Hypothesis Generation with AGATHA

Accelerate Scientific Discovery with Deep Learning

Justin Sybrandt
School of Computing
Clemson University

Ilya Tyagin
School of Computing
Clemson University

Michael Shtutman
Drug Discovery and
Biomedical Sciences
U. of South Carolina

Ilya Safro
School of Computing
Clemson University

Motivation: Drug Discovery

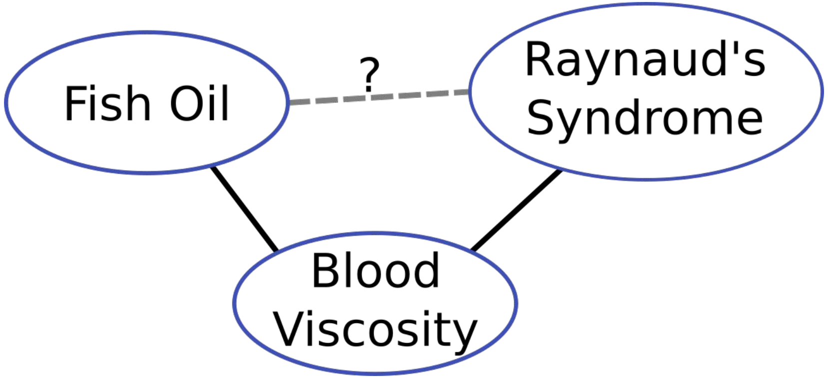
- Solution strategy:
  - Identify ~1000 target substances
  - Determine ~10 candidates
  - Conduct ~1 human trial
- Challenges:
  - Expensive early investment
  - Uncertain results

Available Data

- National Library of Medicine provides public databases
- MEDLINE contains nearly 30 million biomedical abstracts
- Data available through PubMed
- New papers per-year is increasing!
    - Nearly 1 million last year
    - New paper every 35 sec

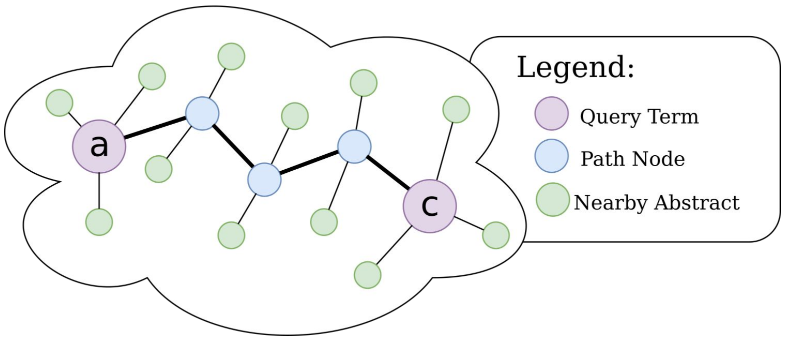
Background: ARROWSMITH (1986)

- Pattern:
  - Given two terms: A, C
  - Find words related to A
  - Find words related to C
  - Find overlap
- Key Limitations:
  - Only simple connections
  - Biased to incremental results

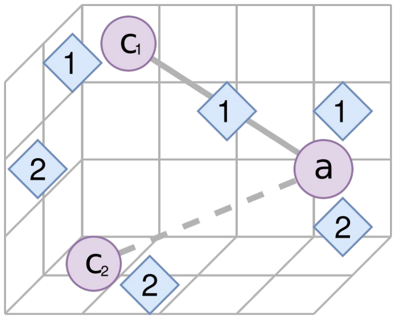


Background: MOLIERE (2017)

- Our first hypothesis generation system
- Process:
  - Construct semantic network
  - Shortest path queries
  - Topic modeling
  - Heuristic analysis
- Limitations:
  - Minutes per query
  - Heuristic analysis



Shortest Path Query

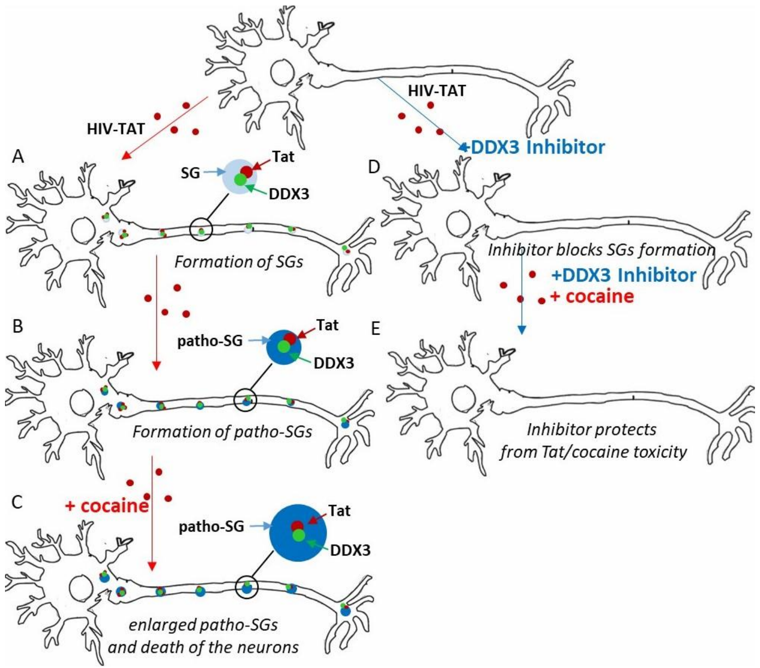


Topic Model Heuristic Analysis

Moliere: Automatic biomedical hypothesis generation system
Sybrandt, Shtutman, Safro
KDD'17

Success with HIV-Associated Dementia

- Moliere ranked 40,000 genes
- DDX3 ranked highly
- Confirmed in laboratory



Inhibition of the DDX3 prevents HIV-1 Tat and cocaine-induced neurotoxicity by targeting microglia activation

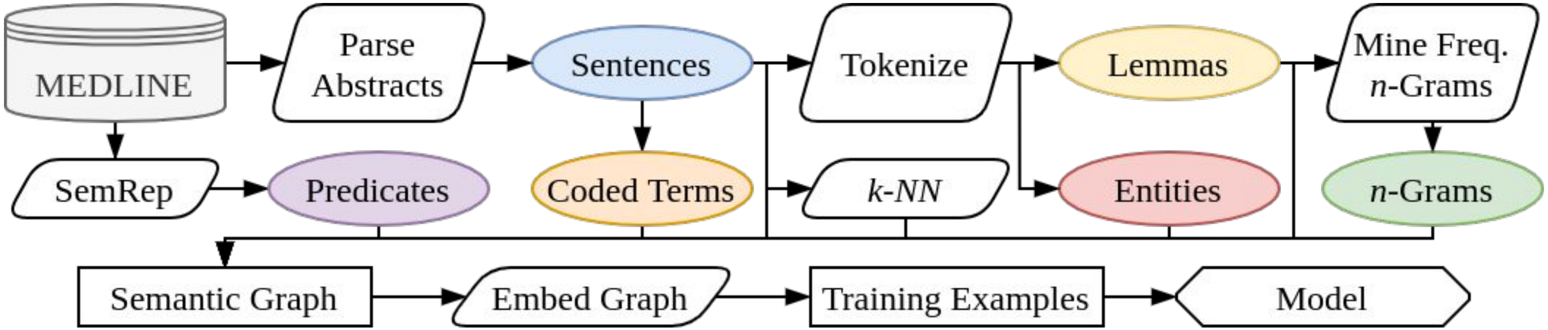
Aksenovam, Sybrandt, Chu, Sikirzhytski, Ji, Odhiambo, Lucius, Turner, Broude, Pena, Lizzaraga, Zhu, Safro, Wyatt, Shtutman

