimental Algorithms (SEA 2018)*, 2018.

[197] Michael A Shepherd, Carolyn R Watters, and Yao Cai. Transient hypergraphs for citation networks. *Information Processing & Management*, 26(3):395–412, 1990.

[198] Chuan Shi, Yitong Li, Jiawei Zhang, Yizhou Sun, and S Yu Philip. A survey of heterogeneous information network analysis. *IEEE Transactions on Knowledge and Data Engineering*, 29(1):17–37, 2016.

[199] Chuan Shi, Yitong Li, Jiawei Zhang, Yizhou Sun, and S Yu Philip. A survey of heterogeneous information network analysis. *IEEE Transactions on Knowledge and Data Engineering*, 29(1):17–37, 2017.

[200] Gail Sinclair and Bonnie Webber. Classification from full text: A comparison of canonical sections of scientific papers. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications*, pages 66–69. Association for Computational Linguistics.

[201] Neil R Smalheiser. Literature-based discovery: Beyond the ABCs. 63(2):218–224.

[202] Neil R Smalheiser and Don R Swanson. Linking estrogen to alzheimer's disease an informatics approach. 47(3):809–810.

[203] Neil R Smalheiser and Don R Swanson. Using ARROWSMITH: a computer-assisted approach to formulating and assessing scientific hypotheses. 57(3):149–153.

[204] Neil R Smalheiser, Vetle I Torvik, and Wei Zhou. Arrowsmith two-node search interface: A tutorial on finding meaningful links between two disparate sets of articles in MEDLINE. 94(2):190–197.

[205] NR Smalheiser and DR Swanson. Assessing a gap in the biomedical literature: Magnesium deficiency and neurologic disease. 15(1):1–9.

[206] Larisa N Soldatova and Andrey Rzhetsky. Representation of research hypotheses. 2(2):S9.

[207] John F Sowa. *Principles of semantic networks: Explorations in the representation of knowledge*. Morgan Kaufmann.

[208] Scott Spangler. *Accelerating Discovery: Mining Unstructured Information for Hypothesis Generation*, volume 37. CRC Press.

[209] Scott Spangler, Angela D Wilkins, Benjamin J Bachman, Meena Nagarajan, Tajhal Dayaram, Peter Haas, Sam Regenbogen, Curtis R Pickering, Austin Comer, Jeffrey N Myers, and others. Automated hypothesis generation based on mining scientific literature. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1877–1886. ACM.

[210] Serena Spudich. HIV and neurocognitive dysfunction. 10(3):235–243.

[211] Padmini Srinivasan. Text mining: generating hypotheses from MEDLINE. 55(5):396–413.

[212] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[213] L.E. Stock. *Strategic Logistics Management*. Cram101 Textbook Outlines. Lightning Source Inc, 2006.

[214] Jeremy Stribling, Max Krohn, and Dan Aguayo. Scigen-an automatic cs paper generator, 2005.

[215] Mianen Sun, Ling Song, Tong Zhou, G Yancey Gillespie, and Richard S Jope. The role of DDX3 in regulating snail. 1813(3):438–447.

[216] Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223*, 2019.

[217] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. Lstm neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*, 2012.

[218] Don Swanson and Neil Smalheiser. Link analysis of MEDLINE titles as an aid to scientific discovery. In *Proceedings of the AAAI Fall Symposium on Artificial Intelligence and Link Analysis*, pages 94–97.

[219] Don R Swanson. Fish oil, raynaud's syndrome, and undiscovered public knowledge. 30(1):7–18.

[220] Don R Swanson. Migraine and magnesium: eleven neglected connections. 31(4):526–557.

[221] Don R Swanson. Undiscovered public knowledge. 56(2):103–118.

[222] Don R Swanson and Neil R Smalheiser. Implicit text linkage between MEDLINE records: using arrowsmith as an aid to scientific discovery. 48(1):48.

[223] Don R. Swanson and Neil R. Smalheiser. An interactive system for finding complementary literatures: A stimulus to scientific discovery. 91(2):183–203.

[224] Justin Sybrandt and Ilya Safro. Heterogeneous bipartite graph embeddings.

[225] Justin Sybrandt, Michael Shtutman, and Ilya Safro. MOLIERE: Automatic biomedical hypothesis generation system. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pages 1633–1642. ACM.

[226] Justin Sybrandt, Micheal Shtutman, and Ilya Safro. Large-scale validation of hypothesis generation systems via candidate ranking. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 1494–1503, 2018.

[227] Shaheen Syed and Marco Spruit. Full-text or abstract? examining topic coherence scores using latent dirichlet allocation. In *The 4th IEEE International Conference on Data Science and Advanced Analytics*, pages 165–174. IEEE.

[228] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1067–1077. International World Wide Web Conferences Steering Committee, 2015.

[229] Lei Tang, Huan Liu, Jianping Zhang, and Zohreh Nazeri. Community evolution in dynamic multi-mode networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 677–685. ACM, 2008.

[230] Aleksandar Trifunovic. *Parallel algorithms for hypergraph partitioning*. PhD thesis, University of London, 2006.

