INSTITUTE FOR APPLIED COMPUTATIONAL SCIENCE AT HARVARD UNIVERSITY

Harvard John A. Paulson School of Engineering and Applied Sciences

KENSHO

Entity Disambiguation via “Congruence”

AC297R Capstone

Will Seaton, Johannes Kolberg, Ruochen Zhao, Hardik Gupta

KENSHO

Kensho builds amazing tools to help people make fact-based decisions.

Founded in 2013, Kensho is headquartered in Cambridge, Massachusetts with offices in Washington, D.C., New York City, and Los Angeles. In 2018, S&P Global acquired Kensho for $550M, the largest AI acquisition in history to date.

AI 100

FINTECH 50

FORTUNE

Our Goal:

Improve Named Entity Disambiguation (NED) by comparing the relatedness of entities within the same sentence using “congruence”

Named Entity Disambiguation

“Chris Tanner met Pavlos in Harvard Yard.”

Named Entity Disambiguation

"Chris Tanner met Pavlos in Harvard Yard."

Step One: Named Entity Recognition (NER) identifies entities or objects in strings of text. SpaCy is a popular package for this.

Named Entity Disambiguation

"Chris Tanner met Pavlos in Harvard Yard."

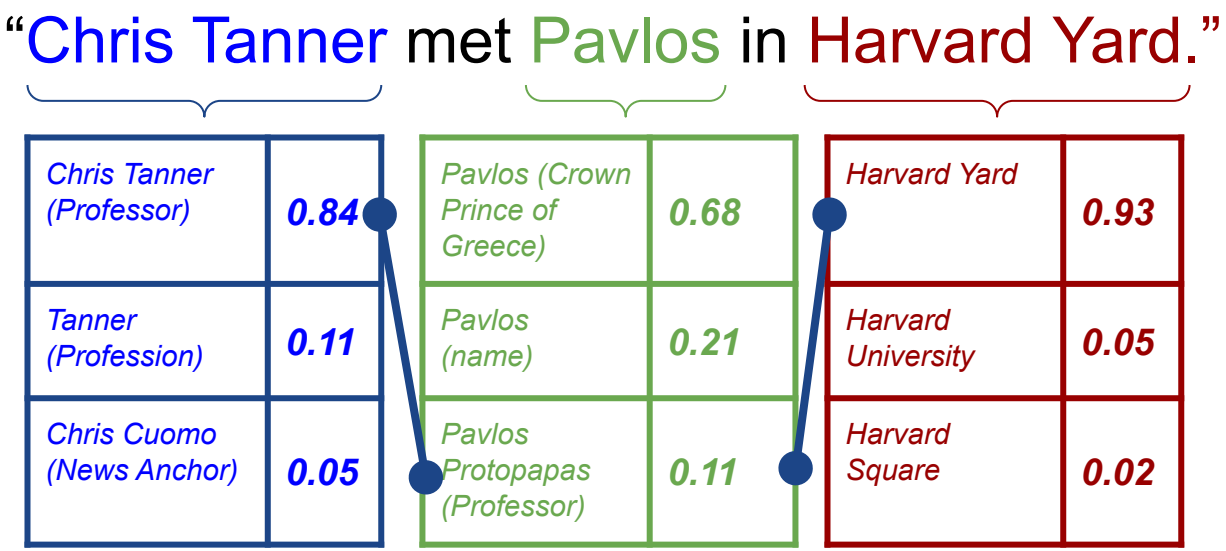
Chris Tanner (Professor)	***0.84***
Tanner (Profession)	***0.11***
Chris Cuomo (News Anchor)	***0.05***

Pavlos (Crown Prince of Greece)	***0.68***
Pavlos (name)	***0.21***
Pavlos Protopapas (Professor)	***0.11***

Harvard Yard	***0.93***
Harvard University	***0.05***
Harvard Square	***0.02***

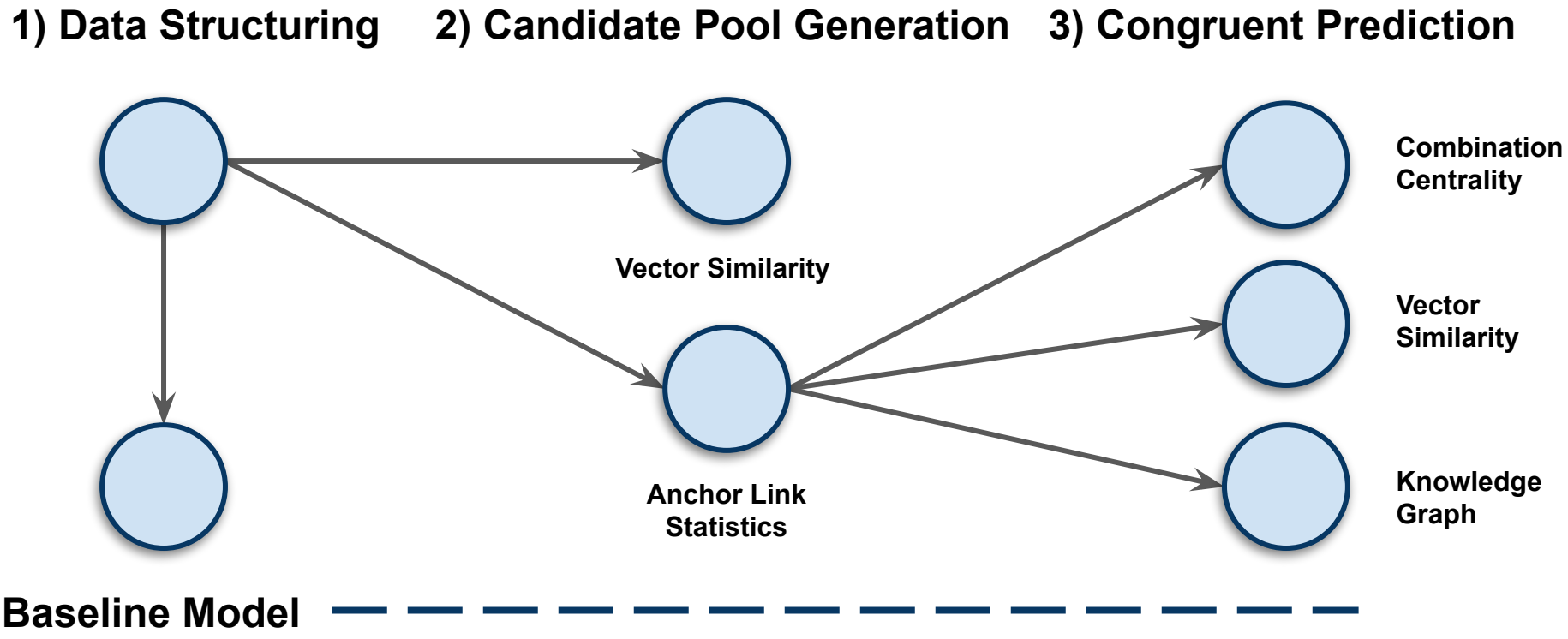
Step Two: Each named entity has a pool of candidate Wikipedia pages that it might be referring to and that can be generated in different ways.

Named Entity Disambiguation



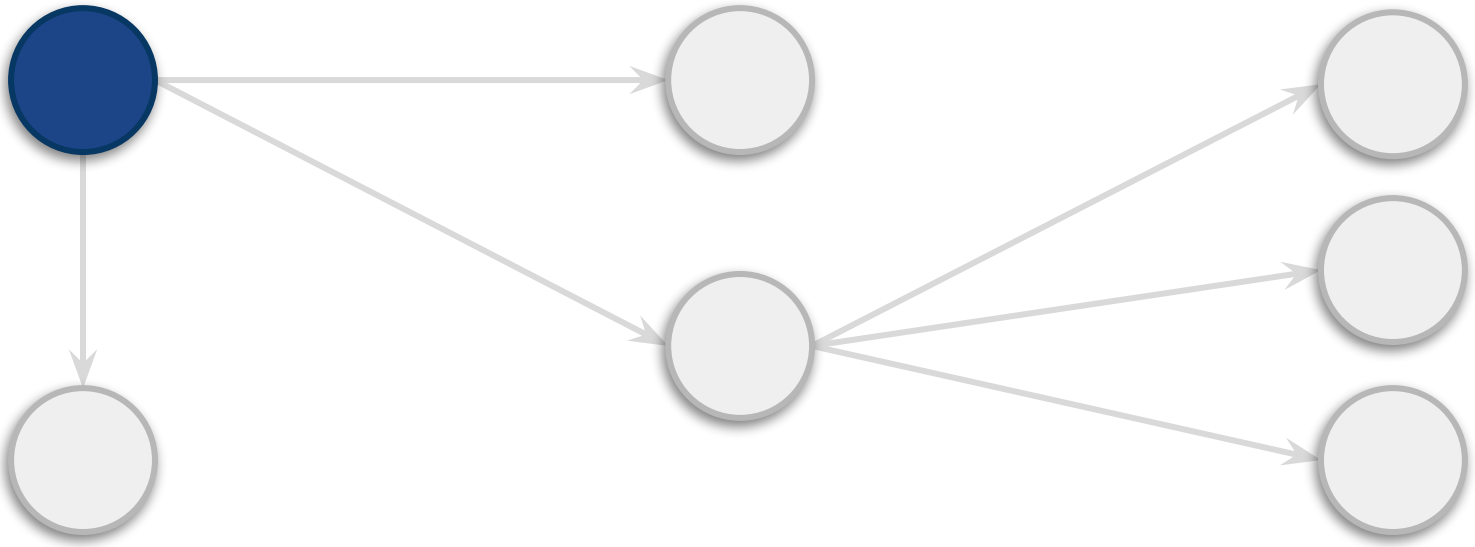
Step Three: Use each candidate's relation to each other ("congruence") to re-order the candidate pools and make a better prediction.

