Collective Named Entity Disambiguation via Personalized Page Rank and Context Embeddings

NED

Collective Named Entity Disambiguation via Personalized Page Rank and Context Embeddings

CS 273 · Data and Knowledge Bases · Term Project Report

Luis Ángel Larios Cárdenas · *6612279*

Abstract

Named entity disambiguation is a key component of knowledge graph construction when information is extracted from large text repositories. In this work, we provide a solution to the disambiguation task by leveraging the traditional techniques of candidate mapping entity generation and local evaluation with some latest developments, such as word embeddings. We also consider a graph-based collective process to establish a topical relatedness metric that helps true mapping entities in a document to disambiguate one another through personalized PageRank. The final mapping entities for the given surface forms are obtained by heuristically reincorporating the candidates' local features with their resulting graph score and performing a maximal discriminant selection. The proposed methodology is capable of reaching up to 80% accuracy when it is evaluated against a well known dataset with around 18,000 named entity mentions.

Introduction

Named entity disambiguation (NED) or *entity linking* is the task of figuring out the true (real-world) mapping entity for each of the mentions that appear in an input text. NED is of particular importance for knowledge graph construction from large text repositories because it helps automate the accurate identification of entities that become part of the *subject - predicate - object* triples. In turn, linking to true mapping entities also enriches the contents of a knowledge graph by incorporating more facts that may be found in other structured knowledge bases.

The NED problem has been widely studied in a variety of settings. There actually exists a well established line of research and traditional heuristics that have proven to be effective in linking entities in scopes ranging from structured, topic-narrowed web lists [3], to topically-coherent documents [4], and to extremely noisy environments like microblogs (e.g. Twitter) [5]. In all of these developments, it is clear that not only is NED vital for knowledge graph construction and bridging of existing KBs (like DBPedia and YAGO). In reality, entity linking is a core process for exploiting the abundance of information in everyday news, podcasts, short posts (e.g. tweets), and blogs. Furthermore, NED helps improve search engines, web page navigation, personalized recommendation systems, trend detection, and question answering systems, among others [5].

Our work hinges on the theory provided in [3, 4, 5] but also incorporates the concept of *maximal discriminant selection* described in [6]. The latter poses NED as a ranking problem whereby the goal is to choose the best mapping entity from a list of candidates, such that the difference between the first and second highest ranked mapping entities is the largest (i.e. maximally discriminant).

On a different but complementary line of research, the recent developments with distributed word representations [8, 9], which represent words as continuous vectors, has spurred a lot of excitement in the NLP and Information Retrieval communities. *Word embeddings*, particularly when available as sets of pre-trained vectors from large corpuses (like Wikipedia), have elicited the appearance of advanced neural language models like ELMO, Transformer, and BERT. In an increasing number of situations, these newer techniques are becoming dominant in many of the Information Retrieval tasks. In regards to ranking documents, for instance, the good, old approach of the *vector model* and *TF-IDF* [2] for text similarity has been recently outperformed by the introduction of a simple, yet very effective framework that employs pre-trained word vectors to compute *sentence embeddings* [9].

We have incorporated the algorithms from the research presented in [9] into our work, where we naturally extend their sentence embeddings method to "longer" textual versions like named entity windowed contexts and whole entity documents from a knowledge base. Thus, we demonstrate that the latest developments in neural language models can play a key role in solving for the true mapping entities in the NED problem. Our results show that *context embeddings* (as we called them) are a very competitive and promising replacement for TF-IDF-based vector documents when it comes to measuring the context similarity between two texts.

We next present our framework in further details. We also show the experiments we performed, and conclude our report with some learning outcomes, challenges, and future work.

Methodology

Preliminaries

We begin by providing a formal description of the NED problem and some of the notation used in this report.

Let d be an input text document which contains a set $M = \{m_1, m_2, \ldots\}$ of previously recognized named entity mentions. The NED problem consists of mapping each $m_i \in M$ to its corresponding real-world (surrogate) entity e_i in a knowledge base. When a mention m_i cannot be mapped to an entity in the knowledge base, it is assigned the value of NIL.

