A Study on Detecting Domain-Specific Senses and its Application to Text Categorization

ATTAPORN WANGPOONSARP

UNIVERSITY OF YAMANASHI

Supervisor: Prof. Fumiyo Fukumoto

A thesis submitted in fulfilment of
the requirements for the degree of
Doctor of Philosophy

Integrated Graduate School
of Medicine, Engineering, and Agricultural Sciences
(Life Information System)
University of Yamanashi
Japan

1 March 2021

Abstract

Recently, Natural Language Processing tasks are widely used from research that has been developed hastily. The text processing field is popular research since users need to process a large number of documents for various applications such as translation, text categorization, and question answering. One of the techniques used to increase performance is to choose the appropriate sense of a word in a document. However, each word can have more than one sense. Especially the word that is used frequently has more senses than the word that is less used.

This thesis consists of three works. The first work presents an unsupervised method for identifying a proper sense in a document called "Domain-Specific Senses (DSS)". This approach is based on the similarity of senses which is obtained by word embedding learning in order to resolve the inefficient semantic capture of the traditional method. The Markov Random Walk (MRW) model is applied as a ranking algorithm to detect DSS. The result of the sense assignment on Reuters corpus demonstrates that my method attained 1.5 improvements on the Inverse Rank Score (IRS) over the Cosine approach.

Because the first work presents unsupervised learning, most of these methods had poor performance. Therefore, in this second work, semi-supervised learning is proposed to resolve the problem. This approach base on "Graph Convolutional Networks (GCN)" and "Bidirectional Encoder Representations from Transformers (BERT) is called APPNP (Approximate Personalized Propagation of Neural Predictions)". My experimental results show that this approach works well and attain a 0.647 Macro F1-score.

The third work of the thesis is the extrinsic evaluation of DSS to see the influence of DSS in both an unsupervised method and a semi-supervised method. DSS is combined into text categorization. DSS initially is replaced proper senses with a target word that appears in a document. The technique used for text categorization is a single channel of Convolutional Neural Network(CNN). However, in both experiments, the number of categories is increased from 6 to 14 instead. The result shows a single channel of CNN with DSS at Topmost 10% gain Macro F-score at 0.832 and is better than CNN at 0.046. There is a comparison between my approach and the WSD approach i.e. Context2vec. The results show my DSS approach obtains a better Macro F-score than Context2vec at 0.053. Moreover, I also applied DSS obtained by a semi-supervised method to text categorization and gained a macro F1-score at 0.918, while the CNN baseline method was 0.776.

Keywords: Domain-Specific Senses; Markov Random Walk; Text categorization; Word Sense Disambiguation; Neural random walk model.

Acknowledgements

I wish to express my sincere appreciation to my supervisor, Professor Dr.Fukumoto Fumiyo, who has the substance of a genius: she convincingly guided and encouraged me to be professional and do the right thing even when progress was slow. Without her persistent help, the goal of this thesis would not have been realized. I am deeply indebted to Professor Dr.Suzuki Yoshimi for permission to use his lab, high-performance server, support for installing software, maintaining the server available 24/7 service, his advice and comments have been a great help in developing my work. I also wish to thank Prof. Li Jiyi for the thesis discussion. I am grateful to the thesis examination committee for reviewing, commenting on my work. I am grateful to Shimura Kazuya, for discussing and supporting my research. With his help, I can test my experiment much time and obtain the result conveniently. Thank lab mates, Mishima, and Goto who support me for any my requested very well.

And of course thanks to the Prince of Songkla University. Without their support and funding, this thesis could not have reached its goal.

Acronyms

APPNP	Approximate Personalized Propagation of Neural Predictions
BCE	Binary Cross Entropy
BERT	Bidirectional Encoder Representations from Transformers
CBOW	Continuous Bag-of-Words
CNN	Convolutional Neural Network
DSS	Domain-Specific Senses
EMD	Earth Mover's Distance
FSH	First Sense Heuristic
GCN	Graph Convolutional Networks
IR	Information Retrieval
IRS	Inverse Rank Score
LDA	Latent Dirichlet Allocation
LSI	Latent Semantic Indexing
MRW	Markov Random Walk
MSF	Multiplicative Scoring Function
MT	Machine Translation
NER	Named Entity Recognition
NLP	Natural Language Processing
NNLM	Neural Network Language Model
OM	Optimal Matching

PPR	Personalized PageRank
POS	Part of Speech
RCV1	Reuters news corpus
SFC	Subject Field Codes
SVM	Subject Vector Machine
VSM	Vector Space Model
WMD	Word Mover’s Distance
WSD	Word Sense Disambiguation

Contents

List of Figures

CHAPTER 1

Introduction

1.1 Identification of Domain-Specific Senses (DSS)

Domain-Specific Senses (DSS) are appropriate senses (meaning) of a target word in a particular context based on a domain. Frequently used word tend to have multiple senses than less used words (McCarthy et al. 2007). The website wordandphrase.info also has reported a ranking of the most frequently used word classified by 5 types of document source (spoken, fiction, magazine, newspaper, academic), i.e., the word "arm" with Part Of Speech (POS) as Noun has the ranking of 491, whereas the word "towel" with POS as Noun has the ranking of 3,710. According to statistics data, both words are used the most in the fiction category. The word "arm" used 49,778 times and 4,170 times for the word "towel". When querying sense in WordNet, the word "arm" has all 8 senses including 6 noun senses and 2 verb senses, while the word "towel" only has 2 senses including 1 noun sense and 1 verb sense. This example indicates a piece of evidence that corresponds to the above statement. The problem is when using the word "arm" in a sentence. It is difficult to identify an appropriate sense for the word "arm" because word the "arm" has more than one sense. Given the following sentences:

(1) "Armed groups attacked the soldiers this morning."
(2) "The sponsor is printed on his shirt arm"

The simplest way to define the sense to the target word "arm" is the First Sense Heuristic (FSH) which is to select the first sense of the target word. However, for both sentences, if the word "arm" is determined by FSH that mean the sense "a human limb" from the table 1.1 is selected which makes the meaning of the sentence is incorrect. One technique used to solve the problem is to applying domain information to the target words. Magnini et al. (Magnini et al. 2002) proposed a method to Word Sense Disambiguation (WSD) based on the hypothesis that domains are a feature used to connect words together as context with SFC (Subject Field Code) in WordNet Domain.

From the above sentences applying with domain information, it is noted that most words (armed, attacked, soldier) appear in the first sentences tend to have the same domain "military", thus the correct sense of the word "arm" from the TABLE 1.1 would be assigned with the third sense while in the second sentence, most words (printed, shirt, arm) tend to have the same domain "fashion". The correct sense of the word "arm" would be assigned with the sixth sense.

Assigning an appropriate sense to a target word using the domain method provides better performance than the traditional First Sense Heuristic (FSH) method (McCarthy et al. 2004). The sense is obtained from the FSH is based on the statistics of sense usage. FSH carries out the correct sense when the given sentence is the most commonly used, however, it ignores the meaning of the context of a sentence, thus assigning sense to a target word in a sentence sometimes are incorrect.

TABLE 1.1. "Arm" sense in WordNet

No.	Sense key	POS	domain	gloss
1	arm%1:08:00::	N	anatomy	(a human limb; technically the part of the superior limb between the shoulder and the elbow but commonly used to refer to the whole superior limb)
2	arm%1:06:03::	N	factotum	(any projection that is thought to resemble a human arm) "the arm of the record player"; "an arm of the sea"; "a branch of the sewer"
3	arm%1:06:01::	N	military	(any instrument or instrumentality used in fighting or hunting) "he was licensed to carry a weapon"
4	arm%1:06:02::	N	furniture	(the part of an armchair or sofa that supports the elbow and forearm of a seated person)
5	arm%1:14:00::	N	administration	(a division of some larger or more complex organization) "a branch of Congress"; "botany is a branch of biology"; "the Germanic branch of Indo-European languages"
6	arm%1:06:00::	N	fashion	(the part of a garment that is attached at the armhole and that provides a cloth covering for the arm)
7	arm%2:33:00::	V	military	(prepare oneself for a military confrontation) "The U.S. is girding for a conflict in the Middle East"; "troops are building up on the Iraqi border"
8	arm%2:40:00::	V	military	(supply with arms) "The U.S. armed the freedom fighters in Afghanistan"

From the above example, it has been observed that DSS influences the proper sense of words. DSS can be obtained from three approaches including

(1) Supervised learning requires the labeled input data (sense inventory), the training corpus, a set of features extracted from the sense inventory, and a classifier model. The model is trained until the best-predicted results are obtained and then the model is evaluated with test data to see the model's performance.
(2) Unsupervised learning is a technique that allows the model to discover data features that are undetected and they require sense inventory without labeled data, the corpus, a set of features extracted from the sense inventory, and a classifier model.

(3) Semi-supervised learning is another technique that relies on a small portion of training labeled data to distinguish unlabeled data. It also requires the corpus, a set of features extracted from the sense inventory, and a classifier model. The model is trained until the best-predicted results are obtained and then the model is evaluated with test data to see the model's performance same as supervised learning.

Recently, this task is widely studied because the advancement of the neural network has been developed rapidly and produces better results than the traditional method.

DSS is essential in NLP, it can be utilized in many applications following as:

(1) Machine translation (MT) is an automatic translation. It is the translation of text from one language to another language. The benefit of MT is that it can translate text in a short time and also save translation costs by humans. The translator must interpret and analyze every word in a sentence and know how each word influences the other. It is not a word-for-word translation. This requires analyzing the sentence structure and meaning of sentences in both the source language and the target language. DSS can be used to interpret the meaning of a word in a sentence more accurately.
(2) Question Answering (QA) is a system build to automatically answer human questions. Many efforts have been made to tackle a wide variety of question types. There are generally two types including Information Retrieval-based (IR-based) and Knowledge-based. IR-based focuses on searching for relevant documents to queries, while knowledge-based that focuses on interpret questions by transforming the semantic representation from queries to the logical representation and then retrieve documents from the database. QA requires DSS to reduce the ambiguity of questions. As a result, QA can retrieve documents related to the queries effectively.
(3) Text categorization, the goal of this task is to assign predefined labels to a text. Applications in this task are popular and widely used nowadays, for example, junk mail detection, article paper categorization, etc. This requires DSS to help a classifier because it is semantic oriented, therefore identifying the correct senses of words improves classification efficiently.
(4) Text extraction, it is an application for scanning and extracting keywords or phrase from various texts such as news, polls, and customer sentiment, etc. to detect trending news or product that customer needs. DSS can make the ambiguous text more clear and it affects text extraction that pulls the most relevant words powerfully.

An automated method (McCarthy et al. 2004) proposed by McCarthy is used for assigning predominant noun senses. They experimented with only two domains including sports and finance. While their method uses a thesaurus acquired from automatically parsed text based on the Lin method(Lin 1998). The k nearest neighbors to each target word, together with the similarity score between the target word and its neighbor. However, the issues of their work are the number of experimental domains is small and the method for determining DSS is based on statistical data.

My study in this thesis experiments more domains than their work to be able to apply to practical NLP application. I focus on the above problems of DSS identification in a context that:

(1) How to extract predominant senses from each category
(2) How to measure the efficacy of predominant senses

I propose the methods to solve the problems consists of (1) to identify DSS with two methods including an unsupervised method and a semi-supervised method, an unsupervised method needs to leverage word embedding, similarity metric, and Markov Random Walk (MRW), whereas a semi-supervised method relies on BERT and a neural random walk model (2) to evaluate the performance of DSS when applying to the downstream application.

1.2 Thesis Outline

In this thesis, I focus on word embedding learning for capturing the sense similarity and propose two techniques for identifying DSS. My proposal is (1) to develop an unsupervised method for leveraging word embeddings and (2) to develop a semi-supervised method using a neural random walk model.

In CHAPTER 2, I review some researches related to Domain-Specific Senses including Domain-Specific Senses, similarity measurement, link analysis, and text categorization.

In CHAPTER 3, I focus on the problem that how to identify DSS from a text corpus. I propose an unsupervised method based on word embeddings and Markov random walk model. I extract only nouns with frequency is greater than 5 from the RCV1 corpus and I use nouns to retrieve glosses from WordNet to generate word embeddings with Word2Vec. I then measure sense similarity with WMD and rank them with Markov random walk model. Moreover, I compare the DSS from my method with Cosine similarity to verify the results of sense assignments.

In CHAPTER 4, I focus on the problem that how to identify the number of DSS more accurately. Firstly, I initially experiment on DSS by modifying word embedding from Word2Vec as the BERT and the Deepwalk to see the quality of them and the result show the BERT is better than the Deepwalk, therefore I applied BERT to a new method. I propose a semi-supervised method base on the APPNP model and connectivity on the graph. The POS is used in the experiments consist of nouns and verbs as I use in the quantitative experiment of CHAPTER 3. APPNP model is based on the GCN model that is a powerful neural network even 2 layers of GCN can generate useful feature representation of nodes on graphs. Finally, I evaluate this method with the CNN method and the unsupervised method.

