WIKIPEDIA ENTITY RETRIEVAL WITH WORD AND ENTITY EMBEDDING SPACE ALIGNMENT

By

Hongshuo Zhang

Senior Thesis in Computer Engineering

University of Illinois at Urbana-Champaign

Advisor: Kevin Chen-Chuan Chang

May 2021

Abstract

User queries used to search on Wikipedia might not always correspond to a Wikipedia article. Searching for such queries on the built-in search engine for Wikipedia does not return as many relevant Wikipedia articles as expected, since it is only text-based, and it overlooks the relationship between different Wikipedia articles. To improve the results of the Wikipedia search engine, with the help of the Wikipedia Link Graph, we propose a simple word – entity embedding space alignment model for searching relevant Wikipedia articles using fringe keywords from the Computer Science domain. By using this model, our problem can be reduced to finding the closest neighbors to a user query translated from the word space to the entity space.

Due to a lack of dataset available for this purpose, a custom dataset constructed from Wikipedia2Vec is used to evaluate the model. Empirically, our word – entity alignment model does not always fair better than the text-based models.

Subject Keywords: Information Retrieval; Search Engine

Contents

1. Introduction 1
    1.1 Thesis Roadmap 3
2. Literature Review 4
    2.1 Word – Entity Embeddings 4
    2.2 Prior Work Inspirations 5
3. Model Overview 6
    3.1 Word Embedding 6
    3.2 Entity Embedding 8
    3.3 Word-Entity Alignment 11
        3.3.1 Negative Sampling 12
        3.3.2 Word-based Reordering 13
4. Implementation Specifics 15
    4.1 Word Embedding 15
    4.2 Entity Embedding 15
    4.3 Word – Entity Alignment 16
5. Results 17
    5.1 Quantitative Evaluation 17
    5.2 Qualitative Evaluation 19
6. Future Work 21
7. Conclusion 22
References 23
Appendix A 24
Appendix B 25

1. Introduction

Many commercial search engines follow a similar pipeline. Most of them infer relevant documents to a user query largely based on the number of keywords matched from all document fields. The idea behind such inference is that if more words from the user query are present in a document, the document is more likely to be relevant. In practice, this kind of approach does work for these search engines.

Table 1: Top 5 search results from searching 'query transformation' on both the Wikipedia Search Engine and Google. Results in red are judged *not* relevant to the user query.

Wikipedia Search Engine	Google Within Wikipedia
Query String	Query Rewriting
Query Rewriting	QVT
QVT	Query Language
SPARQL	Relational Algebra
OGC GeoSPARQL	Transformation Language

Table 1, however, illustrates that the built-in Wikipedia search engine fails to deliver good results for fringe keywords compared to the established commercial search engines like Google. Compared to commercial search engines, the Wikipedia search engine only contains around 10 million documents, which is a very tiny fraction of the number of documents that commercial engines index. With such a small number of documents indexed, the model is more likely to infer irrelevant documents that merely mention some of the query keywords. In addition, since the Wikipedia search engine is powered by Elasticsearch, there exists no mechanism, like Google's PageRank or learning to rank algorithms, that fixes the mistakes that text-based models infer. Therefore, understandably, the built-in Wikipedia search engine fares much worse than commercial search engines in inferring semantically relevant result documents from fringe user keyword queries.

Although the Wikipedia search engine only uses the textual component of Wikipedia articles, there are several other information-rich resources from Wikipedia that we could utilize. Notably, Wikipedia contains a rich relationship structure between entities from Wikipedia. Incorporating the relationship between Wikipedia entities not only solves the problem of inferring irrelevant documents with common keywords as the user query, but it also takes relevant documents without common keywords as the user query into account.

However, compared to text-based searching models, we cannot directly associate user queries to Wikipedia articles since the Wikipedia relationship graph only captures the relationship between Wikipedia entities. There is still a knowledge gap between keywords and Wikipedia entities that needs to be aligned. Therefore, instead of using a text-based model like Elasticsearch to search for relevant articles, we could also align the knowledge of words and Wikipedia articles.

In this thesis, we propose a simple and lightweight method to align word – entity embedding spaces. In this model, the task of finding the most relevant Wikipedia articles to a user query can be reduced to the task of finding the nearest Wikipedia entities of the user query translated from the word embedding space to the entity embedding space.

Our model is largely based on three components: word embeddings, entity embeddings, and a linear regression layer that maps the two. We then tested it on a custom dataset built from Wikipedia2Vec quantitively and we also analyzed the results generated from ten fringe keywords generated from the Computer Science to evaluate our model qualitatively. Empirical results show that our model does not always fare better than text-based search algorithms like Elasticsearch.

1.1 Thesis Roadmap

In chapter 2, we will identify relevant prior works and discuss the similarities and differences to our problem. In chapter 3, we will describe our model in detail and discuss how are these design decisions justified. In chapter 4, we will discuss how are our model implemented. In chapter 5, we will compare and analyze the results generated from our model to the results generated from the text-based baseline. In section 6, we will identify future directions to further improve our model. Finally, in section 7, we will end the thesis with a short conclusion.

2. Literature Review

Since our work involves retrieving relevant Wikipedia articles to a user query by aligning word and entity embedding spaces, there is one area of research that relate to our problem: word – entity embeddings. Specifically, works in word – entity embeddings focus on how words and entities are embedded so that semantically similar words and entities are placed together and vice versa.