[231] Aleksandar Trifunović and William J Knottenbelt. Parallel multilevel algorithms for hypergraph partitioning. *Journal of Parallel and Distributed Computing*, 68(5):563–581, 2008.

[232] Anton Tsitsulin, Davide Mottin, Panagiotis Karras, and Emmanuel Müller. Verse: Versatile graph embeddings from similarity measures. In *Proceedings of the 2018 World Wide Web Conference*, pages 539–548, 2018.

[233] Richard Van Noorden. Global scientific output doubles every nine years.

[234] Marise R Heerma van Voss, Willemijne AME Schrijver, Natalie D ter Hoeve, Laurien D Hoefnagel, Quirine F Manson, Elsken van der Wall, Venu Raman, Paul J van Diest, Dutch Distant Breast Cancer Metastases Consortium, and others. The prognostic effect of DDX3 upregulation in distant breast cancer metastases. pages 1–8.

[235] Brendan Vastenhouw and Rob H Bisseling. A two-dimensional data distribution method for parallel sparse matrix-vector multiplication. *SIAM review*, 47(1):67–95, 2005.

[236] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[237] Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575, 2015.

[238] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

[239] Jessica B Voytek and Bradley Voytek. Automated cognome construction and semi-automated hypothesis generation. 208(1):92–100.

[240] Alex Wang and Kyunghyun Cho. Bert has a mouth, and it must speak: Bert as a markov random field language model. *arXiv preprint arXiv:1902.04094*, 2019.

[241] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

[242] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1225–1234, 2016.

[243] Huijun Wang, Ying Ding, Jie Tang, Xiao Dong, Bing He, Judy Qiu, and David J Wild. Finding complex biological relationships in recent PubMed articles using bio-LDA. 6(3):e17243.

[244] Peng Wang, Bo Xu, Jiaming Xu, Guanhua Tian, Cheng-Lin Liu, and Hongwei Hao. Semantic expansion using word embedding clustering and convolutional neural network for improving short text classification. 174:806 – 814.

[245] Marc Weeber, Henny Klein, Alan R Aronson, James G Mork, LT De Jong-van Den Berg, and Rein Vos. Text-based discovery in biomedicine: the architecture of the dad-system. In *Proceedings of the AMIA Symposium*, page 903. American Medical Informatics Association, 2000.

[246] Marc Weeber, Henny Klein, Lolkje de Jong-van den Berg, Rein Vos, and others. Using concepts in literature-based discovery: Simulating swanson's raynaud–fish oil and migraine–magnesium discoveries. 52(7):548–557.

[247] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big data*, 3(1):9, 2016.

[248] David Westergaard, Hans-Henrik Stærfeldt, Christian Tønsberg, Lars Juhl Jensen, and Søren Brunak. A comprehensive and quantitative comparison of text-mining in 15 million full-text articles versus their corresponding abstracts. 14(2):e1005962.

[249] Stephen Wilson, Angela Dawn Wilkins, Matthew V Holt, Byung Kwon Choi, Daniel Konecki, Chih-Hsu Lin, Amanda Koire, Yue Chen, Seon-Young Kim, Yi Wang, et al. Automated literature mining and hypothesis generation through a network of medical subject headings. *BioRxiv*, page 403667, 2018.

[250] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[251] Jonathan D Wren, Raffi Bekeredjian, Jelena A Stewart, Ralph V Shohet, and Harold R Garner. Knowledge discovery by automated identification and ranking of implicit relationships. 20(3):389–398.

[252] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[253] Meliha Yetisgen-Yildiz and Wanda Pratt. Evaluation of literature-based discovery systems. In *Literature-based discovery*, pages 101–113. Springer.

[254] Muhammed A Yıldırım, Kwang-Il Goh, Michael E Cusick, Albert-László Barabási, and Marc Vidal. Drug—target network. *Nature biotechnology*, 25(10):1119, 2007.

[255] Daniel GR Yim and David M Virshup. Unwinding the wnt action of casein kinase 1. 23(6):737.

[256] Quanzeng You, Hailin Jin, Zhaowen Wang, Chen Fang, and Jiebo Luo. Image captioning with semantic attention. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4651–4659, 2016.

[257] Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv preprint arXiv:1904.00962*, 1(5), 2019.

[258] Chenzi Zhang, Shuguang Hu, Zhihao Gavin Tang, and T-H. Hubert Chan. Re-revisiting learning on hypergraphs: Confidence interval and subgradient method. 70:4026–4034, 06–11 Aug 2017.

[259] Chenzi Zhang, Shuguang Hu, Zhihao Gavin Tang, and TH Chan. Re-revisiting learning on hypergraphs: confidence interval and subgradient method. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 4026–4034. JMLR. org, 2017.

[260] Zi-Ke Zhang and Chuang Liu. A hypergraph model of social tagging networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2010(10):P10005, 2010.

[261] Denny Zhou, Jiayuan Huang, and Bernhard Schölkopf. Learning with hypergraphs: Clustering, classification, and embedding. In *Advances in neural information processing systems*, pages 1601–1608, 2007.

[262] XueZhong Zhou, Jörg Menche, Albert-László Barabási, and Amitabh Sharma. Human symptoms–disease network. 5.