JNP'19

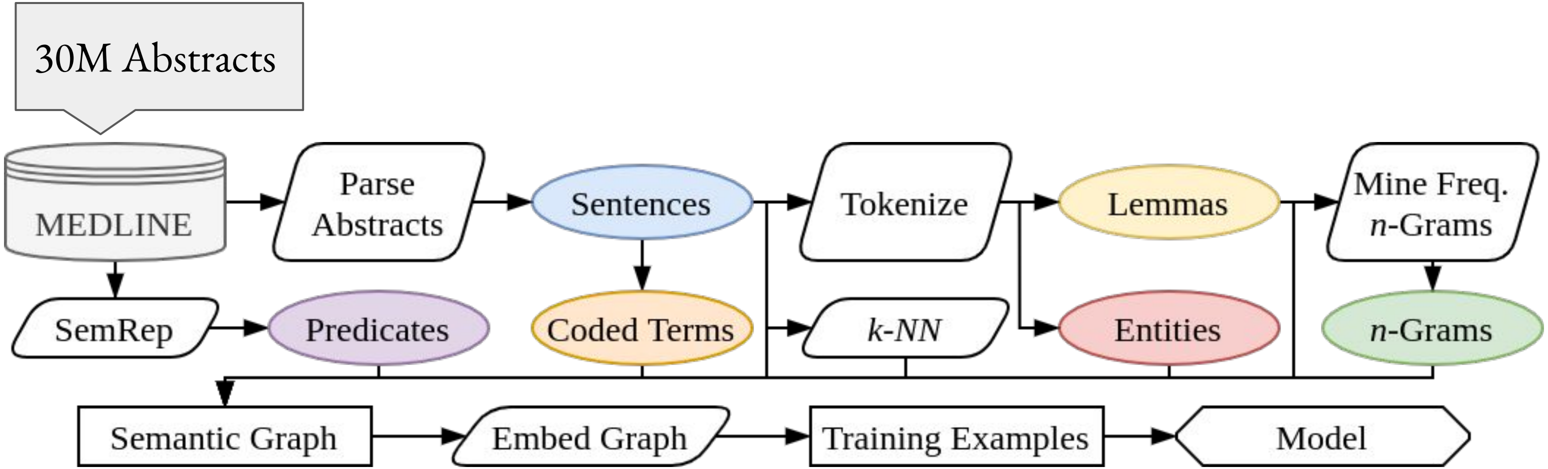
Agatha

- Process:
    - Parse MEDLINE
    - Construct semantic network
    - Train embedding & model
    - Validate through ranking
- Intuition:
    - Remove heuristics
    - Add data-driven insights
    - Speed up hypothesis queries