2.1 Word – Entity Embeddings

There are many studies that either focus on learning word and entity embeddings in a unified vector space or engineer their word-entity embedding models to work for a variety of downstream tasks, such as knowledge graph link prediction [6, 7, 10], named entity disambiguation [8], entity linking [2], knowledge graph triplet classification [6, 7, 10], et cetera. Regardless of their downstream tasks, the methods of bridging entities and words can be categorized into the following two ways.

Many word – entity embedding models first train the word embedding and the entity embedding separately, then utilizes a source where words and entities coexist to align the word and entity embedding spaces. Wikipedia2Vec [8] aligns the two embedding spaces by considering the context words of a Wikipedia in-text anchor, which directs the anchor text to a specific entity. Although the two embedding spaces are aligned, we observed that many of the nearest Wikipedia entities to a word in Wikipedia2Vec are not very semantically relevant, largely because of the noisiness that exists within the Wikipedia anchor context. The model only aligns the two embedding spaces to improve the quality of embeddings for words from the Wikipedia corpus, and for the entities from the Wikipedia Link Graph. A more relevant model to our problem, MPME [1], aligns the word and entity embedding space by considering the duality of Wikipedia anchors, where the anchor can either represent a sequence of words or a Wikipedia entity. Because of the duality of the anchor text, each anchor text is annotated with the sense that it represents. Therefore, the model takes advantage of this by creating multiple instances of the same anchor text in the text embedding space and linking them to different entities in the entity embedding space depending on their senses. However, this setup would not work for our problem since it is very difficult to tag short user queries with the correct sense.

Other word – entity embedding models trains on resources where words and entities co-exist directly. KEWER [5] randomly replaces entries from the knowledge graph with their surface forms. For example, the entity 'Computer science' might be replaced with its entity name 'Computer science', then the model randomly walks on the graph to embed the words and entities in the same space. However, we observe that such approaches perform even poorer than aligning two embedding spaces together, possibly because of the noisiness of the Knowledge Graph itself.

2.2 Prior Work Inspirations

Our word – entity alignment model largely resembles the alignment model from MPME [1]. However, instead of mapping the two embedding spaces with the word – entity mapping pairs where both the word and the entity represent the same sense, we mapped the two embedding spaces with the word – entity mapping pairs where the word can directly refer to the entity. A more specific overview of the model implementation will be available in the next section.

3. Model Overview

Our word – entity embedding space alignment model consists of three components: word embeddings, entity embeddings, and, the most important component of our model, word – entity Alignment. Given a Wikipedia Link Graph *G,* Wikipedia Text Corpus *C,* and Word Space Corpus *WC*, we aim to learn the translation function for keyword embeddings so that the translated keyword is closer to the entity that it refers to, while the translated keyword is further from the entities that it has no relation with.

3.1 Word Embedding

We trained our word embeddings with Word2Vec [3] While it was published many years ago, Word2Vec is still one of the most popular word embedding model used today due to its training efficiency and its decent performance. Word2Vec relies on an intuition that closer keywords are often placed in similar context and vice versa. For example, the keyword 'python' is similar to the keyword 'java' because both keywords commonly refer to the python programming language and the java programming language, which are often mentioned around similar contexts in a sentence.

Two variants were proposed in Word2Vec, namely the Continuous Bag of Words Model (CBOW) and Skip-gram (SG). Both variants share the same intuition, that is to model the predictive relations among a sequence of words. However, their exact implementation differs. The CBOW model aims to predict the current word based on the context words that surround it. Equation 1 is the objective function of the CBOW model, in which ***D*** refers to a set of all words from the word sequence, ***C*** refers to the context words of the current word $\boldsymbol{w_I}$.

$$\mathcal{L} = \sum_{w_i \in \mathcal{D}} \log P(w_i | \mathcal{C}(w_i)) \quad \textbf{(1)}$$

While the SG model aims to predict the context words that surrounds the current word. Equation 2 is the objective function of the SG model, in which ***D*** refers to a set of all words from the word sequence, ***C*** refers to the context words of the current word $\boldsymbol{w_I}$.

$$\mathcal{L} = \sum_{w_i \in \mathcal{D}} \sum_{w_o \in \mathcal{C}(w_i)} \log P(w_o | w_i) \quad \textbf{(2)}$$

In practice, the CBOW model is much easier to train than the SG variant, since CBOW is only responsible for predicting one word based on the surrounding words. However, from our observation, the CBOW model does a worse job in predicting the semantic relationships between different keywords, which is what we will be relying on for our overall model. Therefore, we used the SG variant of Word2Vec to train our word embeddings.

Within our SG model, we also automatically chose negative samples to ensure that words that are not semantically related are kept further apart. Negative sampling is not only used to replace a large amount of training data by randomly choosing keyword pairs that are not relevant, but also to speed up the training process by only updating a fraction of the model's weights.

Finally, the quality of word embeddings not only depends on the Word2Vec variant, or whether we use negative sampling, but also depends on the corpus used for the Word2Vec Model. Since our model is focused on aligning CS domain keywords to Wikipedia entities, therefore, two major factors are considered in constructing our text corpus.

Firstly, the relation between keywords and phrases needs to be semantically related with respect of the computer science domain. This needs to be accounted for since a vast majority of keywords that may have connections to the Computer science domain often have different meanings outside of the computer science domain. For example, the word 'java', often refers to a programming language within the computer science domain. Therefore, the most relevant keywords to the word 'java' should be a set of programming languages. However, outside of the computer science domain, the word ‘java’ can also refer to a large island in Indonesia. Therefore, for our corpus, we used abstracts from computer science papers to make sure that the different keywords are semantically related with respect to computer science.