Research Paths



Research Paths

1) Data Structuring 2) Candidate Pool Generation 3) Congruent Prediction



Baseline Model

Data Structuring

Testing Data: AIDA CoNLL-YAGO

- Provides named entity recognition
- Manual assignment of “true” Wikipedia page

Wikipedia Statistics:

Kensho-Derived Wikimedia Data

- Open-source dataset structured for NLP tasks

Pre-Trained Embeddings:

Wikipedia2Vec

- Open-source package and pre-trained embeddings developed by Studio Ousia

GraphVite

- Pre-trained embeddings over Wikipedia knowledge graph

Data Structuring: AIDA CoNLL-YAGO

~4,400 sentences, ~21,000 named entities

	mention	full_mention	wikipedia_URL	wikipedia_page_ID	wikipedia_title	sentence_id	doc_id	congruent_mentions
0	B	EU	None	None	None	0	0	[EU, German, British]
1	B	German	http://en.wikipedia.org/wiki/Germany	11867	Germany	0	0	[EU, German, British]
2	B	British	http://en.wikipedia.org /wiki/United_Kingdom	31717	United Kingdom	0	0	[EU, German, British]
3	B	Peter Blackburn	None	None	None	1	0	[Peter Blackburn, BRUSSELS, European Commissio...
4	I	Peter Blackburn	None	None	None	1	0	[Peter Blackburn, BRUSSELS, European Commissio...

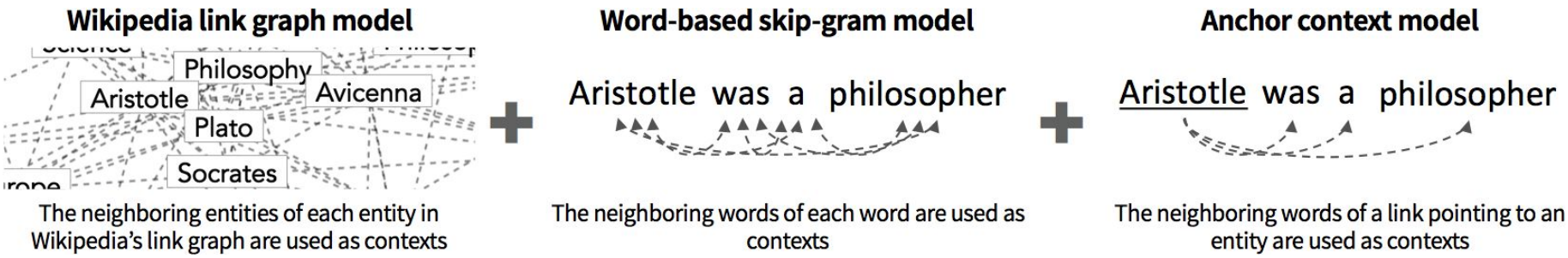
Input

Response

Contextual Entities

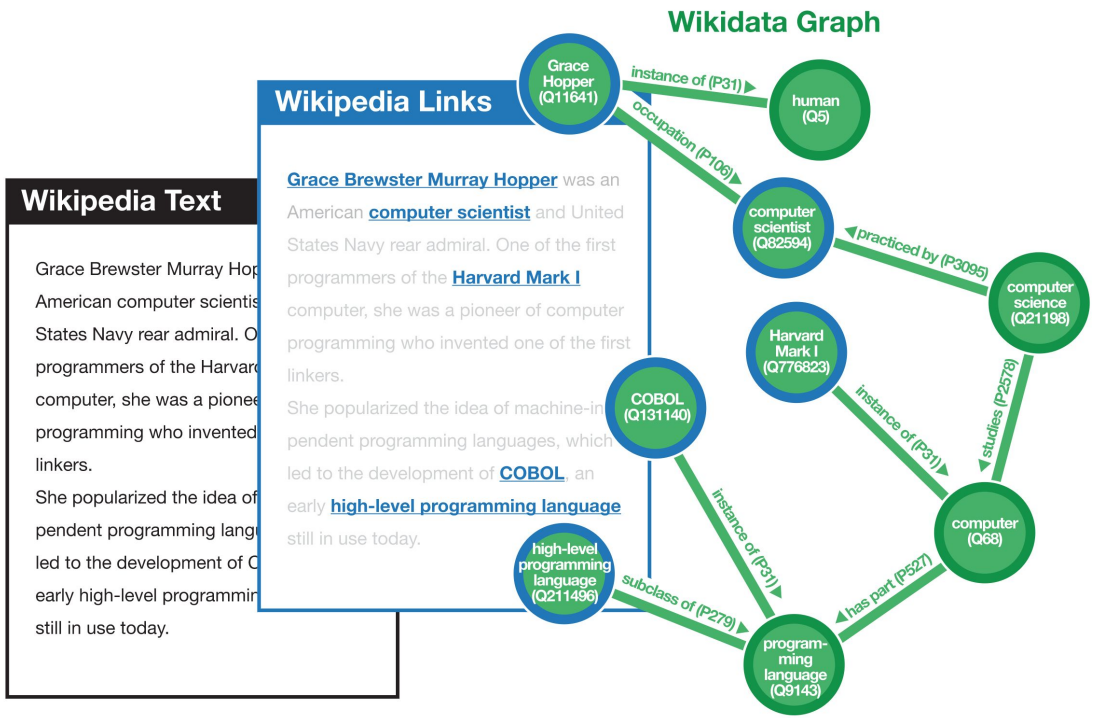
Data Structuring: Wikipedia2Vec

Innovative embeddings represent **entire Wikipedia page**, combining three informational sources including graph relations that avoid traditional computational challenges. Embeddings provided in 100d, 300d and 500d sizes.



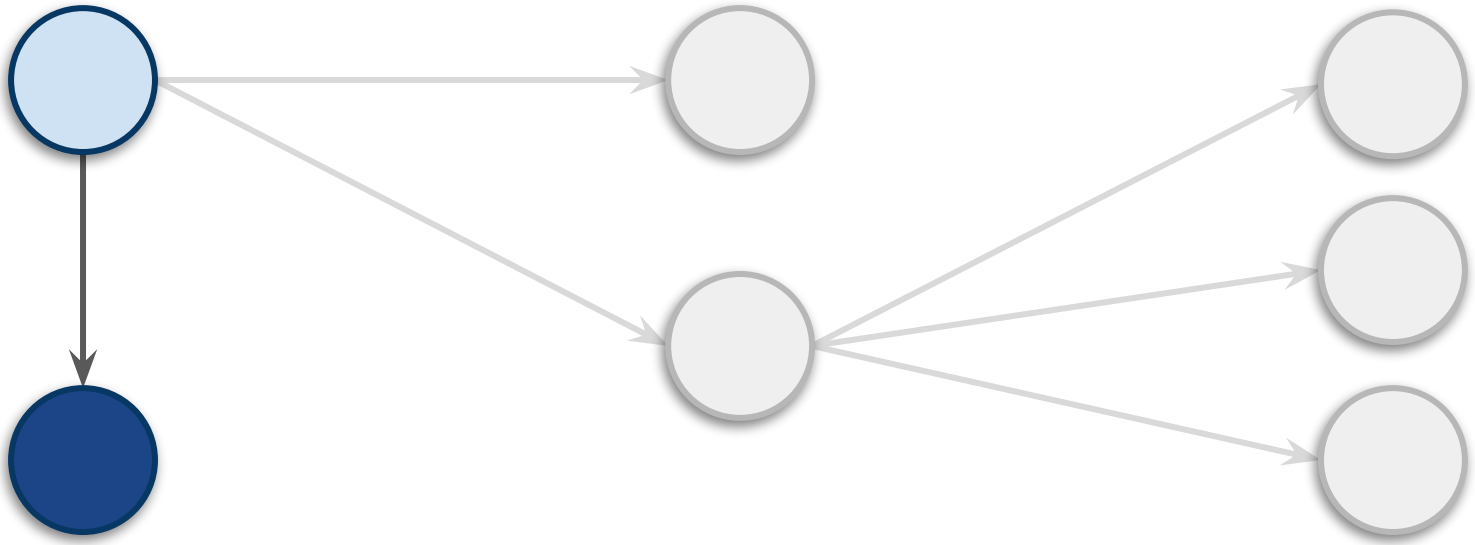
Data Structuring: Kensho-Derived Wikimedia

In February 2020, Kensho released an NLP-structured version of open source Wikimedia data that provides us with “anchor links” and a knowledge graph of page relationships.



Research Paths

1) Data Structuring 2) Candidate Pool Generation 3) Congruent Prediction



Baseline Model

Baseline Model

Goal: What is possible using existing open source packages? Is custom work really necessary?