In CHAPTER 5, I focus on the problem that how DSS influence to extrinsic downstream application, i.e. text categorization. I compare the aforementioned methods from CHAPTER 3 and CHAPTER 4. I divide a text categorization comparison for two sections consisting of 1) a comparison between unsupervised learning and the WSD approach 2) a comparison between semi-supervised learning and unsupervised learning.

In the CONCLUSION, I summarize the thesis results and presents several approaches for future work.

CHAPTER 2

Related Work

In this chapter, I align the related work to five sections, i.e., Domain-Specific Senses, similarity metric, pre-trained language model, link analysis, and text categorization.

2.1 Domain-Specific Senses

2.1.1 Statistical data of context

Each Word possibly has more than one sense. Identifying the correct sense is a great challenge. One of the techniques that use for detecting is DSS that is a technique to determine a suitable sense of a target word in a sentence based on a domain of context. perhaps the first Domain-Specific Senses were proposed by Walker and Amsler that method assigned domain from context using LDOCE (the Longman Dictionary of Contemporary English) with counting a frequency of subject code of each word in a context which subject code had the highest count is assigned to the domain of context (Walker and Amsler 1986). Domain-specific sense of a word has attracted the attention of NLP researchers. It is crucial information for many NLP tasks such as IR, WSD, and Machine Translation(MT).

2.1.2 One Sense Per Discourse to SFC(Subject Field Codes)

Gale *et al.* first observed a tendency to share sense in the same discourse (Gale et al. 1992). To make the best use of the tendency, a method for automatically detecting the one sense given a document is required. Magnini *et al.* presented a lexical resource where WordNet 2.0 synsets were annotated with SFC that is useful in NLP tasks such as Information Retrieval (IR) which SFC are applied to enlarge a query and the keyword to contain more word in the intersection for improving recall and precision. Their procedure exploit WordNet structure (Magnini and Cavaglia 2000, Magnini et al. 2002, Bentivogli et al. 2004). SFC is structured in hierarchical manner where the upper level, the specificity is more general i.e. "art", while the lower level is more specific i.e. "drawing". However, there are a number of synsets in WordNet that are commonly used such as stop words ("the", "I", "in") and are not specific to any particular SFC i.e. man#1 "an adult male person (as opposed to a woman)", date#1 "day of the month". Therefore, FACTOTUM SFC has been created as a label for such synset. The annotated 96% of WordNet synsets of the noun hierarchy. However, mapping domain labels for word senses was semi-automated and required hand-labeling.

2.1.3 DSS automated method

McCarthy *et al.* addressed the SFC problem and proposed an automated method for assigning predominant noun senses (McCarthy et al. 2004). They used a thesaurus acquired from raw textual corpora and WordNet similarity package that includes similarity measures such as Lesk and JCN. They also used parsed data to find words with a similar distribution to the target word. Unlike (Buitelaar and Sacaleanu 2001) method, McCarthy *et al.* evaluated their method using publicly available resources: the hand-tagged resources SemCor and the SENSEVAL-2 English all-words task. The motivation for their work is similar to me, that is to capture predominant senses in different domains by ranking senses among documents. They tested 38 words containing two domains of Sports and Finance from the Reuters corpus, whereas I test 14 domains with 4,488 senses. Fukumoto et al. (Fukumoto and Suzuki 2010) also proposed an approach to acquired Identifying Domain-Specific Senses (IDSS) and applied to text categorization. However, they only focus on nouns. Moreover, their approach is based on the traditional Support Vector Machines (SVMs), that is the selection of the associated sense. Text categorization are also conducted by using SVMs.

2.1.4 The latest DSS work

Recently, Scarlini et al. (Scarlini et al. 2019) proposed OneSeC (One Sense per Wikipedia Category), an automatic method for sense-annotated corpora that were developed the methods from Gale et al. (Gale et al. 1992) and Yarowsky (Yarowsky 1993) to tackle the knowledge acquisition bottleneck because the advance of deep learning that needs a larger amount of data. Their method utilized a Wikipedia corpus along with category information by performing the following three steps:

(1) Category representation, which represents a lemma-category pair (l, C) as the Bag Of Words of the sentences of the category C in which the lemma l appears.
(2) Sense assignment, which annotates a sense of lemma to each lemma-category pair.
(3) Sentence Sampling, which extracts a number of sentences for each sense of each lemma in the lexicon by exploiting the relationship between lemma-category pairs and word sense calculated in the previous step.

They trained IMS (It Makes Sense) (Zhong and Ng 2010) with the data generated by OneSeC and evaluated the results with SOTA methods. The experimental results on English All-Words WSD demonstrated their F-score obtains at 69.0 compared with F-score of Train-O-Matic at 67.3 (Pasini and Navigli 2017) and F-score of OMSTI (Taghipour and Ng 2015) at 66.4. For multilingual All-Words WSD evaluation demonstrates their performance leading a supervised WSD to the state-of-art results on the multilingual WSD task. In this thesis, I proposed a method to identify automatically an appropriate sense for a target word based on domain.

2.2 Similarity metric

Word similarity can be measured base on two categories: Thesaurus-based algorithms and Distributional algorithms.

2.2.1 Thesaurus-based algorithms

In practice, hypernym/hyponym structures in WordNet have separate hierarchical structures, therefore word similarity can calculate within the same structure only, such as Noun-Noun or Verb-Verb. The simplest method based on word/sense is closest to each other only if there is the shortest path between them. Close words are usually parents or siblings. For fewer similar words, the farther the path is. Other examples of the algorithms in this category are as follows:

(1) Path-based similarity can be calculated from the inverse of the path-length as follows:

$$sim_{path}(c_1, c_2) = \frac{1}{pathlen(c_1, c_2)}, \tag{2.1}$$

where $pathlen(c_1, c_2)$ denotes the number of edges in the shortest path in the thesaurus graph between the sense c_1 and c_2

(2) Resnik similarity is computed related to their common information by the information content of the lowest common subsumer of the two nodes as follows:

$$sim_{resnik}(c_1, c_2) = -logP(LCS(c_1, c_2)), \tag{2.2}$$

where $P(LCS(c_1, c_2)$ denotes the probability of the lowest node in the hierarchy that subsumes both c_1 and c_2

(3) Lin similarity extends from Resnik work that the information in common between two concepts is twice the information in the lowest common subsumer $LCS(c_1, c_2$ as follows:

$$sim_{lin}(c_1, c_2) = \frac{2logP(LCS(c_1, c_2))}{logP(c_1) + logP(c_2)} \tag{2.3}$$

(4) Jiang-Conrath similarity is a distance function that returns a score on how similarity two-word senses are based on the information of the lowest common subsumer as follows:

$$sim_{JC}(c_1, c_2) = \frac{1}{2 \times logP(LCS(c_1, c_2)) - (logP(c_1) + logP(c_2))} \tag{2.4}$$

(5) Extended Lesk similarity is a dictionary-based method that two senses in a thesaurus are similar if their glosses contain overlapping words as follows:

$$sim_{eLesk}(c_1, c_2) = \sum_{r,q \in RELS} overlap(gloss(r(c_1)), gloss(q(c_2))), \tag{2.5}$$

where $RELS$ is the set of WordNet relations.

2.2.2 Distributional algorithms

The drawback of thesaurus-based algorithms is a limitation of the thesaurus for every language. They also encounter problems, i.e, a number of missing words, a number of missing phrases, and thesaurus does not work with verbs and adjectives because both of them have fewer hyponymy relations. The distributional algorithm sometimes is called vector-space models (VSM). The early work in the field of distributional semantics is presented by Firth (Firth

1957). His famous quotation is "You shall know a word by the company it keeps". Consider a given context "**Khanohm Jeen** is rice noodle in Thai cuisine. It often serves with curry and vegetables. Thais made Khanohm Jeen from rice flour". From the above context, guessing Khanohm Jeen as noodle cuisine like Somen because both words are similar and they have a similar word context. There are several methods for measuring similarity, such as

(1) Term-document matrix is the simplest way of measuring. This matrix describes the frequency (count vector) of terms that occur in a collection of documents. Consider a given TABLE 2.1, the pair of documents D_1 and D_2 are similar as well as D_3 and D_4 because their vectors are similar. Whereas a pair of word medicine/patient and eagle/dove are similar because their vectors are similar.

TABLE 2.1. term-document matrix

Word \ Doc.	D_1	D_2	D_3	D_4
medicine	17	29	0	1
patient	10	58	1	0
eagle	0	0	12	6
dove	0	0	2	18

(2) TF-IDF (Term Frequency-Inverse Document Frequency) is commonly used instead of raw term counts. It can be computed by multiplying two metrics: how many times a word appears in a document and the inverse document frequency of the word across documents, this indicates common or rare a word is in the documents. therefore, if the word is common and appears in many documents, the score close to 0. Otherwise, it close to 1. TF-IDF is calculated as follows:

$$tf - idf(t,d,D) = tf(t,d) \cdot idf(t,D), where \tag{2.6}$$

$$tf(t,d) = log(1 + freq(t,d)) \tag{2.7}$$

$$idf(t,D) = log(\frac{N}{count(d \in D : t \in d)}) \tag{2.8}$$

t denotes word in the document d from the document set D.

(3) Euclidean distance is a metric to measure dissimilarity between vector x and y is defined as follows:

$$dist(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{2.9}$$

Euclidean distance is only appropriate for data measured on the same scale.

(4) Cosine similarity is a metric used to measure how similar the words are regardless of their size. It computes the cosine of the angle between two vectors projected in a multi-dimensional space. The advantage of cosine similarity is that even if the two words are far apart by Euclidean distance, there are chances closer together. The

smaller the angle, the higher the cosine similarity. it can be calculated as follows:

$$sim_{cosine}(A,B) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{2.10}$$

The above algorithms are not suitable for document representation because of their frequent near-orthogonality (Greene and Cunningham 2006). There have been much attempts to avoid the problem by projecting document dimensional space into a lower-dimensional space, e.g., LSI (Deerwester et al. 1990) and LDA (Blei et al. 2003).

2.2.3 Word2Vec

Researchers have developed a word embedding model to learn the features of words. Learning the word embedding is unsupervised and it can be computed by using the textual corpus. The result is that it can be used to find words that have a similar meaning to a given word, even if they are different words. For example, vector("Japan") - vector("Yen") + vector("Thailand") is close to vector("Baht"). Word2Vec provides two architectures to generate word embedding: CBOW and skip-gram model. The word embedding is commonly used, i.e., Word2Vec (Mikolov et al. 2013), Fasttext (Bojanowski et al. 2016), Glove (Pennington et al. 2014), etc.

Word2Vec is a shallow neural network method for generating word embedding given a text corpus presented by Mikolov et al. (Mikolov et al. 2013). It is a groundbreaking work in the NLP research and commercial because many works apply this method and the NLP field has developed rapidly and increasingly in the last decade. Word2Vec can be utilized in different ways, for example, in Sentiment analysis usually, a sentence may consist of positive words or negative words or both. This task is a prediction overall of a sentence that it is either positive or negative sentiment more accurate because each word can learn from the surrounding words in different features. Named Entity Recognizer(NER) also benefited Word2Vec to analyze which words in a sentence should be named entities and what the classes should be labeled for words. For instance, Given the sentence: "Bob works at IBM in Japan." NER should be tagged Bob as "Person", IBM as "Organization" and Japan as "Location". The recommendation system is another one that can take advantage of Word2Vec using customer personality, ordering behavior, and then choose relevant products for customers to increase sales opportunities.

Word2vec's principle is the words that appear in similar contexts have similar embedding and it consists of 2 models for use, consisting of CBOW and Skip-gram. Consider an example sentence "Durians are smelly and spiky" FIGURE 2.1 shows the CBOW model uses surrounding words to predict a target word within a given window. in this example, the window size is five and surrounding words include durians, are, and, spiky to predict a target word (smelly) while Skip-gram in FIGURE 2.2, conversely, uses a target word (smelly) to predict surrounding words (durians, are, and, spiky).