Secondly, we notice that a vast majority of Wikipedia entities have names of more than one word. To make sure that keywords from the word embedding space can be mapped to Wikipedia entities with multi-word names, we need to calculate the word embeddings for multi-word sequences (i.e., phrases). However, Word2Vec only supports unigram embeddings. Therefore, to retrieve word embeddings for such multi-word sequences, we merged words from each key phrase from a set of key phrases within the computer science domain in our corpus to a single unigram.

3.2 Entity Embedding

We trained our entity embeddings with Wikipedia2Vec. (Yamada et al, 2018). Specifically, the entity embeddings from Wikipedia2Vec are jointly learned by optimizing the word embeddings, the entity embeddings, and the anchor context. It is a popular framework that improves the quality of word embeddings and entity embeddings **separately** by considering information contained from the other Wikipedia representation. Although, Wikipedia2Vec also jointly learns word and entity embeddings in the same embedding space, we observed that, in the Wikipedia2Vec embedding space, the closest Wikipedia entities to a Wikipedia keyword is not as semantically related to the keyword itself. Therefore, we are not able to directly apply this framework to achieve our goal.

Wikipedia2Vec consists of three components: word embeddings, entity embeddings, and the anchor context. While the word embeddings and the entity embeddings are trained separately, the Wikipedia2Vec framework aims to jointly optimize the word embedding model, the entity embedding model, and the anchor context model where entities and words coexist.

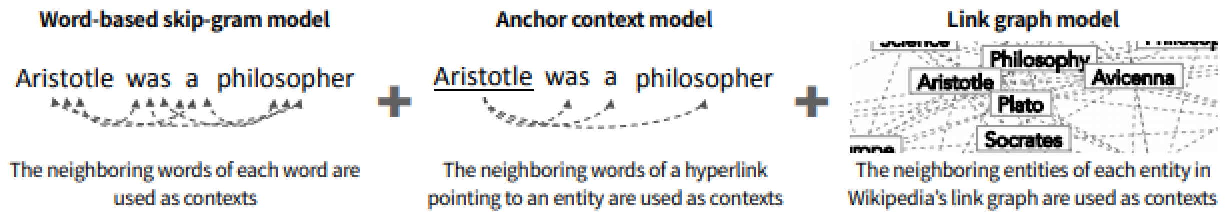

Word-based skip-gram model
Aristotle was a philosopher
The neighboring words of each word are used as contexts
Anchor context model
Aristotle was a philosopher
The neighboring words of a hyperlink pointing to an entity are used as contexts
Link graph model
Philosophy
Aristotle
Avicenna
Plato
Socrates
The neighboring entities of each entity in Wikipedia's link graph are used as contexts

Figure 1: An Overview of the Wikipedia2Vec framework.

The word embedding model for Wikipedia2Vec also uses the Skip-gram model proposed in Word2Vec. Namely, the words that surround a current word are used as contexts. The entity embedding model takes the Wikipedia Link Graph, the graph constructed from connecting anchor connections between different Wikipedia entities as nodes, as the corpus. The Wikipedia entities are embedded in a similar fashion as Skip-gram. For a collection of all edges ***E*** and a collection of all neighboring entities $\boldsymbol{C_e}$, as demonstrated in equation (3), we aim to maximize the conditional log probability of surrounding Wikipedia entities given the current Wikipedia entity.

$$\mathcal{L} = \sum_{e_i \in E} \sum_{e_s \in C_{e_i}} \log P(e_s | e_i) \quad \textbf{(3)}$$

The anchor context aims to place similar words and entities next to each other in the same embedding space, the anchor context model bridges the words from the Wikipedia text corpus and the entities from the Wikipedia Link Graph by using the hyperlink behind an anchor text and its surrounding words as context. Equation (4) shows that, for a collection of Wikipedia in-text anchors ***A*** and a collection of surrounding words ***C***, we aim to maximize the conditional log probability of surrounding context words given the current Wikipedia entity referenced from the anchor text.

$$\mathcal{L} = \sum_{(e_i, C) \in A} \sum_{w_c \in C} \log P(w_c | e_i) \quad \textbf{(4)}$$

Finally, the Wikipedia2Vec framework is jointly optimized by maximizing the combination of equation (2), equation (3), and equation (4).

The original authors of the Wikipedia2Vec framework did not evaluate the semantic similarity between words and entities in the same embedding space, nor did they analyze the reason behind the poor semantic similarity between similar words and entities. We believe that the root of the problem stems from the noisiness of the anchor context model. Consider the following sentences taken from the Wikipedia page for Computer engineering as an example.

Wilhelm Schickard designed and constructed the first working mechanical calculator in 1623. **(5)**

Computer Science departments with a mathematics emphasis and with a numerical operation consider alignment with computational science. **(6)**

Sentence (5)[1] contains two anchor texts – Wilhelm Schickard and Mechanical calculator. Per Wikipedia, for the anchor text Wilhelm Schickard, the context words of it do not accurately reflect the relationship between the entity and words since the sentence will also be semantically and syntactically coherent if we replace Wilhelm Schickard with words like ‘some person’ or ‘some company’, whose semantic meanings are very distant from the semantic meaning of Wilhelm Schickard. The same thing can be said for mechanical calculator in sentence (5) and computational science in sentence (6)[2], many other semantically non-similar words or phrases to the anchor texts can also replace the anchor texts without rendering the sentences semantically and syntactically incoherent.

Other Word2Vec based models also suffer from the same problem. However, for other Word2Vec based models, the effect of this problem is mitigated by the sheer number of words from the text corpus. For Word2Vec, words are placed in many different contexts and surrounded by different keywords. Thus, the Word2Vec model will have a more holistic view of what a keyword should be semantically close to.

