
AN EXPLORATION OF WIKIPEDIA FOR DISAMBIGUATION OF GRAPH ENTITIES

Jeffrey Chiavetta

Department of Linguistics

University of Arizona

Tucson, AZ 85721

jchiavetta@email.arizona.edu

https://github.com/jchiavetta/HLT_internship_report

ABSTRACT

Entity Linking (EL) is the task of connecting salient entities in text with entries in a knowledge base of structured information. Since the rise of Wikipedia as a large open-source knowledge base, attempts have been made to leverage its articles and link structure, both to enrich natural text and to disambiguate entities.

In this report I detail my internship with lum.ai, a company using a large graph of data from machine-read research papers containing potentially ambiguous entities. I survey some of the historical work done in this area and make a contribution with three models for the task of disambiguation to Wikipedia, these being 1. basic joint probabilities, 2. vectors based on article text, and 3. vectors based on link structure.

I find that article text vectors perform best, because the length of an average Wikipedia article gives many contextual clues to determine the semantic domain of the ambiguous entity.

The code used for this research can be found in the Github repository shown above.

Keywords Disambiguation · Wikipedia · Entity

1 Introduction

Word Sense Disambiguation (WSD) is a task that tries to untangle the messiness of natural language with regard to meanings. Words can contain many different meanings despite having the same surface form. A 'play' can be a theatrical production, a coordinated action in a sports match, or a part of a larger structure like 'a play on words'. In fact, this can even extend to the grammar of the language, affecting syntactic representations as in the verb form 'to play a game'.

WSD uses the context of the words in question to determine which of the multiple meanings is the intended one. In this paper I will explore the closely-related task of Entity Linking (EL). The key word is 'linking'; we can visualize WSD as choosing the correct entry out of a dictionary, whereas EL will typically refer to a knowledge base of some sort, more like choosing the entry from an encyclopedia. Moro et al. (2014) use this analogy in their survey of the field, as well as stating that "The aim of EL is to discover mentions of entities within a text and to link them to the most suitable entry in a reference knowledge base." [1] They make the further distinguishing point that the EL task often has to deal with incomplete mentions, for example proper nouns that only include the first name. A mention of "Mario" within a news article about cooking would most often be linked to the knowledge base entry for the chef "Mario Batali", a problem that isn't as common in the standard WSD task. The solution would need to have a representation for the context that encodes the culinary terms and uses that information to differentiate this mention from occurrences of "Mario" in other contexts e.g. the "Mario" from Nintendo's video games.

Every successful EL task will have a way of learning a representation of context, but it will also need a suitable knowledge base to link to. The choice of knowledge base (KB) is important; different KBs will differ not just in pure size but in the structure of the content. There are two main approaches I have encountered which I choose to call

'horizontal' vs. 'vertical'. These are not the only ways to structure a KB but seem to be the most important in the literature.

Horizontal knowledge bases are those that consist of individual articles (of varying lengths) written for specific topics. There are three crucial components to this system: 1. The articles are not organized hierarchically, 2. They are unique, and 3. They contain links.

Horizontal KBs do not belong to an overt system of organization; a given article can be viewed as a separate unit that does not depend on other articles in hypo- or hypernym relationships. In other words, an article for 'cutlery' will not be located within a grouping such as 'tools' > 'cutlery' > 'spoon'.

The articles are unique in that semantic referents are given their own articles regardless of orthographic similarity. Continuing the analogy, 'spoon' will have one unique article for the type of cutlery and a separate article for the band named Spoon.

Last, the horizontal knowledge bases useful for NLP purposes will have links embedded within each article that direct the user to other related articles. These links are typically formatted as hyperlinks on plain text strings called anchor text. The user can explore a near limitless range of related topics by hopping from article to article through clicking on anchor texts.

By far, the largest and most-discussed knowledge base of this type is Wikipedia. There are others, such as the online version of the Encyclopedia Britannica, but none compare to the sheer size and breadth of Wikipedia's content. As of this writing, the English-language Wikipedia has 5.6 million articles, which are not constrained to any specific semantic domains. It takes up about 100 GB of storage space even while compressed [2]. As we will see, many people have recognized the growing influence of the site and are looking for ways to leverage its structured data for various machine learning and NLP tasks.

The other knowledge base paradigm is vertical or hierarchical, meaning that these KBs do choose to represent hypo- and hypernyms for each semantic referent. This data is helpful to set up taxonomies of referents and can be visualized as a tree structure. It is clear that this is a valuable way to represent the data, as it allows users to browse topics by zooming in or out to the level of detail they want. They might want to first survey the more general information on cutlery and then focus on just spoons. The downside is that this information on spoons will likely not include a horizontal slice of the varying meanings for spoons, such as you would find in a Wikipedia disambiguation page.

This type of structure is not the only advantage to using a hierarchical knowledge base. The two largest such KBs are DBpedia and Wikidata, which in fact both rely on Wikipedia. Both are attempts to extract hierarchical data from Wikipedia. But in addition, they are both examples of what is being called Web 3.0 or the Semantic Web. Horizontal KBs, as well as essentially all of the user-directed internet (blogs, social networking sites, etc.) belong to the Web 2.0 framework, which represents data in "pages" that are easy to read by humans but not by computers. Aside from links, Web 2.0 reflects the idea of the internet as a sort of digital newspaper.

Web 3.0 is an emerging framework from the World Wide Web Consortium (W3C) designed to make the Internet more machine-readable [3]. While Wikipedia is stored in XML, vertical KBs like DBpedia and Wikidata represent information in the Resource Description Framework (RDF) format. In short, this format uses a unique URI for every semantic referent, but instead of user-directed prose it consists solely of a predicate 'triple', so that 'spoon' might be represented as 'spoon is-member-of cutlery'. Each subject, object, and predicate have their own URIs that can be linked together in a directed graph. This structure also allows for powerful queries, usually with a specific RDF query language named SPARQL.

2 Related Work

Some early work in EL with Wikipedia relied on comparing the text in an input document to the text in each candidate article, as in Bunescu and Paşca (2006) [5]. A year later, Mihalcea and Csomai (2007) [6] tried two strategies in their Wikify! project. They also used textual overlap, with an algorithm that "attempts to identify the most likely meaning for a word in a given context based on a measure of contextual overlap between the dictionary definitions of the ambiguous word – here approximated with the corresponding Wikipedia pages, and the context where the ambiguous word occurs". In addition, they also tried training a classifier based on the link structure within the articles; "For each ambiguous word, we extract a training feature vector for each of its occurrences inside a Wikipedia link, with the set of possible word senses being given by the set of possible links in Wikipedia." For features they used local context and parts of speech among others.

