Master's Thesis of Computational Science and Technology

A Study on the use of Etymology for Semantic Knowledge Extraction

June 2016

School of Natural Sciences
Seoul National University

Pablo Estrada
(Student Id: 2014-25245)

A Study on the use of Etymology for Semantic Knowledge Extraction

Submitting a master's thesis of Computational Science and Technology

School of Natural Sciences
Seoul National University
Interdisciplinary Major in Computational Science and Technology

Confirming the master's thesis written by
Pablo Estrada
(Student Id: 2014-25245)

June 2016

Chair: Prof. Hyeongin Choi _____ (Seal)

Vice Chair: Prof. Kyomin Jung _____ (Seal)

Examiner: Prof. U Kang _____ (Seal)

Abstract

Etymology is the study of the composition of words through their historical roots. It is a rich area of study that dates back millennia, and that has contributed significantly to our understanding of human cultures and languages. The field of computational linguistics is a much younger field that grew from the advent of the digital era; and that has advanced continuously, even nowadays with the changes brought by Artificial Intelligence and Machine Learning. Computational linguistics have not yet leveraged the knowledge of etymology to its full potential. This work is a step to make etymology another contributor to the field of computational linguistics.

In this work we propose a framework to capture the complex etymological relationships that exist in the vocabulary of a human language by creating a complex network that associates words with their historical roots. We then use this framework to obtain insights into the semantics of the words that are part of the Chinese and Korean languages. We run two tasks: one of supervised learning, and one of unsupervised learning, and show that etymology can be effectively used to extract knowledge.

We believe that this work helps push etymology into the main stage of computational linguistics, and natural language processing.

Keywords: Graph mining, etymology, computational linguistics, chinese language

Student number: 2014-25245

Contents

Chapter 1

Introduction

It has been found that many different systems can be modeled through networks (e.g. the internet, conflicts between countries). Many interesting and significant studies have come out of the creative application of graph theory to research problems that had been tackled in the past, but had not found an approach that could capture their complexity as a graph-based approach is able to.

An interesting example of research area that has been influenced by graph theory is that of computational linguistics. Particularly, the vocabulary of human languages can be modeled through networks in many different ways. There exist co-occurrence models, where two words will be connected if they appear in the same context (i.e. they co-occur). Graphs based on phonetic properties of words, or those that use semantic datasets such as Wikipedia, Freebase, and DBPedia are also common.

In this study we focus on a novel way to model human vocabularies with complex networks. Our approach is able to capture deliberate semantic similarities between words that come from their historical roots, and is able to leverage them to extract semantic knowledge from the network. This is because words are usually formed by mixing and matching smaller *meaning units* (i.e. etymological roots). To determine the etymological roots of words in languages with phonetic writing systems such as English normally requires a lot of intellectual work to trace words back to their ancestors. For this, etymologists rely on the historical written record of a language, to gather knowledge about the evolution of words and their use, and are often able to trace their origins back to their main language family.

Chinese etymology is also an ample field of study, where scholars are able to study the long-ranging and extensive written record of the Chinese language - and other languages that use it. The study of Chinese etymology is based on the use of individual Chinese characters. Chinese characters often have a definite history, and many of them can be traced back to Oracle bones, when they were first encountered. This is why our model is very fit to be used to study Chinese words; because unlike phonetic alphabets, which carry only sound information, Chinese characters carry information about their meaning. Therefore, upon reading a Chinese word (formed by one or more individual characters) it is possible to know the *meaning units* that integrate it, therefore making an etymology-extraction step unnecessary.

By separating words and studying them from their etymological roots, we are able to suggest a natural bipartite network structure. In one partition of the network it is possible to add all etymological roots (or, in the case of the Chinese language, individual characters); and in the other partition it is possible to include all words. An example of this can be seen Figure1.1, where in the left is the set of words, and in the right is the set of roots necessary to form these words.

Given that words are formed by mixing and matching *meaning units*, and that it is possible to know these meaning units from the logo-graphic writing of words with Chinese characters, we have all the necessary pieces to form an etymological network and use it to obtain semantic knowledge. Thus we set out to study whether the techniques of network science can be applied to bring some insight to etymological networks.

In this paper, we shall define a framework for creating etymological networks of any language; and extracting semantic knowledge from them. We use the model to build two graphs: One of Sino-Korean words (i.e. word used commonly in the Korean language that have been borrowed from Chinese; and that conserve their logographic writing), and one of Chinese vocabulary. We then use these graphs in two different tasks to extract semantic knowledge: Synonym pair classification, and word embedding.

1.1 Synonym pair classification

The first task of semantic knowledge extraction that uses our approach and that we shall explore is that of synonym pair classification with a supervised learning

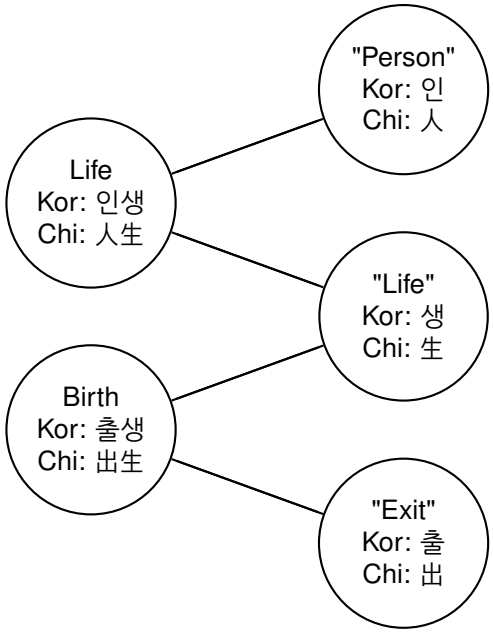


Figure 1.1: Subset of bipartite graph. On the left are the 'words' that are formed by mixing 'roots', which can be seen on the right. In Sino-Korean vocabulary, most words are formed by using two etymological roots, but there are also words formed by three roots, and a few that are formed by more than three. In Chinese vocabulary, words tend to be longer on average, and thus have a larger amount of etymological roots.

strategy. We opted for this task because it is fairly simple to comprehend, and a simple enough approach to demonstrate the power of an etymological graph to a wider audience.

Synonym discovery is a useful task in the field of information retrieval and search engines, as well as sentiment analysis and NLP. Information retrieval systems might rely on synonym substitution to know the meaning of obscure queries, or queries that use obscure words [1]. The same applies to sentiment analysis, where synonym substitution may help calculate a more precise polarity for a text.

In a graph of words such as out approach, the task of discovering unknown pairs of synonyms can be though of as either a task of link prediction, or a task of link classification. This is because, as a particular characteristic of the Chinese language, Chinese characters that represent similar concepts often join together (e.g. 同 'same', and 一 'one' mix to form the word 同一 'sameness').

In a classification task such as this, a set of features are calculated that describe the graph, the pair of nodes to classify, and the neighborhood of this node pair. A problem that we are likely to encounter in a task such as this is that most pairs of nodes (likely over 99%) do not represent synonyms, and thus the precision of the model might suffer.

1.2 Word embedding

One important area of research in the field of natural language processing (NLP) is that of word embedding. The traditional approach to word embedding consists on using a text corpus to characterize and embed words into rich high-dimensional vector spaces. By mining a text corpus, it is possible to embed words in a continuous space where semantically similar words are embedded close together, and different dimensions can express different semantic characteristics. This is important because by encoding words into vectors, it is possible to represent semantic properties of these words in a way that is more expressive and useful for natural language processing tasks where a written word would not carry much useful information for an algorithm to leverage. Word embedding has become an active area of research, and it has been effectively used for sentiment analysis [2], machine translation [3], and other tasks.