The CBOW model is based on the feedforward Neural Network Language Model(NNLM). The model predicts the center word w_t given a representation of the surrounding words. The hidden layer H is obtained by summing the embeddings of the context words:

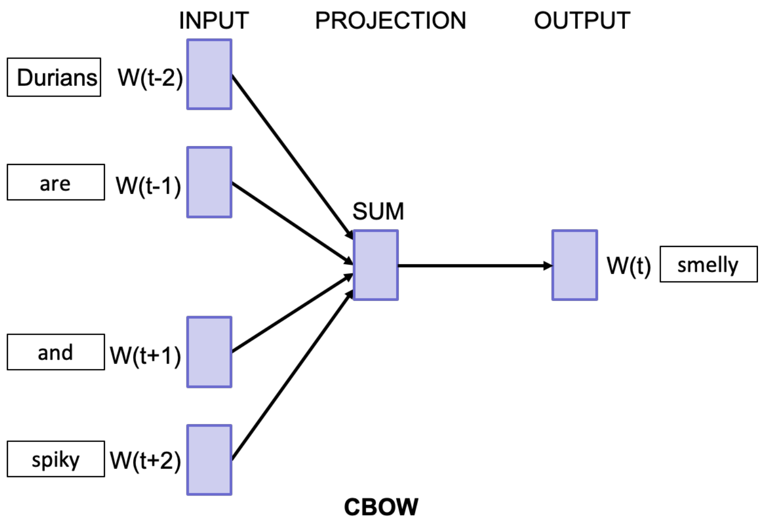

FIGURE 2.1. Continuous Bag-Of-Word architecture

$$H_{CBOWt} = \sum_{-R \le j \le R, j \ne 0} V(w_{t+j}) \tag{2.11}$$

R in Eq. (2.11) refers to the training window. $V(w_t) \in \mathbb{R}^d$ indicates d-dimensional word vector of w_t , and $H_{CBOW} \in \mathbb{R}^d$ shows d-dimensional vector. The model is learned by using negative-sampling approach. The objective of the model is to maximized L:

$$\begin{aligned} L = \sum_{w_t \in V} \Big(& \log(\tau(H_t W(w_t))) \\ & + \sum_{w_i \in N_t} \log(\tau(-H_t W(w_i))) \Big) \end{aligned} \tag{2.12}$$

t in Eq. (2.12) shows the t^{th} sampling, and V refers to the number of all words in the training corpus. N_t indicates negative sampling of k when the center word is w_t. Here, w_t is a positive sample, and all words except for w_t are negative samples. The CBOW is trained using stochastic gradient descent. The gradient is computed using the backpropagation rule (Rumelhart et al. 1986).

The skip-gram model's objective function L is to maximize the likelihood of the prediction of contextual words from given the center word. Given a sequence of training words w_1, w_2, $\cdots$, w_T, the objective of the model is to maximize L:

$$L = \frac{1}{T} \sum_{t=1}^{T} \sum_{-k \le j \le k, j \ne 0} \log p(w_{t+j} \mid w_t) \tag{2.13}$$

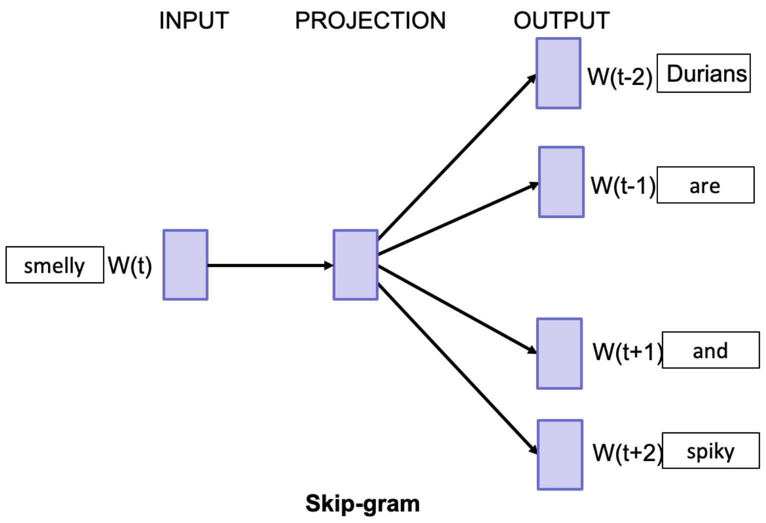

FIGURE 2.2. Skip-gram architecture

k in Eq. (2.13) refers to a hyperparameter defining the window of the training words. Every word w is associated with two learnable parameter vectors, *i.e.*, input vector I_w and output vector O_w of the w. Given the word w_j , the probability of predicting the word w_i is defined by Eq. (2.14).

$$p(w_i \mid w_j) \;=\; \frac{\exp({I_{w_i}}^\top O_{w_j})}{\sum_{l=1}^{n} \exp({I_l}^\top O_{w_j})} \tag{2.14}$$

n in Eq. (2.14) refers to the number of words in the vocabulary. For larger vocabulary size, it is not efficient for computation, as it is proportional to the number of words in the n.

For model training, Word2Vec provides two approaches consisting of Hierarchical softmax and negative sampling. Hierarchical softmax is a technique for estimating the softmax function which reduces the compute cost of training a softmax neural network.

FIGURE 2.3 illustrates a structure of hierarchical softmax model. Hierarchical softmax requires vocabulary to be organized into a binary tree. All internal nodes (white circle) have two branches and vocabulary is a leave node of this tree. In Word2Vec, it used the Huffman tree. As shown in this FIGURE, rare words are down at deeper levels, while frequent words are at shallower levels. For example, If I obtain a pair of words (bauxite, element) in the skip-gram model word "bauxite" is a target word of context window and the word "element" is a context word that I attempt to learn. Input vector "bauxite" compute dot product with the vector at the root node and then apply the Sigmoid activation function to get an activation value between zero and one. If the value is near to one go to the left and if the value is near zero go to the left. Finally, I can reach to word "element" only training on the three outputs and also can compute the probability of a word by multiplying a number from a root node to an "element" node, therefore it means this structure has shared the property together rather than the entire vocabulary.

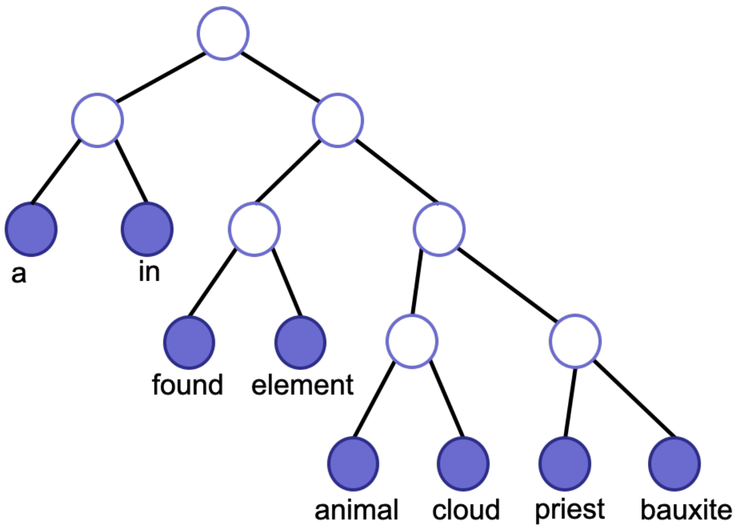

FIGURE 2.3. Huffman tree for the Hierarchical Softmax training model

$$
\begin{aligned}
p(w \mid w_j) &= \prod_{j=1}^{L(w)-1} \sigma([\![n(w, j+1) = ch(n(w,j))]\!] \quad \cdot {v'_{n(w,j)}}^{T} h), \\
\text{such that:}[\![x]\!] &= \begin{cases} 1 & \text{if } x \text{ is true;} \\ -1 & \text{otherwise} \end{cases}
\end{aligned} \tag{2.15}
$$

$ch(n)$ denotes left child of n. $v'_{n(w,j)}$ denotes output vector of the internal node $n(w,j)$ h denotes the output of hidden layer. Negative Sampling is another approach that reduces the compute cost of training a softmax neural network. I select a couple of contexts at random. With Word2Vec, the list of words that need to be similar is assigned as the positive class however negative class is assigned to the words that are not similar to the target word. I use negative sampling as the default is 5.

2.2.4 Word embeddings of a sentence

Domain-Specific Senses identification is a method for detecting the most prominent sense in each domain. Every word in gloss text should be represented the word with word embeddings and the sense similarity is then measured and finally, the most prominent senses are determined through graph analysis.

The most common method is the use of the Vector space model(VSM) that the sentence is represented as vectors through a bag of word or term frequency. However, despite some successes, the first attempts explored for noun sense only (Buitelaar and Sacaleanu 2001). Their approach has tested on three domains corpora consisting of business, soccer, and medical. They used extracting terms from the corpora and GermaNet. Their method defines the domain-specific relevance of synsets that occur within domain corpora. The results show an accurate prediction of DSS.

Other efforts have been acquired DSS (McCarthy et al. 2004). They use automatically parsed data based on the Lin method to capture words with a similar distribution to the target word. The experiment contains 38 words of two domains of Sports and Finance. The method relied on raw textual corpora and WordNet similarity package that consists of similarity metrics i.e., lesk, jcn, lin, res, etc. The experimental results show that the method achieves 64% precision for all-nouns tasks. However, the number of senses used in domains is too small. Their methods represent documents as vectors via a bag of words or term frequency. This can lead to the near-orthogonality problem (Greene and Cunningham 2006) that is two sentences have no common words but the meaning of the sentences are very close.

Consider the following similar sentences: "He buys a hoodie on the market." and "I shop for a jacket at the mall." The two sentences have similar meanings. All stop-word removed (I, he, a, on, the, for, at). the rest of the vocabulary(V) includes buys, hoodie, market, shop, jacket, and mall. A simple measure of similarity is to use one-hot encoding representations. A vector is represented as zeros except for the element at the index representing the corresponding word in the vocabulary. In this case, the size of V = 6, so the encoding should be followed as:

(1) buys=[1,0,0,0,0,0]; hoodie=[0,1,0,0,0,0]; market=[0,0,1,0,0,0]
(2) shop=[0,0,0,1,0,0]; jacket=[0,0,0,0,1,0]; mall=[0,0,0,0,0,1]

The words 'hoodie' and 'jacket' are different which is not true, even though they look similar. All the words are independent of each other. Latent Semantic Indexing (LSI)(Deerwester et al. 1990) and Latent Dirichlet Allocation (LDA)(Blei et al. 2003) are well-known techniques to overcome the problem by acquiring features from a low-dimensional space. One such attempt is the Word2Vec (Mikolov et al. 2013). It uses for generating distributed representations (word vector) that mean some dependence of one word on the other words. On the other hand, the word vector consists of continuous numbers to represent words that appear in a given vocabulary. Word2Vec has 2 models consisting of Skip-gram and Continuous Bag of Words (CBOW). Skip-gram is predicting the context word for a given target word. It is contrary to the CBOW model.

Suppose vocabularies include the following words "Salmon", "Tuna", "Cow" and "Pig". In vector space, each vector has 3 features continuous numbers represent as:

(1) Salmon = [0.8, 0.1, 0.8]
(2) Tuna = [0.7, 0.2, 0.9]
(3) Cow = [0.2, 0.9, 0.3]
(4) Pig = [0.1, 0.8, 0.2]

From the 1^{st} feature represents an animal type (aquatic, terrestrial). "Salmon" and "Tuna" have higher numbers because they live in marine. The 2^{nd} feature denotes blood type (warm-blood, cold-blood). "Cow" and "Pig" have higher numbers which mean warm-blood, while "Salmon" and "Tuna" have smaller numbers. The 3^{rd} element captures breathing type (gill, nose). "Salmon" and "Tuna" have higher numbers that represent breathing system with gill.

The above example shows an advantage of different dimensions in continuous word vector can detect the features of words whereas this feature is not available in one-hot encoding. The word vectors capture features only when they are trained on a large corpus.

Because the vocabulary is huge according to BBC website, Oxford English Dictionary consists of 171,146 words commonly used in English. It is very expensive to annotated by a human, therefore unsupervised learning is developed to learn the meaning of any word by itself. The word vectors are similar to the human brain. At first, the word vectors are randomized and probably have no meaning. However, when they have been trained repeatedly, the word vectors can detect the meaning of the words.