This approach ends up being a much more efficient method, because at the scale we're working with, referring to the text content of entire articles becomes difficult to manage. This problem is only becoming more severe, as Wikipedia

has roughly tripled in size over the past decade. Interestingly, Wikify! not only disambiguates but also performs entity extraction to determine which words in the text should be linked to the KB. In our task we only need the disambiguation component, because the entities are already contained in lum.ai's graph.

Milne and Witten (2008) [7] devised what they call the Wikipedia Link-based Measure (WLM) to measure similarity between two articles using only the links within them, saving a great deal of storage space and processing time. At that time there were over 5 million links harvested throughout all of Wikipedia; it is very common for a single article to have 50+ links, though this number ranges widely because there is no standard article length. The idea is that this is enough information to compare two articles and get a reasonable similarity score. The equation they came up with is as follows:

$$sr(a, b) = \frac{\log(\max(|A|, |B|)) - \log(|A \cap B|)}{\log(|W|) - \log(\min(|A|, |B|))} \quad (1)$$

"where a and b are the two articles of interest, A and B are the sets of all articles that link to a and b respectively, and... W is the entire Wikipedia." By plugging in two articles for a and b we get a single number representing their similarity, based solely on their link structure. What's most interesting here is that this equation doesn't take into account links being made from the articles of interest, but rather only looks at links to them. The authors also experimented with outgoing links but found that this method performed much better.

Figure 1 shows how two articles can be compared based on the links that they have in common, whether incoming or outgoing.

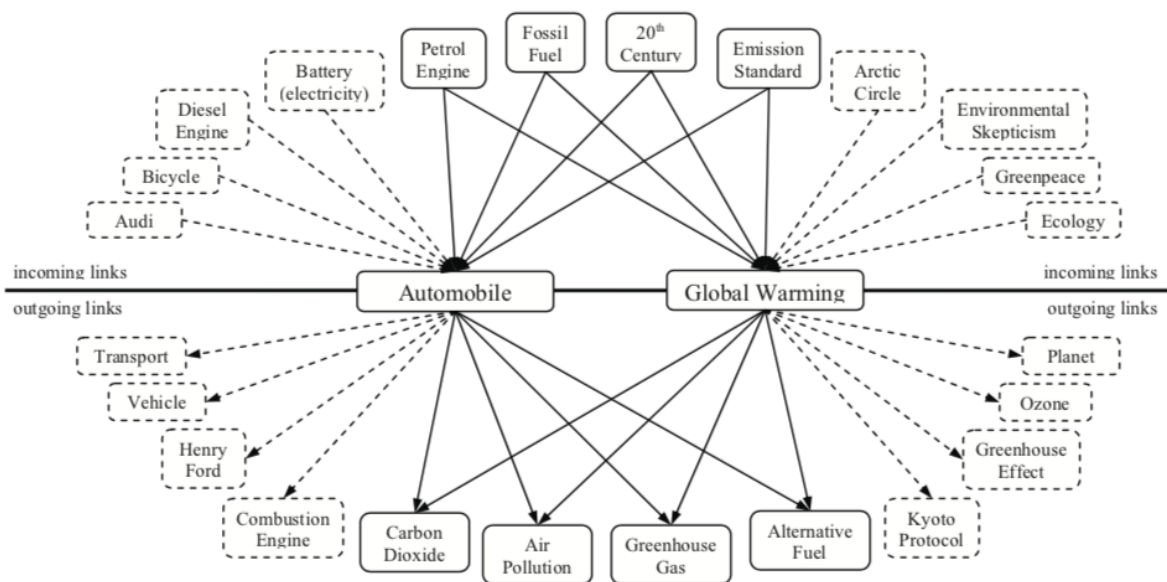


Figure 1: Obtaining a semantic relatedness measure between Automobile and Global Warming from Wikipedia links.

Figure 1: Visualization of two Wikipedia articles and their similarity in terms of shared links. [7]

Both Moro et al. (2014) [1] and Hoffart et al. (2011) [12] take a graph-based approach to the problem instead. Their methods both attempt to disambiguate multiple entities at a time, taking advantage of the link structure as in Milne and Witten (2008) but using it to "build a weighted graph of mentions and candidate entities, and compute a dense subgraph that approximates the best joint mention-entity mapping." [12] This is useful because it encodes the intuition that entities within the same document are likely to belong to the same semantic domains.

3 Task Description

For my internship I worked with a company here in Tucson named lum.ai [4]. The company has developed a sophisticated machine reading system which can scan through text and extract salient entities. They use this in part to read through the medical literature. Crucially, the number of papers published every year is increasing and it is already

far beyond the ability of any human to read through all of the papers that might be relevant to their research. In fact, even beyond the obvious papers there may be important information hidden in papers that they aren't aware of, perhaps those that come from other departments or less well-known authors.

Lum.ai's idea is that machine reading can distill the important parts of millions of papers and present them to the researcher. In addition to the machine learning and NLP tools needed for entity extraction, their system makes causal links between entities. For example, a researcher can search for a given symptom and see all of the entities that purportedly cause that symptom.

These links mean that the knowledge base can be interpreted as a directed graph, with entities as nodes and causal links connecting them as edges. This structure allows for a detailed look at interesting properties like the number of edges on a node (i.e. whether or not it's a hub node). Lum.ai has already used this to create a graph with over 1.5 million nodes and 2.3 million edges. In Figure 2 we can see a visual demo of the company's graph.