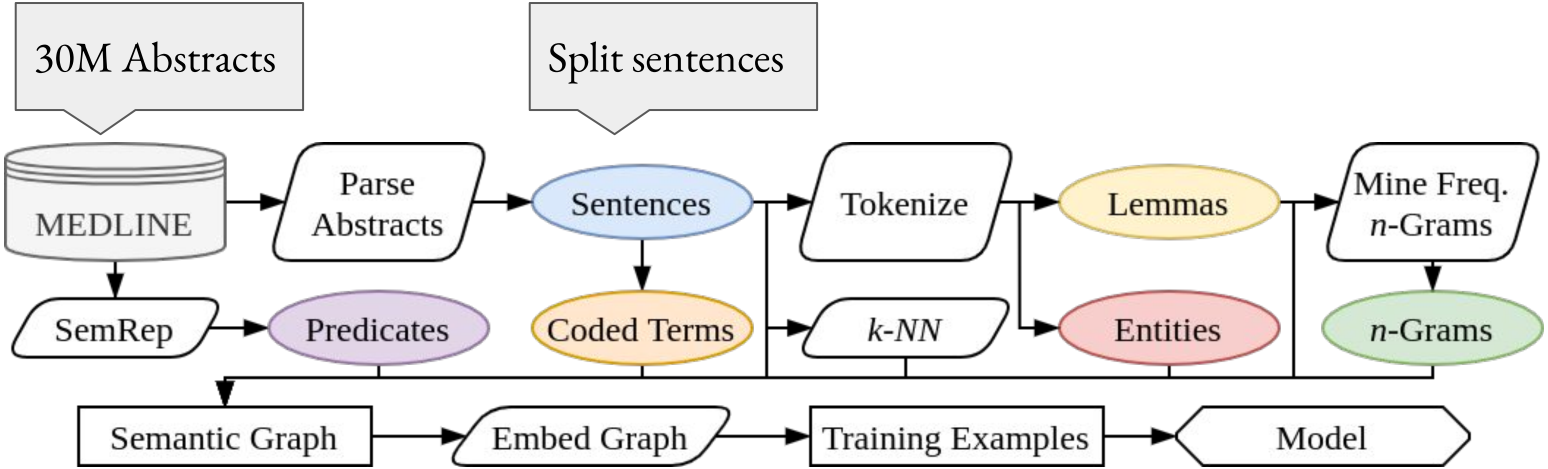
Agatha Pipeline Summary



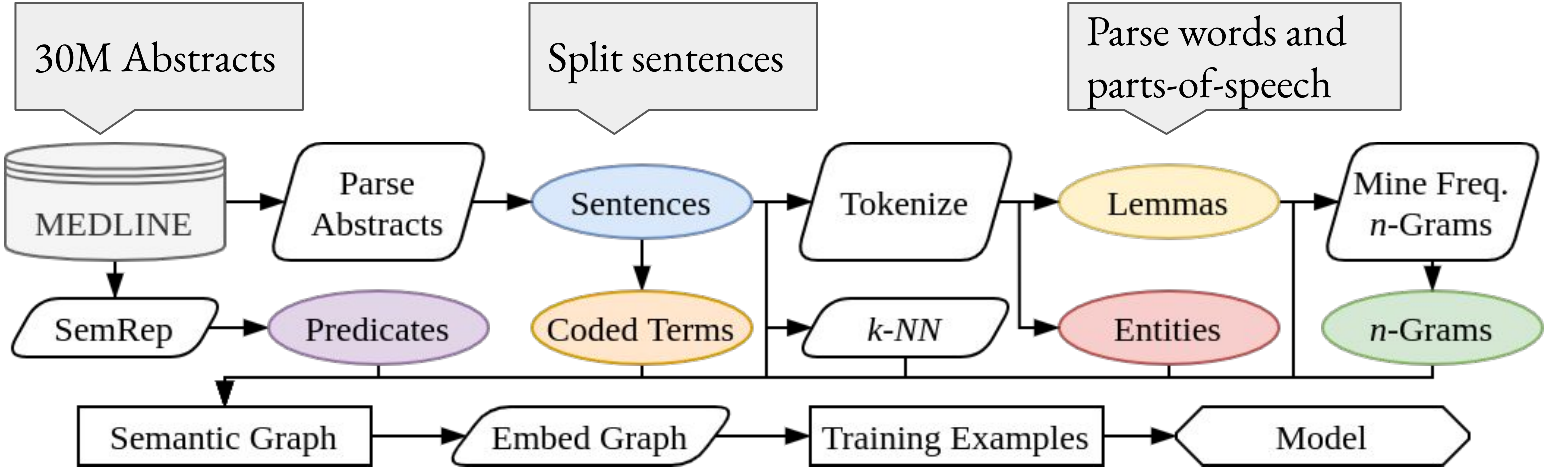
Agatha Pipeline Summary



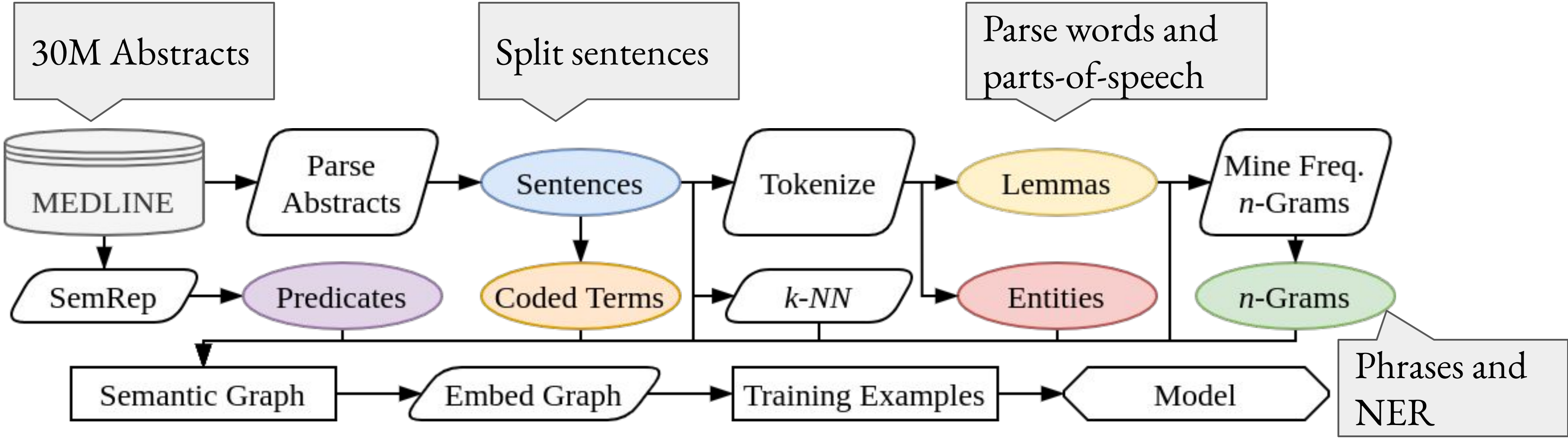
Agatha Pipeline Summary



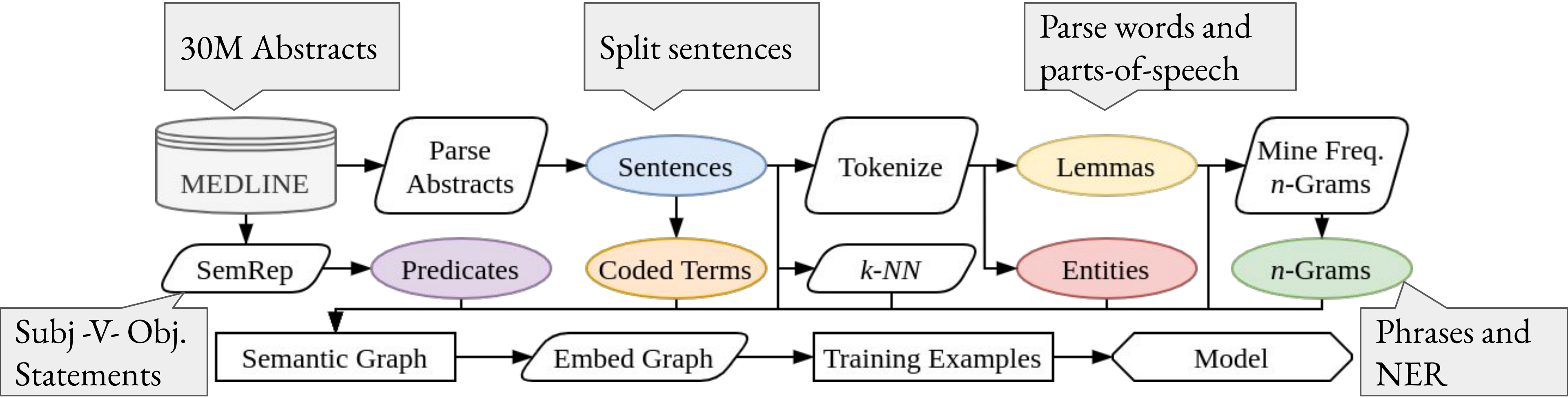
Agatha Pipeline Summary



Agatha Pipeline Summary

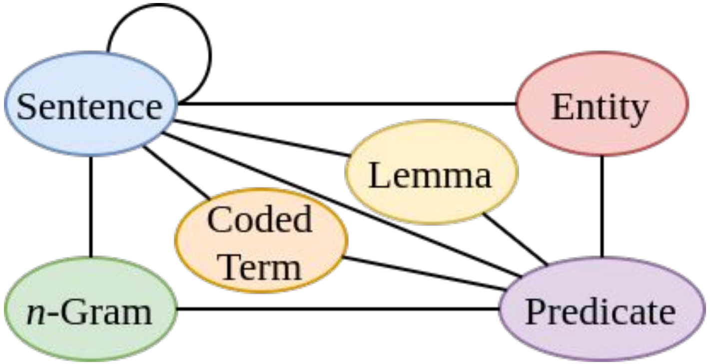


Agatha Pipeline Summary



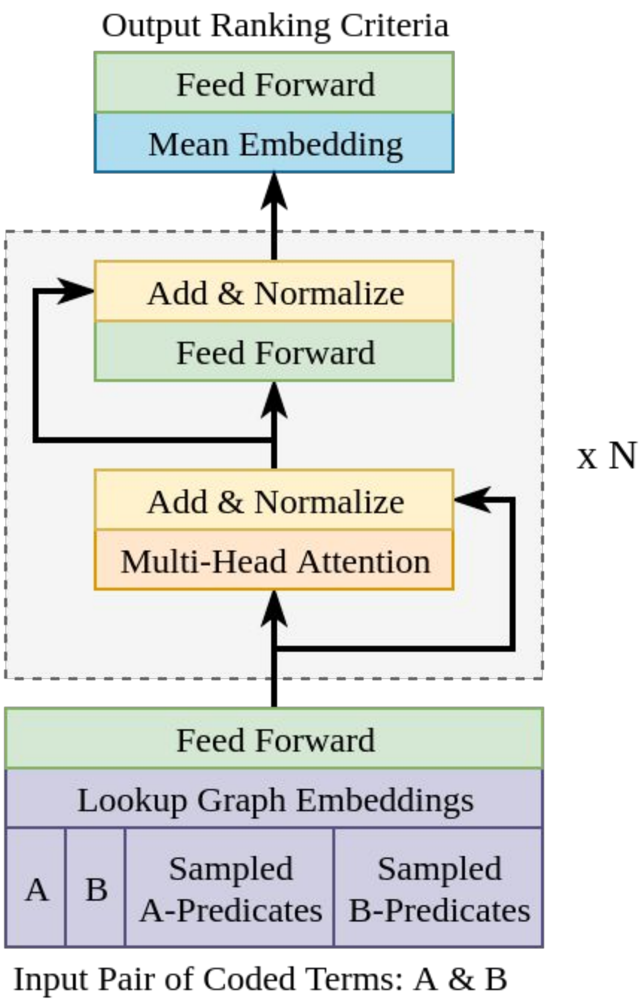
Semantic Graph

- Sentences:
  - Connected by nearest-neighbors
  - Edges to contained elements
- Predicates
  - Edges to info supplied by SemMedDB
- Size:
  - 2015 Release:
    - 188 M. Nodes
  - 2020 Release:
    - 270 M. Nodes



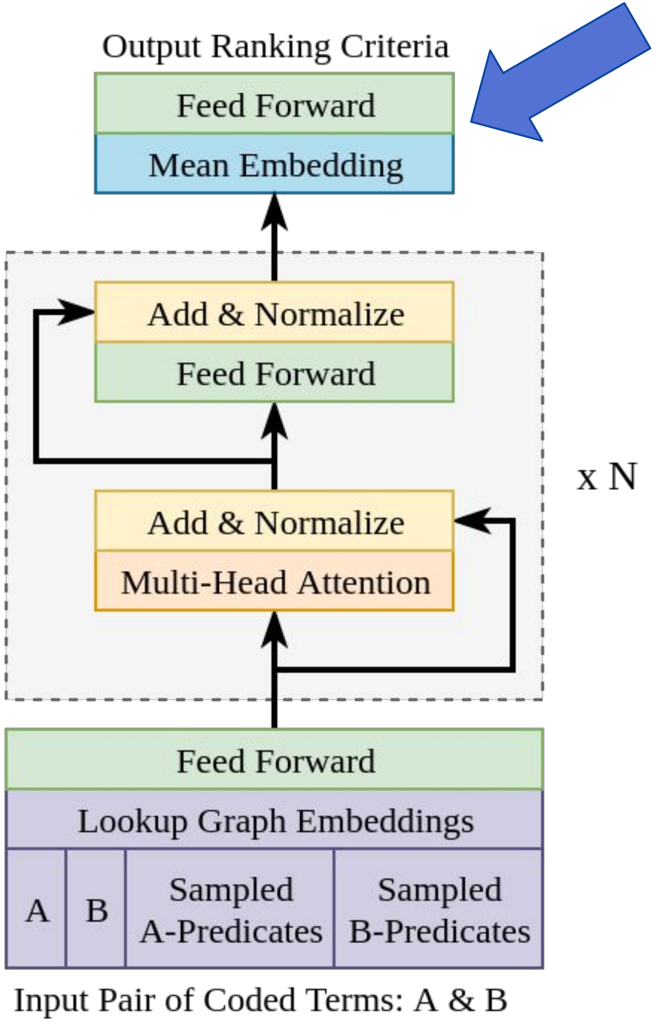
Agatha Deep Learning Model

- Goal: train a transformer encoder to accept two query terms and produce ranking criteria
- Objective: Margin Ranking Loss
- Model: Transformer Encoder
- Graph Embedding



Agatha Deep Learning Model

- Goal: train a transformer encoder to accept two query terms and produce ranking criteria
- **Objective: Margin Ranking Loss**
- Model: Transformer Encoder
- Graph Embedding



Predicate Modeling Objective

$$\mathcal{L}(\alpha, \beta) = \sum_{i=0}^{n} L\left(\mathrm{PS}_{\alpha\beta}, \mathrm{Nscr}_{\alpha\beta}^{(i)}\right) + \sum_{j=0}^{n'} L\left(\mathrm{PS}_{\alpha\beta}, \mathrm{Nswp}_{\alpha\beta}^{(j)}\right)$$

$$\text{where } L(p, n) = \max\left(0, m - \mathcal{H}(p) + \mathcal{H}(n)\right)$$

Predicate Modeling Objective