Utilize Wikipedia2vec API and its function **get_entity** to query full mention text

wikipedia2vec.get_entity("Chris Tanner")

Returns: <Entity Chris Tanner> or None

****

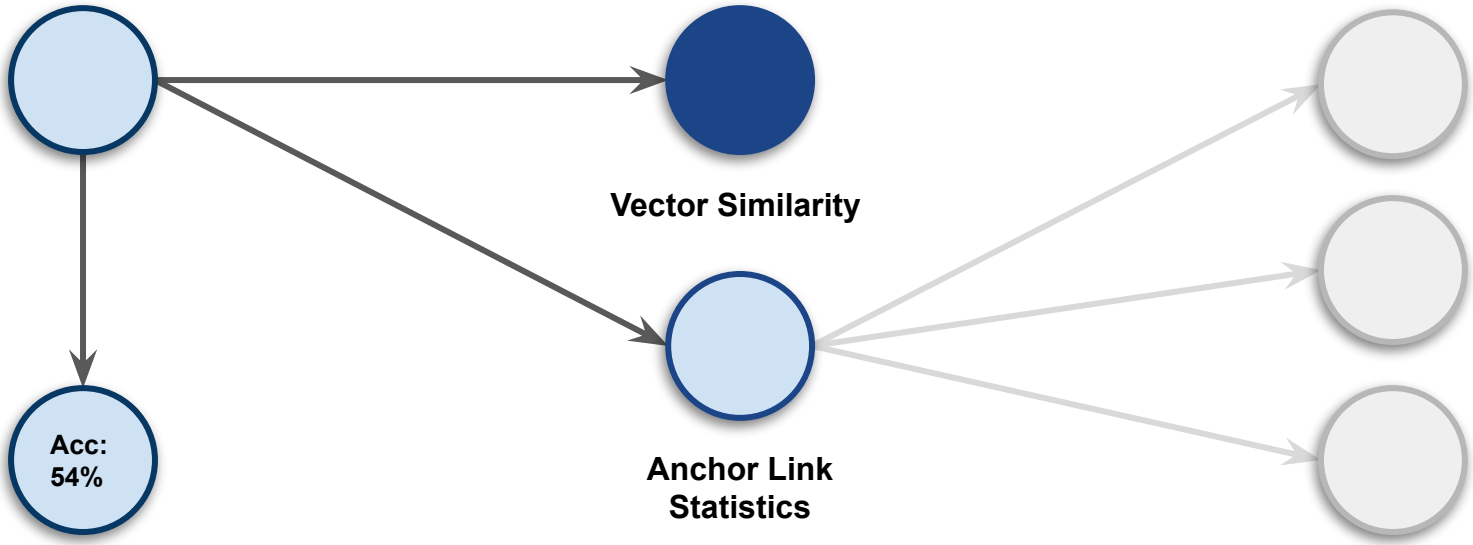
Query Success: 60%
Predictive Accuracy: 54%

Disambiguation Accuracy =

$$\frac{\text{\# Correctly Identified Wikipedia Page IDs}}{\text{Total Test Entities with Known True Value}}$$

Research Paths

1) Data Structuring **2) Candidate Pool Generation** **3) Congruent Prediction**



Baseline Model

Candidate Pool Generation: Vector Similarity

Method:
Utilize Wikipedia2vec API and its function **most_similar_by_vector** to return list of similar embeddings

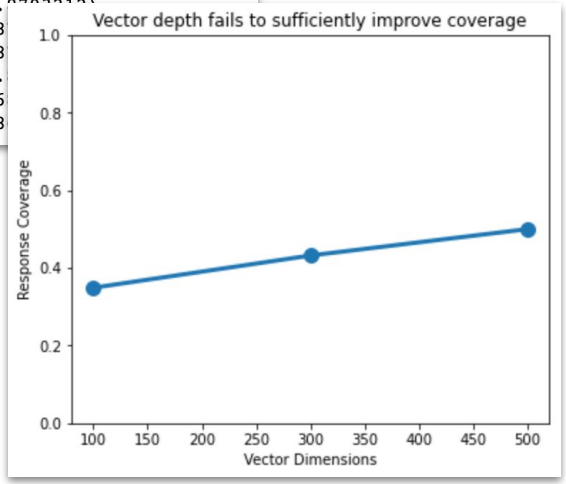
Challenges:

- Multiple words require mean vector
- Inflexible API
- Comparing text vector with minimal information to complex entity vector

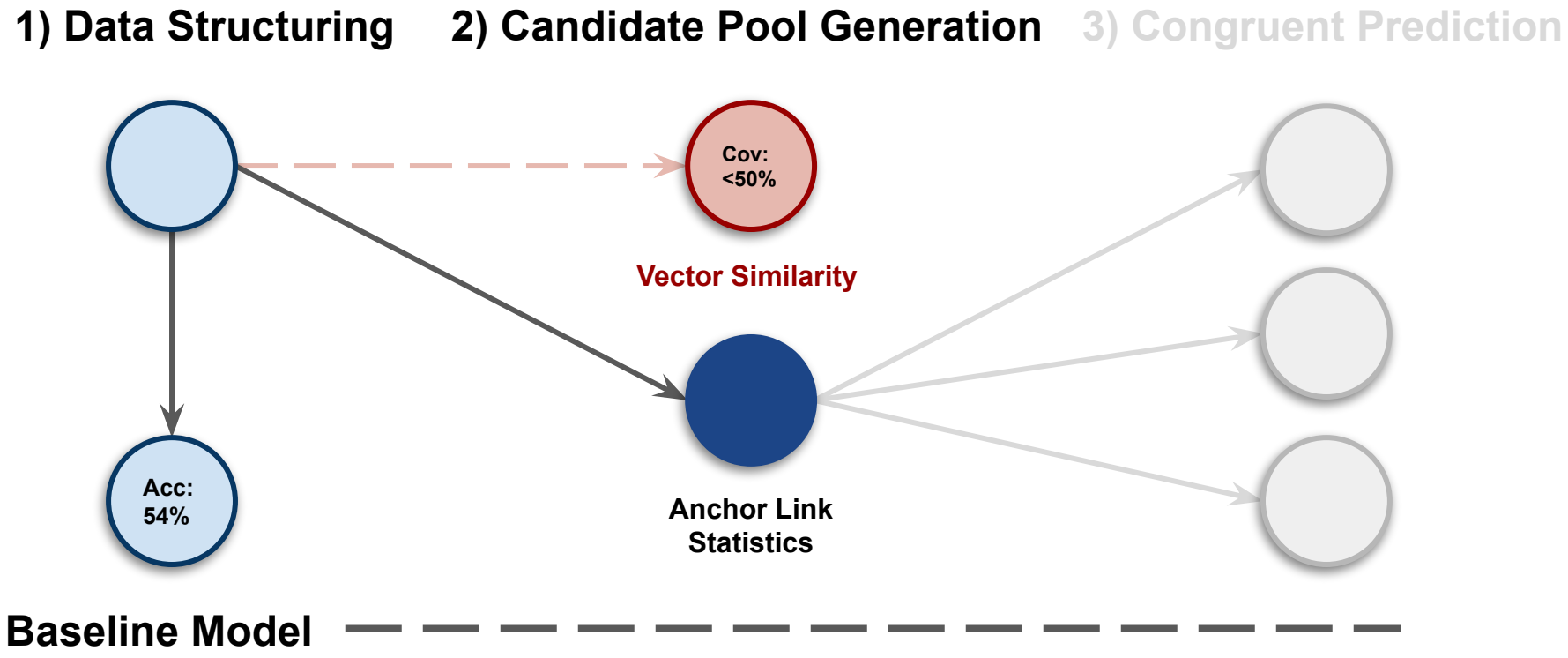
Insufficient "coverage" of correct answers in candidate pools to continue.

```
In [317]: # Return most similar entities
          w2v.most_similar_by_vector(
              (w2v.get_word_vector("chris")\
               + w2v.get_word_vector("tanner"))/2,
              10)

Out[317]: [(<Word chris>, 0.9109982),
           (<Word matt>, 0.8974283),
           (<Word craig>, 0.892782),
           (<Word todd>, 0.8830913),
           (<Word tanner>, 0.
           (<Word jason>, 0.8
           (<Word brett>, 0.8
           (<Word jeremy>, 0.
           (<Word brad>, 0.86
           (<Word scott>, 0.8
```



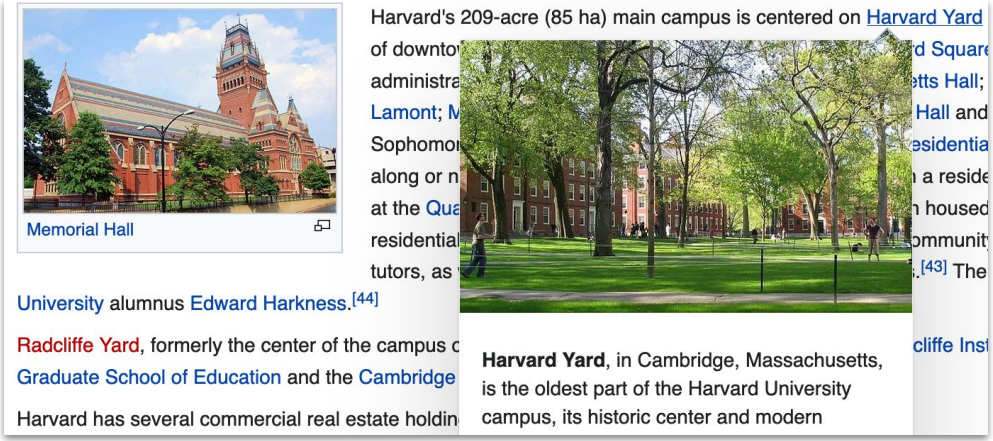
Research Paths



Candidate Pool Generation: Anchor Link Statistics

“Anchor Link” is a text string within a Wikipedia page that hyperlinks to a different Wikipedia page.