We have adopted *Wikipedia* as our knowledge base given its availability, breadth, and wide use as the underlying KB in previous works [3, 4, 5]. For a better exposition of our methodology we are sticking to the example shown in figure 1. It corresponds to an *MTV News*[1] excerpt, when Madonna

[1] http://www.mtv.com/news/1539338/with-no-director-and-broken-ribs-madonna-was-hung-up-vmas-behind-the-camera/

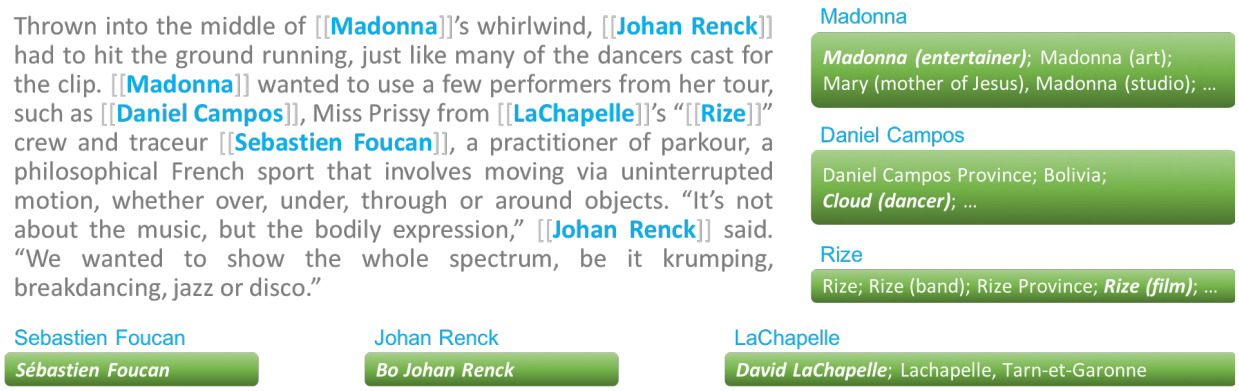

Figure 1: Example input text with labeled named entity mentions and corresponding candidate mapping entities in Wikipedia.

released her single "Hung Up" in 2005. This article illustrates the key problems that need to be addressed when mapping entities to mentions in text. Consider the *surface form* "Madonna": it may refer to several individuals, not just the entertainer that the article talks about. This is an exemplification of ambiguity and, to solve it, it is imperative to begin by constraining the solution to a set of potential mapping entities.

Formally, given an entity mention m_i, the (possibly empty) set of entities that may be referred to by such surface form is denoted by R_i, where each **candidate mapping entity**, $r_{i,j} \in R_i$, corresponds to an entry in the knowledge base. Therefore, the true entity mapping e_i for m_i must be exactly one of the candidate entities gathered in R_i.

For example, the entity mention "Madonna" may refer to several entries in Wikipedia, and figure 1 lists its top 4 *most popular* candidate mapping entities. The corresponding candidate lists for the rest of the surface forms in the text also appear within their green boxes. Notice that the true mapping has been emphasized (in a bold, italicized font) with respect to the rest of the candidates, and it is not always the case that it matches the most *popular* entity for the given mention. In some other instances, an entity mention happens to have just one candidate mapping entity (like "Johan Renck"), while in others (not illustrated here), $|R_i|$ may be overwhelmingly large. It all depends on how *general* a surface form is.

Candidate Mapping Entities Generation

In order to collect the R_i set for each mention m_i, we need to build a dictionary [3]. This is done by processing the articles in Wikipedia in such a way that we extract all name variations, synonyms, nicknames, etc., that editors use to refer to entities within the knowledge base:

- **Entity pages**. Each entity may be referred to by, at least, its Wikipedia canonical name. For example, Madonna, the singer, is uniquely identified by "Madonna (entertainer)". In this case, we create an entry for the entity in the dictionary under the surface form `madonna` after removing the parenthesized expression. Note that we have normalized the surface forms to lowercase to increase the number of candidates for a given mention. For instance, "Algol" and "ALGOL" correspond to two distinct entities in Wikipedia; however, we have chosen to inscribe both of them under `algol` in our dictionary.

- **Redirect pages**. Often times, entities have nicknames, name variations, or name mispellings, and Wikipedia captures these through its redirect pages. For example, the article "Sebastien Foucan" redirects to the entity page "Sébastien Foucan", and in this case, we register a dictionary entry for the latter under the surface form `sebastien foucan`.