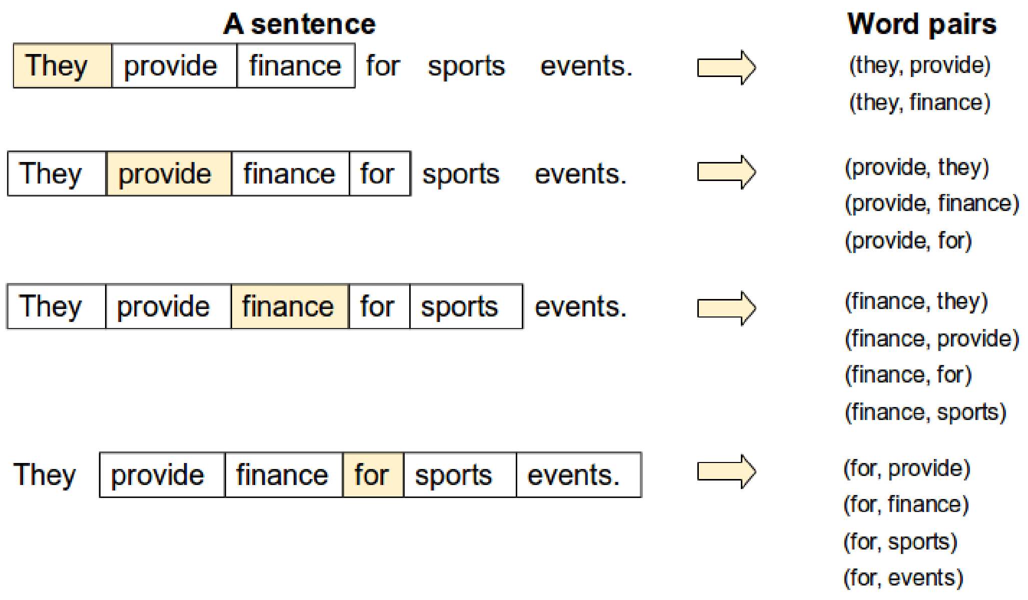

FIGURE 2.4. Training the Skip-gram model with word pairs

FIGURE 2.4 shows word pairs in a sentence that are used for training the Skip-gram model. Whereas training in the CBOW model is also similar to the Skip-gram model. The window size is set to 2. Initially, the 1st word "they" is defined as the target word and then generate word pairs between the target word and the words which are covered in the window ("provide" and "finance"). The generated results have 2 pairs including ("they", "provide") and ("they", "finance"). Next, move to the 2nd word "provide" and match it with words covered in the window ("they", "finance", "for"). Therefore, it can match 3 pairs in total ("provide", "they"), ("provide", "finance"), ("provide", "for"). The pairing between the target word and word in the window continues until the target word reaches the last word. The skip-gram model learns the number of pairing that has occurred. For example, word pairs ("sports", "event") are more common than word pairs ("sports", "zombie"). When the training is over if given the word "sports" as input, the output from the skip-gram model gives a higher probability of the word "event" than the word "zombie".

2.2.5 Word Mover's Distance (WMD)

Kusner *et al.* presented WMD to compute the similarity between two sentences (Kusner et al. 2015). It is based on Word2Vec embeddings that learn semantic representations for words from co-occurrences in sentences. These embedding techniques show that semantic relationships are often preserved in vector operations on word vectors. For example, vector(Bangkok)-vector(Thai)+vector(China) is close to vector(Beijing).

WMD is distances between word vectors are to some degree semantically. It utilizes this property of word vectors and treats text documents as a weighted point cloud of embedded words. It measures the dissimilarity between two sentences as the minimum amount of distance that the embedded words of one sentence reach the embedded words of another sentence as shown in FIGURE 2.5.

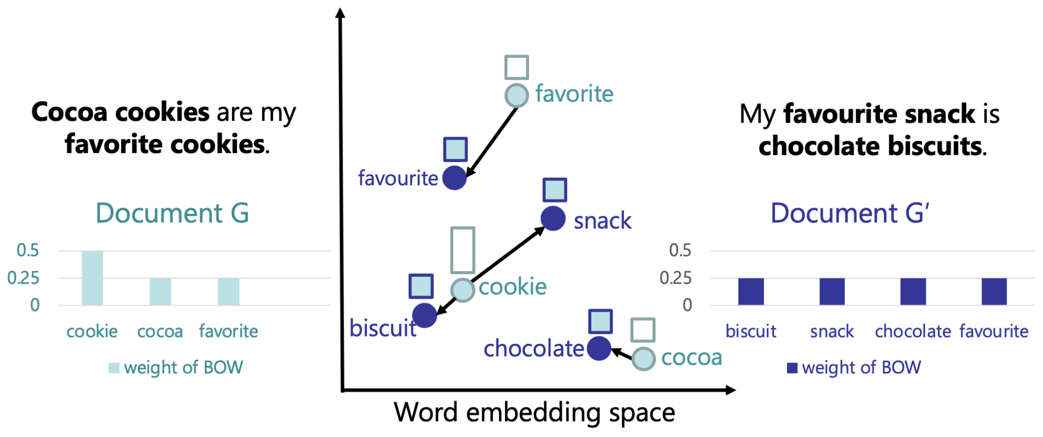

FIGURE 2.5. An illustration of the word's mover distance

From FIGURE 2.5, all non-stop words (bold) of both documents are embedded into word embedding space. The distance between two documents is the minimum cumulative distance that all words in a document G need to move to document G'. It notes that two documents are different: Cocoa cookies are my favorite cookies and My favourite snack is chocolate biscuits. Both documents do not have the same word, however, their meaning of both documents is very close. In this case, the similarity of word pairs: (favorite, favourite); (cookie, biscuit); (cookie, snack); (cocoa, chocolate). In contrast, both documents cannot be represented by the BOW model.

When the word embeddings are obtained, the distance among documents is defined by the following three parts: document representation, similarity metric, and a flow matrix.

(1) Document representation is represented as a vector D in which each member denotes a word's normalized frequency in the document.

$$D = [d_1, d_2, .., d_n]^T, \tag{2.16}$$

$$d_i = \frac{c_i}{\sum_j^n c_j}, \tag{2.17}$$

where c_i denotes word i appears c_i times in a given document

(2) A similarity metric is measured by the Euclidean distance in word embedding space of two given words, x_i and x_j is defined as follows:

$$c(i,j) = ||x_i - x_j||_2 \tag{2.18}$$

where x_i and x_j are different documents and $c(i,j)$ is the "travel cost" from word x_i to x_j

(3) Flow matrix(T) is computed from document A to document B. Each member in the flow matrix, T_{ij} denotes how many times word i in Document A travel to word j in Document B and then normalize the value by the total words in the vocabulary. The flow matrix is as follows:

$$\sum_{j=1}^{n} T_{ij} = d_i \tag{2.19}$$

$$\sum_{i=1}^{n} T_{ij} = d'_j \tag{2.20}$$

Finally, the semantic distance(L) is as follows:

$$L = \sum_{i,j=1}^{n} T_{i,j}\, c(i,j) \tag{2.21}$$

By tuning values in T, the distance between two documents can be obtained. The distance also is the minimum cumulative cost to move all words from one document to the other.

The results using eight real-word document classification datasets including Reuters and 20News in comparison with seven baselines including LDA show that the WMD attained at low k-nearest neighbor document classification error rates.

2.2.6 Measuring similarity with Word Mover's Distance(WMD)

To measure the document similarity in the simplest way first words should be transformed to Bag of Word(BOW), then calculate the average vector for all words in every document and use metrics such as cosine between vectors. If both documents are similar, cosine values are close to 1 which means they are quite similar. In contrast, if values are close to 0 which means they are totally different. However, documents sometimes are closely semantic, but the cosine values point out that they are different.

(1) Document 1: These mobile graphics programs are popular.
(2) Document 2: The photo editor is my favorite on Android.

From the above example, Both documents look similar, however, cosine values equal to 0 because they are no common keyword. Therefore, word embedding is crucial in order to detect the different features of the word. One technique used to measure the dissimilarity between two documents is Word Mover's Distance(WMD) proposed by Kusner (Kusner et al. 2015), which leverages on such word embedding to capture the semantic similarity of documents.

FIGURE 2.6 presents only non-stop words of two documents (D1, D2). The arrow lines represent direction between two words and are assigned with their distance. However, The red line denotes the shortest distance. A WMD similarity score is calculated with a sum of the shortest distance from one word of one document reach to a word of another document. A smaller similarity score means more similar.

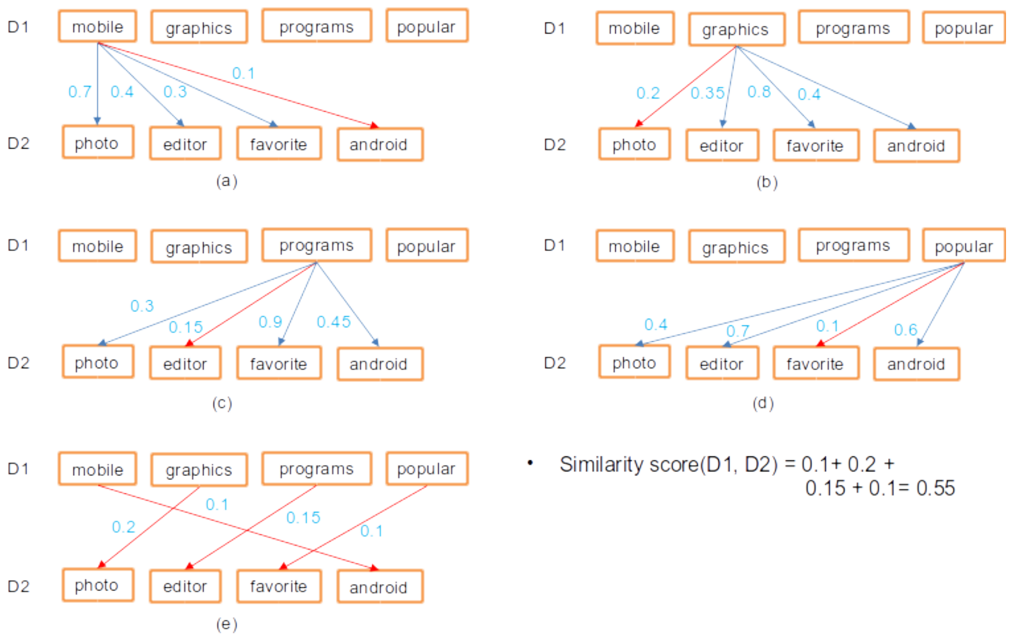

FIGURE 2.6. Acquiring a WMD similarity score

2.3 Pre-training Language Model

A major problem in NLP is that there are not enough large-scale labeled datasets. In contrast, large-scale unlabeled datasets are huge. Therefore, researchers have developed a variety of techniques to create general-purpose language models from unlabeled datasets is called "Pre-training Language Model". to learn universal language representation from datasets. The model can be fine-tuned on smaller task-specific datasets, for example, sentiment analysis and question answering, etc. For this thesis, I have experimented with two types follows as:

2.3.1 Context2Vec

Context2Vec is proposed by Melamud et al. (Melamud et al. 2016) that it is an unsupervised method for learning a generic context embedding using a bidirectional LSTM model. Generally, word embeddings such as Word2Vec are used to learn the semantic and syntactic of words. However, the drawback of Word2Vec is the model learns words only in a fixed window of sentential context, while, Context2Vec can learn every word in the sentential context. Context2Vec architecture is shown in FIGURE 2.8

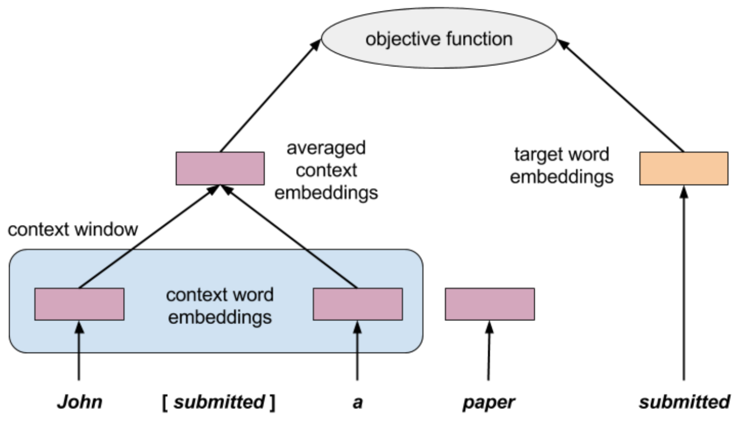

FIGURE 2.7. Word2Vec CBOW architecture (Melamud et al. 2016)

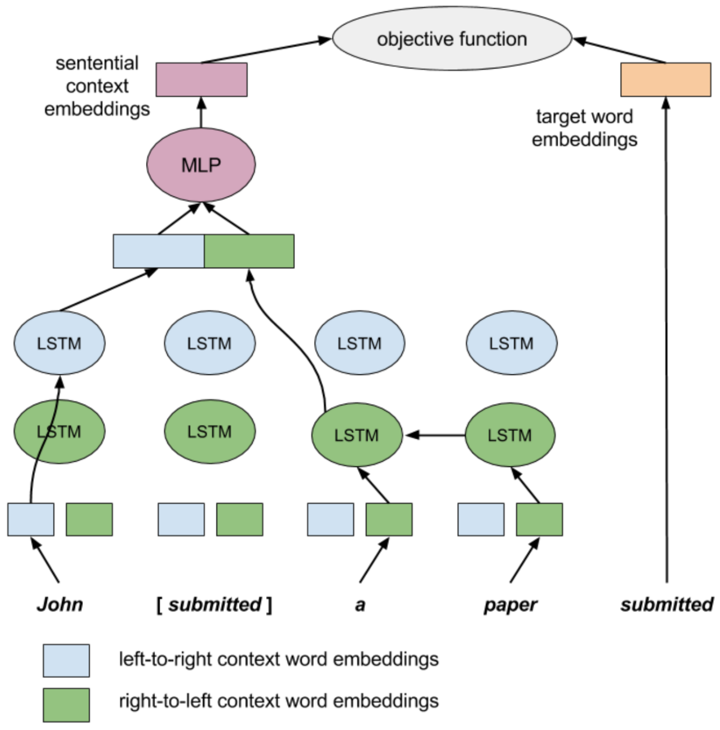
objective function
sentential context embeddings
target word embeddings
MLP
LSTM
John
[submitted]
a
paper
submitted
left-to-right context word embeddings
right-to-left context word embeddings

FIGURE 2.8. Context2Vec architecture (Melamud et al. 2016)

From the FIGURE 2.8, Context2Vec architecture looks similar to Word2Vec CBOW FIGURE 2.7, but instead of context modeling with bidirectional LSTM. However, both models can learn context and target embeddings simultaneously. Sentential context is fed into LSTM in both directions, from left to right and from right to left. The parameters of the two networks are separate from each other. The context of a target word in a sentence is represented by concatenating the LSTM output vector ("John") from left to right with another LSTM output vector ("a paper") from right to left. Therefore, the model can learn every word in the sentential context, and then the concatenated vector is fed into a multi-layer perceptron (MLP). The output from MLP is the joint sentential context surround the target word. At the same time, the target word is represented with its embedding. Finally, the last step is to learn the parameters of a network using Word2Vec's negative sampling as an objective function that enables both context embeddings and target word embeddings. Context2Vec can be used in a variety of NLP applications such as sentence completion, lexical substitution, and supervised WSD, etc. For this thesis, I used Context2Vec as supervised WSD to compare with my method for the second work.