The basic idea behind all methods of word embedding is the distributional hypothesis[4]. This is the idea that words that appear in similar contexts must express similar ideas: "A word is characterized by the company it keeps". Based on this idea, count-based methods such as LSA[5] , and predictive methods that use neural networks to learn the embedding vectors[6][7] were developed, and used in research with success.

In this work, we propose a new approach to learn word embeddings that is based on the etymological roots of words, rather than the context in which they appear in text or speech. Our approach relies on the fact that a shared etymological root between two words expresses a deliberate semantic similarity between these two words; and by leveraging information on these semantic similarities, it is possible to derive the embedding vectors of words. This is akin to extending the distributional hypothesis to consider etymological context as well as textual context: words that appear in similar *etymological* contexts must also express similar concepts.

Based on this hypothesis, we propose an approach that consists on building a graph that captures these etymological relationships, and reducing the dimensionality of its adjacency matrix to learn word embeddings. To verify the validity of the word embeddings learned by our model we use the task of synonym discovery, whereby we analyze if it's possible to identify a pair of words as synonyms only through their embedding vectors. Synonym discovery is an important task in fields such as information retrieval, and sentiment analysis; and it has been used before to test word embedding schemes[2].

Chapter 2

Literature review

Literature on computational linguistics has sometimes looked at etymology[8], but it has rarely used etymological information in tasks related to semantic knowledge extraction. Particularly, [8] specifies the architecture of a new Etymological WordNet, which is developed to build a sort of etymological dictionary for vocabularies from many languages. These efforts are important, as etymological dictionaries can help bridge the gap to incorporate etymology into machine learning and data mining tasks [9], and we believe that with new interest in these dictionaries, research on etymology and computational linguistics will progress significantly.

Research that uses network science to study text, and particularly semantics[10] has also largely ignored etymology, with only a few exceptions. Particularly [11], and [12] use an etymological network-based approach to study movie scripts and movie reviews in English. They find that an etymological network built through either the script, or the reviews of a film can be used to extract important keywords about the film. Hunter et al's research is related to research done by the CASOS group at CMU, particularly with their AutoMap software[1], which exists as a text processing software suite that is equipped with pre-processors, and other tools for network text analysis. AutoMap would be an interesting starting point to look at when considering etymological graph-based research on the English language; and incorporating etymological dictionaries to a text mining package such as this would be a very useful tool.

When it comes to work that studies the Chinese language (or the Chinese

[1] http://www.casos.cs.cmu.edu/projects/automap/

writing system when used in other languages) that is based on network science, the literature is quite extensive and diverse. A popular topic in previous work is to study how radicals combine to form more complex characters [13]. This is interesting, and perhaps relevant to our work, because characters that integrate with other characters in the form of radicals to form more complex compounded characters often carry meaning information - although they also carry phonetic information just as often. Some other studies have created networks based on word co-occurrence [14], which is the standard practice in studies that focus on languages with writing systems that are primarily phonetic to produce graphs and co-occurrence matrices that can later on be used to train neural models, or matrix-based models in word embeddings. In our review of previous art we found only one study that creates a network based on how characters mix to form words [15]. This study is based, specifically, in two-character composite words in the Japanese language, and its main contribution is a random-graph model to produce similar graphs, which is inspired in the way humans generate and connect new words. This study generates a directed graph, unlike our own work which generates a graph that is undirected, but very similar. All the studies that we found studying Chinese language from the perspective of network science have studied the graph-theoretical properties of their networks, but have not attempted knowledge extraction from the graphs.

The task of synonym discovery has also been well explored in previous research. We found that much of it uses graph-theoretic approaches, such as [16], where multilingual synonym graphs are used, and machine learning approaches are taken to leverage the graph structure to discover new pairs of synonyms - much like what we do later on in this work. They also attempted to use the graphs of synonyms in other languages to find new synonym pairs. Other research has resulted from using the WordNet [17], which is a detailed network that shows semantic relationships between words, and also their linguistic relationships. The WordNet has been successfully used in several studies [18] [19] [20] [21] that extract semantic knowledge from it, and particularly [18] looks at semantic similarity between words by studying their ontologies, whcih are trees that represent semantic relationships between them. On this theme, semantic networks such as knowledge graphs may also bring useful knowledge bases for models that use etymology and other semantic relationships between words to infer more semantic knowledge [22].

Emulating the original WordNet developed at Princeton, some researchers

have also started building an analog database of the Chinese language [23]. The database is still young, but some research has already used it, either in considering a multilingual WordNets [24], or work on improving the existing Chinese WordNet itself [25].

Synonym discovery has also been tackled for the particular case of Chinese vocabulary. This is the task we use to test our model initially, with individual character synonyms, and to test the appropriateness of our word embeddings. Previous work exists that tackles this task [26] [1], although we were unable to survey the Chinese-language literature on this topic. These studies use a large corpus from the Chinese Wikipedia, and identify synonyms by building a graph where words are linked if their Wikipedia articles link to each other; or they use hand-coded pattern matching rules to try to find words that might be synonyms. These studies do not report their performance in general, instead reporting some identified synonym pairs, and thus are not a very good measure of the performance of our models. Unfortunately, we did not have access to previous art written in Chinese, but synonym-discovery has likely been tackled in other works written in the Chinese language.

In our own previous work, we defined an etymological graph-based framework, and used it in a supervised classification scheme to find pairs of Chinese characters (e.g. etymological roots) that were synonyms[27]. In this paper we showed that the etymological graph approach can be effectively used to extract knowledge from a complex etymological network. We explore this approach later on on this work, and address its strengths and weaknesses. Particularly, we observe that synonym-pair ground truth is a less useful set of knowledge to extract when compared to word embedding, which we also tackle here.

In this work we use a technique that is very similar to Latent Semantic Analysis (LSA) to learn the embedding vectors that correspond to words in etymological graphs. LSA was first implemented for information retrieval, and was called Latent Semantic Indexing[28] in the late 1980s. The reasoning behind the whole LSA process is that by constructing a matrix that associates documents with the words that these documents contained, and then decomposing the matrix using Singular Value Decomposition (SVD) it would be possible to cluster documents that address similar topics together. The idea is that latent variables exist which are the topics, and by reducing the dimensionality of the original document-term matrix allows to

expose these latent 'topic' variables, and to cluster documents in a vector space[5]. Since its first publication and implementation, numerous papers have been written around the discovery of latent semantic variables, topic modeling, considering documents as topic mixtures, and a number of other lines of research. The topic is still active, and is still used in information retrieval.

Word embedding was defined originally in [29], where the authors use a neural network-based approach to generate a language model of which word embeddings are a byproduct. Since then, numerous studies have been written where both neural networks and count-based models have been used to produce word embeddings [6] [7]. Count-based models rely on reducing the dimensionality, or factorizing a matrix that contains counts of co-occurrence between words in a large text corpus. Neural-network based approaches often rely on converting words into long sparse vectors, and training a neural network to infer context based on an occurring word, or a word given a context. Numerous datasets and software tools are available to borrow and train word embedding models. Word embeddings have settled around dimension counts of 50, 100 and 200, and these are now standard in the research community. Some areas where word embedding vectors have been used to achieve breakthroughs are those of sentiment analysis, where the semantic characteristics of a word that are expressed in its embedding vector can be used to infer polarity, and other sentiment qualities of the word; ontology, where embedding vectors are used to build sets of synonyms, and obtain the lexemes of words [30]; topic modeling, whereby averaging the embedding vectors of the words within a text it is possible to embed documents in a space and use these embeddings when building an index in a database for information retrieval tasks that involve full-text search.