Influence Search

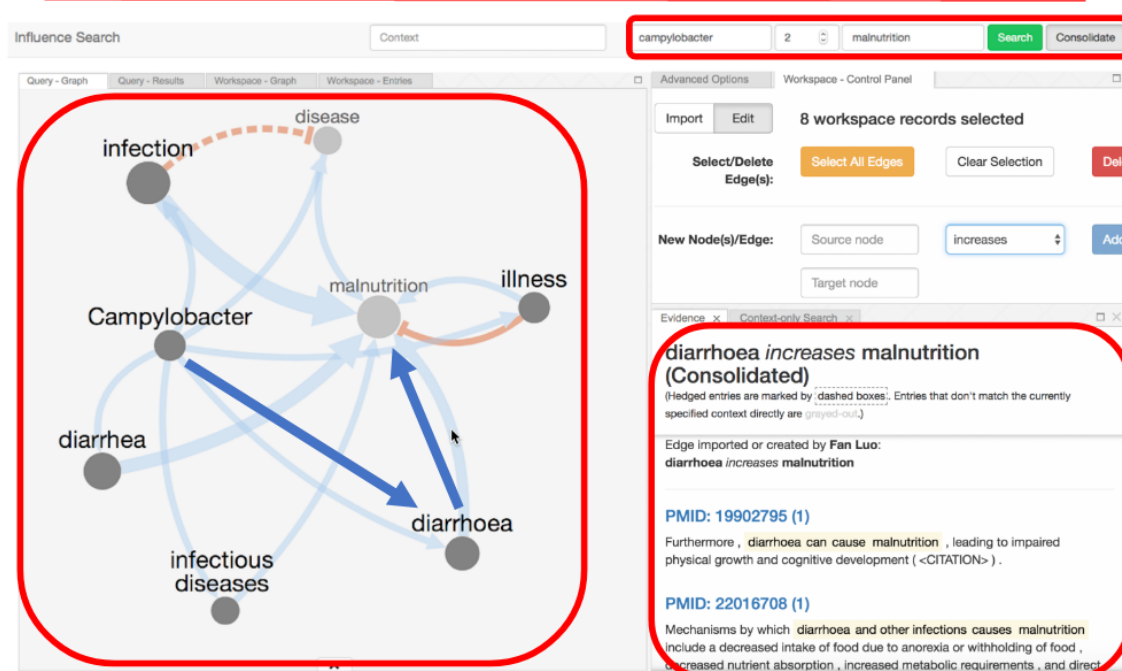


Figure 2: Lum.ai's graph showing the results of a query on the bacteria *Campylobacter* and its indirect effects on malnutrition, from a presentation. [10]

One issue with the approach, however, is that the entities represented by each individual node in the graph (e.g. 'Campylobacter' and 'malnutrition' in Fig 2) often have multiple meanings or senses. This is especially problematic with entities that aren't proper nouns, like many in the dataset. For example, a medical dataset will contain words like 'blood'. A cursory look at the related Wikipedia disambiguation page reveals over 50 distinct pages for 'blood', ranging from the intended biological meaning to the LA street gang to a variety of songs and music albums. In many cases we can simply choose the most common meaning, but it isn't always that simple. For a word like 'tree' this would give the arboreal reading when a medical paper might be more likely to use the data structure.

We want this knowledge to be encoded in the knowledge graph for several reasons. First is that having a representation of the unambiguous meaning of an entity allows for linking between different nodes of the graph. If we know that 'blood' relates to the biological fluid, it can be related to other entities like 'hemoglobin'; in some cases the entities might even be differently-worded versions of the same entity and could therefore be collapsed into one node.

Second, the opposite operation to this idea of node collapse would be to keep nodes distinct even if they look similar. In a large dataset like lum.ai's, it is possible that the entity 'tree' could appear in both the arboreal and the data structure readings. These two should be represented as two separate nodes, likely far apart, even if they both appear as 'tree'. In

other words, the task is to move from a strictly orthographic representation of concepts to one where the concepts are disambiguated and therefore can be referred to by unique IDs.

This is where Wikipedia comes in. As mentioned, Wikipedia has a unique article for each concept. These pages can be enumerated and used as IDs so that 'tree (arboreal)' will be completely different from 'tree (data structure)'. The entity linking task consists of finding the best match between an entity and the corresponding Wikipedia article. My main contribution during the internship period was to survey some of the ways researchers have used Wikipedia for entity linking and to implement them myself to see what challenges and benefits can be found through different methods of approaching this problem.

I ended up creating three models. The first is commonness, meaning whichever sense/article is linked to most often. The second is term vectors, which involves creating a matrix of that encodes the body text of each article. The third is link vectors, which differ because they encode the link structure of an article rather than its textual content. Each of these models is explored below.

4 Commonness

To restate, my goal was to find a way to use Wikipedia to help disambiguate entities from the lum.ai graph. There is no ground truth in this task, the only way to verify success is to manually check the results of test queries on the nodes. Ideally the entities will be linked to pages that you would expect to find in a scientific (specifically medical) research paper. As a baseline it is useful to gather the commonness score for each anchor. In other words, we want to see which pages are most commonly used as link destinations, e.g. that 'tree' goes to 'tree (arboreal)' about 90% of the time.

4.1 Wiki Link Structure

The first step in this process was to learn how to process the articles and grab links from them by downloading the latest Wikipedia dump. Wikipedia stores their data in XML format so that different sections can be tagged and organized in a page hierarchy. Fortunately, what is of interest here is simply the links, so it was not important to navigate the XML tags. Instead, I went through line by line and pulled out only the raw links. Wikipedia formats these in a consistent way that is worth going into, because it informs the entire task. Standard links (from the article on anarchism, the first in the list) look like this:

```
[[Anarchism in Mexico | Mexico]]
[[Anarchism in Brazil | Brazil]]
[[Spanish Civil War]]
```

Every link is surrounded by double square brackets, and some contain a pipe character. The pipe serves to separate the anchor text from the article that it links to. For example, if you go to the anarchism article, you will find the string 'Mexico' as a hyperlink. When you click on that anchor text it takes you to the page titled 'Anarchism in Mexico'. The 'Spanish Civil War' link has no pipe because in that case the name of the page is identical to the anchor. This convention saves space but is conceptually identical to [[Spanish Civil War | Spanish Civil War]].

This standardization means that the links can be extracted with a simple line by line regular expression search for the pattern '\[.*?\]', capturing everything between each pair of double brackets. The only trick is to make sure that the pattern isn't greedy, or else it will run into problems when a single line has more than one link.

4.2 Joint Probabilities

This results in a massive list of links that can be used to build a joint dictionary of (article, anchor) pairs along with their associated counts. At that point, finding the percentage probability or commonness of a given pair just comes down to dividing the count of the pair by the total number of times that anchor appears. The result is an easy baseline to compare against the more complex measures we'll be using later. For entities that are less domain-specific, this should actually be fairly accurate; a lot of the time there's one obvious answer to a disambiguation question. What's more interesting to see is whether a more complicated measure will successfully disambiguate medical terms even if the correct sense is not the most common one.

Here's an example of how commonness works by running it on the word 'disorder':

```
[('disorder (medicine)', 40.74%),
 ('disease', 29.63%),
 ('communication disorder', 7.41%),
```

```
( 'mental disorder', 7.41%),
( 'mental illness', 5.56%),
( 'sleep disorder', 5.56%),
( 'disease\#disorder', 3.7%)]
```

Disorder is a term that likely has a well-defined meaning in medicine and is used in specific contexts. Forty percent of the time that anchor goes to the page for the general medical definition. The other definitions are either more specific (e.g. 'mental disorder') or more broad (e.g. 'disease'). The last item has the pound sign included, which means it's a subheading within the 'disease' page. Unfortunately the answers are sometimes not so clear-cut. Selected senses for the word 'synthesis' include the following:

```
[('chemical synthesis', 22.2%),
( 'synthesis (journal)', 13.32%),
( 'synthesis', 11.72%),
( 'organic synthesis', 10.66%),
( 'biosynthesis', 10.3%),
( 'synthesis (evanescence album)', 6.04%),
( 'sound synthesis', 4.8%),
...
( 'protein synthesis', 0.71%)]
```

The responses here aren't bad, but it's difficult to determine if the top choice is the right one. We probably aren't looking for the Evanescence album or even the journal, and it makes sense that 'chemical synthesis' would be the most common sense. But 'protein synthesis' is a very reasonable answer as well, and it appears near the bottom of the 31 assorted meanings for 'synthesis', at only 0.71% of links. If the node in lum.ai's graph for 'synthesis' came from a paper on the effects of diet on protein synthesis, the commonness results here would be wrong. Clearly we need a method that uses the context of the queried entity to inform the answer. Also worth noting is the relative percentages. The more senses a term has, the more spread-out the percentage of links will usually be. Compare the top result of 'synthesis' with 22.2% of the link probability with that of 'disorder' at 40.74%. Ideally, a better measure will not lose confidence just because there are more potential senses. In other words, a disambiguation decision should be made based on just the context of the term itself rather than the size of the knowledge base.