2.3.2 BERT

BERT (Bidirectional Encoder Representations from Transformers) is a method of pretraining language representations developed by Devlin et al. (Devlin et al. 2018) The BERT concept is to build a language model to solve a problems sentence completion. For example, the given sentence "The woman went to the cafe and bought a [mask] of coffee." The model should fill the word "cup" 80 % of the time and the word "glass" 20 % of the time. Typically, the language model focuses on predicting the next word in a sequence. However, BERT uses a novel technique called Mask Language Modeling (MLM) where BERT randoms the most likely word in the sentence based on all words on the left and right of the mask word to predict what the mask word should be. Another technique is used called Next Sentences Prediction (NSP) to understand the relationship between two sentences. During training the model, 50% of the time comes after the first sentence, while 50% of the time it is a random sentence from the corpus. BERT predicts whether the second sentence is random or not, with the assumption is that the random sentence will be disconnected from the first sentence. The model is trained with MLM and NSP to minimize the loss function of the two techniques.

The high-quality embeddings are obtained from this method and it can apply to various tasks, i.e. text categorization, question-answering, etc. BERT is used to extract features as word or sentence embedding vectors from text data. For the text categorization task, the text document is classified more accurately, even if there's no keyword or phrase overlap. BERT is better than Word2Vec because each word in Word2Vec has a fixed representation regardless of the context within which the word appears, therefore Word2Vec is a context-free model. While BERT generates word representations that are dynamically informed by the words surround them, BERT is a context-based model.

For example, given two sentences: "They pinned a 'kick me' sign on his back." and "Their back showed some impressive running." Word2Vec generates the same word embedding for the word "back" in both sentences, while BERT generates the word embedding differently for each sentence rely on polysemy, the context-informed word embedding which made the result is better than SOTA word embeddings.

BERT is built on the basis of a Transformer model that generally consists of an encoder to read input and a decoder to predict the output. Because BERT's goal is to create a language representation, therefore, only the encoder is used. BERT's input is generated from the sum of the token embeddings, the segment embeddings, and the position embeddings. Token embeddings formatting has two special tokens follows as: [SEP] that is used to indicate the end of a sentence, or used as a separator between two sentences. [CLS] is used to indicate the beginning of a sentence. Segment embeddings is an indicator showing that Sentence A or Sentence B for each token. Positional embeddings use to indicate a token position in the sentence.

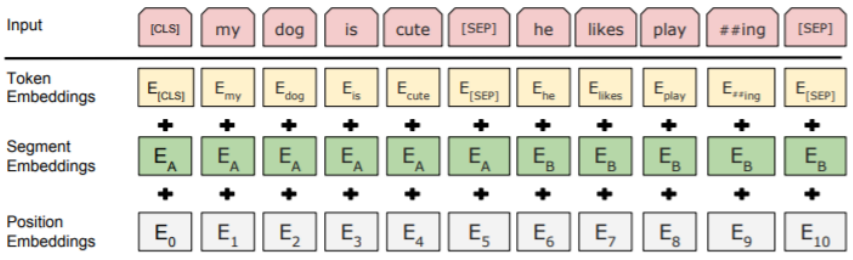

FIGURE 2.9. BERT input representation (Devlin et al. 2018)

From FIGURE 2.10, both BERT and OpenAI GPT architecture utilizes a transformer, however, BERT uses a bidirectional transformer, while, OpenAI GPT uses a left-to-right transformer. ELMo uses a bidirectional LSTM.

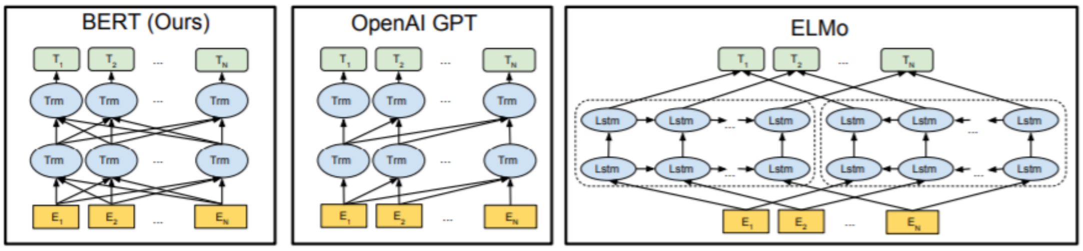

FIGURE 2.10. BERT architecture compared to SOTA architectures (Devlin et al. 2018)

There are currently many versions available from the smallest version with BERT-Tiny, Uncased (L=2, H=128, A=2) which represents a model consisting of 2 layers, 128 hidden units, and 12 attention heads to the largest version with BERT-Large, Cased (L=24, H=1024, A=24) which represents a model consisting of 24 layers, 1024 hidden units, and 24 attention heads. The available BERT model can download from https://github.com/google-research/bert.

For applying to downstream tasks by fine-tuning approaches, it requires a few parameters, and the results of a pre-trained model on GLUE benchmark, SQuAD, and SWAG datasets perform better than SOTA. The ablation tests show that a bidirectional transformer of BERT learns meaningful representation effectively. Furthermore, BERT can be utilized in a feature-based approach. The experiment results show BERT with a feature-based approach can reach the same performance on the Named Entity Recognition task as BERT with a feature-based approach. For this thesis, I have compared BERT with Deepwalk. I also applied BERT to a neural random walk model to acquire DSS.

2.3.3 Gloss Text Learning with BERT

BERT is a pre-training language model that can capture rich information from the text therefore they are useful for many NLP tasks. BERT utilizes a large-scale of unlabeled datasets to learn universal language representation from datasets and it is based on the transformer encoder model (Vaswani et al. 2017). BERT is very powerful and effective for many downstream tasks. Many researcher recognize the benefits of BERT and they have adapted BERT for any application, i.e., SciBERT (Beltagy et al. 2019) for scientific data, bioBERT (Lee et al. 2019) and BlueBERT (Peng et al. 2019) for biomedical data, ViLBERT (Lu et al. 2019) for vision-and-language task, MobileBERT (Sun et al. 2020) for mobile devices.

BERT architecture consists of two steps: pre-training and fine-tuning which are displayed in FIGURE 2.11 in the pre-training step, a sentence pair is an input where each sentence is divided as a token sequence. Each token is split as a word piece by using WordPiece embedding vocabulary (Wu et al. 2016). The representation of each token is summed up in the corresponding token, segment, and position embedding that has flowed to the encoder part of the transformer. On the left side of FIGURE 2.11 the output of the last hidden states of tokens is regarded as features that are utilized to the predicted masked tokens of the two-sentence pair. The hidden state of [CLS] is utilized to train the Next Sentence Prediction (NSP) task. After the pre-training step, the model can be utilized on many downstream tasks, such as Question Answering, Named Entity Recognition (NER) along with DSS identification which is demonstrated on the right side of FIGURE 2.11. The input of the model is labeled data and the format can be either a sequence of text, a sentence, or a pair of the sentence and the special format which shows the task-dependent. For example, in the DSS identification task, the input format is a sequence of the text.

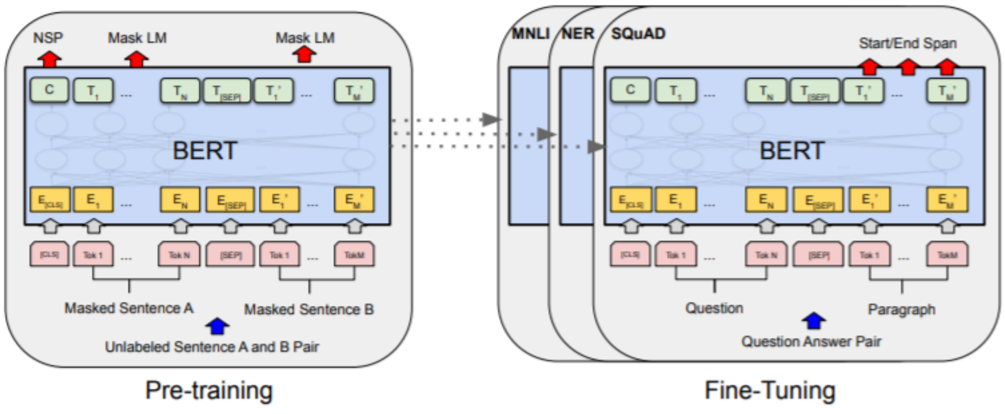

FIGURE 2.11. BERT architecture (Devlin et al. 2018)

The BERT goal is to represent a variable-length vector into a fixed-length vector, i.e. "soft drink" to [0.2, 0.6, 0.7]. Each attribute of the vector should encode some semantic of a sentence. The process for obtaining sentence embedding and word embedding is demonstrated in FIGURE 2.12 BERT can obtain as input either one or two sentences and applies the special token [SEP] to distinguish them while another token [CLS] always inserts at the start of the text. BERT provides its own tokenizer to split each word into word pieces (subwords and characters) For example, the word "falsifying" in a given sentence "He was caught falsifying financial accounts." is represented as ['fa', '##ls', '##ifying']. The double hash signs denote the subwords are part of a larger word and preceded by another subword. The reason for dividing the word into word pieces because the WordPiece model creates a fixed-size vocabulary of individual characters, subwords, and words that best fits language data containing English characters, 30,000 most common words, and subwords in the English language corpus. All of the hidden states of word pieces corresponding to every word are averaged, and the final output which corresponds to sentence embeddings are obtained. In my work, I used this approach to obtain gloss text embeddings from senses in FIGURE 4.4.

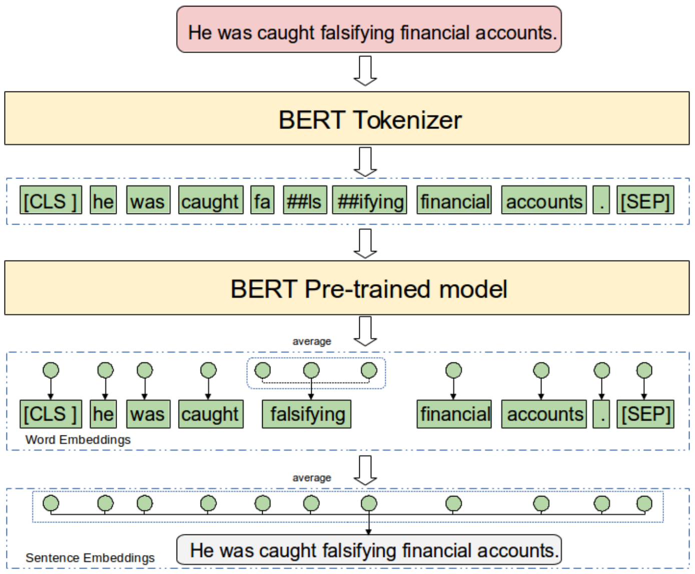
He was caught falsifying financial accounts.
BERT Tokenizer
[CLS] he was caught fa ##ls ##ifying financial accounts . [SEP]
BERT Pre-trained model
average
[CLS] he was caught falsifying financial accounts . [SEP]
Word Embeddings
average
Sentence Embeddings
He was caught falsifying financial accounts.