- **Disambiguation pages**. When the same surface form refers to many entities, Wikipedia provides a disambiguation page with links to the corresponding, distinct articles. For example, the Wikipedia page *Rize (disambiguation)* lists all entities that go by the name "Rize", including the one we are interested in: "Rize (film)". In this case, we remove the parenthesized expression and record entries for all listed entities under the surface form `rize`.

- **Wikilinks**. Wikipedia editors can point or refer to entities within the knowledge base via Wikilinks of the form `[[entity|anchor text]]`. These links provide lots of (possibly noisy) information about the variety of ways that people use when they refer to Wikipedia entities. We exploit Wikilinks by inserting entries in the dictionary, where the anchor text becomes the surface form referring to the corresponding entity. For example, from `[[David LaChapelle|LaChapelle]]` we take the surface form `lachapelle` and match it with the entity "David LaChapelle".

A portion of the dictionary with surface forms corresponding to the example of figure 1 is shown in table 1. Note that we have added a *count* column. This field keeps track of the number of times a surface form is used to refer to the indicated Wikipedia entity. The *count* value is calculated during the (long) process of surface forms extraction from Wikipedia dump archives and is later used, at runtime, as part of a metric for ranking the candidates R_i of every m_i.

Surface Form	*Candidate Entities*	*count*
madonna	**Madonna (entertainer)**	4805
	Madonna (art)	178
	Mary (mother of Jesus)	55
	[16 more candidates]	[...]
johan renck	**Bo Johan Renck**	1
lachapelle	**David LaChapelle**	1
	Lachapelle, Tarn-et-Garonne	1
daniel campos	Daniel Campos Province	2
	Bolivia	1
	Cloud (dancer)	1
	[4 more candidates]	[...]
sebastien foucan	**Sébastien Foucan**	2
rize	Rize	113
	Rize (band)	30
	Rize Province	16
	Rize (film)	14
	[5 more candidates]	[...]

Table 1: Part of the dictionary for working example.

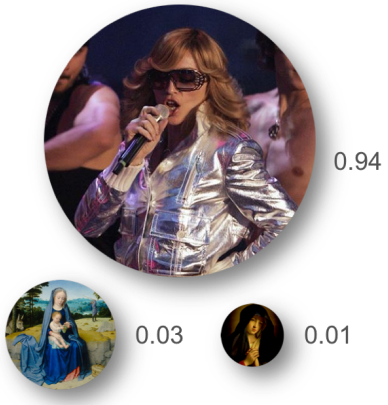

Figure 2: Prior probability of the top three candidate mapping entities for the surface form `madonna`. (Images courtesy of Wikimedia Commons).

Candidates Evaluation Overview

Once a set of candidate mapping entities R_i for mention m_i becomes available, we have to select the best candidate $r_{i,j}$ and assign it to e_i. We claim that the process of evaluating the quality of the candidates for a named entity mention in d can be posed as a *ranking* problem. Followed the suggestions in [6, 5], we determine the qualification of each $r_{i,j}$ by considering two composite measurements:

- An **initial score**, or $p(r_{i,j})$, which is based on the local features of $r_{i,j}$ and its textual mention m_i in d, and
- A **propagation score**, or $s(r_{i,j})$, which is based on the global, topical interaction between $r_{i,j}$ and all of the candidate mapping entities R_ℓ of the rest of the mentions m_ℓ in d, for all $\ell \neq i$.

The final candidate selection and assignment to e_i is made from the maximal discriminant combination of p and s, such that the difference between the best ranked candidate mapping entity combined scores and its closest contestant's is the largest [6]. We next describe in depth how to arrive at e_i and the steps we followed to compute the initial and propagation scores for R_i.

Initial Score

The **initial score** of a candidate mapping entity is a composite quality measurement that focuses on $r_{i,j}$ and m_i local or intrinsic features. In other words, the initial score considers each candidate and its associated textual mention in d in isolation with respect to the other surface forms and candidate sets. Specifically, $p(r_{i,j})$ is the convex combination of the candidate entity's *prior probability* and *context similarity* with respect to the surroundings of m_i in d.