4.3 Dataset

Lum.ai's machine reading model is designed to scan through natural text in order to extract entities and their directed relationships. Certainly it uses the context of the entities in some way to inform its decisions. Fortunately, this contextual information is saved for each edge in the graph; we can use that to help in the EL task. As an example, an edge is formatted as:

```
"4192111153", "increases", "2976004250", "4198194237"
```

where the first number is the unique edge ID and the others are node IDs corresponding to the gene CXCR3 and the protein zonulin, respectively. This denotes that some treatment or observation involving CXCR leads to an increase in zonulin. We can look up the exact context for this claim by looking up the edge ID, which gives the following:

"Initially gut paracellular permeability is increased in CD in part due to peptide induced CXCR3 activated upregulation of zonulin , an intestinal peptide involved in epithelial tight junction control."

The context sentences vary in length and are a guaranteed indicator of relatedness. By looking at the text, it should be possible to choose a disambiguation more accurately. But to do this we need to re-cast this information in a machine-readable form, typically in a vector of numbers.

In this context, a vector refers essentially to an array of numbers that characterize a given object. An object can be a picture, a sentence of text, or even an entire document, which will be the case in this task. There are plenty of different ways to build vectors for a collection of data, with assorted pros and cons. We will see how this works below.

Unfortunately, building vectors of any type can be limited by space and time constraints. Usually these constraints are not a very big issue for modern hardware, but as we've seen, Wikipedia is a very large dataset; it is likely that the entire set of articles cannot be loaded into memory at one time if you are working on a personal computer and this was certainly true in my experience. Because of this, I needed to find a package that could work with a dataset of this size instead of building the vectors myself.

5 Term Vectors

There is a python package named Gensim that can handle the Wikipedia corpus. Gensim is a tool for topic modeling and NLP but is designed for large datasets. Per their website: "Gensim can process large, web-scale corpora, using incremental, online training algorithms. There is no need for the whole input corpus to reside fully in RAM at any one time." [13] This is exactly what is needed to work with the Wikipedia articles.

The first step to building vectors for the articles is to create a dictionary of word types. We need to scan over every article (along with all the other data in the corpus e.g. user pages and disambiguation pages) and keep track of every unique word type. This is what determines the length of the vectors; a matrix of article vectors will be of size $M \times N$, where M is the number of articles and N is the length of the vocabulary. Creating the dictionary took about 12 hours on my machine. Even though Gensim allows us to go without storing everything in memory, it still takes a lot of time to parse the data and scan through all 5+ million articles. This is a one time cost, however; because it only stores unique words the size of the resulting dictionary is manageable and can be saved to disk and loaded up when needed, using about 4.5 GB of RAM.

5.1 Bag of Words and tf-idf

The dictionary will be used to create Bag of Words (BoW) vectors. Because the vectors are of length equal to the size of the dictionary, that means there is one dimension corresponding to each word type. For BoW vectors we simply go through and gather all the words in an individual article, and then increment the dimensions corresponding to each word. This results in a very sparse count of the words in an article in a format that is amenable to vector operations like cosine similarity and Euclidean distance.

Relying on pure counts of occurrences is a good start, but there are more sophisticated statistical methods that can improve the level of detail in the vectors. The most common way to do this is to use term frequency-inverse document frequency (tf-idf). A Bag of Words model only counts term frequency; tf-idf weights this against the overall document frequency. The idea is that many high frequency words have very low domain specificity. A function word like 'the' will appear in every article, regardless of the domain of that article. Because of this it should be considered less important; surely a word that appears in every article is not helpful for a model that tries to use words as evidence to characterize an article.

Instead we want to maximize term frequency while also minimizing document frequency. A word like 'medicine' will appear with high frequency in medical domain pages but will have low document frequency elsewhere in the corpus, indicating that it is highly domain-specific and should be emphasized in the corresponding vectors. Gensim can convert BoW vectors to tf-idf, but it can also be done manually by taking the term frequency and weighting it by the total number of documents divided by the number of documents including the term, as such:

$$idf(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|} \quad (2)$$

where N is the total number of documents and is equal to $|D|$, and the denominator can be read as "all of the documents in D such that the term is in the document". If the term appears in every document, e.g. 'the', then this will equal 1, making no difference. When the divisor is smaller than this, the term will be weighted higher.

5.2 Latent Semantic Analysis

There is another improvement we can make to the vectors so that they will more accurately encode the documents. These vectors are very sparse; the vast majority of dimensions will be zero because each article contains only a small subset of the overall vocabulary. We want to lower the dimensionality of the vectors, making them more dense. Gensim provides this in the form of latent semantic analysis (LSA). LSA is essentially the technique of applying singular value decomposition (SVD) to the tf-idf vectors in order to condense the sparse vectors into a pre-set number of "topics" representing groupings of semantic relatedness amongst terms and documents. SVD operates on the matrix of document vectors, factoring them into three smaller matrices that tease apart the term/document co-occurrences. These matrices are shown in Figure 3.