A very interesting attempt to use word embeddings is that of using aligned embeddings for machine translation [31], particularly [3] attempts translation between English and the Chinese language by producing two sets of word embeddings in Enlish and in Chinese, and then aligning them together so that words in Chinese will align to similar words in English. This is quite interesting because instead of considering a word-to-word translation model, using word embedding vectors it would be possible to express several words that can accurately translate a single word while carrying some nuances. These studies use neural network-based models, and they make their embeddings available in a 200-dimension format on their website[2]. We use their word embeddings in this work to compare the performance

of their model with our own model.

To the best of our knowledge, there are no previous studies that explore a data mining task based on etymology.

[2] http://ai.stanford.edu/~wzou/mt/

Chapter 3

An etymological graph-based framework

We shall now present the steps to set up the framework. The main step in the process is to build a network that captures the associations between words and their etymological roots. In the following section we shall describe the steps to do this.

3.1 Building an Etymological Graph

To build an etymological graph, it is necessary to have a list of words, and these words must be annotated with their etymological roots. These words may be obtained by scraping, as we did for the Sino-Korean set of data, or by obtaining a dictionary in a single list, as we did for the Chinese data with the ADSO dictionary[1].

Algorithm 1 shows how to build an etymological graph from a list of words annotated with their etymological roots. As can be seen constructing the graph from these inputs is quite simple. As it has been mentioned earlier, the challenge lies more on obtaining the etymological root annotations.

Algorithm 2 presents how to build a graph by scraping from a root-word dictionary, where it is possible to obtain all words that borrow the same etymological root. This approach has the particular feature that the graph produced is strongly connected (i.e. for every node, there exists a valid path to any other node). The

[1] http://adsotrans.com/

Algorithm 1 Building etymological graph

Require: Empty graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
Require: List of words $\mathcal{W}$ annotated with etymological roots.

```
for each w ∈ W do
    V ← V ⋃ {w}
    for each root ∈ w do
        if root ∉ V then
            V ← V ⋃ {root}
        end if
        E ← E ⋃ {{root, w}}
    end for
end for
```

graphs obtained in [15] displayed connectedness and small-world effect, therefore it is reasonable to expect algorithm 2 to result in graphs that cover most of the vocabulary of a given language - though this consideration might be thought twice in vocabularies that are more sparse, or that have more than one linguistical source.

For example, given a vocabulary with the words Life (Kor: 인생 Chi: 人生), and Rebirth (Kor: 출생 Chi: 出生), our algorithm might start scraping from the character for Exit (Kor: 출 Chi: 出), and one-by-one traverse and build the graph as shown in Figure1.1.

Using this approach we built two graphs: One of Chinese words that we obtained from the ADSO dictionary, and another of Sino-Korean words (Korean words that have been borrowed from Chinese, and that keep their Chinese writing). Table 3.1 contain basic statistics about the two graphs.

It is interesting to note that the distribution of word lengths is significantly different. While the Chinese language graph has a more widely spread distribution of word lengths, and a higher average length; the Sino-Korean vocabulary graph has most of its words concentrated around 2 and 3 characters. This is perhaps due to the way Chinese words were borrowed into the Korean language, and the way in which they are used.

Sino-Korean words are often used as two-syllable verbs (e.g. 인정 'to acknowledge', 예약 'to book', 취직 'to look for employment'), as three-syllable adjectives (e.g. 애국적 'nationalist', 소극적 'passive', 소심적 'introvert'), or as four-syllable proverbs (e.g. 일석이조 'one stone, two birds'). These categories do not completely explain all the words in the Sino-Korean vocabulary, but they do cover several of

Algorithm 2 Building etymological graph by scraping

Require: Empty graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
Require: Single character $\rfloor$ to start scraping from
let $\mathcal{Q} \leftarrow \rfloor$ be a queue
while $|\mathcal{Q}| > 0$ **do**
    $ch \leftarrow pop(\mathcal{Q})$
    $\mathcal{W} \leftarrow words(ch)$
    for each $word \in \mathcal{W}$ **do**
        if $word \notin \mathcal{V}$ **then**
            $\mathcal{V} \leftarrow \mathcal{V} \bigcup \{word\}$
        end if
        for each $root \in word$ **do**
            if $root \notin \mathcal{V}$ **then**
                $\mathcal{V} \leftarrow \mathcal{V} \bigcup \{root\}$
                $push(\mathcal{Q}, root)$
            end if
            $\mathcal{E} \leftarrow \mathcal{E} \bigcup \{\{root, w\}\}$
        end for
    end for
end while

them. This stands in contrast with Chinese words, which obviously constitute the whole vocabulary of the Chinese language, and therefore they take many forms, and possible word lengths.

Something interesting about these two graphs is that they show a small-world effect[32], despite their bipartite nature. The high degree of most root notes likely is the main contributor to this phenomenon. This characteristic is perhaps not so noticeable in etymological graphs derived from languages that have many sources of etymological influence (e.g. English having Germanic and Latin influence on its vocabulary); which might cause the graph to be more sparse, and less interconnected.

3.2 Obtaining semantic knowledge from the graph

One of the major strengths of our framework is that a graph that pairs etymological roots and words is quite versatile, and it can be modified to fit diverse learning techniques; but despite its versatility, the graph contains a substantial amount of information about a vocabulary.

Language	Chinese	Korean
Num. of Words	117,568	136,045
Num. of Characters	5,115	5,972
Avg. word length	3.36	2.56
Avg. degree of a root-node	76.45	58.2
Avg. shortest path length	4.308	4.831
Bipartite density	0.000718	0.000427
Average bipartite clustering	0.178	0.239
Words by length		
1 character	2,082	-
2 characters	25,001	77,891
3 characters	35,108	40,024
4 characters	39,249	18,130
5 characters	16,128	-

Table 3.1: Statistics from the Sino-Korean vocabulary graph, and the Chinese vocabulary graph.

When it comes to a graph, several techniques have been explored to analyze and understand its structure, as well as derive insight from it. Techniques in the graph mining literature are very wide in range and versatility. From identifying common substructures [33]; analyzing, or predicting links, which we try in our first experiment; finding community structure, which might be important among words that could likely be grouped by semantic closeness. In general, we believe that a task to extract semantic knowledge from an etymological graph may be based on one of the following factors:

- **The adjacency matrix or biadjacency matrix of the graph.** Tasks based on the matrices that describe the graphs may be related to word embedding, which we attempt on this work, or finding communities of words through PCA. With more detailed information, such as the first known use of a character, it would be possible to study them with tensor-based strategies.

- **An unipartite projection of the bipartite graph.** Tasks based on an unipartite projection of the existing graph could study common substructures (closing triangles, common tri-nodes), or attempt to classify over nodes, or edges; or as we try later on this work, to classify over sets of nodes. Community discovery is also a possible area of work.

- **The bipartite etymological graph itself**. Tasks that use the full bipartite graph may be based off some bipartite algorithms such as co-clustering[34], bipartite structure analysis (based not on triangles, but perhaps on squares and others).

In the following chapter we shall explore two strategies to obtain semantic knowledge from an etymological graph. These tasks are based on (1) a unipartite projection of the original bipartite etymological graph, and (2) the bi-adjacency matrix of the graph.

Chapter 4

Two use-cases of the framework

In this chapter we provide two interesting examples of semantic knowledge extraction techniques that are based on an etymological graph. The first of them is a fairly simple classification scheme that allows to obtain previously unknown pairs of synonyms - a sort of toy example that demonstrates the framework.

The second task has much more interesting results - and a more consequential outcome. In this task we use the etymological graph's biadjacency to embed words in the vocabulary into a rich high-dimensional space that represents semantic properties spatially. This is clearly a count-based method to obtain word embeddings.