Prior probability, $\Pr(r_{i,j})$, is a metric that expresses how popular the candidate mapping entity $r_{i,j}$ is within the set R_i. It is given by equation (1) and depends on the *count* field that appears in the dictionary (see table 1):

$$\Pr(r_{i,j}) = \frac{count(r_{i,j})}{\sum_{c=1}^{|R_i|} count(r_{i,c})} \tag{1}$$

From our running example's dictionary in table 1, one can observe that the prior probability of the candidate mapping entity *Madonna (entertainer)* should be far larger than that of *Madonna (art)* and *Mary (mother of Jesus)* with regards to the surface form `madonna`. The accurate numbers may be seen in figure 2.

The second, but not less important, local feature of a candidate mapping entity is its **context similarity**, $\text{Sim}(r_{i,j})$. It expresses how compatible a candidate mapping entity entry in Wikipedia is with respect to the syntactic context of its mention, m_i, in d.

Our approach for evaluating the context compatibility of a candidate mapping entity relies first on *tokenizing* d and the Wikipedia pages for all entities across the KB. This yields a working vocabulary $\mathcal{V}$. Formally, a context c is a map of tokens $w \in \mathcal{V}$ to term frequencies. Unlike the traditional TF-IDF method regularly used to determine the similarity between two contexts, c_1 and c_2, we calculate **context embeddings**, $\mathbf{v}_{c_1}$ and $\mathbf{v}_{c_2}$, by incorporating pre-trained word embeddings $\{\mathbf{v}_w \mid w \in \mathcal{V}\}$. Our context embeddings are a natural extension to the *Smooth Inverse Frequency* or SIF-based sentence embeddings described in [9].

For our purposes, we extract 50 tokens around each occurrence of mention $m_i \in M$ in d into a *bag of words* to construct its context c_{m_i}. Then, we follow the steps in algorithm 1 to compute the context embeddings, $\mathbf{v}_c$, for all c in d and in R_i, $i = 1, 2, \ldots, |M|$. Later, the context similarity between two embeddings is calculated via *cosine similarity* with a subsequent re-scaling to ensure that the result lies in the range of $[0, 1]$. That is,

$$\text{Sim}(r_{i,j}) = \frac{1}{2}\left(1 + \frac{\mathbf{v}_{m_i}^T \mathbf{v}_{r_{i,j}}}{\|\mathbf{v}_{m_i}\|\|\mathbf{v}_{r_{i,j}}\|}\right) \tag{2}$$

where $\mathbf{v}_{m_i}$ and $\mathbf{v}_{r_{i,j}}$ are the context embeddings of the textual mention m_i and its candidate mapping entity $r_{i,j}$, respectively.

Data: Word embeddings $\{\mathbf{v}_w \mid w \in \mathcal{V}\}$, contexts $\mathcal{C} = \{c_{m_i} \mid m_i \in M\} \cup \{c_{r_{i,j}} \mid r_{i,j} \in R_i\}$, $i = 1, 2, \ldots, |M|$, parameter a, and estimated probabilities $\{p(w) \mid w \in \mathcal{V}\}$ of the words
Result: Context embeddings $\{\mathbf{v}_c \mid c \in \mathcal{C}\}$

forall $c \in \mathcal{C}$ **do**
  $\mathbf{v}_c \leftarrow \frac{1}{|c|} \sum_{w \in c} \left(\frac{a}{a + p(w)} \mathbf{v}_w \right)$
end
Form a matrix X whose columns are $\{\mathbf{v}_c \mid c \in \mathcal{C}\}$, and let $\mathbf{u}$ be its first singular vector
forall $c \in \mathcal{C}$ **do**
  $\mathbf{v}_c \leftarrow \mathbf{v}_c - \mathbf{u}\mathbf{u}^T \mathbf{v}_c$
end

Algorithm 1: Context embeddings.

Finally, we formulate the initial score $p(r_{i,j})$ of a candidate entity mapping as the convex combination of metrics (1) and (2):

$$p(r_{i,j}) = \alpha \mathrm{Pr}(r_{i,j}) + \beta \mathrm{Sim}(r_{i,j}) \tag{3}$$

where α and β are hyperparameters, such that $\alpha + \beta = 1$. We experimentally found that $\alpha = 0.4$ and $\beta = 0.6$ performed well for our datasets; nevertheless, some previous works have demonstrated that max-margin machine learning techniques may be used to search for their optimal values [3, 4, 5].