FIGURE 2.12. The procedure for computing sentence embeddings

2.4 Link analysis

A graph is a well-known technique is to analyze the strength of a relationship between nodes(vertices) through edges(links) and the direction of a relationship (undirected graphs, directed graphs) that can lead to detect the latent information in the graph. Each vertex initially votes other vertices and then applied the ranking algorithm to measure the importance of vertex in the graph. Many authors adopted graph-based model to their works that is, text semantic similarity (Ramage et al. 2009), document summarization (Mihalcea 2005) WSD (Sinha and Mihalcea 2007). Reddy attempted to use the Personalized PageRank algorithm (Agirre and Soroa 2009) over a graph representing WordNet to disambiguate ambiguous words (Reddy et al. 2010). They combined sense distribution scores and keyword ranking scores into the graph to personalize the graph for the given domain. The results showed that exploiting domain-specific information within the graph-based methods generated better results than using them as an individual. However, sense distribution scores were based on the frequency of neighbors of the target word from the thesaurus which was difficult to capture the distance between individual words. Recently, Dongsuk (Dongsuk et al. 2018) proposed a word similarity method based on the semantic structure of BabelNet from a knowledge-based graph. They evaluated the SemEval-2013 and SemEval-2015 datasets and the results indicated their method performed better than the state-of-art method in the SemEval-2013 dataset. Kutuzov (Kutuzov et al. 2018) presented Path2vec which encoded synset paths between graph nodes into dense vectors. Their results were better than graph embedding baselines. Perozzi et al. presented DeepWalk to learn latent representations of vertices in a network (Perozzi et al. 2014). They used local information obtained from truncated random walks to learn latent representations by treating walks as the equivalent of sentences. They applied DeepWalk to several multi-label network classification tasks including Flickr and Youtube and showed that it outperforms baseline methods.

Advantage from word embedding that can capture rich features between words and powerful WMD similarity metric. My method should benefit from both methods to identify similarities between sense. Whereas Markov Random Walk (MRW) model is used to decide the importance of nodes(senses) within a graph based on global information from the whole graph. An edge between nodes represents a vote cast from one node to the other. The higher score of nodes denotes that nodes are important. Finally, predominant senses are obtained from a graph.

2.4.1 Markov Random Walk

The MRW is a model that decides the importance of a vertex within a graph based on global information drawn recursively from the entire graph (Bremaud 1999). The essential idea is that of "voting" between the vertices. The model constructs a graph that demonstrates relationships between nodes. The graph-based ranking algorithm then is applied to compute the rank scores for nodes. The nodes with large rank scores are selected as important nodes. This PageRank, which is Brin and Page (Brin and Page 1998) work also applied this idea. For this thesis, I apply the MRW to the first work and the second work. Pasini and Navigli (Pasini and Navigli 2019) presented Train-O-Matic, Supervised WSD that is a method based on a graph $G = (V, E)$ where V represents synsets and E represents the semantic relations between synsets. Train-O-Matic can label to sense automatically and it requires only a minimum knowledge, WordNet-like resource, and a corpus. They apply it to perform on-the-fly domain adaptation by learning the sense distribution of the text to disambiguate. Train-O-Matic includes three parts follow as:

(1) Lexical profiling: A vector of synset is created from calculating the probability distribution of every synset over a graph using Personalised PageRank (PPR) as a propagation scheme and is calculated as follows:

$$v^{(t+1)} = (1-\alpha)v^0 + \alpha M v^{(t)}, \tag{2.22}$$

where M denotes the row-normalized adjacency matrix, v^0 is the restart probability, v^t is the probabilities of each node at time step t, and α is the damping factor set to 0.85. The result is a probability distribution over the knowledge graph for each word sense.

(2) Sentence scoring: the computation of the probability of a target sense appears in a given sentence.
(3) Sentence ranking and selection: the ranking of each sentence for a sense of a target word.

The motivation for their work is similar to mine, which is to identify predominant senses in different domains using PPR over a graph.

2.4.2 DeepWalk

A graph can capture relationships among the nodes which is a difficult task in a traditional data structure. However, graphs cannot be used directly in a learning model. Features have to first be created from the graph before the model can be utilized. DeepWalk is a learning algorithm for node embeddings that can capture the feature about the context of a node (the surrounding node).

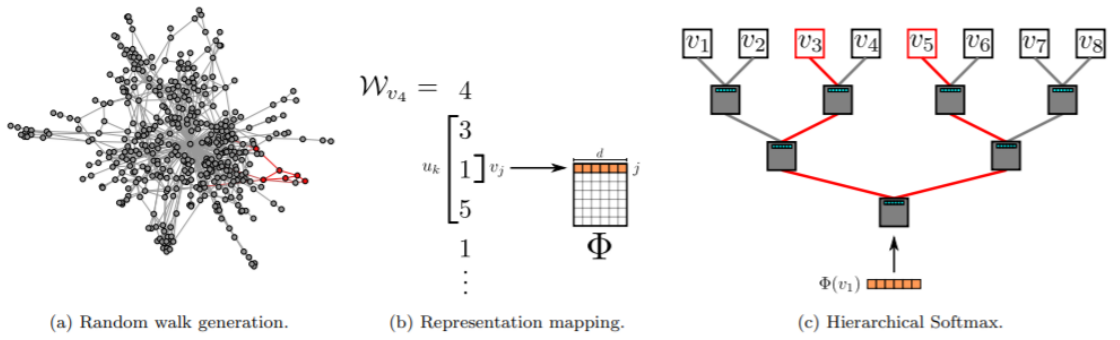
(a) Random walk generation.
$\mathcal{W}_{v_4}$ = 4 3 1 5 1
u_k
v_j
d
j
Φ
(b) Representation mapping.
v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8
$\Phi(v_1)$
(c) Hierarchical Softmax.

FIGURE 2.13. Deepwalk framework (Perozzi et al. 2014)

DeepWalk is proposed by Perozzi et al. (Perozzi et al. 2014). It has two parts including a random walk generator and an update procedure. The random walk generator 2.13(a) obtains a graph and samples a random node as the root of the random walk. The neighbor nodes are chosen at random of the last node visited until reach the maximum length. An update procedure utilizes with SkipGram 2.13(b) that use a sliding window of length 2w+1 over the random walk W_{v4} and mapping node v_1 to its embedding $\Phi(v_1)$. Hierarchical Softmax then is used for maximize the probability of v_1 co-occurring with its context $\{v_3, v_5\}$. Their results show DeepWalk representation performs F1 scores up to 10% higher than competing methods. In this thesis, I also have tested a comparison between DeepWalk embedding with BERT embedding and Word2Vec embedding as a preliminary test before the second work.

2.4.3 Graph Convolutional Network (GCN)

Neural networks over the graph have attracted the attention of NLP researchers because they can learn latent representation between nodes over the graph. One of the popular techniques is Graph Convolutional Networks (GCN).

GCN is a model that utilizes a convolution on a graph and it is proposed by Kipf and Welling (Kipf and Welling 2016). Traditional convolution is used to split an image into small images to perform feature extraction. The features are extracted from small images. They are useful for classification tasks. Similar to CNN, GCN uses a filter over the graph to extract important vertices and edges that can categorize nodes within the graph.

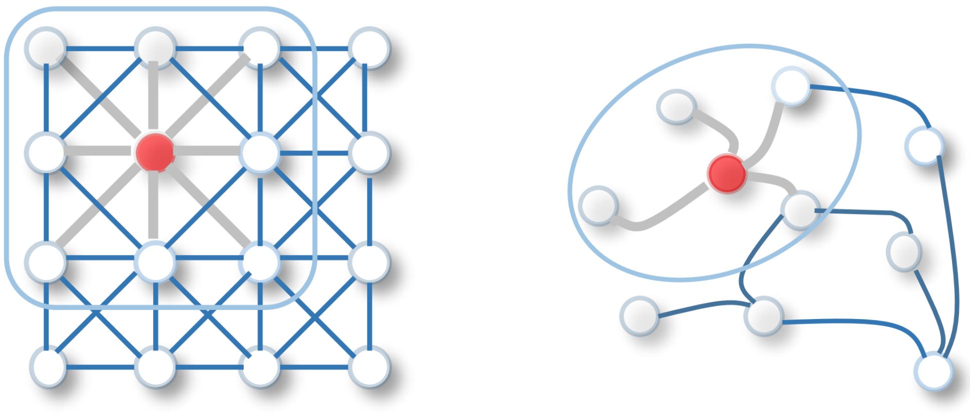

FIGURE 2.14. 2D Convolution(left) vs. Graph Convolution(right) (Zonghan et al. 2019)

From the FIGURE 2.14 the left side of the FIGURE shows 2D convolution, each pixel of the image represent as a node and the filter is used to determine neighbors. The 2D convolution utilizes a weighted average of pixel values of the red node together with its neighbors. The right side of the FIGURE shows graph convolution that utilizes the average value of node features of the red node together with its neighbors to get a latent representation of the red node. The difference from 2D convolution is the neighbors of a node are unordered and have a variable size.

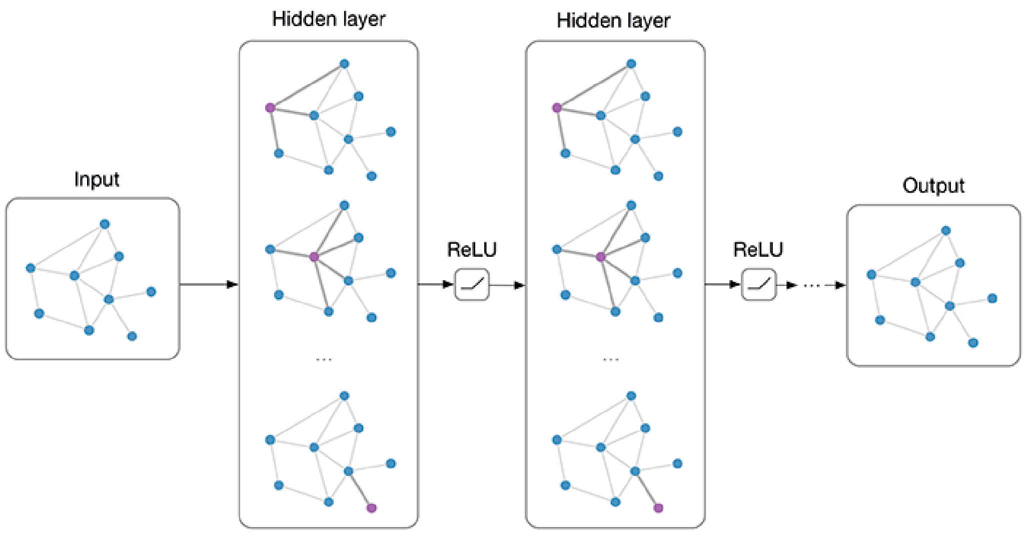

FIGURE 2.15. Multi-layer Convolutional Network (GCN)

The FIGURE 2.15 shows a GCN that operates on graphs. Given a graph $G = (V, E)$ is a graph that represents the relationships between nodes by adjacency matrix $A \in \mathbb{R}^{n\times n}$ where n is a node set. $\tilde{A} = A + I_n$ represents the adjacency matrix with added self-loops. V is the vertices set consisting vertex v_i that is a sense gloss texts. E is a edge set. Each edge e_{ij} denotes co-occurrence relationship of v_i and v_j in documents. The sense gloss texts v_n are represented by the feature matrix $X \in \mathbb{R}^{n\times f}$ where f denotes the number of features, and the category matrix $Y \in \mathbb{R}^{n\times c}$ where c denotes the number of categories. GCN model with two layers is defined by

$$Z_{\text{GCN}} = \text{softmax}\Big(\hat{\tilde{A}}\ \text{ReLu}\ \big(\hat{\tilde{A}}XW^{(0)}\big)\ W^{(1)}\Big), \tag{2.23}$$

where $Z_{GCN} \in \mathbb{R}^{n\times c}$ is the prediction of each node, $\hat{\tilde{A}} = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$ is a normalized adjacency matrix with self-loops, $\tilde{D}_{ij} = \Sigma_k \tilde{A}_{ik}\delta_{ij}$ is the diagonal matrix, and $W^{(0)}$ and $W^{(1)}$ are weight matrix.

The experimental results on four datasets with GCN that performs categorization better than related methods by a significant margin.

2.4.4 (Approximate) Personalized Propagation of Neural Predictions (PPNP, APPNP)

However, The propagation of neural networks with message passing loses its focus on the local neighborhood (Li et al. 2018) when many layers are applied. Klicpera et al. (Klicpera et al. 2018) addressed the problem and proposed the Personalized PageRank (PPR) as a propagation scheme instead to solve the problem.