We can see that r is a hyperparameter that represents the number of semantic topics and serves to condense the factorized matrices into smaller dimensions. The choice of r depends on the dataset and the task but for a large corpus like Wikipedia it is common to pick a number around the 200-300 range. The larger the number, the more fine-grained the topics. Factoring the matrix apart gives us a lot of information: 'U' shows how each Wikipedia article fits into the topics. This gives us a vector for each article of length ' r ', which is greatly compressed from the original ' n '. 'S' is a diagonal matrix showing the strength or confidence of each topic. This can be used to rank the topics and is in

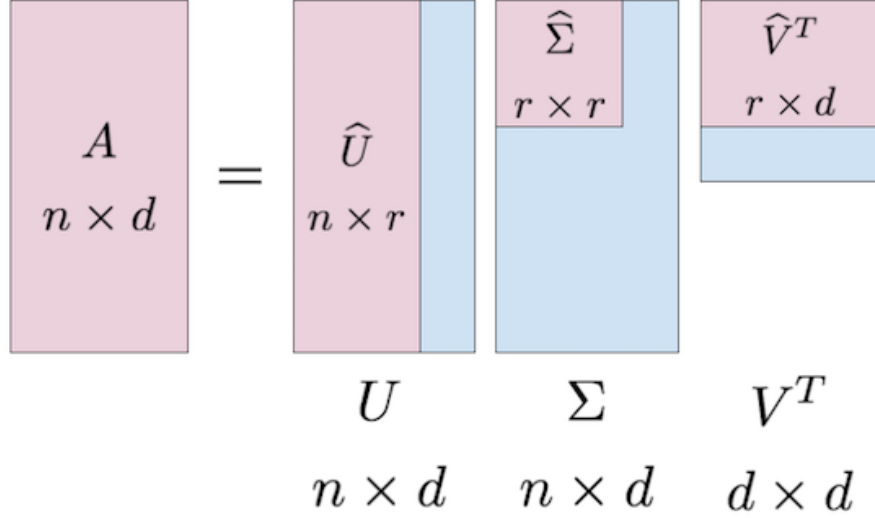


Figure 3: Factorization and compression of a matrix into lower dimensions, from a web article on the topic [9].

descending order from the top-left to the bottom-right. Last, 'V' shows the relation of terms to topics. For this task we only need the document vectors, but in certain cases it can be helpful to know the topic scores of individual words.

Armed with the LSI vectors, we can compare similarities between Wikipedia articles using various methods e.g. Euclidean distance:

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^n (q_i - p_i)^2} \quad (3)$$

Two vectors compared in this way will yield a score that equals zero if the vectors are identical and is larger if they are farther apart. Therefore we can choose an article and compare it against all others to get a list of the lowest numbers, which will be the most similar.

The big problem with this is that the task is not to compare articles with other articles. Rather we should be comparing nodes from the lum.ai graph to articles and choosing the most similar one as the disambiguation. This is difficult because the nodes are not structured documents of text. But we saw earlier that each edge of the graph contains the contextual evidence linking those two nodes. Because of this, my idea here was to build "articles" out of the context sentences. For each node, I gathered each edge connected to it and concatenated the evidence sentences from each of those edges together into one long string. Doing this gives a textual representation for each node based on its neighbors in the graph and how they connect to one another. Note that these pseudo-articles could be made even larger by including edges of edges. This could potentially increase performance, but I found that just by using first-order edges the pseudo-articles often approximate the length of an average Wikipedia article, because the graph is dense and can include hundreds of edges for any given node. This length makes these documents a good fit for comparing similarity.

The last step is converting the pseudo-articles to LSA vectors in the same way as before, making sure to use the same vocabulary that was used for the real articles. Now they can be compared to effectively see which articles have the greatest textual overlap with the evidence from the nodes, and choose the best match as the disambiguation. We have no ground truth for this task; there are no gold labels that can be used to calculate the accuracy of predictions. Instead we can do a manual error analysis, which shows some interesting results.

5.3 Error Analysis

So how well do the Gensim-generated LSA vectors help disambiguate entities from lum.ai's node contexts? To get a rough idea, I took a sample of 100 random nodes and checked the similarity results. Out of these, 58 of them were already unambiguous. That is to say, the anchor text corresponding to that node only ever linked to one Wikipedia article. That leaves 42 where the model had to make a choice. Overall the results were good, often picking what we would think of as the common sense choice. In fact, in some cases the model even chose a more specific meaning from the medical domain over the most common general-case scenario. For example, see the following:


```
In [9]: similarity(3537364982) #‘low temperature’
Out[9]:
[(array([[0.83764703]]), ‘hypothermia’),
 (array([[0.77624216]]), ‘cryogenics’),
 (array([[0.68448465]]), ‘refrigeration’),
 (array([[0.63521367]]), ‘lowest temperature recorded on earth’)]
```

Here I passed in the ID for the node containing the string ‘low temperature’. The highest similarity score between the articles and the node context was ‘hypothermia’. This is a good result because it shows that the LSA vector for the node learned a good representation of its meaning from the context pseudo-document and chose a medical-domain sense. Another example of this is the word ‘welfare’:

```
In [179]: error_analysis[28]
Out[179]:
(‘welfare’,
 [(array([[0.89238783]]), ‘quality of life’),
  (array([[0.84245253]]), ‘welfare state’),
  (array([[0.82455504]]), ‘welfare’),
  (array([[0.80394897]]), ‘welfare fraud’),
  (array([[0.79407724]]), ‘well-being’)])
```

where the sense most likely intended in the medical papers was chosen even over the existence of a more general article with the exact same title as the anchor text. For comparison, ‘welfare’ is the top choice in the commonness metric. It is also worth noting that the more general articles still score well, with ‘welfare’ getting a score of 82. For space I’ve only shown some of the top results here, but some nodes have dozens of potential senses. To get a sense of the scale, the lowest ranked sense for ‘welfare’ was a score of 11 for an English soccer player named Harry Welfare.

Out of the 42 ambiguous nodes, I judged 26 of them to be clearly correct like the above two examples. Happily for the model, none of the remaining 16 gave bad results. But there are some trends that show a difficulty in the underlying task.

Two of the answers might have been a bit off: one mapped ‘dexamethasone’ to the article ‘dexamethasone suppression test’ rather than the more general ‘dexamethasone’ article (88% vs 81% similarity score). The second mapped ‘ultrasound’ to ‘intravascular ultrasound’ rather than ‘medical ultrasound’ (89% vs 87% similarity). In both those cases it narrowly chose the more specific article when the node text suggests the general case. For ‘ultrasound’, it is likely that the papers were indeed talking about about intravascular ultrasounds, and this seems to reflect an intuition that research will try to be as specific as possible. But that exact bigram might not have actually appeared in the text, or the machine reading algorithm simply chose to extract the word ‘ultrasound’ by itself. It is hard to decide what the correct answer or the correct node text “should” be in a systematic way.

Similarly, five of the nodes presented difficulties because of the vagueness of the intended sense. For example:

```
In [207]: error_analysis[6]
Out[207]:
(‘model’,
 [(array([[0.83486322]]), ‘model organism’),
  (array([[0.82587689]]), ‘physical model’),
  (array([[0.81382142]]), ‘conceptual model’),
  (array([[0.80369831]]), ‘computational model’),
  (array([[0.78444925]]), ‘economic model’),
```

the top 5 (out of 61) senses for the node ‘model’ are all good answers, but the word ‘model’ does not give a very strong sense of the meaning because it encompasses so many different concepts. Because of this, it is a very dense node, with 4,600 edges connected to it. We can assume that all of those top sense choices are represented in those edges many times over, but Wikipedia does not have an article to accommodate all of them under one neat heading. In these cases it might be best to chose the disambiguation page (the page containing links to all ‘model’-related articles) as the correct answer, but it would be difficult to do this automatically because those pages have hardly any body text and so would have particularly weak-scoring vectors.