$$\mathcal{L}(\alpha, \beta) = \sum_{i=0}^{n} L\left(\mathrm{PS}_{\alpha\beta}, \mathrm{Nscr}_{\alpha\beta}^{(i)}\right) + \sum_{j=0}^{n'} L\left(\mathrm{PS}_{\alpha\beta}, \mathrm{Nswp}_{\alpha\beta}^{(j)}\right)$$

$$\text{where } L(p, n) = \max\left(0, m - \mathcal{H}(p) + \mathcal{H}(n)\right)$$

Loss associated with two connected terms

Predicate Modeling Objective

Positive Sample

Negative Sample (Scramble)

Negative Sample (Swap)

$$\mathcal{L}(\alpha, \beta) = \sum_{i=0}^{n} L\left(\mathrm{PS}_{\alpha\beta}, \mathrm{Nscr}_{\alpha\beta}^{(i)}\right) + \sum_{j=0}^{n'} L\left(\mathrm{PS}_{\alpha\beta}, \mathrm{Nswp}_{\alpha\beta}^{(j)}\right)$$

$$\text{where } L(p, n) = \max\left(0, m - \mathcal{H}(p) + \mathcal{H}(n)\right)$$

Loss associated with two connected terms

Predicate Modeling Objective

Positive Sample

Negative Sample (Scramble)

Negative Sample (Swap)

$$\mathcal{L}(\alpha, \beta) = \sum_{i=0}^{n} L\left(\mathrm{PS}_{\alpha\beta}, \mathrm{Nscr}_{\alpha\beta}^{(i)}\right) + \sum_{j=0}^{n'} L\left(\mathrm{PS}_{\alpha\beta}, \mathrm{Nswp}_{\alpha\beta}^{(j)}\right)$$

$$\text{where } L(p, n) = \max\left(0, m - \mathcal{H}(p) + \mathcal{H}(n)\right)$$

Loss associated with two connected terms

margin ranking loss

margin

Model output

Predicate Formulation

$$\mathrm{PS}_{\alpha\beta} = \left\{\alpha, \beta, \gamma_1^{(\alpha)}, \ldots, \gamma_s^{(\alpha)}, \gamma_1^{(\beta)}, \ldots, \gamma_s^{(\beta)}\right\}$$

where $\gamma_i^{(\alpha)} \sim \{\Gamma(\alpha) - \Gamma(\beta)\}$, and $\gamma_i^{(\beta)} \sim \{\Gamma(\beta) - \Gamma(\alpha)\}$

