INSTITUTE FOR APPLIED COMPUTATIONAL SCIENCE AT HARVARD UNIVERSITY

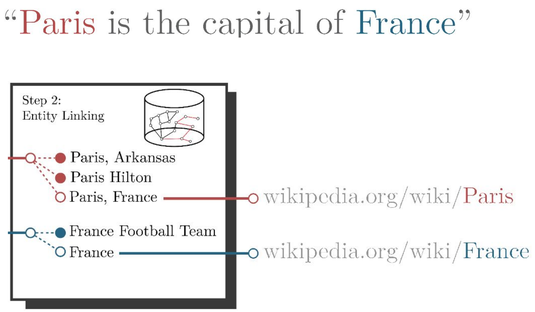
KENSHO

Entity Linking using Kensho-Derived Wikimedia Dataset

Weiru Chen, Dean Hathout, Tyler Yoo, David Zheng

Introduction and Background

In natural language processing, entity linking[1] is the process of assigning a unique identity to entities (such as famous individuals, locations, or companies) in a text. For example, when given the sentence "Paris is the capital of France", the idea is to determine that "Paris" refers to the city of Paris and not to Paris Hilton or any other entity that could be referred to as "Paris".



Also known as entity disambiguation, entity linking is important to the broad task of extracting abstract representations from text, and separating relevant concepts from non-meaningful text. As such, entity linking finds application in fields such as text analysis, recommender systems, semantic search, and chatbots.

Problem Statement & Scope

Our goal is to generate a model that can accurately execute named entity linking, using the Kensho-Derived Wikimedia Dataset (KDWD)[2].

The KDWD is a multi-layered, structured rendition of the Wikidata knowledge graph, and has been made available by Kensho to assist researchers in tackling many of the NLP problems explored by Kensho R&D, including entity linking. We aim to use the KDWD to experiment with a fairly novel approach of entity linking via joint disambiguation over an entity's context window.

What are we focusing on?	What are we not focusing on?	Future possibilities
Named Entity Disambiguation via Context Window Type Predictions	Named Entity Recognition	State-of-the-art baseline (e.g. OpenAI)
KDWD	Text-based entity linking	Speed improvements

The table above summarizes our intended scope. As mentioned, we focus specifically on the task of entity linking aka entity disambiguation via joint linking over the context window, and use the KDWD. We will not be exploring entity linking based on text features, nor entity recognition, the process of identifying entities in text.

Looking beyond this project, we see interesting opportunity to develop our approach further by making it more computationally efficient, and comparing it to state-of-the-art baselines which do not employ context window predictions,, such as OpenAI's deep learning approach to disambiguation.

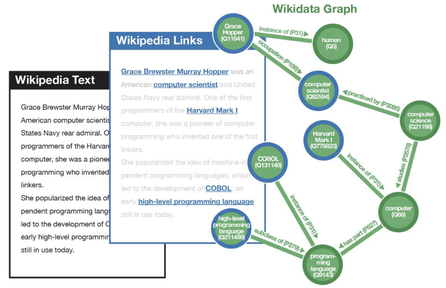
1. https://en.wikipedia.org/wiki/Entity_linking
2. https://blog.kensho.com/announcing-the-kensho-derived-wikimedia-dataset-5d1197d72bcf

Exploratory Data Analysis

How do we create the training/test dataset?

Each Wikipedia article contains links that point to their *page_ids*. The text of each link is what we aim to disambiguate and the ground truth is the *page_id* of the link.

The candidate id values for each link are generated from all the links in the Wikipedia corpus that share the same text but point to different *page_ids*.



Parsing 58 million links is slow…

The main issue we ran into when constructing an exploratory model was the long run-time it took to search for all the candidate ids for a given link's text value. We solved this be aggregating all the links and their ids and constructing a dictionary with each ***key*** **:** ***value*** pair as ***link_text*** **:** ***candidate_ids*****.**

Constructing graph for traversal

We faced a similar runtime constraint when traversing a graph with 101 million **source-property-target** entries 51 million unique *source_ids*. First we assigned a weight to each property. Then we sorted each *source_id*'s children by the property weights. Finally we restructured the graph as a dictionary with each ***key*** **:** ***value*** pair as ***source_id*** **:** ***sorted_target_ids*****.**

Baseline Model

Algorithm of the baseline model:
In our two baseline models we just simply choose the **most popular entity** as entity prediction for each mention.

- In the first baseline model, the most popular entity is defined as the entity with the highest Wikipedia **page views**.
- In the second baseline model, the most popular entity is defined as the entity that has **most links** directed to it.

In the baseline model with **word embeddings**, the entity that has the minimum cosine distance with the neighboring named entities in the context window identified by the spaCy package is predicted as the entity for each mention.

- The distance between two entities is defined as the cosine distance between their pre-trained word embeddings from the google2vec model.

Metric for model evaluation:
We randomly sample k=20,000 mentions and compute the overall **accuracy rate**.

Results:

Model	Accuracy Rate
Model with highest page views as popularity	61.83%
Model with most links directed as popularity	59.135%
Model with word embeddings	66.74%

Developed Model

Disambiguation as Combinatorial Optimization

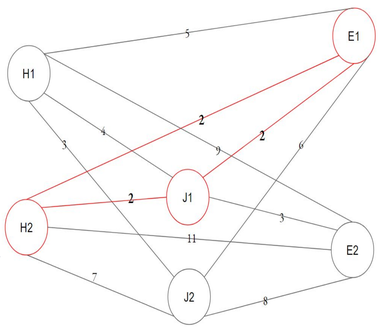
Disambiguates the given context window as a whole
Assumption: the correct set of disambiguation minimizes the the sum of pairwise distances between nodes on the knowledge base (i.e. semantic distance)

Example:
"Honda is competing against Jaguar in EV industry"

Honda: [H1: the entrepreneur, H2: the car brand]
Jaguar: [J1: the car brand, J2: the animal]
EV: [E1: electric vehicle, E2: expected value]

All possible combinations:
(H1,J1,E1),(H1,J1,E2),(H1,J2,E1),(H1,J2,E2),(H2,J1,E1,(H2,J1,E2),(H2,J2,E1),(H2,J2,E2)

For each combination:
Measure = sum(all pairwise semantic distances)
# e.g. measure = D(H1,J1)+D(H1,E1)+D(E1,J1))
Update least if measure < least
Return combination with least measure



Results

Overall Accuracy Score Achieved

On the same sentence dataset (n = 10,000) we worked with in the baseline models, our developed model achieved a final accuracy score of 85.5%

Major implementation details

Performance enhancement and speedup was achieved through caching the values of previously calculated
distances for use later in other combinations
(e.g. D(H1,J1) value cached initially when calculating M(H1,J1,E1), accessed directly in M(H1,J1,E2).)

Future Steps

Speed Improvements: To test the viability of this approach on a large scale, improvements must be made to computational efficiency. We see potential improvements to the bottleneck of graph traversal for prediction and evaluation via techniques such as graph embeddings and graph truncation.

Comparison to State-of-the-art baseline models: Armed with greater computational efficiency, we could make a fruitful comparison between advanced learning techniques for disambiguation that do not employ context window predictions and similar techniques using our context approach.

Acknowledgements

We would like to thank Gabriel Altay and Georg Kucsko at Kensho for their graciousness in sharing their time and resources with us throughout this project.

Finally, we thank Chris Tanner of IACS for his invaluable guidance to our group throughout the semester and for his leadership throughout the Capstone experience.