Propagation Score

The true entities for mentions in a document should help one another to be correctly mapped [6]. That is, to rank a candidate mapping entity $r_{i,j}$ for mention m_i, it is desirable to introduce some way to gauge the *topical relationship* between $r_{i,j}$ and the other true mapping entities, e_ℓ, $\ell \neq i$. The issue, however, is that during the NED process we do not yet have access to any e_k, $k = 1, 2, \ldots, |M|$. To address this situation of incomplete information, we resort to a graph-based approach [5, 6] that models the interdependence of candidates within a topically coherent document d, via **Personalized PageRank**. In some sense, we are interested in the *propagation* of candidate entities' initial score, such that those that are strongly related can *collectively* reinforce their qualification. We refer to the PageRank, convergent candidate mapping entity qualification simply as **propagation score**, $s(r_{i,j})$.

Let $G(V, W)$ be an undirected graph where the vertices $v_k \in V$ correspond the candidate mapping entities $r_{i,j} \in R_i$, for all textual mentions $m_i \in M$ in d, $i = 1, 2, \ldots, |M|$; and where W is a symmetric, adjacency matrix. Further, $w_{k,k'} \in W$ is a nonnegative weight that denotes the topical relationship strength between nodes v_k and $v_{k'}$.

We construct G by linking pairs of entity nodes, $r_{i,j} \in R_i$ and $r_{i',j'} \in R_{i'}$, whenever their **topical relatedness** is nonzero and $i \neq i'$. We have adopted the *Wikipedia Link-based Measure* (WLM), which is formulated as [5]:

$$\mathrm{TR}(u_1, u_2) = 1 - \frac{\log(\max(|U_1|, |U_2|)) - \log(|U_1 \cap U_2|)}{\log(|WP|) - \log(\min(|U_1|, |U_2|))} \tag{4}$$

where u_1 and u_2 are two entity entries in Wikipedia, U_1 and U_2 are the sets of pages respectively linking to them, and WP is the total number of articles in the KB. Notice that if $|U_1 \cap U_2| = 0$, then u_1 and u_2 are completely unrelated, and, therefore, $TR(u_1, u_2)0$ should be zero.

Figure 3 exhibits the graph constructed for our working example from MTV news. Observe that some entities like *Daniel Campos Province* are not topically related to any candidate mapping entity, whereas others like *Madonna (entertainer)* are highly connected. Notably, and as expected, she is more related to *Rize (film)* (directed by *David LaChapelle*) than to *Rize* (a region in Turkey).

We next describe how to compute the propagation score using Personalized PageRank. For each node v_k ($k = 1, 2, \ldots, \sum_{\ell=1}^{|M|} |R_\ell|$) associated to a candidate entity mapping $r_{i,j}$, we normalize its initial score as follows:

$$np_k = \frac{p_k}{\sum_{c=1}^{|V|} pc} \tag{5}$$

where $p_k = p(r_{i,j})$ from equation (3). Then, for each edge $(v_k, v_{k'})$, we normalize the score propagation strength $w_{k,k'} = W(v_k, v_{k'})$ from node v_k to $v_{k'}$ by using the formula

$$NW(v_k, v_{k'}) = \frac{W(v_k, v_{k'})}{\sum_{v_c \in V_k} W(v_k, v_c)} \tag{6}$$

where $V_k \subset V$ is the set of nodes that share an edge with v_k. In other words, equation (6) normalizes the weights in edges departing from v_k, such that np_k is proportionally distributed to all nodes v_c pointed to by v_k.

We are now ready to formalize the computation of propagation scores for candidate mapping entities in graph G. Let $\mathbf{p} = [np_1, np_2, \ldots, np_k, \ldots, np_{|V|}]^T$ be a vector of node normalized initial scores as given by equation (5). Let B be the $|V| \times |V|$ stochastic, column-normalized propagation strength matrix, such that $b_{k',k} = NW(v_k, v_{k'})$, as shown in equation (6). Then, the propagation score, $s_k = s(r_{i,j})$, for each candidate entity is collected in $\mathbf{s} = [s_1, s_2, \ldots, s_k, \ldots, s_{|V|}]^T$ via the Personalized PageRank algorithm

$$\mathbf{s} = \lambda \mathbf{p} + (1 - \lambda) B \mathbf{s} \tag{7}$$

where $\lambda \in [0, 1]$ is a hyperparameter inherited from the "random surfer model." We found that $\lambda = 0.4$ was good at