Kensho provides frequency of links and linked page popularity, both ways of generating a pool.



norm_anchor_text	target_page_title	target_page_views	anchor_target_count	prior_target_count	prior_page_views
harvard university	Harvard_University	58385	24229	0.995480	0.370821
harvard university	Harvard_Crimson_men's_ice_hockey	447	22	0.000904	0.002839
harvard university	Harvard_Crimson_football	1013	14	0.000575	0.006434
harvard university	Harvard_Law_School	7410	12	0.000493	0.047063
harvard university	Harvard_Crimson	1504	12	0.000493	0.009552

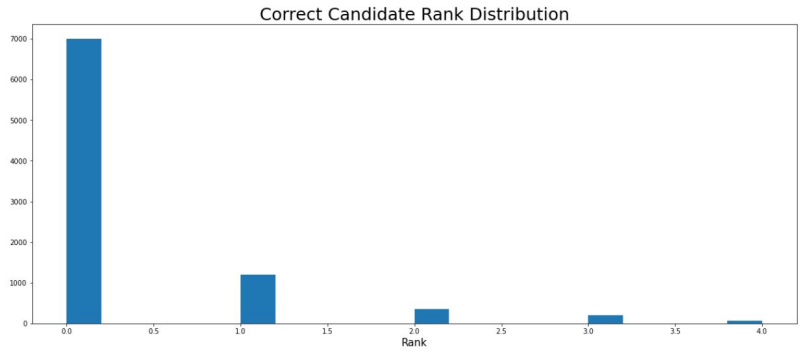
Candidate Pool Generation: Anchor Link Statistics

Anchor Link Frequency

****

Coverage: 90.26%

Predictive Accuracy: 72.03%

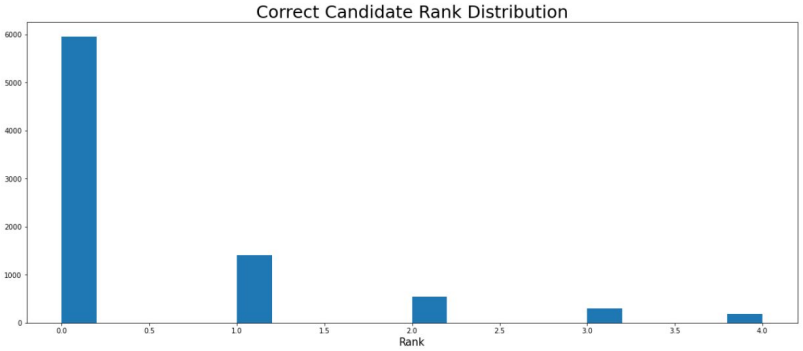


Linked Page Popularity

****

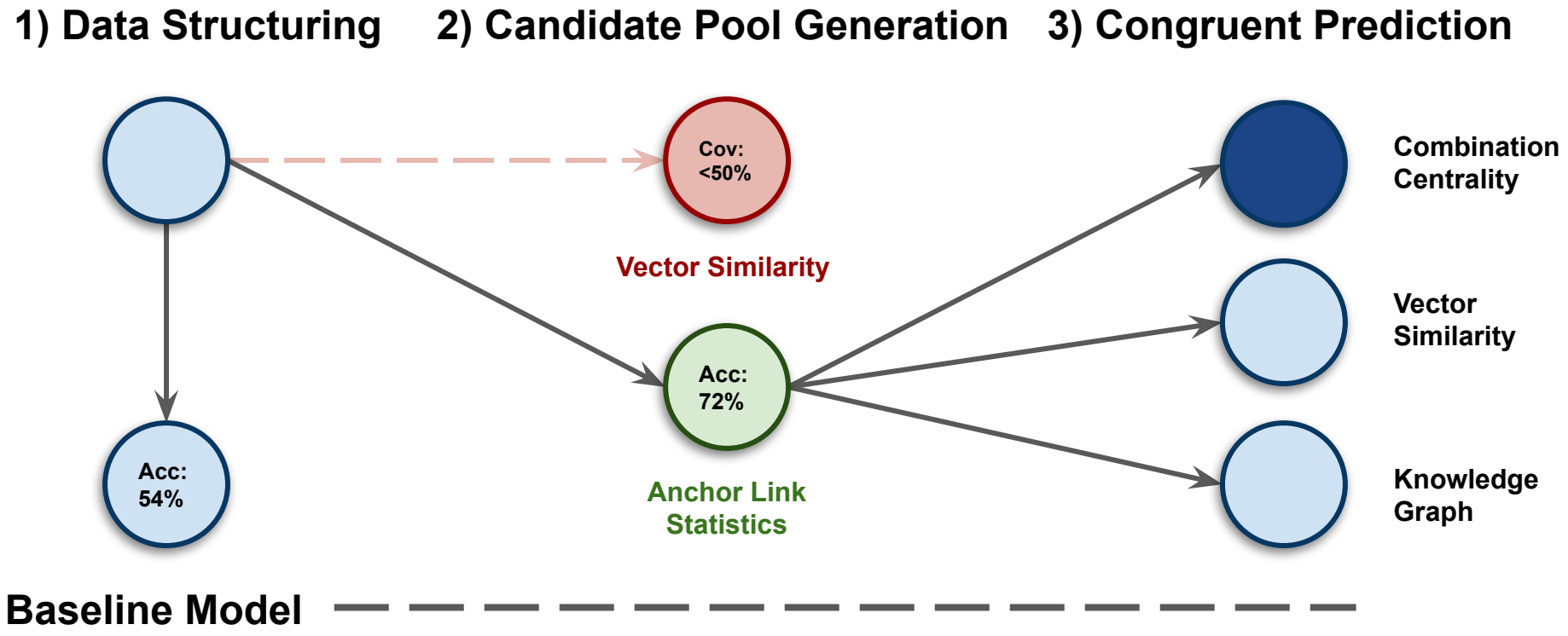
Coverage: 85.99%

Predictive Accuracy: 61.87%



Prior Confidence	
Frequency	**Popularity**
88.2%	65.0%
17.7%	16.6%
7.6%	9.8%
4.2%	6.2%
2.4%	5.8%

Research Paths



Congruent Prediction: Incorporating Priors

"Chris Tanner met Pavlos in Harvard Yard."

Chris Tanner (Professor)	***0.84***
Tanner (Profession)	***0.11***
Chris Cuomo (News Anchor)	***0.05***

Pavlos (Crown Prince of Greece)	***0.68***
Pavlos (name)	***0.21***
Pavlos Protopapas (Professor)	***0.11***

Harvard Yard	***0.93***
Harvard University	***0.05***
Harvard Square	***0.02***

Congruent Prediction: Incorporating Priors

"Chris Tanner met Pavlos in Harvard Yard."

Chris Tanner (Professor)	***0.84***	*Pavlos (Crown Prince of Greece)*	***0.68***	*Harvard Yard*	***0.93***
Tanner (Profession)	***0.11***	*Pavlos (name)*	***0.21***	*Harvard University*	***0.05***
Chris Cuomo (News Anchor)	***0.05***	*Pavlos Protopapas (Professor)*	***0.11***	*Harvard Square*	***0.02***

Indirectly Incorporate:

Priors define cut-off for candidate pool, then congruence measure makes selection

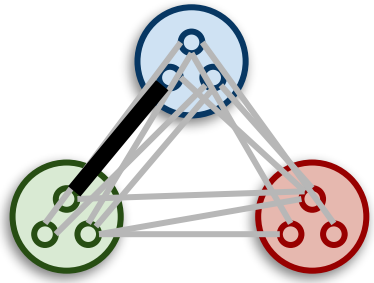
Congruent Prediction: Incorporating Priors

"Chris Tanner met Pavlos in Harvard Yard."

Chris Tanner (Professor)	***0.84***
Tanner (Profession)	***0.11***
Chris Cuomo (News Anchor)	***0.05***

Pavlos (Crown Prince of Greece)	***0.68***
Pavlos (name)	***0.21***
Pavlos Protopapas (Professor)	***0.11***

Harvard Yard	***0.93***
Harvard University	***0.05***
Harvard Square	***0.02***



Congruence * Mean(**0.84**, **0.21**)

Directly Incorporate:

Discount congruence measure with candidate prior confidence when making selection

Congruent Prediction: Combination Centrality

"Chris Tanner met Pavlos in Harvard Yard."

Method: Calculate entity congruence by:

1. Identifying all unique combinations of candidates
2. Calculate each combination's "centroid vector"
3. Calculating the average distance from each entity to its centroid vector
4. Selecting the combination with the lowest average distance

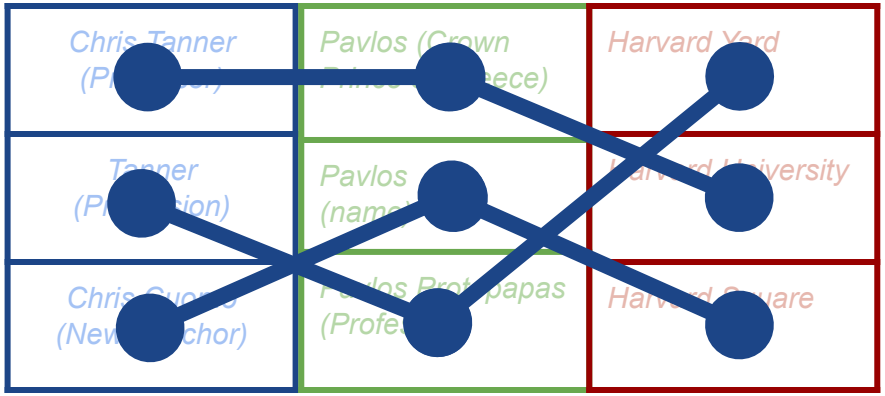
Congruent Prediction: Combination Centrality

"Chris Tanner met Pavlos in Harvard Yard."