[1] Referenced from https://en.wikipedia.org/wiki/Computer_science#History

[2] Referenced from https://en.wikipedia.org/wiki/Computer_science#Etymology

However, each anchor text, and by its extension its referent entity, appears in much fewer contexts because it occurs less frequently. Therefore, the relations between words and entities appear much less semantically similar.

3.3 Word-Entity Alignment

After training both word embeddings and the entity embeddings, we align two embedding spaces in order to place the words next to their semantically similar entities. The intuition behind such alignment is inspired from the finding that multi-language words represented in different embedding spaces follow a similar distribution, even though the words are in represented in different human languages. Therefore, multi-language embedding spaces can be mapped linearly with a translation matrix by minimizing the sum of squared errors[4]. We assume that relationship between words and entities is akin to the relationship between words in represented in multiple languages, thus the intuition also applies. Figure 2 illustrates our intuition for our word – entity alignment model. The dashed line in cyan indicates a confirmed edge, an edge between a keyword and the Wikipedia entity that it directly refers to. The dotted line in dark blue indicates the translation from the word embedding space to the entity embedding space. In this figure, we can clearly see that the relation between different entities is just a mirrored version of the inter-keyword relations. Therefore, we can correctly infer the expected referent entity for cse – Computer science and engineering - by only learning a subset of the word – entity mappings.

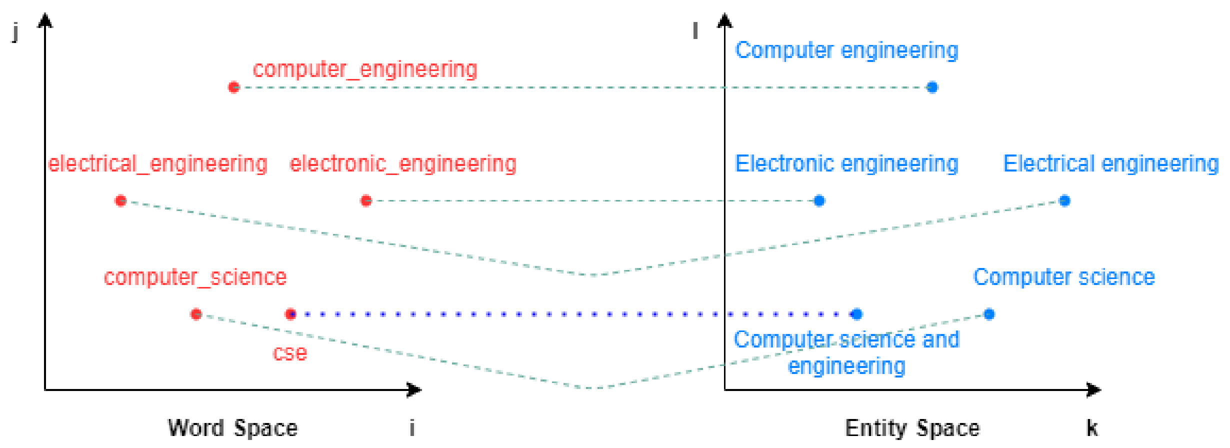


Figure 2: An idealistic overview of our alignment model

In order to align words with entities, we implemented a linear regression model to learn the linear transformation matrix between the two embedding spaces. Word – entity mapping pairs used for training is extracted by finding each keyword with a direct Wikipedia entity correspondence from the Word2Vec dictionary as well as the redirect links extracted from the Wikipedia database. Equation (7) shows how our alignment model is optimized: For a collection of words ***W***, a collection of entities ***E,*** and a function for expected cosine similarity ***ex(w, e)***, the model is optimized by tuning the actual cosine similarity between the translated word vector and the entity vector towards the expected cosine similarity.

$$\mathcal{L} = -\sum_{w \in W} \sum_{e \in E} |ex(w,e) - \frac{\hat{v_w} \cdot v_e}{||\hat{v_w}|| \times ||v_e||}| \quad \textbf{(7)}$$

In our attempt to align the two embedding spaces, we also experimented with two variants to our model: negative sampling and word-based reordering.

3.3.1 Negative Sampling

Inspired by the negative sampling technique used in Word2Vec, we used a similar technique to fine-tune our embedding translation function by randomly selecting irrelevant entities from the collection of all entities. The expected similarity function of our objective function outputs 1.0 if the word - entity pair comes from the training dataset and 0.0 otherwise. Although, it is possible that some of the entities chosen as negative samples are relevant to the source keyword, it is still highly unlikely since our entity collection consists of millions of entities. In our experiment, we tried different negative samples per source keyword from a word – entity mapping to find out whether negative sampling improves the quality of alignment, and how many negative samples per source keyword should we choose.

3.3.2 Word-based Reordering

For many fringe user-queries, many Wikipedia articles retrieved by merely finding the nearest neighbors in the entity embedding space do not appear semantically relevant to the user query itself. For example, within the computer science domain, the keyword 'python' predominantly refers to the entity Python programming language. However, in many cases, after translating the keyword embedding to the entity embedding space, the closest entities to a keyword might be a collection of other languages, like Java programming language, JavaScript, or Scala. These programming languages, although also very semantically relevant to the user query, are not as relevant as the entity that the user query should be semantically relevant to. Thus, we acknowledged that it might be not sufficient to merely find relevant entities based on cosine similarity.