Last, there were 9 examples like the following:

```
In [209]: error_analysis[8]
```

```

Out[209]:
('separation',
 [(array([[0.61566707]]), 'separated shoulder', 156),
  (array([[0.57899335]]), 'flow separation', 156),
  (array([[0.53637176]]), 'separation of concerns', 156),
  (array([[0.51821689]]), 'separation process', 156),
  (array([[0.48722487]]), 'postage stamp separation', 156),

```

The top choices for the node 'separation' look fairly good, and in fact the winner is a medical term, but it is not a good disambiguation (and we can see that even the top score of 61 is lower than in other examples). In the 'model' case there was an idea of what the answer should be but it was too vague to be found in Wikipedia. These examples are subtly different; there is no real answer because the terms aren't being used to refer to a particular concept. A sentence from the node context will illustrate: "RNA and DNA helicases are considered to be enzymes that catalyze the separation of double stranded nucleic acids in an energy dependent manner often coupled to ATP hydrolysis." This usage of 'separation' could not have a Wikipedia article because it isn't a salient entity. It was chosen by the machine reading algorithm as a node because it is part of a causal relationship with another object, but the word cannot stand alone as a concept. This could be fixed by proposing a different node labeled "the separation of double stranded nucleic acids". This would lower the number of edges attached to the node but restrict them to only those edges pertaining to that specific medical concept.

To summarize, 100 nodes were tested and 42 required disambiguation. Out of those, 26 were clearly correct, preferring medical senses as expected. Two were likely overly specific. Five were vague, representing too many possible meanings in the context, and nine were not salient entities.

6 Link Vectors

It seems to be the case that the Gensim LSA are good enough to produce disambiguations for every term that has a single obvious answer, and that further improvement to the system will not rely on changing those vectors but by changes in the task description and the structure of the graph. That being said, I felt like there was something missing in this representation, namely the link structure of the articles. As we've discussed, one of the main advantages of a horizontally-organized knowledge base like Wikipedia is that articles are interlinked with one another. In the task above the vectors were built only with the text of the documents. In other words, the similarity scores are computed without any link knowledge. This seems like a waste of potential. I decided to see what could be done to include these valuable pieces of information.

We can still represent our articles and nodes with vectors. The difference is that this time, each will be represented by incoming links. In the Gensim case, the length of each vector was equal to the length of the vocabulary; it was based on terms. These vectors will be equal to the number of articles instead, where each dimension counts whether or not that one contained an outgoing link to the article in question. This will be even more sparse than the term-based vectors, which is why it will be run through LSA. For performance reasons I limited this experiment to incoming links only, but it would almost certainly be more accurate if the vectors kept track of both incoming and outgoing links. This modification would double the length of each vector, because each article would have an 'incoming' dimension and an 'outgoing' dimension.

Making link vectors for Wikipedia articles is straightforward, though it requires a more accurate representation of links. I mentioned earlier for the commonness measure that we could use just the raw links taken line from line from the Wiki dump. Now we need to keep track of which links come from which article. This means working with the XML tags, which I did with Python's ElementTree package. With this package I could pull out each individual page, discarding user pages, redirects, etc. and keeping only the links. This can be put into a dictionary mapping articles to a list of links. However, that describes a list of outgoing links. To track incoming instead, we need to go through all of the lists and create a new dictionary with the link as the key and the list of articles it came from as the value. In other words, taking the mapping of article > link list and changing it into link > article list.

This dict is all that's needed to create the article link vectors, and then we can use Sci-Kit Learn's SVD module to perform the dimensionality reduction. It's worth mentioning again that this reduction represents a hyperparameter in the number of the dimensions. Here I chose to compress the vectors into dimensions of 300 as that is a common baseline number, but changing this could result in performance increases depending on what we know about the domain and task. Decreasing the number of dimensions has the effect of pooling together a wider variety of semantic information into each, whereas using more results in more specific, but smaller, topics.

6.1 Link Detection

Representing articles with link vectors works fine for the Wikipedia data, but there's a huge problem with the node context information, which is that they don't contain any links! I decided to solve this by creating artificial links, similar to how I created pseudo-documents earlier. That process was easily done by concatenating the edges' text, but creating links is more difficult.

A thorough exploration of the problem is out of scope here, but there have been efforts to do this as far back as Mihalcea and Csomai (2007) with their "Wikify!" project [6], described in section 2. Their idea was to take input documents and do both keyword extraction and disambiguation, thereby enriching any document with artificial links. The authors did not make a Wikify! API, but I found a different, more recent approach with the same idea called Wikifier [11]. Using Wikifier we can pass in the pseudo-documents and retrieve a list of links generated from what the system identifies as the most salient entities.












Text	Annotations			
<p>The beneficial effects of <i>S.boulardii</i> may be due to <u>antitoxin</u> effect , <u>antibacterial activity</u> , modulation of <u>intestinal flora</u> , increasing the short chain <u>fatty acids</u> in lumen , increased <u>enzymes</u> against <u>viral infection</u> , increased <u>IgA activity</u> , and decreased <u>synthesis of inflammatory cytokines</u> . Some elegant works have demonstrated that the use of the bioactive nanobodies for <u>antitoxin</u> , anti-infection , <u>anti-inflammation</u> , or <u>enzyme inhibition</u> is a potentially feasible way for novel <u>therapeutic</u> development (XREF_TABLE) . Some elegant works have demonstrated that the use of the bioactive nanobodies for <u>antitoxin</u> , anti-infection , <u>anti-inflammation</u> , or <u>enzyme inhibition</u> is a potentially feasible way for novel <u>therapeutic</u> development (XREF_TABLE) . It is critical to note that <u>antitoxin</u> is unable to enter poisoned <u>cells</u> and reverse the <u>paralysis</u> . The administration of <u>antitoxin</u> can <u>neutralize</u> unbound <u>neurotoxin</u> , but can not reverse existing <u>symptoms</u> [XREF_BIBR] . The <u>antitoxin</u> can arrest the progression of <u>paralysis</u> and decrease the duration of <u>paralysis</u> and needs good ventilation support system</p>	PR	Annotation	Annotation (en)	
	0.0232	Antitoxin 	Antitoxin	>>
	0.0204	Enzyme 	Enzyme	>>
	0.0144	Enzyme inhibitor 	Enzyme inhibitor	>>
	0.0134	Cytokine 	Cytokine	>>
	0.0117	Antibiotic 	Antibiotic	>>
	0.0114	Gut flora  	Gut flora	>>
	0.0113	Neurotoxin 	Neurotoxin	>>
	0.0113	Paralysis 	Paralysis	>>
	0.0111	Fatty acid 	Fatty acid	>>
	0.0109	Anti-inflammatory 	Anti-inflammatory	>>