Method: Select unique candidate combination with lowest mean distance to a representative "centroid vector"

Congruent Prediction: Combination Centrality

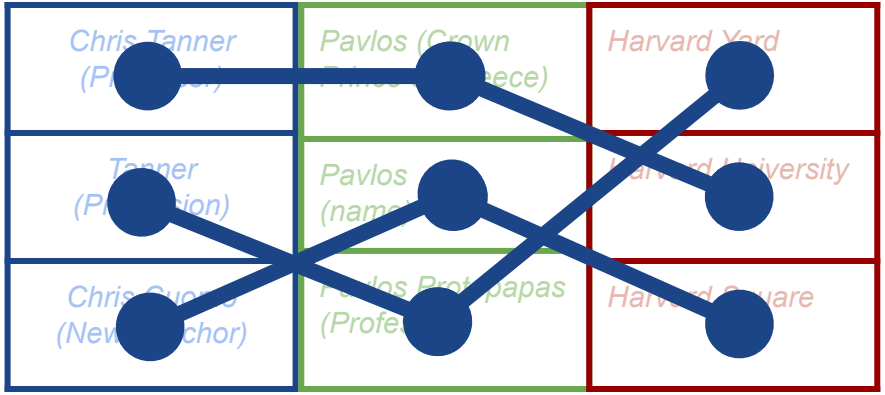
1. Identifying all unique combinations of candidates

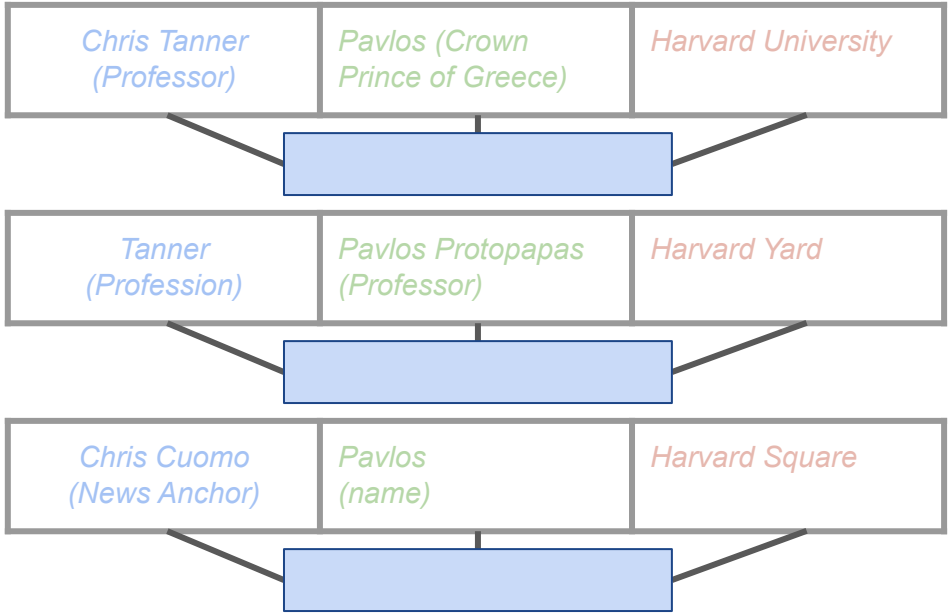


Chris Tanner (Professor)	Pavlos (Crown Prince of Greece)	Harvard University
Tanner (Profession)	Pavlos Protopapas (Professor)	Harvard Yard
Chris Cuomo (News Anchor)	Pavlos (name)	Harvard Square

Congruent Prediction: Combination Centrality

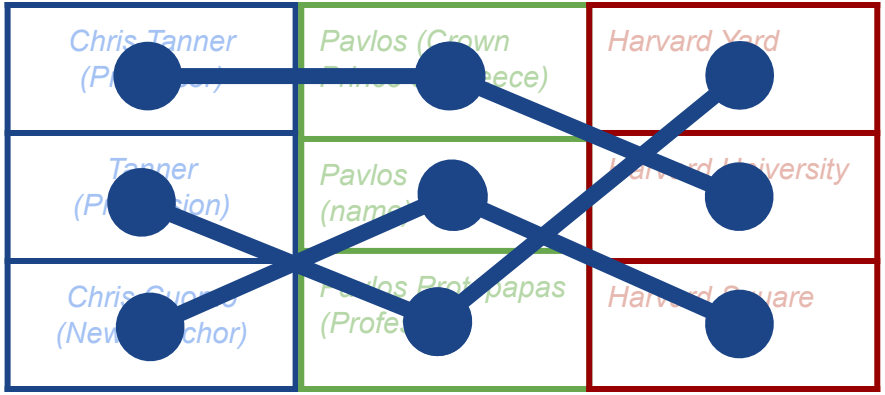
2. Calculate each combination's "centroid vector"

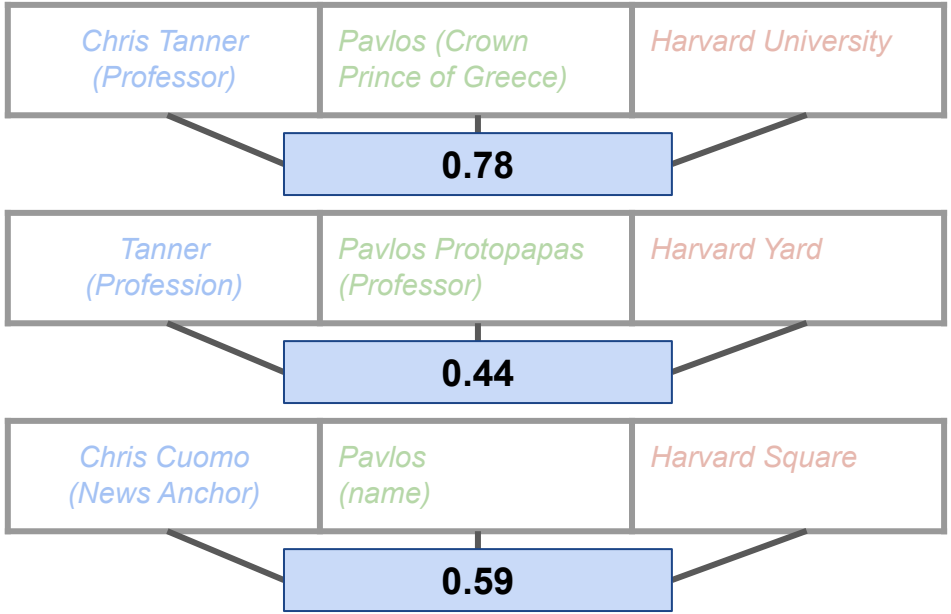




Congruent Prediction: Combination Centrality

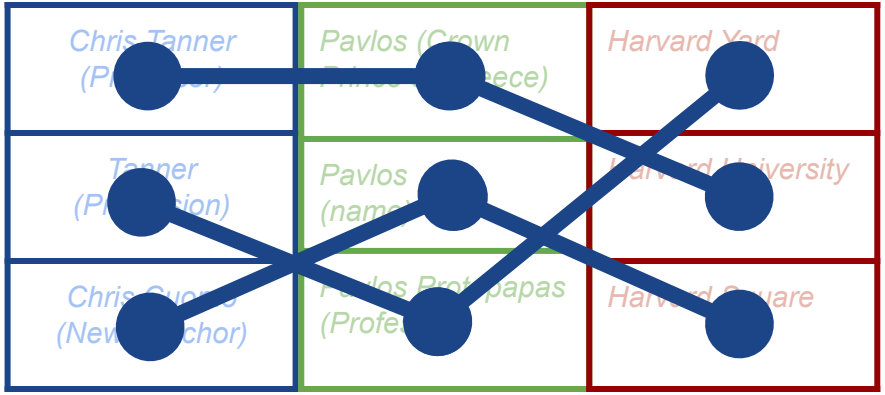
3. Calculating the average distance from each entity to its centroid vector

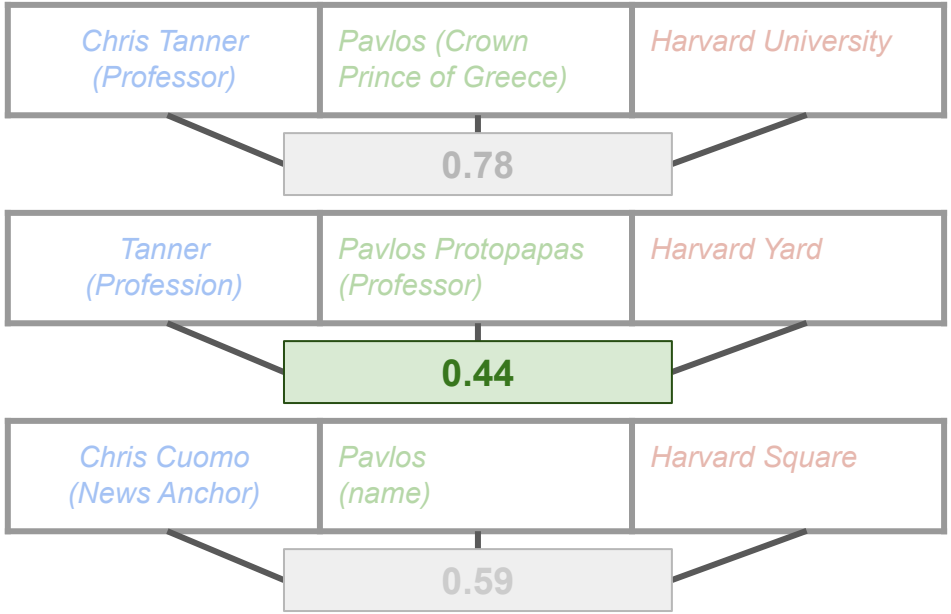




Congruent Prediction: Combination Centrality

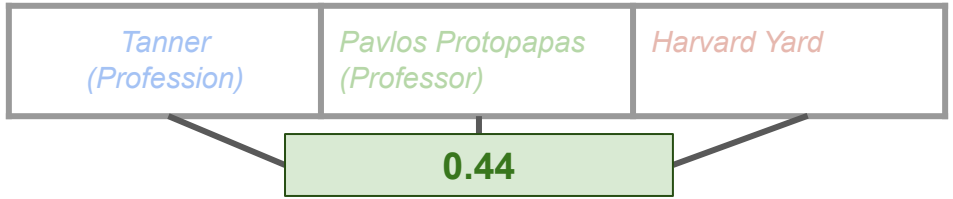
4. Selecting the combination with the lowest average distance





Congruent Prediction: Combination Centrality

"Chris Tanner met Pavlos in Harvard Yard."