4.1 Supervised learning: Finding synonyms though classification

We attempted to extract synonym knowledge from the graph structure. Particularly, we decided to focus on finding synonyms between pairs of Chinese characters from the etymological graph of the set of Sino-Korean words, because for a set of N nodes, the number of node pairs grows at a rate of $(N-1) \times N \times \frac{1}{2}$. Since the set of Chinese characters is much smaller than the whole set of words, the processing time would be much shorter.

To realize the classification, we applied supervised learning. To do the supervised training we needed to know ground-truth synonym or non-synonym values for

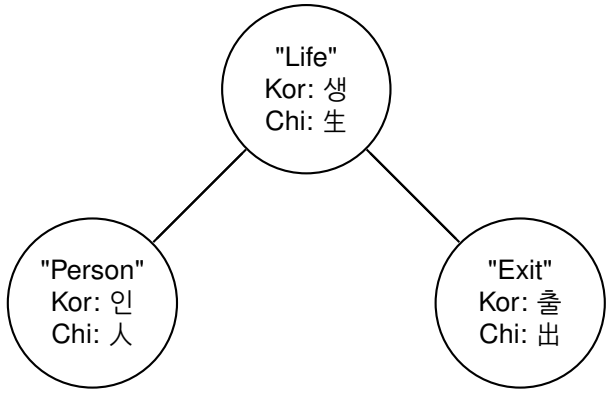


Figure 4.1: Chinese character unipartite projection of the bipartite graph shown before. The character '生' participates in both words, and is therefore connected to all other characters. The characters '人' and '出' do not form a word together, and are therefore not connected.

several pairs of nodes. For this we collected synonym data about Chinese characters from the dictionary of an online Korean portal. After scraping this data, we obtained approximately 5000 pairs of Chinese characters that are known to be synonyms.

Because we were working only with pairs of Chinese characters, and ignoring pairs of words; we transofrmed the source etymological graph into a graph that would contain only Chinese characters. We realized the root-unipartite projection of the original graph [35], in which every node represents a single Chinese character, and two nodes are connected only if they can form a word together. Figure 4.1 shows the root-unipartite projection of the small graph from figure 1.1.

We then set up a classification scheme for each pair of nodes in the root-unipartite projection of the network over Chinese characters. For each pair of nodes, we computed 19 features[36], that are detailed in Table B.1. These features aimed to describe characteristics of the two nodes' relationship (e.g. reciprocity, degree, PageRank scores), their neighborhood (e.g. shared neighbors, two-step walks, dispersion), and their place in the graph (e.g. centrality, clustering).

To improve our classification scheme, we constructed two models: One for pairs of nodes that shared an edge (i.e. that formed a word together), and another for pairs of nodes that did not share an edge. This is because, unlike in the English language, Chinese characters *often* pair with synonym characters to express the same idea (e.g. the word 同一 ('sameness'), where 一 means 'one', and 同 means 'same'). By

constructing two models, we allow each model to focus and improve its predictive power.

We explored several classification techniques that would perform well in both tasks. We obtained best results when using boosted Random Forests ($N_{trees} = 10$, $size_{tree} = \sqrt{N}$) using the Python scikit-learn package[37]. This result likely comes from (1) the fact that classification between synonyms and non-synonyms does not show linearity, but rather is defined by a complex space; and (2) the possibility that the hand-coded features used in the classification scheme may have failed to capture the boundaries between synonyms and non-synonyms properly.

4.1.1 The edge classification problem

Because we were only considering pairs of nodes that share an edge, this section can be though of as an edge classification problem; where we want to determine if an edge is linking two synonyms, or not. Out of the two problems that we consider, the edge classification problem is relatively easier, because nodes that do share an edge have a much higher likelihood of being synonyms, and their neighborhood features are also more likely to carry information that helps classify them appropriately.

This search space is of manageable size. The Chinese character uniprojection has a total of 74,893 edges that must be classified. Because the space of edges is of manageable size, the recall and the precision of the classification scheme were very high, as can be seen in Table 4.1.

It is important to consider that in a different language, this particular result would not exist, because synonym etymologies rarely mix in languages other than Chinese. Indeed, it is likely that the betweenness of most nodes will be significantly lower, because most languages have been influenced by a large number of sources, and therefore, it is likely that several sections of an etymological graph representing a European languages might be loosely connected among themselves.

4.1.2 The synonym-link prediction problem

This can be thought of as a particular case of link prediction problem. In this case, we are looking to predict only synonym-typed edges, and ignore the rest. This problem is significantly more challenging when instead of classifying existing edges, it is about classifying every pair of nodes that does not share an edge. The search space is much larger, since for 5,972 Chinese characters there can be a total

of 17,829,406 different combinations that must be classified, out of which only very few yield real synonyms.

Because of this disparity in search-space sizes, we had to adjust the classification scheme to yield only pairs of nodes that are very likely to be synonyms; otherwise the precision of the classification scheme would be too low. In our experiments we found that $P(synonym) \geq 0.8$ proved to be a reasonable probability cutoff that provided discovery of new synonyms while keeping a high precision.

The performance of the classifier for free pairs of nodes should be a good predictor of the performance expected form attempting the same model on an etymological graph representing vocabulary from another language - particularly for European languages.

4.1.3 Results of the classification schemes

Table 4.1 shows the result of both classification schemes, and the precision of both of them. As expected, the precision for the link prediction problem did suffer as a consequence of having a much larger search space, resulting in synonyms making up a much smaller proportion of the set of pairs.

Since we do not have ground-truth values for all synonym pairs for Chinese characters, it is impossible to provide a recall measure, though when the model was run with 20-validation (i.e. training with N-20 values and testing with the remaining 20 values), the recall was relatively low.

Another interesting result is that the model misidentified some pairs of antonyms (e.g. 剛 'hard' and 柔 'soft'; 始 'start' and 終 'end'; 母 'mother' and 父 'father') as being synonyms. This might be due to the way in which etymologies with opposite meanings have a special use in Chinese morphology. Antonym etymologies in Chinese characters are often interchangeable in words (e.g. 'black' and 'white' in 黑人 'black person' and 白人 'white person'). This particular feature of the language could end up making their features exist very closely, and thus their points could appear very close to one another in the search space - thus leading them to be classified as synonyms.

Although this exercise was relatively simple, computing 19 features for a large search space of over 17 million data points proved to be a time-consuming task. For this, we had to develop a Python script that took advantage of parallel computing capabilities to split the task into 4 processes, and yet computing all 19 features for

Table 4.1: Features calculated for each pair of nodes

Problem	Classification of free pairs	Link classification
Pairs of nodes to classify	16,315,784	74,892
Known synonyms (training)	2,889	1,618
Discovered synonyms	2,132	1,565
Precision	0.73	0.84

each pair of nodes took about 4 days every time - not counting two more features that were not considered in the model due to the low speed at which they could be computed.

As part of our research, this was the first task that we accomplished using the etymological graph, and it was a good proof that the framework could be used to extract semantic knowledge. As mentioned earlier, although the model was able to discover real new pairs of synonyms, the recall of the known synonyms was less than desirable. We believe that this is due to (1) the features that we chose not being able to properly describe the neighborhood of a pair of nodes, or (2) the *synonym space* being too large to be described by our available known data.

4.2 Unsupervised learning: Word embedding with etymology

The second task that was attempted for knowledge extraction from an etymological graph is that of learning the spatial embeddings of words from the etymological graph. This task is more useful, as word embedding is an active area of research. Though it may seem more challenging, we believe that it is a natural task to attempt given the characteristics of an etymological graph, where is biadjacency matrix closely mirrors the co-occurrence matrix that is used in count-based models similar to word embedding such as LSA [5].