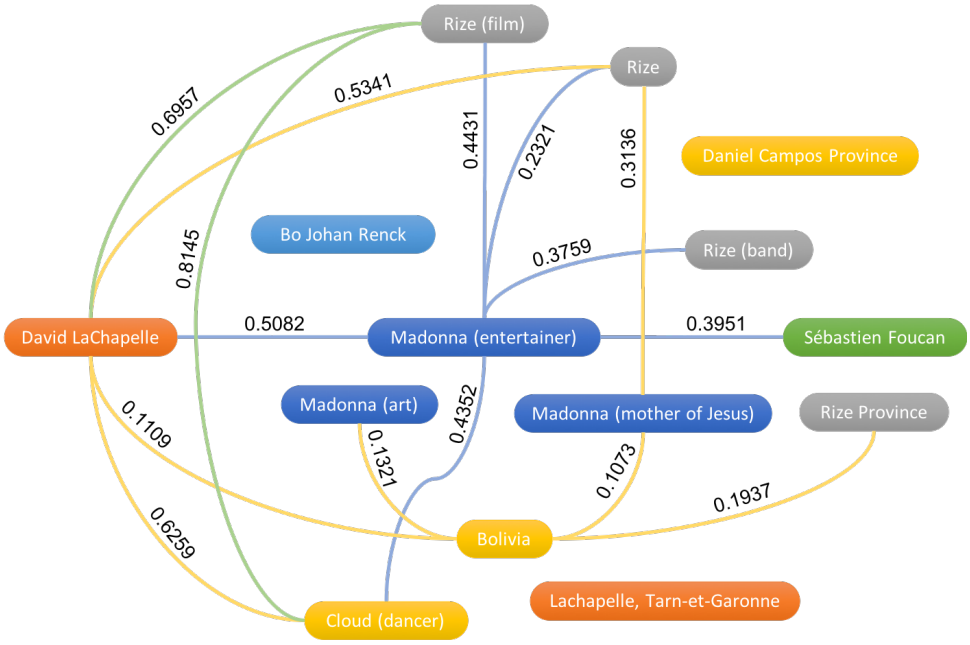

Figure 3: Graph for running example with candidate mapping entities colored by corresponding textual mention.

yielding accurate results. Moreover, since equation (7) is recursive, we start the iterative process with $\mathbf{s} = \mathbf{p}$, and at each stage we make sure to keep $\mathbf{s}$ normalized (i.e. $\sum_{k=1}^{|V|} s_k = 1$).

Maximal Discriminant Selection

One problem with PageRank [6] in NED is that it tends to (over) dissipate the (normalized) initial score over outgoing nodes. For this reason, we opt for reincorporating the initial score with the propagation score through an additive scheme and a multiplicative scheme. Then, we make our final decision on e_i depending on which scheme is more discriminative for the top two candidate entities for each m_i.

Let $\mathbf{l}_s(R_i) = \mathbf{p}(R_i) + \mathbf{s}(R_i)$ and $\mathbf{l}_m(R_i) = \mathbf{p}(R_i) * \mathbf{s}(R_i)$ be two combination schemes for the normalized initial and propagation scores corresponding to the candidate entities of mention m_i, where $*$ means element-wise multiplication. The assignment of the best candidate mapping entity to e_i for the NED problem is finally carried out as depicted in algorithm 2.

Data: Two lists: L_s and L_m of candidate mapping entities R_i for mention m_i, where L_s is ranked using $\hat{\mathbf{l}}_s = \mathbf{l}_s / \sum_{j=1}^{|R_i|} l_{s_j}$, and L_m is ranked using $\hat{\mathbf{l}}_m = \mathbf{l}_m / \sum_{j=1}^{|R_i|} l_{m_j}$
Result: Final entity mapping e_i for m_i

```
Sort L_s and L_m in descending order
δ_s ← L_s[0] − L_s[1]
δ_m ← L_m[0] − L_m[1]
if δ_s > δ_m then
    return R_i[L_s[0]]
else
    return R_i[L_m[0]]
end
```

Algorithm 2: Maximal discriminant selection.