Challenges:

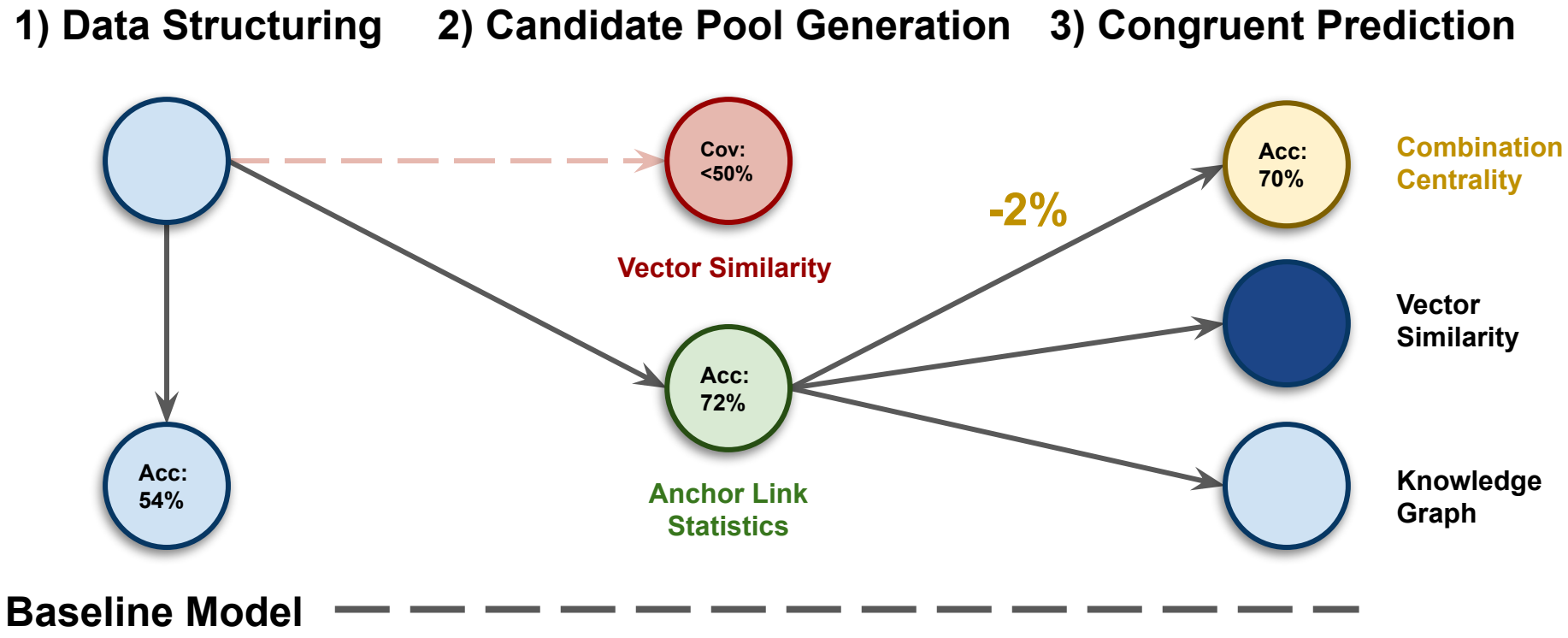
- Centroid not a sufficiently qualified representation of each combination
- Combinatorial computation prevents use with sentences with >5 named entities

Combinatorial implementation refined design but failed to achieve accuracy improvement

****

Predictive Accuracy: 70%

Research Paths



Congruent Prediction: Vector Similarity

"Chris Tanner met Pavlos in Harvard Yard."

Method: Calculate entity congruence by:

1. Calculating pairwise distance between every candidate for every named entity
2. Identify most similar pair and save those candidates as predictions
3. Iteratively find next most similar candidate(s) until all named entities predicted

Congruent Prediction: Vector Similarity

"Chris Tanner met Pavlos in Harvard Yard."

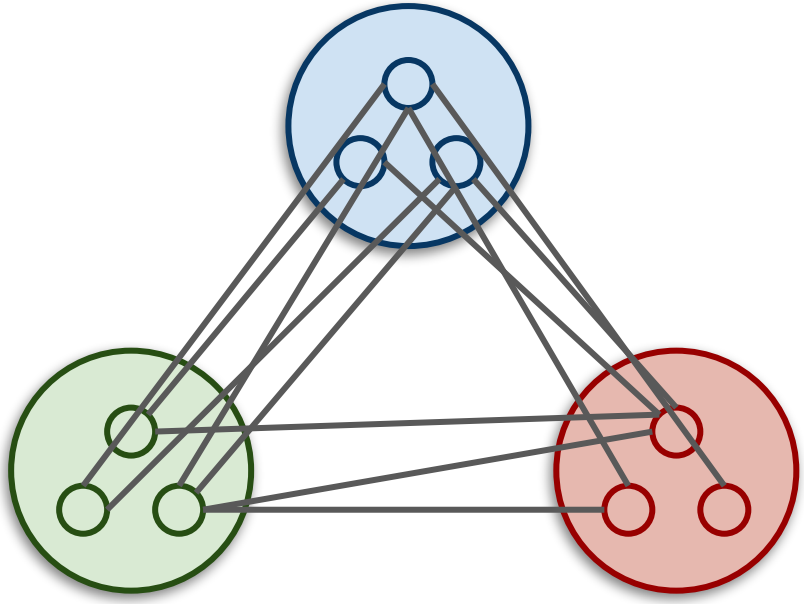
Method: Select most similar entity embeddings,

one named entity at a time

Congruent Prediction: Vector Similarity

"Chris Tanner met Pavlos in Harvard Yard."

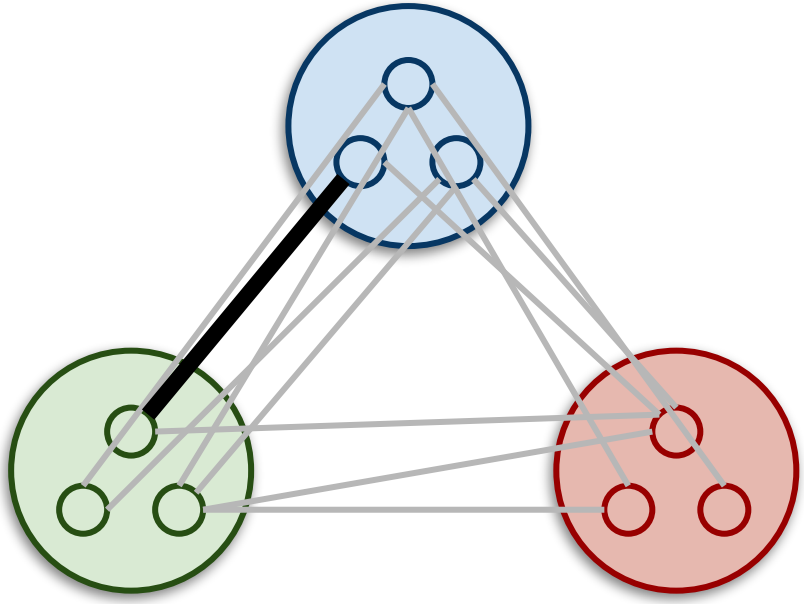
1. Calculating pairwise distance between every candidate for every named entity



Congruent Prediction: Vector Similarity

"Chris Tanner met Pavlos in Harvard Yard."

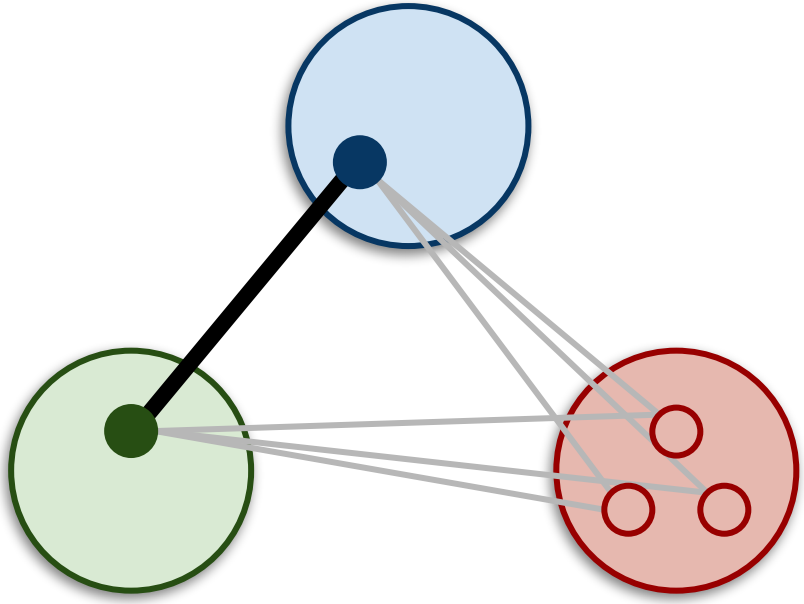
2. Identify most similar pair and save those candidates as predictions



Congruent Prediction: Vector Similarity

"Chris Tanner met Pavlos in Harvard Yard."