Predicate Formulation

Set containing two terms and other associated predicates

$$\mathrm{PS}_{\alpha\beta} = \left\{\alpha, \beta, \gamma_1^{(\alpha)}, \ldots, \gamma_s^{(\alpha)}, \gamma_1^{(\beta)}, \ldots, \gamma_s^{(\beta)}\right\}$$

where $\gamma_i^{(\alpha)} \sim \{\Gamma(\alpha) - \Gamma(\beta)\}$, and $\gamma_i^{(\beta)} \sim \{\Gamma(\beta) - \Gamma(\alpha)\}$

Set of predicates using a given term

Sampled predicate

Negative Samples

- Scramble (easy):

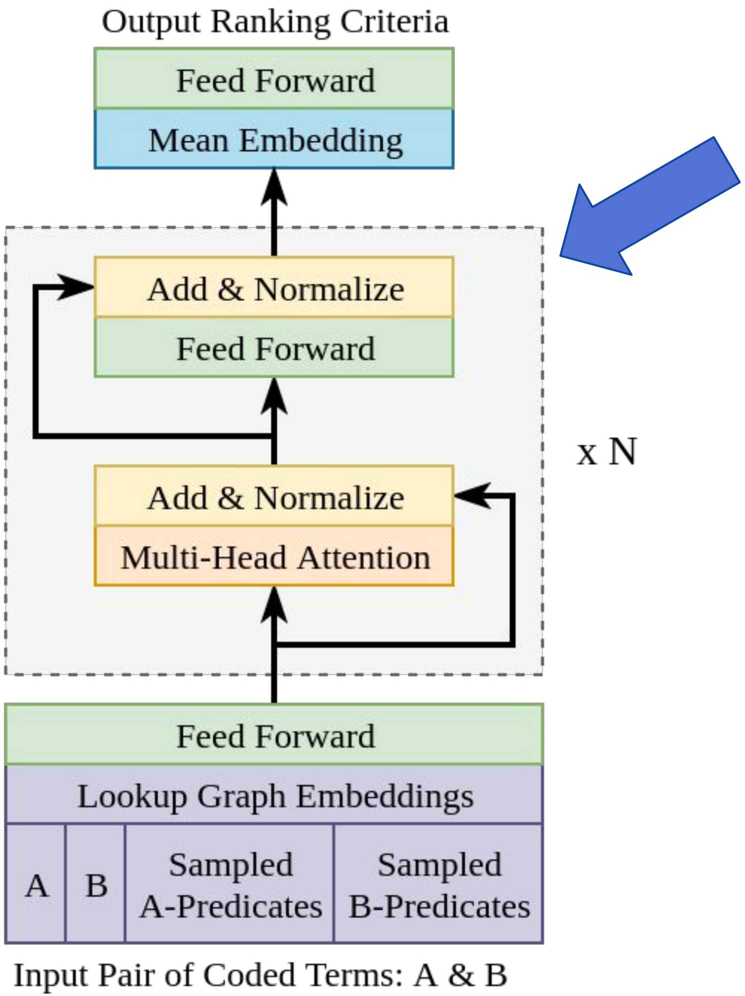
$$\begin{aligned}\mathrm{NScr}_{\alpha\beta} &= \{x, y, \gamma_1, \ldots, \gamma_{2s}\}\\ &\text{where } x, y \sim T,\\ &\text{and } \gamma_i \sim P,\\ &\text{s.t. } \Gamma(x) \cap \Gamma(y) = \emptyset\end{aligned}$$

- Swap (hard):

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

Agatha Deep Learning Model

- Goal: train a transformer encoder to accept two query terms and produce ranking criteria
- Objective: Margin Ranking Loss
- **Model: Transformer Encoder**
- Graph Embedding



Model Formalism

- Prediction Model:

$$\mathcal{H}(X) = \text{sigmoid}(\mathcal{M}W)$$

$$\mathcal{M} = \frac{1}{|X|} \sum_{x_i \in X} E_N(FF(e(x_i)))$$

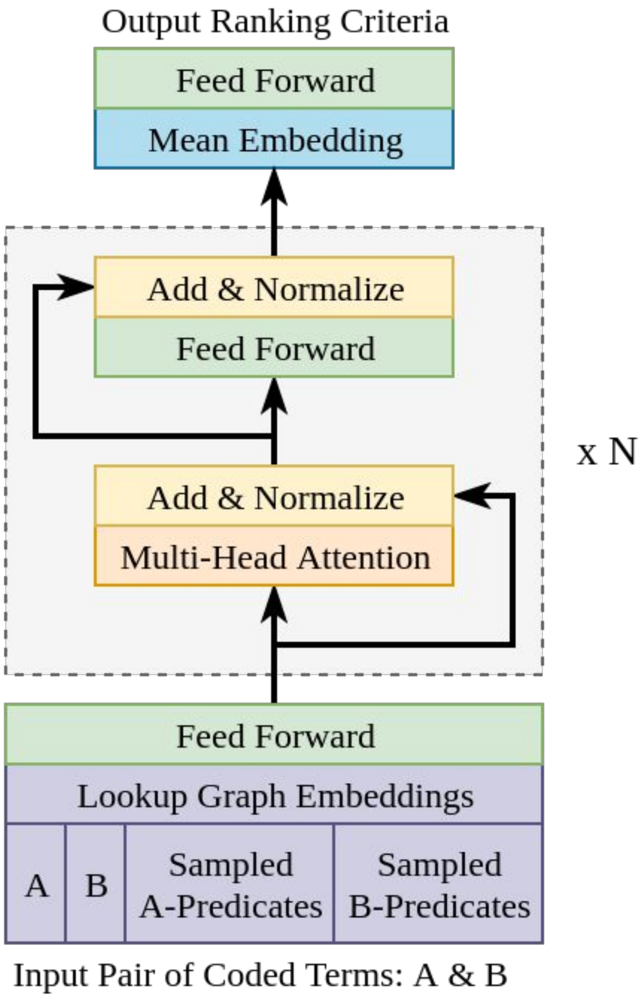
$$E_{i+1}(x) = \mathcal{E}(E_i(x)), \text{ and } E_0(x) = x$$

- Encoder Block:

$$\mathcal{E}(X) = \text{LayerNorm}(FF(\alpha) + \alpha)$$

$$\text{where } FF(Y) = \max(0, YW)\,W'$$

$$\text{and } \alpha = \text{LayerNorm}(\text{MultiHead}(X) + X)$$



Predicate Modeling

- Attention: learned weighted averages

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

Predicate Modeling

- Attention: learned weighted averages

Think: if key matches query

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

... then add in value

Predicate Modeling

- Attention: learned weighted averages

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

- Multi Head Self Attention:

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

Predicate Modeling

- Attention: learned weighted averages

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

- Multi Head Self Attention:

$$\text{MultiHead}(X) = [h_1; \ldots; h_k]W^{(4)}$$

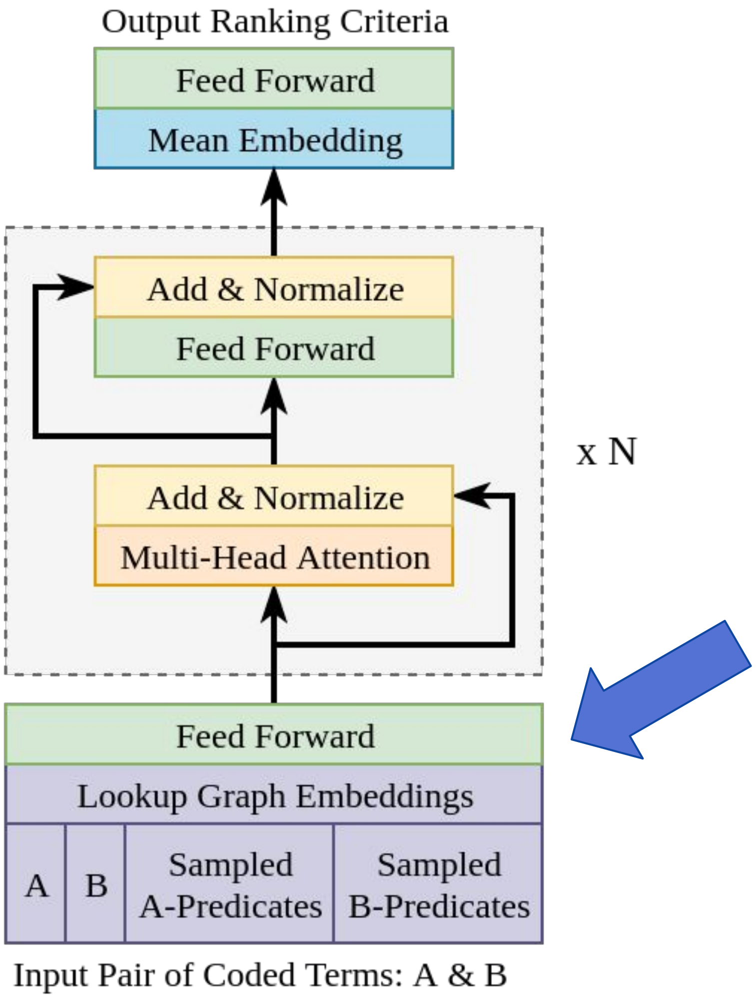
Compute multiple times and merge

$$\text{where } h_i = \text{Attention}\left(XW_i^{(1)}, XW_i^{(2)}, XW_i^{(3)}\right)$$

Derive Q, K, and V from X

Agatha Deep Learning Model

- Goal: train a transformer encoder to accept two query terms and produce ranking criteria
- Objective: Margin Ranking Loss
- Model: Transformer Encoder
- **Graph Embedding**



Graph Embedding Objective

- PyTorch-BigGraph (PTBG) distributed embedding
- Produce embedding per node in network
- Trained separate from Agatha ranking model
- Minimizes Softmax Loss:
  - Positive probability close to 1
  - All negative probabilities close to 0

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

Similarity assoc. w/ existing edge

Similarity assoc. w/ absent edge

Graph Embedding

- Similarity measure:
  - biased transformed dot product of nodes
  - includes typed translation

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

Graph Embedding

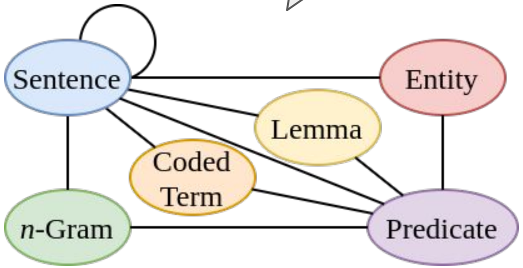
- Similarity measure:
  - biased transformed dot product of nodes
  - includes typed translation

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

Estimated sim.
btwn. i and j

Graph Embedding

Each is a type



- Similarity measure:
  - biased transformed dot product of nodes
  - includes typed translation

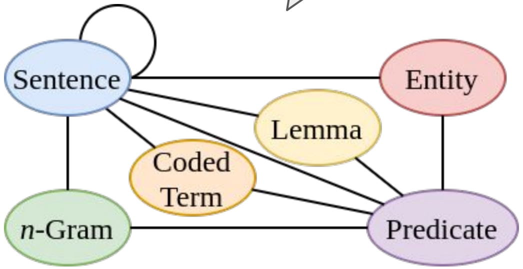
$$s(ij) = e(i)_1 + e(j)_1 + T_1^{(t_i t_j)} + \sum_{k=2}^{N} e(i)_k \left(e(j)_k + T_k^{(t_i t_j)} \right)$$

Estimated sim. btwn. i and j

T translates between types

Graph Embedding

Each is a type



- Similarity measure:
  - biased transformed dot product of nodes
  - includes typed translation

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

Estimated sim. btwn. i and j

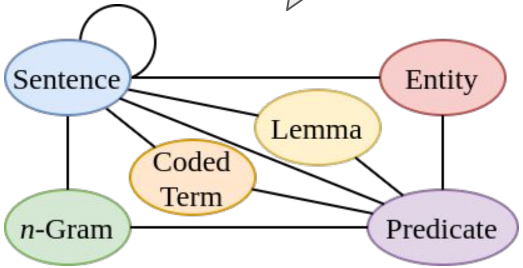
Translated dot product

T translates between types

Graph Embedding

Each is a type

- Similarity measure:
    - biased transformed dot product of nodes
    - includes typed translation



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

Estimated sim. btwn. i and j

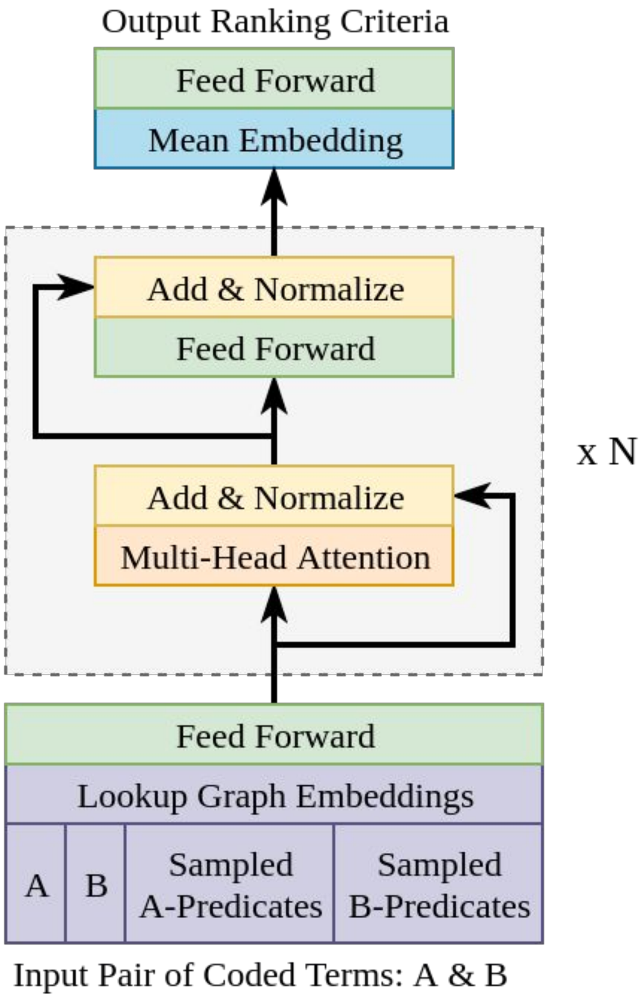
First dim. is bias

Translated dot product

T translates between types

Agatha Deep Learning Model

- Goal: train a transformer encoder to accept two query terms and produce ranking criteria
- Objective: Margin Ranking Loss
- Model: Transformer Encoder
- Graph Embedding