Therefore, to remedy this issue, after finding the closest entities to a translated keyword embedding in the entity embedding space, we added a naïve word-based candidate reordering step to reorder the results based on words in the user query and words in each Wikipedia entity's title. The underlying assumption to the reordering step is that relevant entities are not only positioned in similar neighborhoods in a knowledge graph, but also have semantically similar words that make up both the user query and the name for each Wikipedia entity.

To implement the word-based reordering, we first split both the user query and each Wikipedia title into sequences of individual words. Then, we generate the word-level embeddings for both the user query and the Wikipedia entity name by aggregating the embeddings for all words that make up the user query or a Wikipedia entity title.

After generating a new word-level embedding for both the user query and each Wikipedia entity title, we then calculate the cosine similarity between two embeddings. The calculated cosine similarity between the two embeddings is then used as a reranking score between the user query and each Wikipedia entity.

4. Implementation Specifics

4.1 Word Embedding

Word embeddings are trained using the gensim[3] implementation of Word2Vec. Many training parameters of Word2Vec were set to default values, besides the number of workers and the specific variant of Word2Vec used. The number of workers is set to 5 to allow parallel threads to speed up the training process, while we used the Skip-gram variant in lieu of the Continuous Bag of Words variant (CBOW) for the reasons outlined in section 3.1. The training corpus for the Word2Vec model comes from the publication abstracts from all subfields [4]of Computer Science. The corpus is tokenized into different sentences using the sentence tokenizer from NLTK[5] using the default settings, punctuations are removed, all letters are lowercased, potential computer science key phrases from a given computer science keyword list are merged into one token, finally, each individual sentence is tokenized into a sequence of word tokens using the word tokenizer from NLTK. In total, 2.09 million sentences are indexed by Word2Vec, 152,317 words with occurrences greater than five are included in the model vocabulary. The training takes around 50 minutes, and the model files occupy 132.4 MB in memory.

4.2 Entity Embedding

Entity embeddings are trained using the Wikipedia2Vec software using the default settings. We used the March 2021's Wikipedia database release as the corpus to train our model. The training process takes 30 hours on a 16-inch MacBook Pro, consists of 2.69 million Wikipedia entities, and all Wikipedia2Vec database files occupy 22.98 GB in memory. Then, we extracted entity names and entity embeddings,

[3] https://radimrehurek.com/gensim/models/word2vec.html

[4] Available in Appendix A

[5] https://www.nltk.org/

converted them into NumPy arrays, and saved them in two separate files, which occupy 3.05 GB in memory all together.

4.3 Word – Entity Alignment

The alignment between the word embedding space and the entity embedding space is done by training a linear regression model that learns the linear translation function from the word embedding space to the entity embedding space. Our model, implemented with the PyTorch package, contains one linear layer that has 100 dimensions on both sides and uses Adadelta as the optimizer. The model is then trained for 200 epochs. For all variants of our model, the training process will take less than ten minutes.

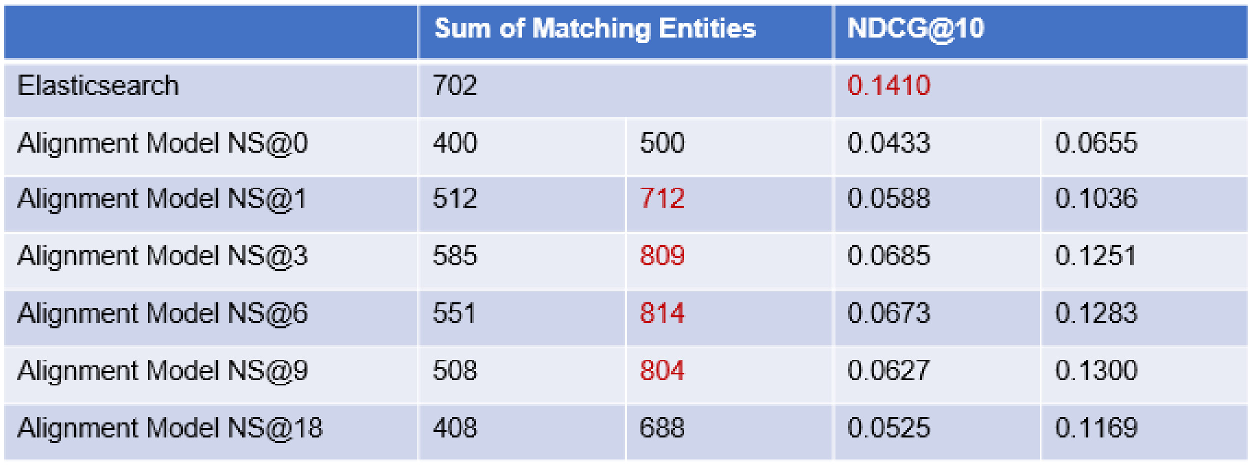
The mapping data between the embedding spaces is obtained from two sources. Direct Wikipedia word – entity correspondences and Wikipedia redirect links. Direct Wikipedia word – entity correspondences are obtained from finding a keyword that has the same name as a Wikipedia entity. Wikipedia redirect links are mined from the same Wikipedia database file we used for training Wikipedia2Vec. In total, there are 31,795 pairs available for training, out of all available training pairs, 981 pairs are randomly selected as testing set. Finally, a batch size of 1,000 samples is used for training, and the number of batches per epoch depends on the number of negative samples per keyword, which are chosen randomly from a collection of all Wikipedia entities.

5. Results

To evaluate our alignment model, we have used two types of evaluation. We first used the dataset generated from Wikipedia2Vec to evaluate our model quantitatively. Then, we randomly picked ten computer science keywords that do not have corresponding Wikipedia articles to evaluate our model qualitatively.