Figure 4: Example of the Wikifier system on the pseudo-document for "antitoxin". [11]

Figure 4 shows an example of Wikifier on a somewhat short pseudo-document made from the edges surrounding the lum.ai node labeled 'antitoxin'. It performs well, detecting plenty of links from the passage. Any detection task is essentially a set of subjective opinions, because deciding which elements of the text should be linked requires an idea of their contextual salience. This is an issue both for computational approaches and for the human editors who make the linking decisions while writing an article. Wikipedia has a manual of style for these writers with suggestions on what should and should not be linked. For example, they recommend linking a language like Tohono O'odham but not one like Spanish, because most users will be less aware of the former and therefore may be more interested in finding out further information about it. There are many of these guidelines and they help to ensure that an article is not "overlinked" or "underlinked".

In our case, it is not crucial to link every single potential entity. We can see by looking at the figure that the system succeeds in linking a large percentage of the passage and that these links are a good indicator of the medical domain. This means they should be good for constructing similarity vectors. Interesting, Wikifier also includes links to DBpedia concepts (the orange D logos in the annotation window), allowing for comparison between Wikipedia and DBpedia.

7 Evaluation

The following table shows a comparison of results for the three methods on a handful of test nodes. It contains the entity nodes from the lum.ai graph along with the three top scoring Wikipedia article matches for each measure.

lum.ai node entities	Commonness	Term Vectors	Link Vectors
'spider'	Spider Spider (2002 film) Roadster (automobile)	Spider Angioma Spider Monkey Cultural depictions of spiders	#1 Johann Wilhelm Meigen Zones of Nepal
'welfare'	Welfare Welfare (financial aid) Social welfare provision	Quality of life Welfare state Welfare	Market (economics) Economic growth Natural resource
'low temperature'	Low temperature Cryogenics Low temperature physics	Hypothermia Cryogenics Refrigeration	Red blood cell Marrow_node Antibody
'marrow'	Marrow (comics) Bone marrow Marrow	Bone marrow Marrow (vegetable) Marrow (comics)	Blood plasma Antibody Macrophage
'darkness'	Darkness Darkness (2002 film) Darkness (poem)	Darkness The Darkness series Ignorance	Isfahan province Cambridge University Press N-terminus
'amnesia'	Amnesia Amnesia (nightclub) Amnesia (5 Seconds of Summer song)	Amnesia Psychogenic amnesia Retrograde amnesia	Action potential Hippocampus Multiple Sclerosis
'tissue death'	Necrosis Tissue necrosis Gangrene	Infarction Necrosis Hypoxia (medical)	1001 Albums You Must Hear... HipHopDX Television in Canada
'life cycle'	Biological life cycle Life cycle Life cycle (biology)	Biological life cycle Cell cycle Lambda phage	antitoxin_node genetic testing_node mRNA
'cognitive processes'	Cognition Cognitive processes Mental function	Cognitive psychology Mental process Cognition	Thalamus acetylcholine paranoia
'surface area'	Surface area Area Surface area to volume ratio	Surface area Body surface area Area	low temperature_node marrow_node biochemical analysis_node
'family history'	Family history Family history (medicine) Family history society	Family history (medicine) Heredity Genealogy	Neurology Cardiovascular disease Medication

This graph has a lot of information in it, but there are certain patterns to look for. We can go column by column to see how the three different methods work and what kind of answers they gravitate towards.

The commonness measure works well as a baseline, in most cases choosing the correct sense. An important pattern here is that many of the winning terms are exact matches. For example, 'spider' links to the generalized Wikipedia article for spiders. 'welfare' links to Welfare, 'amnesia' to Amnesia, etc. When a generalized article exists they tend to be weighted very highly; 'spider' links to 48 different articles, but Spider makes up 91.16% of those links. The takeaway is that the commonness measure does not learn any representation of the data, but still ends up being accurate because of the high probability associated with more general topics.

Moving to the term vector column, we can see why it is important to learn from the data. Though the intended sense is often the most common one, there are still many occasions where this isn't true. The pattern to look for in this case is that the senses corresponding to medical terms often win, even in cases where a more general article exists. The row for 'medical history' shows this very clearly, wherein Family history (medicine) wins over Family history, even though Family history is the winner in the commonness column. The comparison between the two columns for this term shows that the term vectors are learning a semantic representation from the contextual pseudo-documents that steers them strongly toward the medical domain.

See also the second and third choices for this term, Heredity and Genealogy, showing that this measure groups related concepts together. In the commonness column, the top three choices for each term are often totally unrelated, because

link probability has nothing to do with semantic content. For 'amnesia', both measures chose Amnesia as the top term, but looking at the top three choices shows that all three of the term vector choices are in the medical domain and represent more specific types of amnesia.

This suggests that the term vectors are better and furthermore that it might be best to consider multiple disambiguations for each term rather than just one. After all, the node in lum.ai's graph for 'amnesia' deals with the general topic of amnesia, but the appearance of Psychogenic amnesia and Retrograde amnesia in the top three shows that these terms are talked about extensively in the node's neighborhood. That is a valuable source of information that shouldn't be neglected.

The last column is for the link vectors, and the results here are not very good at all. The top choices are often completely unrelated, e.g. Isfahan province for 'darkness' and #1 for 'spider'. But there are still some patterns here that shed some light on why this is happening.

The first observation is that many of the high-ranking answers are still in the medical domain, even if they are not related directly to the query term. 'marrow' returns the Wiki articles for Blood plasma, Antibody and Macrophage. These are clearly not correct disambiguations for the term, but they are at least in the same field. This indicates that the link vectors are indeed learning a semantic representation for the nodes by looking at the contextual data surrounding them; the difference is that they simply don't do it as well as the term vectors.

The other pattern to note in this column is that many of the top choices point to other nodes used in the comparison rather than to Wiki articles. To clarify, "Low temperature" is a Wikipedia article, but "low temperature_node" is the name I gave to the link vector made from the query term 'low temperature'. In fact, all three of the top answers for 'surface area' do this. There seems to be something in these vectors that connects them even though their semantic content is different, though it's not immediately clear what that property is. Theoretically, the links in 'surface area' and those in 'low temperature' should not show much overlap and therefore shouldn't score closely together.

I drew two main conclusions from the data in this comparison. The first is that it is important to have a representation of semantic meaning for the nodes. This is what leads the term vectors to the medical domain for 'welfare' with the correct answer, Quality of life. The commonness measure stays in the economic and social domains for the same term.