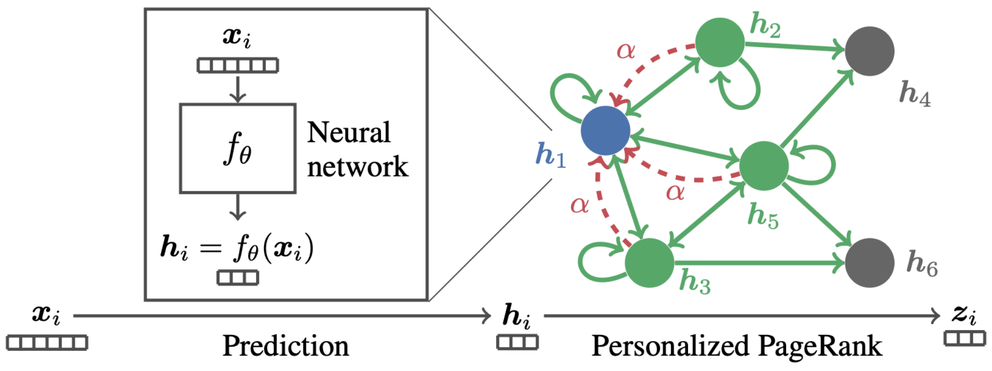

FIGURE 2.16. Personalized Propagation of Neural Predictions (Klicpera et al. 2018)

The FIGURE 2.16 demonstrates APPNP framework has two stages including node predictions and propagation. Firstly, each node has its own features (x_i) and then feeds them into the neural network to generate the predictions of nodes (h_i). Secondly, the predictions are propagated to neighbors using an adaptation of personalized PageRank. α is a teleport probability that allows a node can jump to any other node in the graph.

The propagation in this graph model derived from PageRank Brin and Page 1998 that is defined by $\pi_{pr} = A_{rw}\pi_{pr}$ with $A_{rw} = AD^{-1}$ where D denotes the diagonal matrix. The Personalized PageRank adapts from PageRank for recognizing the connection between nodes and is defined by $\pi_{ppr}(i_x) = \alpha\big(I_n - (1-\alpha)\hat{\tilde{A}}\big)^{-1} i_x$ where x is root node and i_x denotes teleport vector with teleport probability $\alpha \in [0,1]$.

APPNP applies the above ideas and produces the first prediction for each node and then propagates it with PPR to produce the final prediction. It is defined by

$$\begin{aligned} Z^{(0)} &= H, \quad H = f_\theta(X), \\ Z^{(k+1)} &= (1-\alpha)\hat{\tilde{A}}Z^{(k)} + \alpha H, \\ Z^{(K)} &= \text{softmax}\Big((1-\alpha)\hat{\tilde{A}}Z^{(K-1)} + \alpha H\Big), \end{aligned} \tag{2.24}$$

where $Z \in \mathbb{R}^{n\times c}$ is the prediction of each node. H is the prediction for each node and represents both the starting vector and teleport set, f_θ is a neural network with the number of parameters θ. K denotes the number of power iteration steps for approximate topic-sensitive PageRank and $k \in [0, K-2]$.

The experimental results show this model outperforms several recent methods for a classification task. In this thesis, I apply the model to identifying DSS in the second work.

2.5 Text categorization

Most of the data used for communication is unstructured text data, therefore, automated classification tasks are essential. Text categorization is about dividing the text into smaller to learn features and assign a meaningful label, such as topics, sentiment, and language, etc. In the past, Researchers use techniques i.e., Naive Bayes or Support Vector Machines. Currently, novel techniques based on neural networks consisting of CNN.

Many authors have attempted to apply deep learning techniques including CNN (Wang et al. 2018), the attention-based CNN (Yang et al. 2016), bag-of-words based CNN (Johnson and Zhang 2014), and the combination of CNN and recurrent neural network (Zhang et al. 2016) to text categorization.

A celebrated technique was proposed by Kim (Kim 2014) that applied the CNN technique which was commonly used in computer vision into sentence categorization. The idea of a convolution is to use filters over text to determining important features and predicting a meaningful label.

From the FIGURE 2.17 illustrate CNN architecture for binary text categorization. Red and Yellow box represent filter that performs convolutions over the sentence matrix and generates feature maps and then applies 1-max pooling over each map to form a feature vector for the second-to-last layer. Finally, softmax uses a feature vector to classify the sentence.

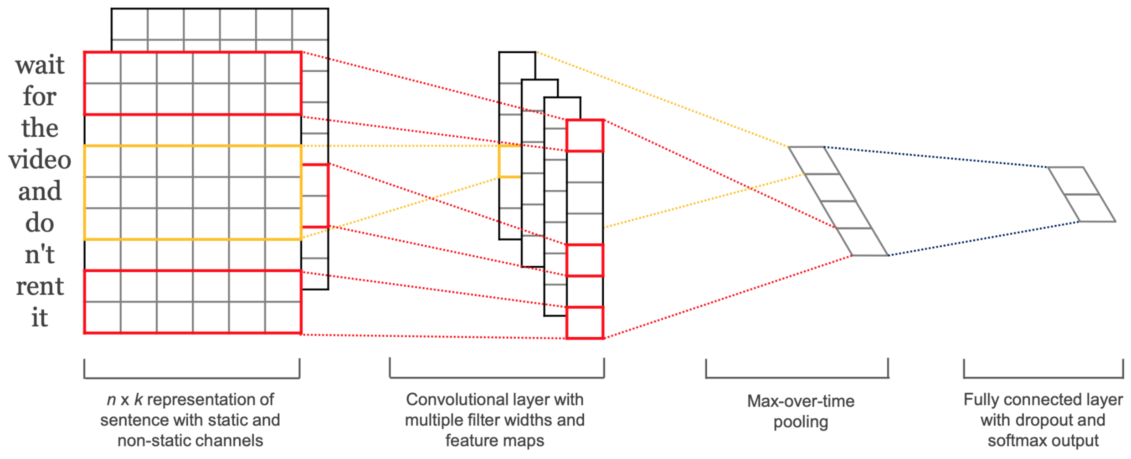
wait
for
the
video
and
do
n't
rent
it
n x k representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

FIGURE 2.17. CNN Model architecture (Kim 2014)

He reported a simple CNN which was tuned with little hyperparameters that outperform than SOTA model and it could obtain impressive results on multiple datasets.

Zhang and Wallace (Zhang and Wallace 2015) reported tuning for the number of feature maps and filter region size are the important factors. Furthermore, They proposed grouped weight sharing for external resources such as Brown cluster, WordNet, etc. into text categorization. They used the two-channel model as input. The first channel was word embedding from external resources and the second channel was weight-sharing embedding among resources. The result showed two-channel performs better than single-channel. Most of them demonstrated that neural network models were powerful for learning features from texts, while they focused on single-label or a few labels problem. Several efforts had been made to multi-labels (Johnson and Zhang 2015). Liu *et al.* explored a family of new CNN models which were tailored for extreme multi-label classification (Liu et al. 2017).

2.6 The main contributions

The problem of the prior DSS work was that those methods of converting documents to vector were an inefficient method i.e. a bag of words that was unable to capture the different features of sense. Also annotated resources are expensive and required hand-labeling. Another problem is the prior work only evaluates DSS against the gold standard i.e. SENSEVAL-3 and it does not cover the NLP application.

This thesis focuses to solve these problems. The main contribution can be summarized:

(1) I propose an unsupervised method for identifying DSS which makes use of distributed representations of words and thus captures large semantic context. An unsupervised method does not require manual annotation of data, while Magnini *et al.* method required a considerable amount of hand-labeling. The method is automated and required only documents from the given domain/category such as the

Reuters corpus, and thesaurus with gloss texts such as WordNet. Therefore, it can be applied easily to a new domain or sense inventory, given sufficient documents.

(2) I propose a semi-supervised method based on a neural random walk model for identifying DSS. A semi-supervised method requires a few labeled data for training, it can improve a number of DSS correctly.

(3) I empirically evaluated my DSS model from an unsupervised learning and that the result of DSS is effective than WSD in the text categorization task, i.e., I examined my hypothesis DSS from a semi-supervised method that can improve the performance of the text categorization.

CHAPTER 3

Domain-Specific Senses Identification based on Word Embedding Learning

As I reviewed in CHAPTER 2, the current DSS issue is that traditional methods do not perform well to transform documents to vectors, therefore they cannot detect an abundance of sense features. Moreover, scarcity and expensive annotated resources are a major obstacle for developing an effective DSS identification system. Most related works base on WSD and they are intrinsic experiments whereas my method base on domain and the experiment is extrinsic based. In this work, I propose an automated method for detecting DSS with unsupervised learning to resolve the issue.

My method is used to assigned labels to each sense. Firstly, I experiment with the number of RCV1 categories at 6 and I focus Part-Of-Speech (POS) only nouns. The Word2vec training model is CBOW model because it trains faster than the skip-gram. To ensure that the method works well, I decide to increase the number of RCV1 categories from 6 to 14. Moreover, I extend Part-Of-Speech (POS) as verbs for a quantitative evaluation. The Skip-gram training model is selected instead of CBOW model because the results outperform than CBOW model according to (Mikolov et al. 2013).

3.1 Domain-Specific Senses Identification

My method includes three stages, (i) Pre-processing, (ii) Calculating sense similarity, and (iii) Ranking sense scores. FIGURE 3.1 demonstrates an overview of the method.

3.1.1 Pre-processing

The goal of text processing is to prepare the data needed to be processed in a task. Typically, the data may be in a format that is not suitable for use or it may have some inappropriate data values. Therefore, data must be prepared before being used in each task which may be a different format. Inappropriate data values are caused by many factors, for example, data entry without carefully(e.g., Age: -5), or missing values, or multi-valued attributes (e.g., Skill: guitar, tennis), etc. If the above-mentioned data is used as input to the task, it may lead to low effective results. Currently, Most of the NLP tasks i.e. text classification, question answering, and Machine Translation, require this process as the first stage of a task.

For my work, RCV1 corpus is in XML document format. Each document is clearly structured including a category as stated in the automated processing of documents (Cristani et al. 2018).

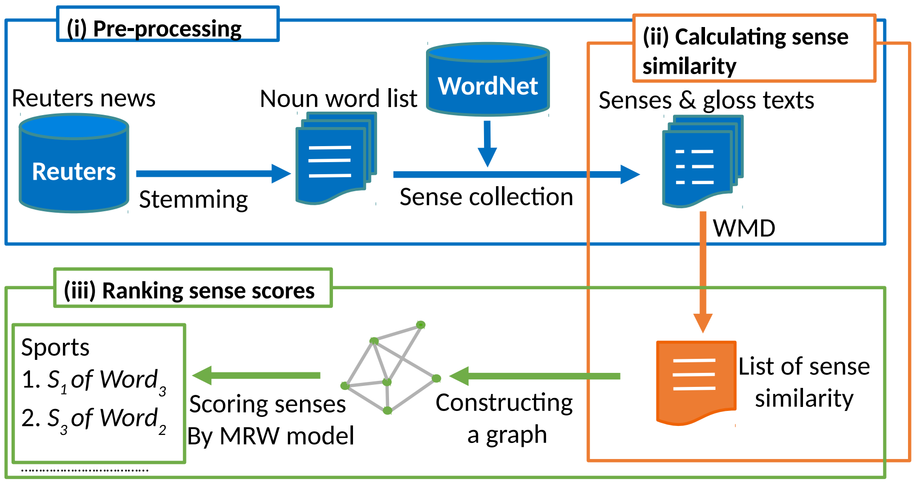

FIGURE 3.1. Overview of the system

I initially process sentences with tokenization tool (i.e. Tree Tagger by Schmid (Schmid 1994), Stanford CoreNLP by Manning (Manning et al. 2014) that use to identify words with POS from a sentence. I also consider using filtering and cleansing methods, such as choosing only letters, converting letters to lowercase, choosing words that have more frequency than minimum values, only selecting words that match my desired POS(Noun, Verb), selecting proper nouns such as name, person (e.g., "Magaret Thatcher"), location (e.g., "New York"), organization (e.g., "Berkshire Hathaway") are identified by NER (Finkel et al. 2005) to make more suitable, select only words that are not in the stopword list ("a", "an", "the"). Appendix A2 is the most common word in a sentence and it can affect performance in applications.

After I obtained the related words from the above processing, I used these words as keywords to retrieve all gloss texts of words from WordNet thesaurus. WordNet is a lexical database invented by Miller (Miller 1995) which consists of the lexical categories nouns, verbs, adjectives, and adverbs. It currently has been applied in more than 200 different languages and frequently used by Linguists, Psychologists, Artificial Intelligence developers, NLP developers. I used WordNet version 3.0 which includes 117,798 nouns, 11,529 verbs, 22,479 adjectives, and 4,481 adverbs. All kinds of lexical categories are assembled into groups of synsets (synonym sets) that represent the same concept. Each synset includes a gloss(short definition) with an example sentence or more than one sentence expressing the usage of synset members, i.e., the fourth sense of word "foot" has synsets follow as "foundation", "base", "groundwork", etc. Generally, WordNet is used to present word senses. Each word has many different meanings.