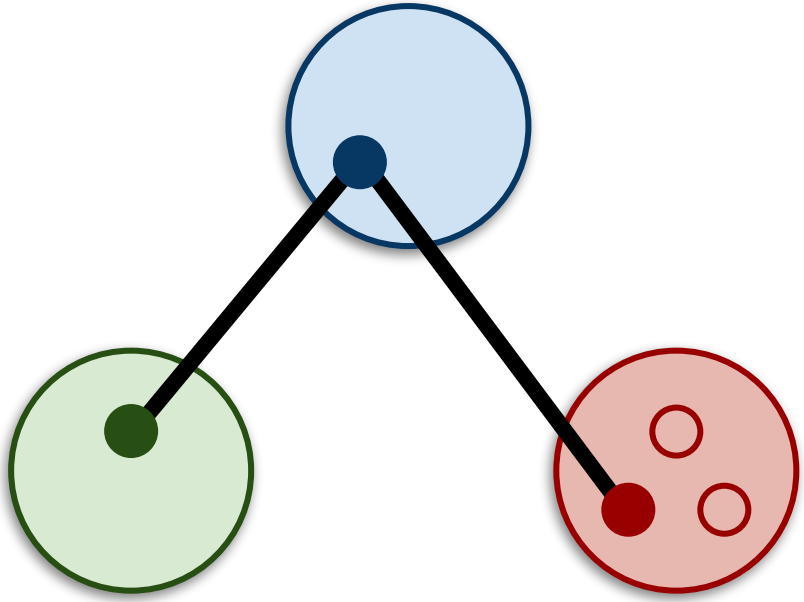
3. Iteratively find next most similar candidate(s) until all named entities predicted



Congruent Prediction: Vector Similarity

"Chris Tanner met Pavlos in Harvard Yard."

3. Iteratively find next most similar candidate(s) until all named entities predicted



Congruent Prediction: Vector Similarity

"Chris Tanner met Pavlos in Harvard Yard."

Congruence provided a performance increase of 2.1-3.7% depending on the size of the sentence.

Increasing Vector Dimension:

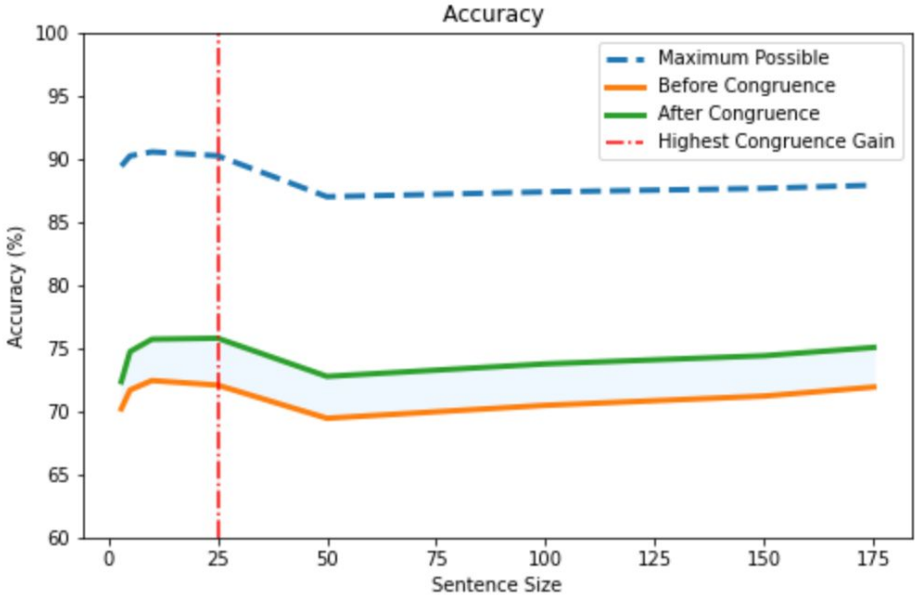
100D: 75.795%

300D: 76.304%

500D: 76.291%

****

Max Predictive Accuracy: 76.3%



Congruent Prediction: Vector Similarity

"Chris Tanner met Pavlos in Harvard Yard."

Alternatives:

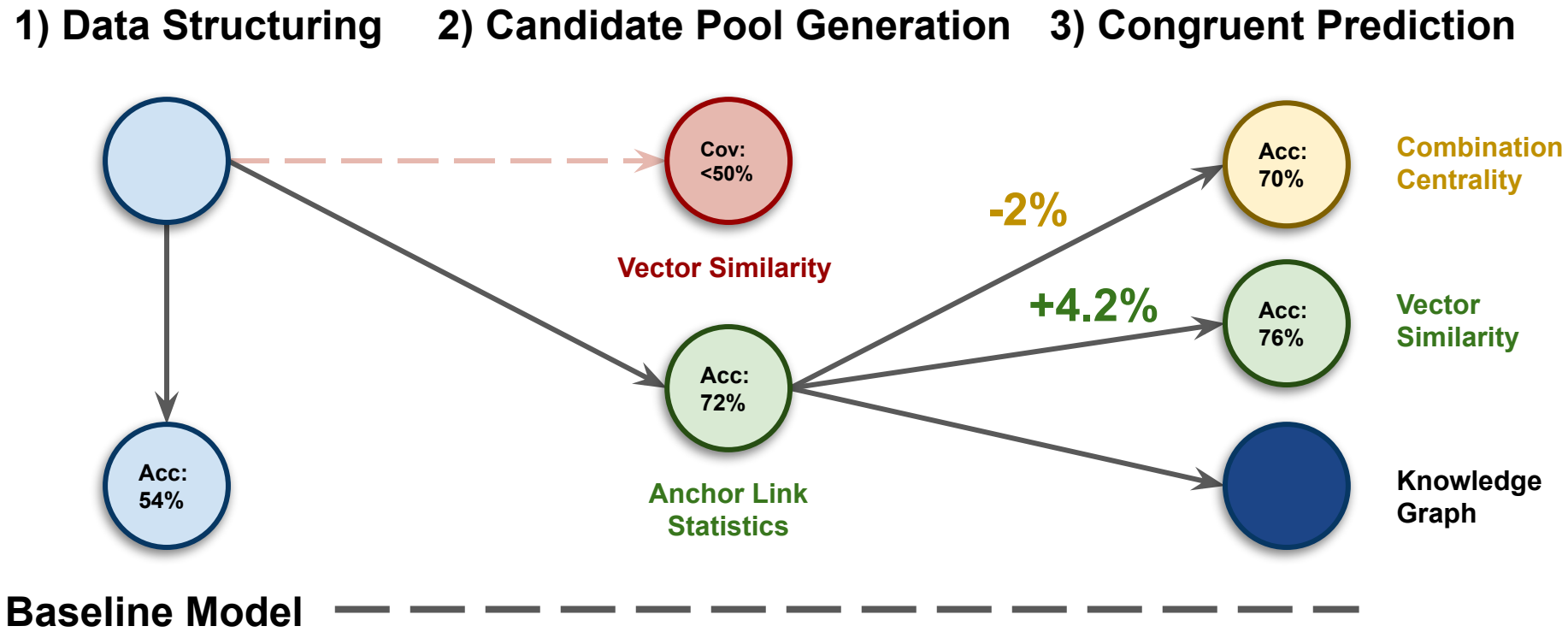
- Don't build off prior predictions - just find most similar pairs (accuracy -2%)
- Increase dimensionality of vectors - minimally improves accuracy (+1%)
- Vary size of candidate pools - Optimal: 5-10 candidates. Unlimited pool: (-3%)

Challenges:

- Congruence can overcorrect when right answer required more general candidate
- Dated Wikipedia pages don't react well to congruence

Incorporating named entity "congruence" using well-trained entity embeddings successfully improves predictive accuracy by a meaningful level!

Research Paths



Congruent Prediction: Knowledge Graph

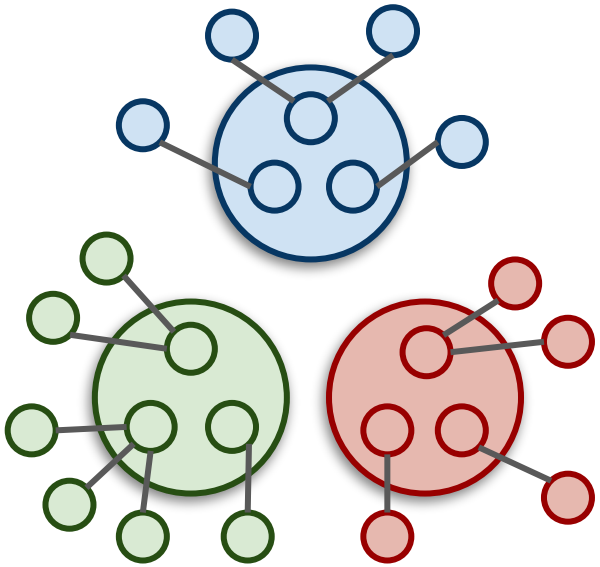
"Chris Tanner met Pavlos in Harvard Yard."

Method: Iteratively select candidates with the shortest path between them on a filtered "dendrite graph"

Congruent Prediction: Knowledge Graph

"Chris Tanner met Pavlos in Harvard Yard."