The second conclusion is that quantity is more important than quality. The links included in an article are guaranteed to be salient and have a connection to the meaning of the document in question. Counting every term in the document means we are including less helpful text such as function words, ambiguous words, and less salient words. But the sheer number of terms in an average Wikipedia article seem to be enough to encode accurate term vectors that perform much better than their link-only counterparts.

The Wikifier had a limit on the input size, so all of these nodes were of "medium" length. I predict that with a custom implementation for the link detection task, the nodes with thousands of edges would have much more accurate predictions. In addition, shorter nodes could have their pseudo-documents augmented with second-order edges to make them longer. Regardless, in its current state I posit that term vectors are by far the best choice for the task of disambiguation to Wikipedia.

8 Conclusion and Future Work

When starting with lum.ai I knew that I would be working on their graph network. We decided that a good task would be to do a sort of survey of methods for entity linking, specifically by disambiguating to Wikipedia. This was my first time working with both graphs and large data sets, and I learned a lot about them.

One of my strongest takeaways was that big data really is big; I ran into frequent roadblocks with the time and space limitations of my personal hardware. As my goal here was to test rather than develop a system ready for deployment, I got around this by taking a few shortcuts, such as pruning the number of articles if they didn't have a certain length. A suggestion for myself and for any future work on this would be to not shy away from using external high-performance computing and cloud storage options. The extra time spent up front to familiarize myself with these resources would have saved a huge amount of time and trouble down the road.

My results, however, were satisfying. I built three working models for the task: a baseline commonness measure using joint probabilities, link vectors with artificially created node links, and most importantly, term vectors with concatenated pseudo-documents. The results for this last method were good and I feel comfortable recommending this approach to others working on the task. Given a node representing a clear semantic entity, running a Euclidean-distance similarity on each Wikipedia article vector consistently produced the correct results, preferring in-domain answers and on the other hand giving very low scores to undesired senses such as those representing athletes, songs, etc. with superficially-similar names.

Future work on this area seems to revolve around the nodes themselves. The quality of results depends on the length of the pseudo-documents. As mentioned, shorter documents can be augmented with farther-away edge connections. The danger in this is that the farther away we step from the original node, the weaker the semantic ties become. There may be a point of diminishing returns for this method.

Longer nodes mean more connected edges. This can have the side effect of "watering down" the semantic content because the contextual evidence is coming from so many different sources. The added length insures that the model will point in the right direction but could cause a failure to find a suitably generic disambiguation, as we saw for the node 'model'. There might be a point where a node is considered too popular and should be broken apart into several smaller nodes that can represent more specific senses, like 'physical model' vs 'computational model'.

One area of future work involves the vectors themselves. I created separate methods for the terms and the links, but of course the best representation would be one that makes use of both. There are probably cases where the links would be the deciding factor in choosing the best sense. Vectors could be made with both terms and links, and perhaps a machine learning model could learn the weights that best balance the contributions that each type of information makes.

In the beginning I made a comparison between what I called horizontal and vertical knowledge bases. I narrowed my scope for this research to just a horizontal KB, but I think it's important to note how much work has gone into vertical KBs like DBpedia and Wikidata. These are still-evolving resources that have large and detailed ontologies, are user-friendly to access, and are specifically designed to be machine-readable through the use of the RDF framework. Anyone interested in this task should consider them, either by themselves or in some sort of combination with horizontal knowledge bases.

Last, graph structure is a crucial component that I did not fully explore here, though I mentioned a couple of prior works in section 2 [1, 12]. I did attempt to implement a framework called node2vec [8]. As the name implies, it takes the graph as input and creates a vector representation that encodes the structure of each node. Nodes aren't just a collection of edges; the centrality of the node in the overall structure, the depth vs. breadth of its connections, and if applicable the direction of its edges are all important information that could inform the disambiguation task. My own implementation of node2vec was stymied by space and time constraints and I decided to leave it out for now. It's worth noting however that Wikipedia itself is a giant graph, with articles as nodes and links as edges. One interesting idea is to model this graph and get node2vec representations for both Wikipedia and lum.ai's nodes. We could then get similarity scores between these node structure vectors just as we did for the terms and links.

In summary, the problem of disambiguation and entity linking is important in multiple ways. As an academic research topic it is a testbed for many different approaches, from simple link counts and intersections to cutting-edge vector representations to the proposed future of Web 3.0. And for real-world applications it is greatly beneficial to link data to knowledge bases like Wikipedia. These resources help to clarify node content, allow for easy exploration of related topics, and are constantly growing. I'm thankful to lum.ai for the opportunity to gain hands-on experience with this task and to my professors and peers for teaching me how to do it.

References

- [1] Moro, Andrea et al. "Entity Linking meets Word Sense Disambiguation: a Unified Approach." *Transactions of the Association for Computational Linguistics* 2 (2014): 231-244.
- [2] "Size of Wikipedia" *Wikipedia*, Wikimedia Foundation, 9 Apr. 2019, en.wikipedia.org/wiki/Wikipedia:Size_of_Wikipedia.
- [3] "Semantic Web." W3C, World Wide Web Consortium, 9 Apr. 2019, www.w3.org/standards/semanticweb/.
- [4] "Lum.ai." *Lum AI*, 9 Apr. 2019, lum.ai/.
- [5] Bunescu, Razvan, and Marius Paşca. "Using encyclopedic knowledge for named entity disambiguation." *11th conference of the European Chapter of the Association for Computational Linguistics*. 2006.
- [6] Mihalcea, Rada, and Andras Csomai. "Wikify!: linking documents to encyclopedic knowledge." *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*. ACM, 2007.
- [7] Milne, David, and Ian H. Witten. "Learning to link with wikipedia." *Proceedings of the 17th ACM conference on Information and knowledge management*. ACM, 2008.
- [8] Grover, Aditya, and Jure Leskovec. "node2vec: Scalable feature learning for networks." *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2016.
- [9] "How Are Principal Component Analysis and Singular Value Decomposition Related?" *Intoli*, 9 Apr. 2019, https://intoli.com/blog/pca-and-svd/.

- [10] Luo, F., Valenzuela-Escárcega, M., Hahn-Powell, G. and Surdeanu, M., Scientific Discovery as Link Prediction in Influence and Citation Graphs., *Text Graphs Workshop*, 2018.
- [11] Janez Brank, Gregor Leban, Marko Grobelnik. Annotating Documents with Relevant Wikipedia Concepts. *Proceedings of the Slovenian Conference on Data Mining and Data Warehouses (SiKDD 2017)*, Ljubljana, Slovenia, 9 October 2017.
- [12] Hoffart, Johannes, et al. "Robust disambiguation of named entities in text." *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2011.
- [13] Rehurek, Radim, and Petr Sojka. "Software framework for topic modelling with large corpora." In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. 2010.