5.1 Quantitative Evaluation

Table 2: Evaluative results for the baseline and all variants of our alignment model. Number after NS@ refers to the number of negative samples per keyword.



	Sum of Matching Entities		NDCG@10	
Elasticsearch	702		0.1410	
Alignment Model NS@0	400	500	0.0433	0.0655
Alignment Model NS@1	512	712	0.0588	0.1036
Alignment Model NS@3	585	809	0.0685	0.1251
Alignment Model NS@6	551	814	0.0673	0.1283
Alignment Model NS@9	508	804	0.0627	0.1300
Alignment Model NS@18	408	688	0.0525	0.1169

We aim to evaluate how well our model performs by ignoring the direct Wikipedia entity that each testing set keyword refers to, finding the closest entities of each testing set keyword by translating the keyword embedding to the entity embedding space, and comparing the model-generated results to the ground truth generated with Wikipedia2Vec. Idealistically, the results generated from the alignment model should be identical to the Wikipedia2Vec ground truth, which indicates that the word embedding space and the entity embedding space are exactly mapped.

We evaluated all variants of our alignment model and the Elasticsearch baseline over the Wikipedia2Vec dataset on two different metrics – Sum of matching entities, which counts the total number of ground truth entities also exist in each result set, and NDCG, which not only takes in the above criteria to account, but also cares about the quality of result set ranking. Six different negative samples per keyword are used, and the model is evaluated both with and without the word-based reordering step.[6]

Table 2 illustrates the evaluation results. For sum of matching entities, we observe that if we train our model with one to nine negative samples per keyword and reorder the results after we find the nearest neighbors, the model performs better than Elasticsearch. However, for NDCG, none of our models performs better than the Elasticsearch baseline.

We attribute the low performance of our alignment model mainly to three reasons. Firstly, there exists no dataset that suits our problem and the ground truth dataset we generated from Wikipedia2Vec is not tailored to our problem either. Although Wikipedia2Vec performs well on capturing the semantic relationship between different entities, and, as shown in table 3, we observed that the closest entities to a less connected entity of the link graph might not always be the most semantically relevant. Therefore, in some cases, evaluation results with Wikipedia2Vec might not be the best indicator of the model performance.

Table 3: Top 5 nearest entities of a frequent entity (Computer science), and a non-frequent entity (Object language) from Wikipedia2Vec

Rank	Entity: Computer science	Entity: Object language
1	Entity: Computer engineering	Entity: λProlog
2	Entity: Software engineering	Entity: Unlambda
3	Entity: Computer science and engineering	Entity: Higher-order abstract syntax
4	Entity: Algorithmics	Entity: Formation rule
5	Entity: Mathematics	Entity: Binary combinatory logic

[6] Left column indicates without reordering, right column indicates with reordering.

Secondly, we acknowledge that the semantic relations between words and the semantic relations between different entities does not exactly mirror each other. In our implementation of word embeddings and entity embeddings, the word embeddings are trained using sentences exclusive to the computer science domain, while the entity embeddings are trained using the entire Wikipedia corpus. Therefore, it is expected that inter-word relations and the inter-entity relations are not the same. If the two embedding spaces exhibit different structures, it is highly likely that the alignment model will not perform as well as expected.

Lastly, we also observe that the source keywords of word – entity mappings in our training data are not evenly distributed in the word embedding space. Ideally, the distribution of source keywords does not matter as much if the semantic relation between different words and entities in both embedding spaces resembles each other. However, this is not the case in our model. Specifically, we notice that frequent words tend to congregate around other frequent words in the word embedding space, which are more likely to be mapped to corresponding Wikipedia entities. Thus, our model will perform better for such keywords since they will have more Wikipedia entities to reference from. On the other hand, infrequent words tend to be further away from words that can be mapped to corresponding Wikipedia entities. Thus, our model will perform worse for such keywords since they will have less Wikipedia entities to reference from.

5.2 Qualitative Evaluation

Besides evaluating both models quantitatively, we also randomly sampled ten Computer Science keywords without direct Wikipedia entity correspondence to find out which type of keywords can be helped by our alignment model and vice versa. Since all keywords we used do not have a Wikipedia entity equivalent, we used the google search engine to represent the ground truth for each keyword instead of Wikipedia2Vec.

Table 4 – 7 in Appendix B outlines relevant Wikipedia articles generated from multiple models for two fringe keywords. acyclic digraph and parsing network. All models used in quantitative evaluation are also evaluated qualitatively.

For the first query, acyclic graph, according to entries from table 4 and table 5 in Appendix B, we observed that results generated from the reordered version of our alignment model are more semantically related to the user query than Elasticsearch. While, without reordering, results generated from our alignment model are noisy, and they are hardly semantically relevant to the user query. For the second query, parsing network, according to entries from table 6 and table 7 in Appendix B, we notice that results generated from Elasticsearch are actually better than the reordered variant of our alignment model, while, results generated from our alignment model without reordering are also not semantically relevant.

The results shown in table 4 – 7 not only reaffirms that our alignment model trained with the reordering step performs better than the model without reordering, it also shows that our alignment model does not always perform better than text-based search engines like Elasticsearch. For user queries like acyclic digraph, we hypothesize that our alignment model performs better than Elasticsearch since, for Elasticsearch, many Wikipedia articles attain a similar BM25 score, which often indicates that many Wikipedia articles uses a similar number of query words to describe articles that are not related to the user query. On the other hand, for user queries like parsing network, we hypothesize that our alignment model performs worse than Elasticsearch since, for elasticsearch, few Wikipedia articles attain a similar BM25 score, which might indicate that the articles with more query word mentions are more relevant to the user query than others.