[2] https://github.com/imyoungmin/NED
[3] https://fasttext.cc/docs/en/pretrained-vectors.html

Implementation and Results

We developed our NED framework in Python 3.6 (Anaconda environment) and used *MongoDB* as the underlying database. The code, datasets, and sample text documents may be accessed on GitHub[2], where we provide full instructions on how to launch the system from start to end.

The NED system is divided into two modules: one for building the necessary MongoDB collections, and another for disambiguating named entity mentions in an input document or evaluating our model's accuracy against a well-known dataset.

We built the NED collections (e.g. dictionary) from the **November, 2014, English Wikipedia dump archives** [7], which we parsed using our own adaptation of the **2003 CoNLL** NER tagging task tokenizer [1]. We use this tokenizer to also process the mention contexts within the input document in order to maintain compatibility across the different tasks within the framework. In total, we extracted 4'589,969 entities and registered 11'964,048 surface forms after preprocessing each Wikipedia article with the help of WikiExtractor [10].

The Wikipedia tokenization further outputted 2'519,370 different terms, without removing any stop words or punctuation marks (which is typical for the vector model and TF-IDF approach). For every token we stored its 300-dimensional, pre-trained word embedding [8] after downloading a publicly available set from Facebook's **fastText** repositories[3]. These are precisely the distributed vectors that we employed in the generation of context embeddings for the context similarity metric computations. Moreover, we approximated word probabilities for the SIF coefficients in algorithm 1 by keeping track of token frequencies across the whole Wikipedia corpus.

The grand total of words and symbols was 2,071'472,445 for the Wikipedia version we utilized in our system. And, in the end, the database collections accounted for around 27GB of disk space.

Accuracy Evaluation

We validated our NED model against the CoNLL 2003 dataset for NER tagging [1]. This dataset consists of **1387** *Reuters* newswire articles from late 1995 and contains manually labeled named entity mentions that are linked to their corresponding Wikipedia entries (whenever possible). In our experiments, we only considered surface forms for which there existed at least one candidate mapping entity in the dictionary collection. Thus, we obtained **18,407** unique (and potentially linkable) surface forms, where **399** of them were mapped to NIL. Table 2 shows the results for three different experiment settings.

Test	***Correctly Mapped***	***Macro Accuracy***	***Micro Accuracy***
Full (initial score + propagation score + re-ranking)	13,926	0.773	0.8028
No re-ranking	13,541	0.752	0.7802
No propagation	13,248	0.736	0.7549

Table 2: Experiment results.

First, notice that the effective number of linkable mentions in the dataset was 18,308 after removing the 399 unmapped surface forms. From these, we considered two accuracy metrics: a macro perspective and a micro perspective. *Macro Accuracy* refers to the proportion of correctly mapped named entity mentions with respect to the total number of linkable surface forms. On the other hand, *Micro Accuracy* refers to the average number of correctly mapped mentions per (news) document.

In the experiments we evaluated the accuracy of the NED system when all of the features described in the previous sections are considered, and when some of them are removed –one at a time. In table 2 we can observe that local features (i.e. initial score) are already good metrics for disambiguating up to 75.49% of mentions in a document. When we add the collective, graph-based approach (i.e. propagation score), the accuracy increases almost three points to reach 78.02%. Finally, when local and global features and the maximal discriminant selection are considered, accuracy increases further by more than 2%, and up to 80.28%. These results demonstrate that not only intrinsic properties contribute to mapping entities to mentions, but also candidate entities help one another to solve for the true entity assignments.

Conclusions and Future Work

We learned in this project how to solve the NED problem, from start to end, through a combination of traditional heuristics with some of the latest technologies, such as word embeddings, and other IR methods, like Personalized PageRank. We also learned how to make use of Python multiprocessing to expedite computations and how to incorporate NoSQL collections via MongoDB for rapid access to the underlying NED data.

Since our framework rests on the foundations laid out in no less than three research papers, our first concern was related to achieving a decent degree of accuracy. We believe that the results presented in the previous section are indicative that our proposed method did meet the expectations and is on par with some other works [6]. Nevertheless, there is still room for further investigation to increase the mapping accuracy by either redefining some metrics or by completely reengineering the solution from a machine learning perspective.