4.2.1 Learning word embeddings

To obtain the word embeddings from the graphs, truncated Singular Value Decomposition(SVD) was applied to their biadjacency matrices[35]. We use truncated SVD inspired by the techniques of LSA [5], where it's possible to map words and documents to 'hidden concepts'.

The biadjacency matrix A of a bipartite graph is a matrix of size $n \times m$ where each column represents a node from one bipatite set, and each row represents a node from the other bipartite set. In the case of etymological graphs, each row represents a root node, while each column represents a word node; therefore the matrix A has dimension $\#roots \times \#words$. In count-based word embedding algorithms, columns and rows represent embedding-words, and context-words; and in LSA columns and rows represent words and documents. In our matrix A, the element a_{ij} will be 1 if the word represented by column j contains the etymological root represented by row i.

In schemes such as LSA, where the number of documents in which a word appears can vary immensely, it is common to apply weightings to the matrix that reflect the popularity of the terms - and in this case, to reflect the popularity of an etymological root, as well as how much of the word is 'represented' by the etymological root.

$$a_{ij} = L(i,j) \times G(i,j)$$

Where $L(i,j)$ is the local weighting of root i in word j, and $G(i,j)$ is the global weighting of root i. In this particular case, the local weighting of roots in a word is not as significant as topic modeling, but it is still relevant. We define these weightings as:

$$L(i,j) = log(len(w_j)/A_{ij})$$

and:

$$G(i,j) = A_{*j}/A_{ij}$$

In our experiments, the weighted version of our matrix delivered results that were very similar to the unweighted version of it.

By applying the truncated SVD, we attempt to approximate the biadjacency matrix A with a matrix $\widetilde{A}$ which is of a lower rank, and can be expressed as the product of three matrices $U\Sigma V^*$, where Σ is a diagonal matrix with the k largest singular values in the diagonal, and the matrices U and V^* are approximations of unitary matrices of size $\#roots \times k$ and $k \times \#words$ respectively; where k is the dimension into which we chose to reduce matrix A. The following equation shows the basic result of SVD applied to A:

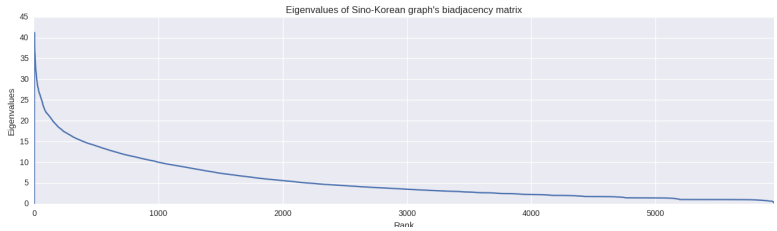


Figure 4.2: Eigenvalues from the biadjacency matrix of the Sino-Korean etymological graph

$$\widetilde{A}_{\#roots \times \#words} = U_{\#roots \times k} \times \Sigma_{k \times k} \times V^*_{k \times \#words}$$

By performing this truncated factorization it is possible to derive the latent semantic structure that exists in the matrix. The matrices V and U contain singular vectors of the matrix $\widetilde{A}$, and these matrices express a breakdown of the original graph structure into a linear combination of a base of smaller rank, which expresses the latent semantic variables. In a sense, by truncating the SVD computation, we are able to capture the underlying semantic structure, while removing 'noise'.

To drive this point home, in figure 4.2 shows the singular values from factorizing the Korean graph in our data. Because each singular value belongs to each one of the dimensions in the rank of the matrix, each singular value is a sort of proxy that tells us the value of each of the semantic dimensions that result from factorizing the matrix with SVD. The rank of the matrix was of 5,952, which 20 dimensions fewer than the actual row count of it.

We use the dimension-reduced column vectors in V^* as embeddings for each word in our vocabulary. Each row in the V^* matrix is one of the embedding dimensions which 'condense' semantic information brought from the etymological graph. It is also possible to use the rows from matrix U as embedding vectors for etymology, where the columns represent single dimensions in the embedding space resulting from reducing the dimensionality of the original biadjacency matrix. This strategy is also useful because it can be used to obtain embeddings for new words: If a new word w comes into the dataset, with a list of etymologies w_e, then it is possible to define the embedding vector of word $V(w)$ as the sum of the embedding vectors of its etymological roots: $V(w) = \sum_{e \in w_e} V(e)$.

Another matrix decomposition technique worth considering for future work is CUR factorization [38]. We're specially interested in its sparsity-maintaining characteristic; since large matrices such as ours can be managed more easily if they are sparse - and SVD eliminates the sparsity of our source matrices.

4.2.2 Verifying the word embeddings: Synonym discovery

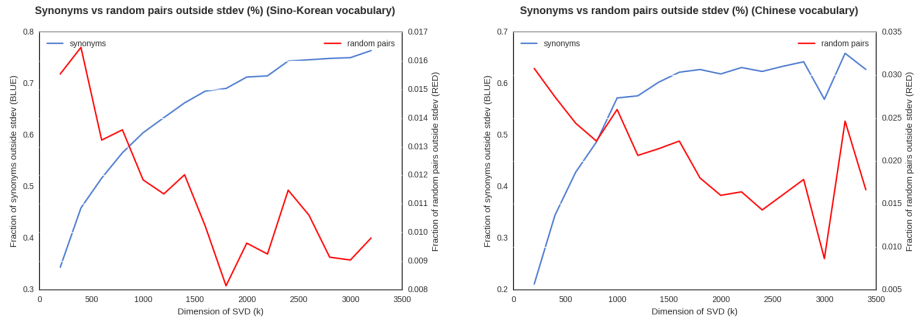
To verify the validity of the embeddings, we selected the task of synonym discovery. To assess whether two words are synonyms, we measure their similarity as proposed in [39]. We expect synonyms to show similarity score above a threshold, which we decide by studying the distribution of the similarity between random pairs of words. In other words, we obtain the dot product of vectors from random pairs of words, and compare them to the dot product of vectors from pairs of synonyms. As random pairs of words are expected to have little semantic relationship, the dot product of their embedded vectors is expected to be close to 0; while the dot product of vectors representing pairs of synonyms is expected to be far from 0 due to the semantic similarity between pair of synonyms, which should be expressed by their embedding in a vector space. For comparison, we used the dataset of Chinese word embeddings that was released as part of [3], which contains embeddings of Chinese words in 50 dimensions. We used this data set on the same task: Synonym discovery by measuring their similarity score as the internal product between vectors.

To obtain the 'ground truth' of synonym pairs, we collected pairs of synonyms from online dictionaries for both Chinese and Sino-Korean vocabulary. In Korean we collected a total of 38,593 pairs of synonyms, while in Chinese we collected 45,731 pairs.

Another task that is often used when working with word embeddings is that of pluralization. We judged this task to be inappropriate for our model, since pluralization in Chinese-based languages is expressed as part of the sentece, and not as part of a word itself.

4.2.3 Performance of synonym discovery task

Experiments show that we were able to reliably identify pairs of synonyms by comparing the dot product between embeddings of pairs of synonyms in both the languages that we tested. Performance was specially good in the Korean language graph, as can be seen in FigureC.1, where we plot distributions of dot product



(a) Sino-Korean vocabulary (b) Chinese vocabulary

Figure 4.3: Proportion of random pairs of words where the dot product of their embedding vectors is far from zero (BLUE) and proportions of pairs of synonyms where the dot product of their embedding vectors is far from zero (RED). Only 1% of the random pairs place far from zero, while about 73%(a) and 65%(b) of synonyms are far from zero.

between random pairs and pairs of synonyms. As shown in the figure, up to 70% of all synonyms have a similarity measure that places them outside the range covered by 99% of random pairs of synonyms. Figure4.3 helps drive this point by showing the variation of the proportion of synonyms that are placed outside the standard deviation of the distribution of dot products of embeddings of random pairs of words when we vary the dimension of our embeddings. Interestingly enough, only about 1% of random pairs of words appear outside of this range, and the vast majority of them consistently concentrated around zero. We found that increasing the number of dimensions does reduce this proportion, albeit slightly.