6. Future Work

Several future directions can be undertaken to improve our model. Firstly, the semantic relationship between different words from the word embedding space and different entities from the entity embedding space can be further fine-tuned to make sure that the underlying semantic structure of two embedding spaces roughly mirrors each other before alignment. For training and evaluating our dataset, crowd sourcing a more accurate training and testing set with human knowledge instead of directly using the one generated from Wikipedia2Vec would also help; we could also experiment with different methods to reorder and to denoise the result set generated from the alignment model not only by considering the semantic similarity between the user query and the name for each Wikipedia entity, but also other available attributes for Wikipedia entities such as the textual description for each Wikipedia entity.

7. Conclusion

In this paper, we reduced the problem of finding relevant Wikipedia entities for fringe keywords to an embedding space alignment problem and studied whether the retrieved Wikipedia entities by alignment are more semantically relevant than the ones retrieved with a traditional text-based search engine like Elasticsearch. We observed that, from both the quantitative results evaluated on a custom made Wikipedia2Vec dataset and the qualitative results retrieved from ten fringe keywords from the computer science domain, our alignment model does not always deliver a better performance than purely text-based models.

We attributed the low performance of our model to three possible reasons: relative dissimilarity between the structure of two embedding spaces, the lack of tailor-made dataset to both train and test our model, and the uneven distribution of source keywords of the word – entity mapping in the word embedding space. To remedy the low performance of our alignment model, we also propose three future directions to improve: word and entity representation, training and testing set quality, and new implementations of text-based results reordering.

References

[1] Y.Cao, L.Huang, H.Ji, X.Chen. Bridge Text and Knowledge by Learning Multi-Prototype Entity Mention Embedding. In ACL, 2017

[2] O.Ganea, T.Hofmann. Deep Joint Entity Disambiguation with Local Neural Attention. In Arxiv, 2017

[3] T.Mikolov, K.Chen, G.Corrado, J.Dean. Efficient Estimation of Word Representations in Vector Space. In ICLR, 2013a

[4] T.Mikolov, Q.V.Le, I.Sutskever. Exploiting Similarities among Languages for Machine Translation. In Arxiv, 2013b

[5] F.Nikolaev, A.Kotov. Joint Word and Entity Embeddings for Entity Retrieval from a Knowledge Graph. In ECIR, 2020

[6] Z.Wang, J.Zhang, J.Feng, Z.Chen. Knowledge Graph and Text Jointly Embedding. In EMNLP, 2014

[7] J.Xu, K.Chen, X.Qiu, X.Huang. Knowledge Graph Representation with Jointly Structural and Textual Encoding. In IJCAI, 2017

[8] I.Yamada, H.Shindo, H.Takeda, and Y.Takefuji. Joint Learning of the Embedding of Words and Entities for Named Entity Disambiguation. In CoNLL, 2016

[9] I.Yamada, A.Asai, J.Sakuma, H.Shindo, H.Takeda, Y.Takefuji, Y.Matsumoto. Wikipedia2Vec: An Efficient Toolkit for Learning and Visualizing the Embeddings of Words and Entities from Wikipedia. In EMNLP, 2018

[10] H.Zhong, J.Zhang, Z.Wang, H.Wan, Z.Chen. Aligning Knowledge and Text Embeddings by Entity Descriptions. In EMNLP, 2015

Appendix A

The ArXiv Paper abstracts are extracted from the following Computer Science subfields:

'cs.AI', 'cs.CC', 'cs.CG', 'cs.CE', 'cs.CL', 'cs.CV', 'cs.CY', 'cs.CR', 'cs.DB', 'cs.DS', 'cs.DL', 'cs.DM','cs.DC', 'cs.ET', 'cs.FL', 'cs.GT', 'cs.GL', 'cs.GR', 'cs.AR', 'cs.HC', 'cs.IR', 'cs.IT', 'cs.LG', 'cs.LO', 'cs.MS', 'cs.MA', 'cs.MM', 'cs.NI', 'cs.NE', 'cs.NA', 'cs.OS', 'cs.OH', 'cs.PF', 'cs.PL', 'cs.RO', 'cs.SI', 'cs.SE', 'cs.SD', 'cs.SC', 'cs.SY'.

Computer Science abstracts are extracted from the 9/26/2020 release of the arXiv database.

Appendix B

Table 4: Search results for the query ‘acyclic digraph’ on alignment models without reordering, Elasticsearch and Google.

Type	1st	2nd	3rd	4th	5th
NS@0	Star world	Voltage graph	Edge and vertex spaces	Bidirected graph	Christofides algorithm
NS@1	Cut (graph theory)	Slack variable	Constraint graph	Kinetic heap	Transitive reduction
NS@3	Cut (graph theory)	Cartesian product	Slack variable	Maximal and minimal elements	Linear inequality
NS@6	Cartesian product	Maximal and minimal elements	Cut (graph theory)	Glossary of graph theory terms	Cayley graph
NS@9	Glossary of graph theory terms	Cut (graph theory)	Graph (discrete mathematics)	Cartesian product	Maximal and minimal elements
NS@18	Cartesian product	Maximal and minimal elements	Complete graph	Glossary of graph theory terms	Metric space
Elasticsearch	Acyclic orientation	Acyclic coloring	Acyclic space	Th (digraph)	Dz (digraph)
Google	Directed acyclic graph	Acyclic graph	Directed graph	Directed acyclic word graph	N/A