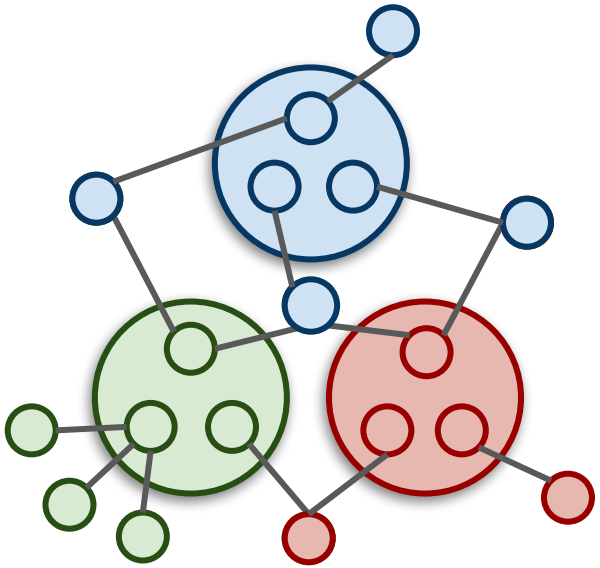
1. Retrieve first-degree in/out connections for all candidates for named entities from Kensho-derived Wikipedia knowledge graph



Congruent Prediction: Knowledge Graph

"Chris Tanner met Pavlos in Harvard Yard."

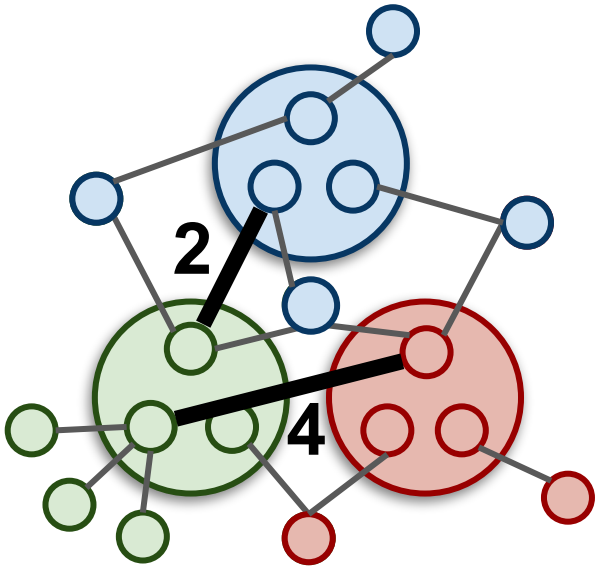
2. Create "dendrite graph" of this expanded pool using overlapping nodes



Congruent Prediction: Knowledge Graph

"Chris Tanner met Pavlos in Harvard Yard."

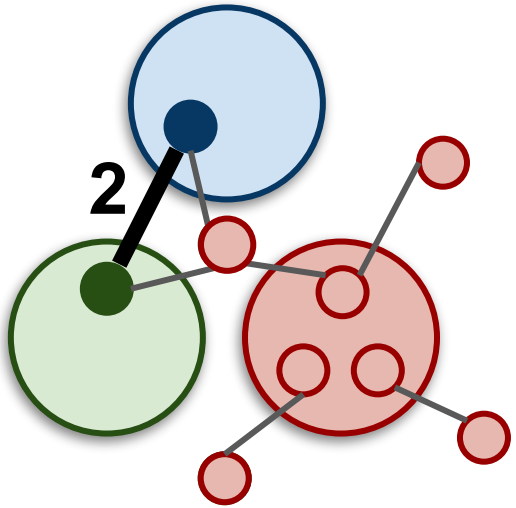
3. Calculate shortest distance between original candidates using "dendrite graph"



Congruent Prediction: Knowledge Graph

"Chris Tanner met Pavlos in Harvard Yard."

4. Select candidates with shortest distance, then iterate through remaining



Congruent Prediction: Knowledge Graph

"Chris Tanner met Pavlos in Harvard Yard."

Takeaways:

- Computational challenges greatest here. Only able to test over 50 named entities in reasonable time.
- Dendrite approach reasonable method to generate manageable sub-graphs on-the-fly
- Multiple graph metrics possible, but currently fastest one wins
- Ontological abstractions add greater separation between similar nodes compared with using anchor links.

Deploying dedicated database technology like Neo4J might increase computational speed enough to make improved congruence calculations feasible, but our lower-level approach proved non-rewarding with 60.163% accuracy.

Congruent Prediction: Graph Embeddings

"Chris Tanner met Pavlos in Harvard Yard."

Method:

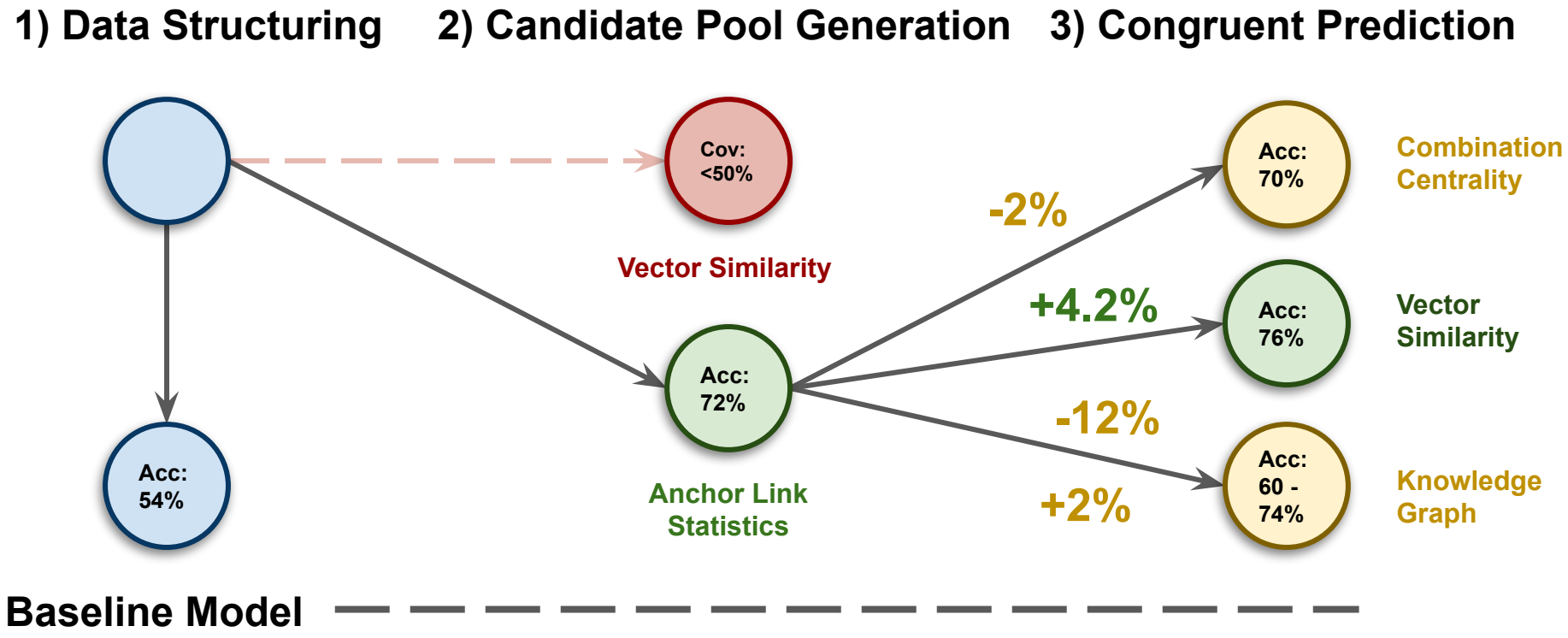
- Experimented with isolating power of knowledge graph using pre-trained embeddings from GraphVite developed over only Wikipedia's knowledge graph

Takeaways:

- Predictive accuracy achieved 74.5%, better than anchor links alone but not as good as Wikipedia2Vec's embeddings trained over three sub-models representing information on graph and text corpus

Embeddings trained over just knowledge graph enable improved accuracy with congruence.

Research Paths



Concluding Remarks

Context clues captured using informative entity embeddings for Named Entity Congruence can improve predictive accuracy on Named Entity Disambiguation by 4.2% (*23% of available gain*).

Future Research

- Improve computational performance of graph calculations using dedicated hardware like Neo4J
- Develop method of training informative embeddings over arbitrary knowledge base
- Improve coverage rate provided by anchor link statistics

Takeaways

- Requires intelligent embedding design over the target knowledge base
- Ontological intermediaries in graph work increase separation compared with entity page embeddings

Literature Review

Literature on this sphere of NLP is vast and growing, with some of the most relevant papers for "congruence" being accepted to conferences *this* month. We've highlighted some of our key papers informing our approach.

Pair-Linking for Collective Entity Disambiguation: Two Could Be Better Than All. Minh C. Phan, Aixin Sun, Yi Tay, Jialong Han, and Chenliang Li. URL.

Wikipedia2Vec: An Efficient Toolkit for Learning and Visualizing the Embeddings of Words and Entities from Wikipedia. Ikuya Yamada, Akari Asai, Jin Sakuma, Hiroyuki Shindo, Hideaki Takeda, Yoshiyasu Takefuji, Yuji Matsumoto. URL.

Improving Entity Linking through Semantic Reinforced Entity Embeddings. Feng Hou, Ruili Wang, Jun He, Yi Zhou. URL.

A Primer in BERTology: What we know about how BERT works. Anna Rogers, Olga Kovaleva, Anna Rumshisky. URL.

Robust Disambiguation of Named Entities in Text, Johannes Hoffart, Mohamed Amir Yosey, Ilaria Bordino, URL.