As a baseline, we used the cosine similarity for the 'raw data', which in this case refers to the vector representations of words in the non-factorized biadjacency matrix. Figure 4.4 shows the histogram of internal products between raw embedding vectors between pairs of synonyms, and between random pairs of words. As it can be seen from this figure, raw vectors of synonyms do not have a significantly higher similarity when compared to raw vectors of random pairs of words.

On the other hand, figure C.1 shows the performance of our model when we use the factorized version of our matrix, with different dimensions for the outcome. Also, figure C.1(d) shows how the data from Zou *et al.* performs when trying to identify pairs of synonyms.

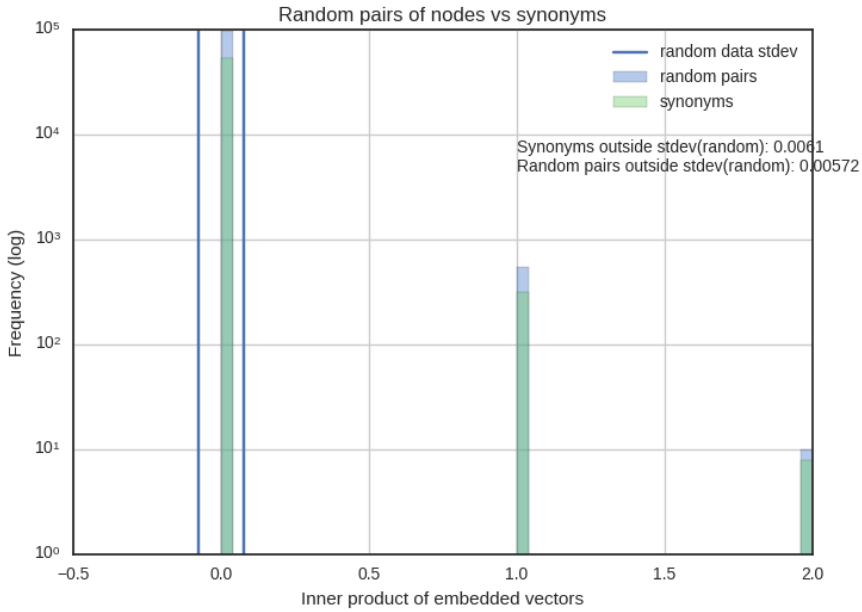


Figure 4.4: Histogram of the cosine similarity between raw vectors of pairs of synonyms and random pairs of words.

Our embeddings also proved to perform better than our benchmark dataset. FigureC.1(d) shows the distribution of the similarity measure between pairs of synonyms and random pairs of characters in the benchmark dataset. In this sample, almost 32% of synonyms show a similarity score that places them away from zero, while 5% of random pairs of words are placed outside of that range. Table 4.2 compares the performance, and the dimensionality of both strategies to learn embeddings.

Table 4.2: Performance for each model and language

Model	**Language**	**Dimensions**	**Correctly classified synonyms**	**Misclassified random pairs**
Our model	Korean	2000	70%	1%
Our model	Chinese	2000	64%	1.5%
Zou, *et al.*	Chinese	50	32%	5%

We have not looked at micro-effects of our model, such as the behavior of

antonym words that contain all the same characters plus a negative character. In this case, the fact that a word and its antonym share all but one character might cause their embeddings to have a high similarity to each other. Another result that we have not addressed directly is the random pairs of words that had high similarity scores. Reviewing these should be interesting, as they may either be unknown synonyms, antonyms, or perhaps an anomaly in our model.

4.2.4 Computation speed of our model

An interesting feature of word embedding models based on matrix factorizations is that training time can be significantly shorter when compared with the time it may take to train a multi-layered neural network. This particular characteristic of deep-learning models has increasingly been a topic of note in the research community, and libraries have started addressing it with models that leverage the sheer array of computing power that can be found in GPUs. Some companies have also worked on purpose-specific hardware. Therefore, a model that can obtain word embeddings in a reasonable amount of time would have a significant advantage over a deep learning model that could take several days to be trained.

Our model had a good performance in training, although it was still rather slow at its highest perofmance. For dimensions under 500, SVD can run very quickly, but as the dimension count rises, the factorization step becomes significantly slower. Our model reaches its best performance at around 2000 dimensions, for which the matrix factorization takes over 5 minutes of computation.

Code for our model was developed in Python 3, which is an interpreted language and therefore slower than languages such as Java or C which are compiled. Paticularly, we used the NetworkX python package to manage and analyze our graphs [36], which is a pure-Python framework. The linear algebra features came from the SciPy[40] and NumPy[41] libraries, which we used to work with matrices and vectors, and to run the matrix decomposition. SciPy and NumPy are libraries coded for high performance, and they use Python interfaces to C-coded routines, which make them perfect for taks that require heavy computation. Our code ran on a Intel Core i7-4790 clocked at 3.60GHz and 16 MB of RAM. Table 4.3 shows the running time of the factorization of both our graphs and different values for the dimension of the matrix decomposition.

These running times stand in contrast with the rather large times it takes to train

Table 4.3: Running time of matrix factorization

	Time (seconds)	
SVD k	Sino-Korean vocabulary	Chinese vocabulary
200	6.3	4.6
600	36.7	29.2
1000	71.3	56.9
1400	144.9	113.36
1800	266.2	215.7
2200	407.2	312.4
2400	484.6	337.2
2600	566.4	346.2
3000	737.4	369.1

a neural network model. Nonetheless, given that our embeddings require a higher number of dimensions to be effective, SVD on the dimension that we require has a relatively slow performance of over 5 minutes when running 2000-rank SVD on our matrix.

Software tools exist to run graph-mining tasks in distributed environments that use MapReduce [42], Apache Spark [43], or the more recent Apache Flink[1].

[1] https://flink.apache.org/news/2015/08/24/introducing-flink-gelly.html

Chapter 5

Discussion, Conclusion, and Future Work

5.1 Discussion

The framework has worked properly on the two tasks that we have attempted. Some details are left to be studied. Particularly, it is rather puzzling that the validation recall from the first task of synonym discovery was so low. We attribute this to a possibly irregular distribution of synonym pairs in the feature space, and the features possibly being unable to capture the essence of a synonym pair for some of the data. More experiments would be necessary to figure out how the recall can be improved without affecting the precision significantly. Perhaps clustering, or clustering-based visualizations could be used to understand a bit better how the groups of synonym pairs are distributed along the search space.

Another weakness of our model is that it does not account for character ordering in words - which does matter in the Chinese language when words are composed. The work in [15] works with a directed graph where the first character of a word points at the second one, but this study has the larger pair of limitations that (1) it builds a graph that consists only of etymological roots, and (2) it only considers two-character word, which would be a very significant limitation in our model. New research might improve our model by addressing the character-order issue.

On the task of word embedding, it would also be interesting to figure out why such a large proportion of words had a similarity close to 0 (about 30%),

independently of the dimensionality of the matrix factorization. Perhaps their similarity is not well expressed by etymology but by daily use. This is a consequence that comes from the fact that we are using what we have decided to call a 'top-down' approach to word embeddings, which is based only on their etymological roots; or in other words: our approach obtains word embeddings of words by how they were *intended* to be used, rather than how they are used in daily speech and text. This is a weakness of our word embeddings against embeddings based on large text *corpi*.