How to Validate

- Challenge: looking for novel information
- Existing methods:
  - Recreate 7 experiments from early 90's
  - Domain-specific statistics
  - Expert interpretation
  - Publish in medicine
- Complications:
  - Too narrow: Only specific domains
  - Too slow: Human in the loop
  - Too small: Datasets of <10 hypotheses

How to Validate

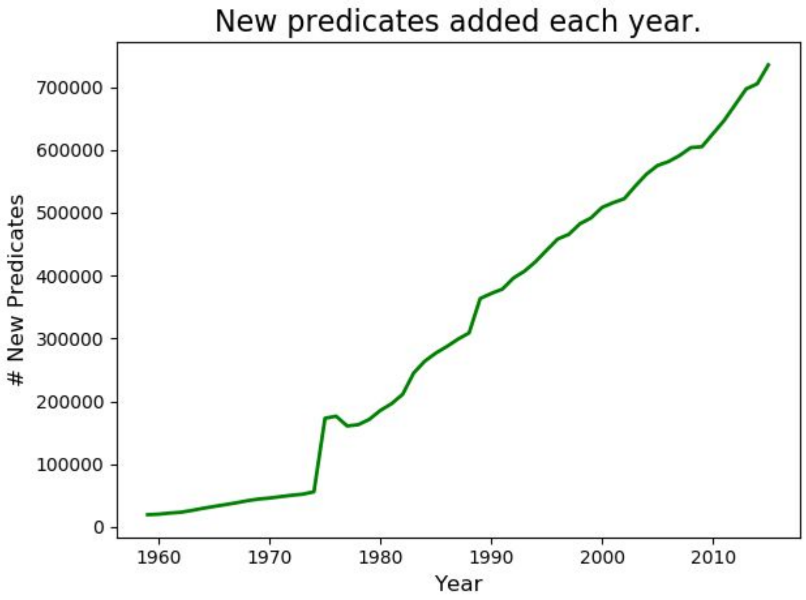
- Challenge: looking for novel information
- Existing methods:
  - Recreate 7 experiments from early 90's
  - Domain-specific statistics
  - Expert interpretation
  - Publish in medicine
- Complications:
  - Too narrow: Only specific domains
  - Too slow: Human in the loop
  - Too small: Datasets of <10 hypotheses

Large-scale validation of hypothesis generation systems via candidate ranking
Sybrandt, Shtutman, Safro

BigData'18

Validation via Ranking

- Drug discovery is ranking
- Positive test set
  - Recently introduced predicates
- Negative test set
  - Never introduced predicates
- Perform temporal holdout

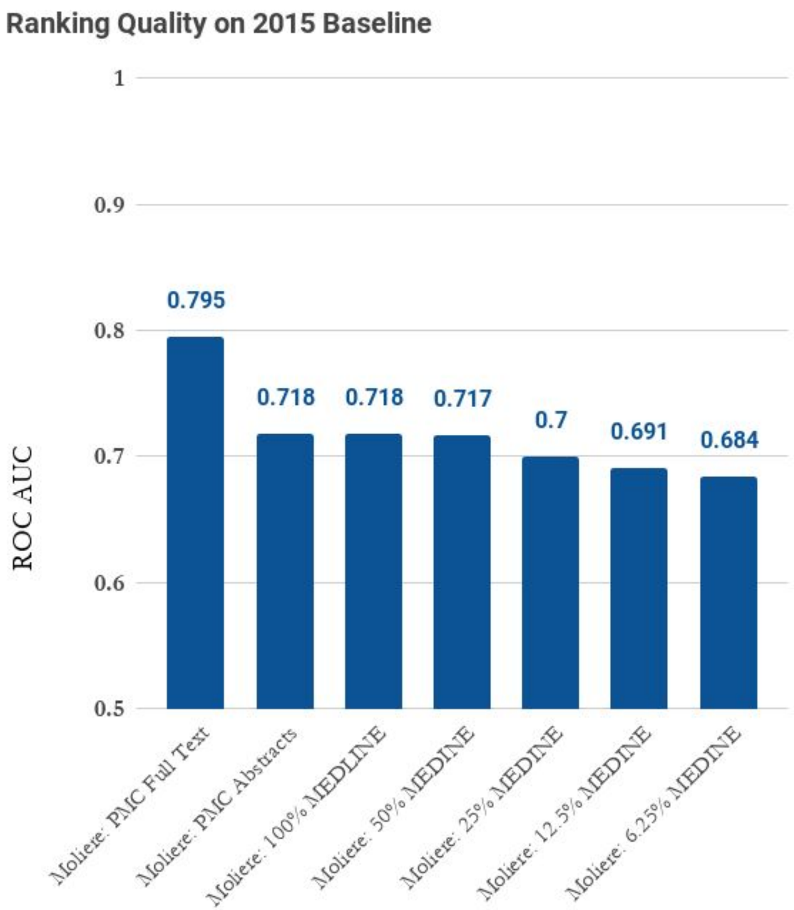


Moliere Baseline

- Train on data before 2015
- Explore different training datasets
- Rank recent predicates with heuristic criteria
- Found that full text papers outperform abstracts by 10%