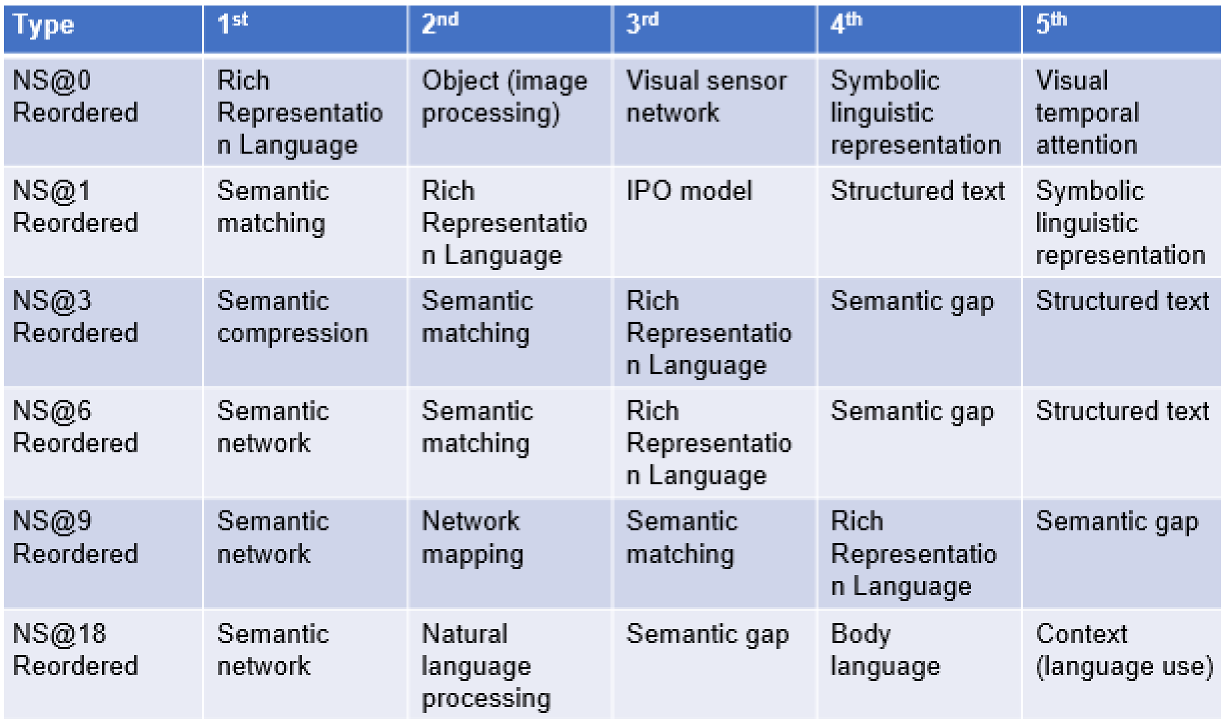
Table 5: Search results for the query ‘acyclic digraph’ on alignment models with reordering

Type	1st	2nd	3rd	4th	5th
NS@0 Reordered	Biconnected graph	Bidirected graph	Acyclic orientation	Aperiodic graph	Permutation graph
NS@1 Reordered	Biconnected graph	Bidirected graph	Cayley graph	Ordered graph	Hypercube graph
NS@3 Reordered	Biconnected graph	Bidirected graph	Cayley graph	Hypercube graph	Graph isomorphism
NS@6 Reordered	Biconnected graph	Bidirected graph	Halin graph	Cayley graph	Hypercube graph
NS@9 Reordered	Bipartite graph	Halin graph	Cayley graph	Graph coloring	Multigraph
NS@18 Reordered	Bipartite graph	Cayley graph	Graph isomorphism	Cycle graph	Hypergraph

Table 6: Search results for the query ‘parsing network’ on alignment models without reordering, Elasticsearch and Google.

Type	Rank 1	Rank 2	Rank 3	Rank 4	Rank 5
NS@0	Benny Omer	1975 FIG Artistic Gymnastics World Cup	Regulus Grammar Compiler	Protochonetes	Mariko Okamoto
NS@1	Mouse tracking	Modality (human–computer interaction)	Post-WIMP	Data truncation	Dialog manager
NS@3	Mouse tracking	Modality (human–computer interaction)	Dialog manager	Post-WIMP	Data truncation
NS@6	Dialog manager	Mouse tracking	Modality (human–computer interaction)	Context model	Post-WIMP
NS@9	Dialog manager	Mouse tracking	Modality (human–computer interaction)	Input (computer science)	Semantic network
NS@18	Syntax	Dialog manager	Semantic network	Speaker recognition	Mouse tracking
Elasticsearch	Parsing	Semantic parsing	Scannerless parsing	Statistical parsing	Parsing expression grammar
Google	Parsing	Semantic parsing	Augmented transition network	Top-down parsing	Parser (programming language)

Table 7: Search results for the query 'parsing network' on alignment models with reordering



Type	1st	2nd	3rd	4th	5th
NS@0 Reordered	Rich Representation Language	Object (image processing)	Visual sensor network	Symbolic linguistic representation	Visual temporal attention
NS@1 Reordered	Semantic matching	Rich Representation Language	IPO model	Structured text	Symbolic linguistic representation
NS@3 Reordered	Semantic compression	Semantic matching	Rich Representation Language	Semantic gap	Structured text
NS@6 Reordered	Semantic network	Semantic matching	Rich Representation Language	Semantic gap	Structured text
NS@9 Reordered	Semantic network	Network mapping	Semantic matching	Rich Representation Language	Semantic gap
NS@18 Reordered	Semantic network	Natural language processing	Semantic gap	Body language	Context (language use)