A noticeable difference between our word embeddings and existing ones is that ours require a much higher number of dimensions to perform well in synonym discovery. Publicly available datasets with word embeddings provide vectors with 25,50 and 100 dimensions [2]; but our embeddings reach their highest effectiveness at around 2,000 dimensions. This is likely a consequence of our data being very sparse: while words in word co-occurrence models can have an almost limitless set of contexts in which they appear, words in etymological graphs have a defined number of etymological roots. All the words in our graphs are formed by 5 characters or less.

Sparsity has been a challenge of co-occurrence models for a while. In co-occurrence models, it can be fought with larger *corpi*, but in our case, because the etymological roots of a word would not change no matter what, our model can be thought of as *doomed* to sparsity.

5.2 Conclusion and Future Work

Research has very rarely approached the problem of semantic knowledge through etymological networks [12] [11]. We believe that this technique has potential to bring new insights, and that etymology has a lot more to bring into the playing field of computational linguistics.

One important challenge that remains for etymological approaches to be applied on languages with phonetic alphabets is access to relyable etymological annotations. The research community has made some efforts such as the Etymlogical WordNet, and the research by the CASOS group at CMU. Perhaps more work should be done in this area before progress can be made.

Regarding the etymological graph framework that we presented, an important feature of it is that it can support several different techniques to extract semantic

knowledge, and both the supervised learning-based synonym discovery, and unsupervised learning of word embeddings are only two tasks that can be tackled. We are specially excited and interested on seeing other researchers use etymological graphs for other tasks, and specially it is important for us to see them applied in other languages.

Future work will focus on exploring in detail the weaknesses and downfalls of our model in the two tasks that we implemented. We also plan to explore other techniques to derive semantic knowledge from the etymological graph of Sino-Korean words.

Bibliography

[1] Lu Yong, Zhang Chengzhi, and Hou Hanqing. Using multiple hybrid strategies to extract chinese synonyms from encyclopedia resources [j]. *Journal of Library Science in China*, 1:010, 2010.

[2] Yanqing Chen, Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. The expressive power of word embeddings. *arXiv preprint arXiv:1301.3226*, 2013.

[3] Will Y Zou, Richard Socher, Daniel M Cer, and Christopher D Manning. Bilingual word embeddings for phrase-based machine translation. In *EMNLP*, pages 1393–1398, 2013.

[4] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*, pages 2177–2185, 2014.

[5] Susan T Dumais. Latent semantic analysis. *Annual review of information science and technology*, 38(1):188–230, 2004.

[6] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[7] Marco Baroni, Georgiana Dinu, and Germán Kruszewski. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *ACL (1)*, pages 238–247, 2014.

[8] Gerard De Melo. Etymological wordnet: Tracing the history of words. In *LREC*, pages 1148–1154. Citeseer, 2014.

[9] Eric Partridge. *Origins: A short etymological dictionary of modern English*. Routledge, 2006.

[10] Jose Emilio Labra Gayo, John P McCrae, Menzo Windhouwer, and Gerard de Melo. Lexvo. org: Language-related information for the linguistic linked data cloud. *Semantic Web*, 6(4):393–400, 2015.

[11] Starling David Hunter and Saba Singh. A network text analysis of fight club. *Theory and Practice in Language Studies*, 5(4):737, 2015.

[12] Starling Hunter et al. A novel method of network text analysis. *Open Journal of Modern Linguistics*, 4(02):350, 2014.

[13] Jianyu Li and Jie Zhou. Chinese character structure analysis based on complex networks. *Physica A: Statistical Mechanics and its Applications*, 380:629–638, 2007.

[14] Shuigeng Zhou, Guobiao Hu, Zhongzhi Zhang, and Jihong Guan. An empirical study of chinese language networks. *Physica A: Statistical Mechanics and its Applications*, 387(12):3039–3047, 2008.

[15] Ken Yamamoto and Yoshihiro Yamazaki. A network of two-chinese-character compound words in the japanese language. *Physica A: Statistical Mechanics and its Applications*, 388(12):2555–2560, 2009.

[16] Emmanuel Navarro, Franck Sajous, Bruno Gaume, Laurent Prévot, Hsieh ShuKai, Kuo Tzu-Yi, Pierre Magistry, and Huang Chu-Ren. Wiktionary and nlp: Improving synonymy networks. In *Proceedings of the 2009 Workshop on The People's Web Meets NLP: Collaboratively Constructed Semantic Resources*, pages 19–27. Association for Computational Linguistics, 2009.

[17] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[18] M Andrea Rodríguez and Max J Egenhofer. Determining semantic similarity among entity classes from different ontologies. *Knowledge and Data Engineering, IEEE Transactions on*, 15(2):442–456, 2003.

[19] Michael N Jones and Douglas JK Mewhort. Representing word meaning and order information in a composite holographic lexicon. *Psychological review*, 114(1):1, 2007.

[20] Christiane Fellbaum. Wordnet. In *Theory and applications of ontology: computer applications*, pages 231–243. Springer, 2010.

[21] George A Miller. Nouns in wordnet. *WordNet: An electronic lexical database*, pages 24–45, 1998.

[22] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. *Dbpedia: A nucleus for a web of open data*. Springer, 2007.

[23] JR Huang, Shu-Kai Hsieh, Jia-Fei Hong, Yun-Zhu Chen, I-Li Su, Yong-Xiang Chen, and Sheng-Wei Huang. Chinese wordnet: Design, implementation, and application of an infrastructure for cross-lingual knowledge processing. *Journal of Chinese Information Processing*, 24(2):14–23, 2010.

[24] Francis Bond and Ryan Foster. Linking and extending an open multilingual wordnet. In *ACL (1)*, pages 1352–1362, 2013.

[25] Shan Wang and Francis Bond. Building the chinese open wordnet (cow): Starting from core synsets. In *Proceedings of the 11th Workshop on Asian Language Resources, a Workshop at IJCNLP*, pages 10–18. Citeseer, 2013.

[26] Hou Hanqing Lu Yong. Research on automatic acquiring of chinese synonyms from wiki repository. 3:287–290, Dec 2008.

[27] Pablo E. and Kyomin Jung. Knowledge extraction through etymological networks: Synonym discovery in sino-korean words. In *Proceedings of the 5th International conference on Information and Knowledge Management (ICIKM 2016)*, 2016.

[28] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391, 1990.

[29] Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186. Springer, 2006.

[30] Sascha Rothe and Hinrich Schütze. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. *arXiv preprint arXiv:1507.01127*, 2015.

[31] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[32] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393(6684):440–442, 1998.

[33] Diane J Cook and Lawrence B Holder. *Mining graph data*. John Wiley & Sons, 2006.

[34] Inderjit S Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 269–274. ACM, 2001.

[35] Armen S Asratian, Tristan MJ Denley, and Roland Häggkvist. *Bipartite graphs and their applications*, volume 131. Cambridge University Press, 1998.

[36] Daniel A Schult and P Swart. Exploring network structure, dynamics, and function using networkx. In *Proceedings of the 7th Python in Science Conferences (SciPy 2008)*, volume 2008, pages 11–16, 2008.

[37] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research*, 12:2825–2830, 2011.

[38] Christos Boutsidis and David P Woodruff. Optimal cur matrix decompositions. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 353–362. ACM, 2014.

[39] Vincent D Blondel, Anahí Gajardo, Maureen Heymans, Pierre Senellart, and Paul Van Dooren. A measure of similarity between graph vertices: Applications to synonym extraction and web searching. *SIAM review*, 46(4):647–666, 2004.

[40] Eric Jones, Travis Oliphant, and Pearu Peterson. Scipy: Open source scientific tools for python. *http://www. scipy. org/*, 2001.