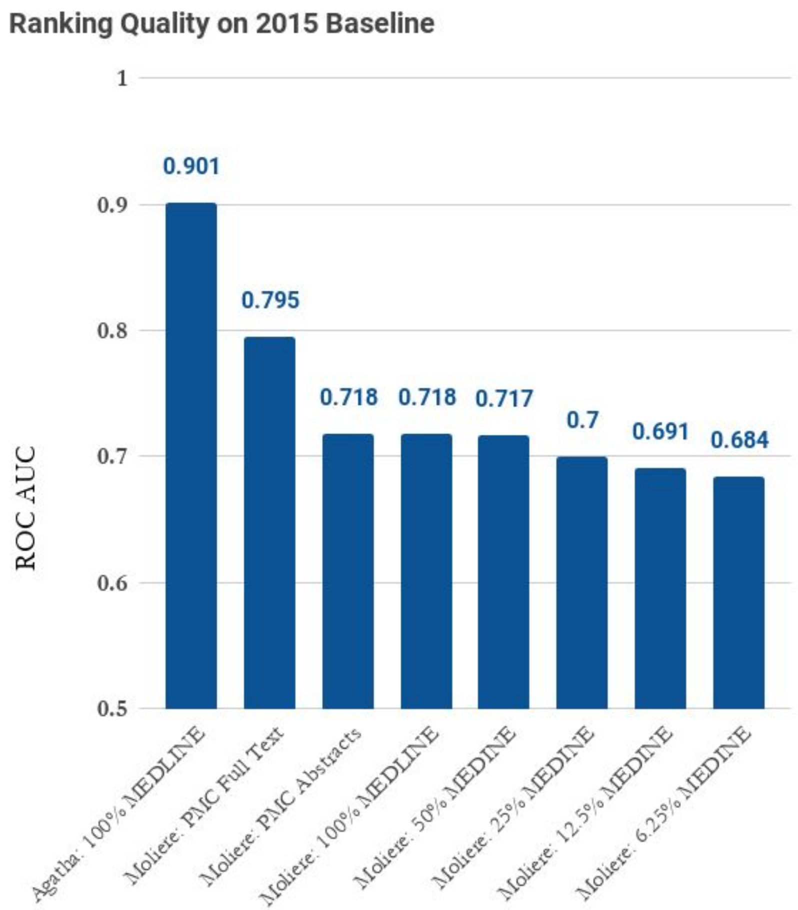
Are abstracts enough for hypothesis generation?
Sybrandt, Carrabba, Herzog, Safro

BigData'18



Agatha performance on Moliere Benchmark

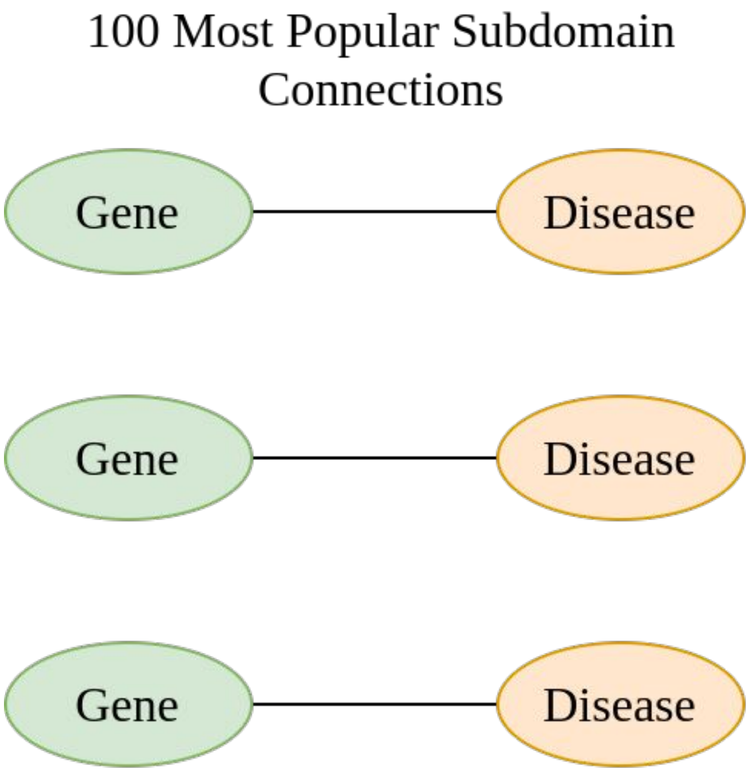
- Trained on same holdout as Moliere experiments (2015)
  - Used only abstracts
- Same set of predicates
- 100's queries per minute

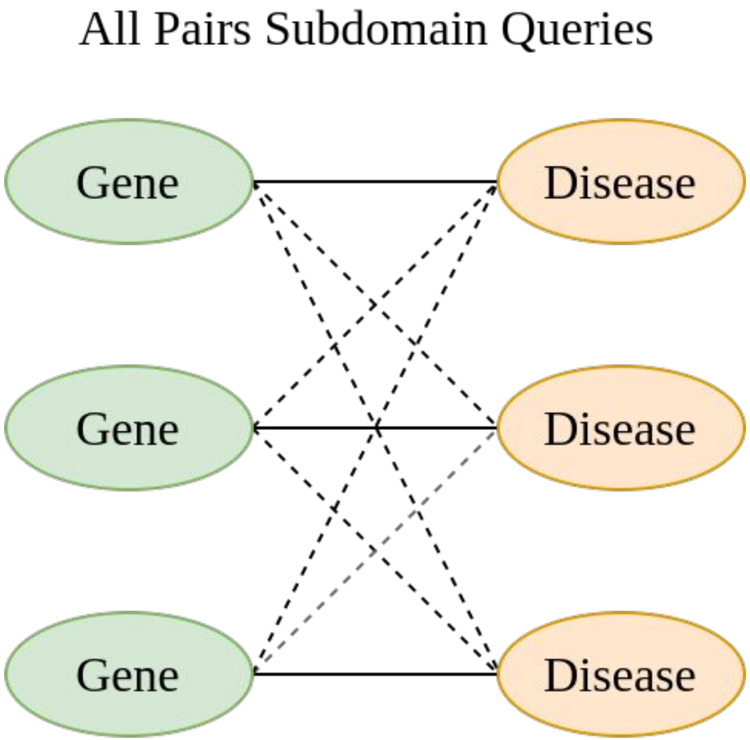


Beyond the Moliere Benchmark

- Moliere benchmark had significant issues
  - Balanced classes
  - Non-representative negative samples
- New validation task
  - Subdomain all-pairs recommendation
- Procedure:
  - Identify popular types of predicates
  - Find 100 most popular new findings within each predicate type
  - Predict all pairs of queries within popular entities
  - Rank
  - Compute recommendation system metrics

All Pairs Recommendation Example





Gene to Cell Function

Top 100 predicates of this type.

- Area under curves:
  - PR: 0.44
  - ROC: 0.62
- Top ranked predicate is positive
- Half of the top-10 are positive
- Each one-to-many query on average:
  - Positive result within first two

Gene to Neoplastic Process

Top 100 predicates of this type.

- Area under curves:
  - PR: 0.34
  - ROC: 0.65
- Second ranked predicate is positive
- Half of the top-10 are positive
- Each one-to-many query on average:
  - Positive result within first two

Try it yourself

```python
# Its now incredibly easy to run your own
# Hypothesis Generation Queries!
import torch
agatha_data_dir = "..."
model = torch.load(f"{agatha_data_dir}/model.pt")
model.set_data_root(agatha_data_dir)
model.init()
# Now you can rank arbitrary UMLS term pairs!
# Keywords: Cancer(C0006826), Tobacco(C0040329)
model.predict_from_terms([("C0006826", "C0040329")])
>>> [0.78358984]
# Check us out at github.com/jsybrandt/agatha
```

In Summary

- Agatha:
  - Deep learning model built on graph embeddings and the transformer
  - Ranks user queries
  - High quality recommendation
- Learn more:
  - Email: jsybran@clemson.edu
  - Project: sybrandt.com/2020/agatha
  - Github: github.com/jsybrandt/agatha