One area that deserves corroboration is the use of distributed word vectors for composing context embeddings. Indeed, the theory in [9] proves that pre-trained word vectors work very well at generating sentence embeddings and exploiting similarity features inherited from the distributed word representations. However, it still remains to formally demonstrate whether the same approach works when it is extended to longer texts (e.g. multiple-sentence documents).

Another challenge that requires a good solution in the graph-based setting concerns disambiguating named entities when the same candidate exists for different surface forms. We have observed that in this scenario the propagation algorithm tends to over-qualify this candidate and, as a result, the same final mapping entity is assigned to all mentions that share such candidate.

Some other opportunity areas for improvement regard increasing accuracy. One possibility includes the incorporation of more features for evaluating the qualifications of candidate mapping entities. We have shown that prior probability, context similarity, and topical relatedness are an excellent baseline, but there is no doubt there are other metrics that we could try out to boost the percentage of correctly mapped entity mentions in the test dataset. Particularly, [4] introduces category-related metrics that are worth integrating into our framework as a venue for additional experimentation.

Furthermore, we are convinced that accuracy can really benefit from a more robust Wikipedia compilation and parsing system. Essentially, if the dictionary is contaminated with noisy candidate entity entries, the NED framework will find it very difficult to overcome the noise and achieve higher accuracy mapping rates. We refer to "noise" in the dictionary as the presence of candidate mapping entities that are highly uncorrelated to the surface forms that some editors chose as anchor texts in Wikilinks. Take, for example, the surface form `title track` that appears in the *Madonna (entertainer)* Wikipedia article: despite of the phrase being so general, it is used in a Wikilink to refer to the entity *Who's That Girl (Madonna song)*. Like this, lots of entities get registered "by mistake" in our dictionary and ultimately end up affecting the accuracy of the NED framework.

Finally, we would like to point out that a more promising line of research and future work appears to be in the realms of machine learning. We can definitely begin by posing parameter tuning as an optimization process, given that we already have the CoNLL dataset divided into training and testing components. However, we know of extensions of supervised learning that have been used for ranking documents, and we

think it is possible to transfer those practices into named entity disambiguation. Thus, it is perhaps more reasonable to look into AI approaches to turn NED into a classification problem. Only then, possibly, we could get accuracy levels like those reported in state of the art frameworks.

References

[1] Conference on Computational Natural Language Learning. *CoNLL 2003 Shared Task*. https://www.clips.uantwerpen.be/conll2003/ner/. May 31 - June 1, 2003. Edmonton, Canada. Retrieved in April, 2019.

[2] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press. 2008.

[3] Wei Shen, Jianyong Wang, Ping Luo, and Min Wang. *LIEGE: Link Entities in Web Lists with Knowledge Base*. KDD '12, Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. August 12 - 16, 2012. Beijing, China.

[4] Wei Shen, Jianyong Wang, Ping Luo, and Min Wang. *LINDEN: Linking Named Entities with Knowledge Base via Semantic Knowledge*. WWW '12, Proceedings of the 21st International Conference on World Wide Web. April 16 - 20, 2012. Lyon, France.

[5] Wei Shen, Jianyong Wang, Ping Luo, and Min Wang. *Linking Named Entities in Tweets with Knowledge Base via User Interest Modeling*. KDD '13, Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. August 11 - 14, 2013. Chicago, IL, USA.

[6] Ayman Alhelbawy and Robert Gaizauskas. *Graph Ranking for Collective Named Entity Disambiguation*. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Short Papers). June 22 - 25, 2014. Baltimore, MD, USA.

[7] Wikimedia Foundation. *2014 English Wikipedia Dump Files*. https://archive.org/download/enwiki-20141106. Retrieved in April, 2019.

[8] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. *Enriching Word Vectors with Subword Information*. Transactions of the Association for Computational Linguistics, vol. 5, pp. 135–146. 2017.

[9] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. *A Simple but Tough-to-Beat Baseline for Sentence Embeddings*. ICLR '17, International Conference on Learning Representations. April 24 - 26, 2017. Toulon, France.

[10] Giuseppe Attardi. *WikiExtractor: A tool for extracting plain text from Wikipedia dumps*. https://github.com/attardi/wikiextractor. Retrieved in April, 2019.