[41] Stefan Van Der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, 2011.

[42] U Kang, Charalampos E Tsourakakis, and Christos Faloutsos. Pegasus: A peta-scale graph mining system implementation and observations. In *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*, pages 229–238. IEEE, 2009.

[43] Reynold S Xin, Joseph E Gonzalez, Michael J Franklin, and Ion Stoica. Graphx: A resilient distributed graph system on spark. In *First International Workshop on Graph Data Management Experiences and Systems*, page 2. ACM, 2013.

[44] Jari Saramäki, Mikko Kivelä, Jukka-Pekka Onnela, Kimmo Kaski, and Janos Kertesz. Generalizations of the clustering coefficient to weighted complex networks. *Physical Review E*, 75(2):027105, 2007.

[45] Lars Backstrom and Jon Kleinberg. Romantic partnerships and the dispersion of social ties: a network analysis of relationship status on facebook. pages 831–841, 2014.

[46] Linton C Freeman. A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41, 1977.

[47] Linton C Freeman. Centrality in social networks conceptual clarification. *Social networks*, 1(3):215–239, 1978.

[48] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: bringing order to the web. 1999.

Abstract in Korean

어원학은 단어의 역사적 뿌리를/역사적 어근을 통해 단어의 구성을 연구하는 학문이다. 이 넓은 학문의 영역은 그 시작을 수백년 전으로 거슬러 올라가며, 인류의 문화와 언어의 이해함에 있어 크게 기여하였다. 디지털 시대로부터 시작한 컴퓨터 언어학은, 이러한 어원학과 비교해 그 나이를 따졌을 때 상당히 어린 분야이지만, 계속해서 진보해 왔으며 최근에는 인공지능과 머신러닝이 가져온 변화 등에 의해 발전을 이루었다. 그럼에도 불구하고, 컴퓨터 언어학은 아직 어원학 지식의 유용성을 모두 발휘하지 못 한 것이 사실이다. 이 연구는 어원학을 컴퓨터 언어학에 기여하기 위한 시도이다.

본 연구에서는, 어원들을 단어로 연결짓는 연결망을 만듦으로써, 인간 언어의 어휘 사이에 존재하는 복잡한 어원적 관계를 사용하는 기술을 제안한다. 이 기술은 한자와 한글에 속하는 단어들에 관한 의미론적 통찰력을 얻기 위해 사용되었다. 본 연구에서는 지도학습, 비지도 학습 두 가지의 과제를 시행하였으며 이를 통해 어원학이 의미론에 관한 지식을 추출함에 있어 효과적임을 보였다.

Appendix A

Unipartite projection

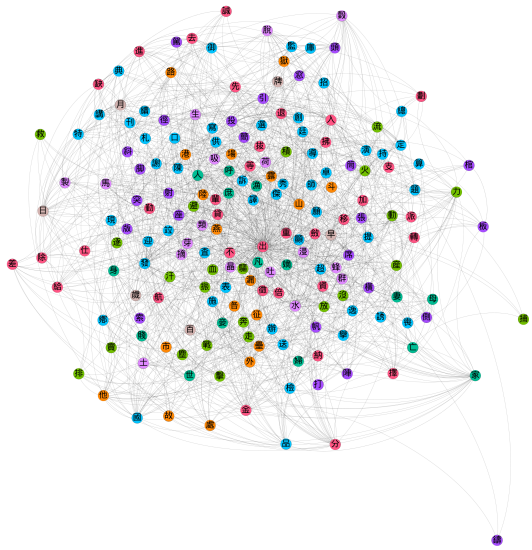


Figure A.1: Ego-network of the character 出 'Exit' in the root-unipartite projection of the source graph.

Appendix B

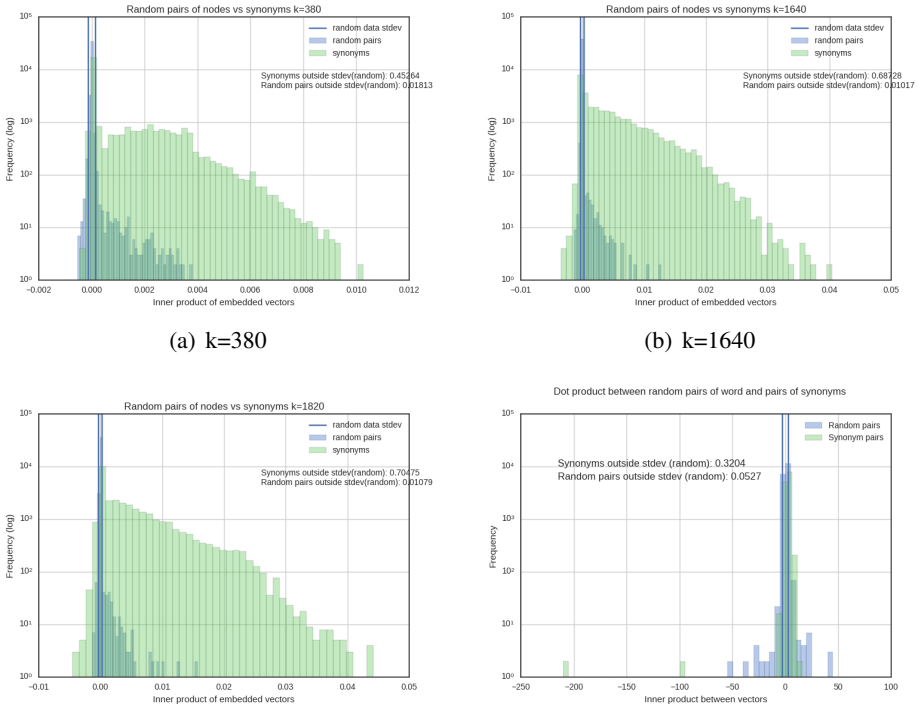
Supervised learning features

Table B.1: Features calculated for each pair of nodes

Feature name	**Formula**
Reciprocity	$A \in neighbors(B)$
Self-neighbor degree ratio	$\left\lvert \frac{degree(A)}{avg\_deg(neigh(A))} - \frac{degree(B)}{avg\_deg(neigh(B))} \right\rvert$
Normalized two-step walks fr. A to B	$\frac{two_step_walks(A,B)}{min(degree(A),degree(B))}$
Absolute degree difference	$\lvert degree(A) - degree(B) \rvert$
Sum of their degrees	$degree(A) + degree(B)$
Normalized degree ratio	$\frac{min(degree(A),degree(B))}{max(degree(A),degree(B))}$
Sum of clustering coefficients [44]	$c_A + c_B$
Abs. difference of clustering coef.	$\lvert c_A - c_B \rvert$
Product of clustering coefficients	$c_A \cdot c_B$
Dispersion between nodes [45]	$disp(A, B)$
Sum of betweenness centrality [46]	$betweenness(A) + betweenness(B)$
Abs. difference of betweenness scores	$\lvert betweenness(A) - betweenness(B) \rvert$
Sum of their closeness centrality [47]	$closeness(A) + closeness(B)$
Abs. difference of closeness centrality	$\lvert closeness(A) - closeness(B) \rvert$
Sum of PageRank scores [48]	$PR(A) + PR(B)$
Abs. difference of PageRank	$\lvert PR(A) - PR(B) \rvert$
Distance between nodes	$length(path(A, B))$
Belong to same 6-clique	
Belong to the same 7-clique	

Appendix C

Performance of embeddings



(a) k=380

(b) k=1640

(c) k=1820

(d) Result from embeddings released from [3]

Figure C.1: Log-scale histograms of dot product between embedding vectors between (green) pairs of synonyms, and (blue) random pairs of words. The vertical lines (blue) represent the standard deviation of similarity of random pairs.