3.1.2 Calculating sense similarity

Markov Random Walk (MRW) model is used to identify domain-specific senses for each category. The input of the MRW model is a graph consisting of vertices and edges with a similarity value between vertices. As shown in (ii) in FIGURE 3.1, I calculated sense similarity by using WMD (Kusner et al. 2015). WMD measures the dissimilarity between two sentences as the minimum amount of distance that the embedded words of one sentence need to travel to reach the embedded words of another sentence. The word embedding is learned by using Word2Vec (Mikolov et al. 2013). More precisely, Word2Vec learns vector representation of words from gloss text as the training documents. It is provided two models, CBOW and skip-gram.

I used the CBOW model as it is reported to be faster in the preliminary experiment. The CBOW model is based on the feedforward Neural Network Language Model(NNLM). The model predicts the center word given a representation of the surrounding words. Whereas, the Skip-gram model is utilized in the quantitative experiment because it performs better than CBOW (Mikolov et al. 2013). The skip-gram model's objective is to maximize the likelihood of the prediction of contextual words from given the center word.

Let $\mathbf{X} \in \mathbb{R}^{d \times n}$ be a Word2Vec embedding matrix for vocabulary size of n words. The i^{th} column, $\mathbf{x_i} \in \mathbb{R}^d$ refers to the embedding of the i^{th} word in d-dimensional space. I represent gloss text of each sense as normalized bag-of-words (nBOW) vector, $g \in \mathbb{R}^n$. The objective of the model is to minimize cumulative cost C of moving the gloss text g to g':

$$\begin{aligned} L &= \sum_{i,j=1}^{n} \mathbf{T}_{i,j}\, c(i,j), \\ &\text{subject to:} \sum_{j=1}^{n} \mathbf{T}_{ij} = g_i,\ \forall i \in \{1, \cdots, n\}, \\ &\qquad\qquad \sum_{i=1}^{n} \mathbf{T}_{ij} = g'_j.\ \forall j \in \{1, \cdots, n\}. \end{aligned} \tag{3.1}$$

$\sum_{j=1}^{n} \mathbf{T}_{ij} = g_i$ indicates that outgoing flow from word i equals g_i. Similarly, $\sum_{i=1}^{n} \mathbf{T}_{ij} = g'_j$ shows that incoming flow to word j mush match g'_j. $c(i,j)$ in Eq. (3.1) refers to word travel cost which is defined by $c(i,j) = \|\ \mathbf{x}_i - \mathbf{x}_j\ \|_2$ (Hitchcock 1941).

3.1.3 Ranking sense score

The final procedure for detecting Domain-Specific Senses which are shown in (3) in FIGURE 3.1 is to score each sense for each domain/category. The MRW model I used decides the importance of a vertex within a graph based on global information drawn recursively from the entire graph (Bremaud 1999). The essential idea is that of "voting" between the vertices. An edge between two vertices is considered a vote cast from one vertex to the other. The score associated with a vertex is determined by the votes that are cast for it, and the score of the vertices casting these votes.

Given a set of senses S_d in the domain d, as shown in (iii) of FIGURE 3.1, I construct a graph. G_d = (V, E) is a graph reflecting the relationships between senses in the set. V is the set of vertices, and each vertex v_i in V is the gloss text assigned from WordNet. E is a set of edges, which is a subset of $V \times V$. Each edge e_{ij} in E is associated with an affinity weight $f(i \to j)$

between senses v_i and v_j $(i \neq j)$. The weight is computed using the standard cosine measure between the two senses. Two vertices are connected if their affinity weight is larger than 0 and I let $f(i \to i)$= 0 to avoid self transition. The transition probability from v_i to v_j is then defined by

$$p(i \to j) \;=\; \begin{cases} \frac{f(i \to j)}{\sum_{k=1}^{|V|} f(i \to k)}, & if\ \Sigma f \neq 0 \\ 0, & otherwise. \end{cases} \tag{3.2}$$

I used the row-normalized matrix U_{ij} = $(U_{ij})_{|V| \times |V|}$ to describe G with each entry corresponding to the transition probability, where U_{ij} = $p(i \to j)$. To make U a stochastic matrix, the rows with all zero elements are replaced by a smoothing vector with all elements set to $\frac{1}{|V|}$. The matrix form of the saliency score $Score(v_i)$ can be formulated in a recursive form as in the MRW model,

$$\vec{\lambda} \;=\; \mu U^T \vec{\lambda} + \frac{(1-\mu)}{\mid V \mid} \vec{e}, \tag{3.3}$$

where $\vec{\lambda}$ = $[Score(v_i)]_{|V| \times 1}$ is the vector of saliency scores for the senses. $\vec{e}$ is a column vector with all elements equal to 1. μ is the damping factor. I set μ to 0.85, as in the PageRank (Brin and Page 1998). The final transition matrix is given by

$$M = \mu U^T + \frac{(1-\mu)}{\mid V \mid} \vec{e}\vec{e}^T. \tag{3.4}$$

Each score of the sense in a specific domain is obtained by the principal eigenvector of the matrix.

I applied the algorithm for each domain. I note that the matrix M is a high-dimensional space. Therefore, I used a ScaLAPACK, a library of high-performance linear algebra routines for distributed memory MIMD parallel computing (Netlib 2007), which includes routines for solving systems of linear equations, least squares, eigenvalue problems.

I selected the topmost $K\%$ words (senses) according to rank score for each domain and make a sense-domain list. For each word w in a document, find the sense s that has the highest score within the list. If a domain with the highest score of the sense s and a domain in a document appeared in the word w match, s is regarded as a domain-specific sense of the word w.

3.2 Experiments

The experiments have divided for two sections following as:

(1) Preliminary experiment: the goal of this section is to explore a preliminary on the proposed method whether it is effective or not.

(2) Quantitative experiment: for this section, I would like to test the proposed method quantitatively cover more a number of RCV1, and POS.

3.2.1 Preliminary experiment

Corpus and sense inventory
I used Reuters'96 corpus from 20th Aug. 1996 to 19th Aug. 1997, and WordNet 3.1. The corpus consists of 806,791 documents organized into 126 categories. There are no existing sense-tagged data for domains that could be used for evaluation. I thus used the Subject Field Codes (SFC) resource (Magnini and Cavaglia 2000), which annotates WordNet 2.0 synsets with domain labels. The SFC consists of 115,424 words assigning 168 domain labels with a hierarchy. It contains some Reuters categories. I tested Reuters six categories corresponding to the SFC labels which are shown in TABLE 3.1. "The number of doc" in TABLE 3.1 shows the number of documents in each category. I set parameters used in the Word2Vec, *i.e.* the number of dimensions is 100, the window size is 5. I used the CBOW learning model. For each category, I built individual models and I chose words whose frequencies in all of the documents are more than five.

TABLE 3.1. The Reuters and SFC category correspondences

SFC	Reuters	The # of doc
Tourism	Travel	680
Sports	Sports	35,225
Military	War	32,580
Law	Legal/Judicial	32,194
Economy	Economics	117,501
Politics	Politics	56,834

Evaluation measure
I have made a comparison with cosine similarity as a baseline. I also investigate my Inverse Rank Score (IRS) which is a measure of system performance by considering the rank of correct senses within the candidate collections. It is the sum of the inverse rank of each matching collections, and the higher the IRS value, the better the system performance.

Results
TABLE 3.2 and 3.3 show the results obtained by the topmost 20% senses according to rank score. I also tested a baseline system that does not use WMD as a similarity measure but instead uses cosine measure and compared it with my system. "S" shows the total number of senses which should be assigned to each category. "DSS(Domain-Specific Senses)" refers to the results obtained by my system. "SFC" indicates the number of senses appearing in the SFC resource. "Cor" denotes the number of senses appearing in both of the systems (WMD/Cos) and SFC. "Prec" means the ratio of correct assignments by my system divided by the total number of the system's assignments. "Rec" is the ratio of correct assignments by the system divided by the total number of correct assignments. The F measure which combines recall(r) and precision(p) with an equal weight is F(r,p) = $\frac{2rp}{r+p}$. "IRS" refers to Inverse Rank Score. "P_IRS" indicates the perfect correct value of IRS.

TABLE 3.2. The results of sense assignments from WMD method (The top 20% words according to rank score)

SFC/Reuters	S	DSS	SFC	DSS(WMD)					P_IRS
				Cor	Prec	Rec	F	IRS	
Tourism/Travel	78	16	18	15	.938	.833	**.882**	3.24	3.38
Sports/Sports	297	59	63	50	.847	.794	.820	4.40	4.66
Military/War	548	110	115	85	.773	.739	.756	4.95	5.28
Law/Law	740	148	153	101	.682	.663	.671	4.06	5.58
Economy/Economics	577	115	120	77	.670	.642	.655	4.63	5.33
Politics/Politics	815	163	174	107	.656	.615	.635	4.92	5.67
Average	509	102	109	72	.761	.714	**.737**	**4.37**	**4.98**

TABLE 3.3. The results of sense assignments from Cosine similarity method (The top 20% words according to rank score)

SFC/Reuters	S	DSS(Cos)					P_IRS
		Cor	Prec	Rec	F	IRS	
Tourism/Travel	78	16	1	.941	**.970**	3.38	3.38
Sports/Sports	297	45	.763	.763	.763	3.42	4.66
Military/War	548	74	.673	.632	.652	4.58	5.28
Law/Law	740	40	.270	.268	.269	1.21	5.58
Economy/Economics	577	65	.565	.551	.558	3.74	5.33
Politics/Politics	815	20	.123	.113	.118	0.89	5.67
Average	509	43	.566	.545	**.555**	**2.87**	**4.98**

Discussions
From TABLE 3.2 and TABLE 3.3 that the overall performance obtained by WMD was better than those by Cos except for "Tourism/Travel" domain as the average F attained at 0.737, while that of Cos was 0.555. Performance depends on the categories. In both methods, I obtained the best F score when I used the category "Tourism/Travel". In contrast, the worst results obtained by WMD were categories "Economy/Economics" and "Politics/Politics", while these results were better than those of Cos. This is not surprising because they are semantically close with each other.

I also examined how the topmost ratio of ranking affects the overall performance of the system. 3.2 shows F-score against the ratio of topmost ranking. When the ratio increased, the F-score obtained by both methods dropped. However, the results obtained by WMD were still better to those obtained by Cosine similarity. This shows that WMD works well to detect Domain-Specific Senses.

3.2.2 Quantitative experiment

Corpus and sense inventory
The Reuters corpus consists of 806,791 documents organized into 126 categories. There

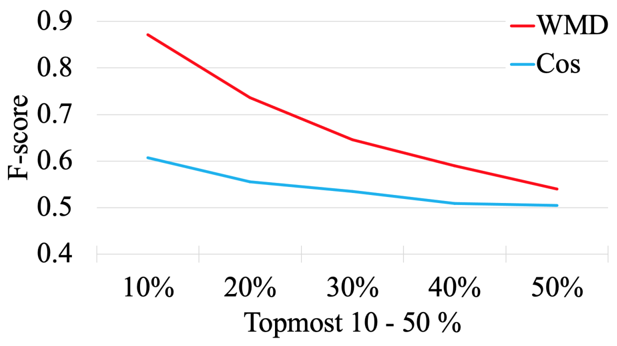

FIGURE 3.2. F-score against the ratio of topmost ranking

are no existing sense-tagged data for domains that I can utilize for evaluation. Therefore, I used the Subject Field Codes (SFC) resource which semi-automatically annotates WordNet 2.0 synsets with domain labels (Magnini and Cavaglia 2000). The SFC consists of 115,424 words assigning 168 domain labels which include some of the Reuter's categories. I manually assigned Reuter's categories to SFC labels which are shown in TABLE 3.4. "# doc" in TABLE 3.4 refers to the number of documents in each category. "# min" and "# max" indicate the minimum and the maximum number of word frequencies that appear in each category respectively.

TABLE 3.4. The Reuters and SFC category correspondences

SFC	Reuters	# doc	# min	# max
Law	Legal	11,944	140	18,781
Finance	Funding	41,829	151	63,973
Industry	Production	25,403	197	21,957
Publishing	Advertising	2,084	27	5,821
Admin.	Management	11,354	56	18,587
Economy	Economics	117,539	1,139	99,197
Art	Arts	3,801	67	4,562
Fashion	Fashion	313	7	775
Politics	Politics	56,878	967	94,970
Religion	Religion	2,849	54	3,746
Sports	Sports	35,317	222	26,898
Tourism	Travel	680	8	1,423
Military	War	32,615	546	45,085
Meteorology	Weather	3,878	17	4,602

In a pre-processing step, a POS tagger and lemmatizer from Stanford CoreNLP (Manning et al. 2014) were applied to the Reuters corpus and then I divided it into three for the experiments including training dataset for 3 months, test dataset for 6 months, and validation dataset for 3 months. The test dataset was fed in DSS. The initial step was to choose the first 20,000 words with the highest frequency from each category. Every word was converted to lowercase, removed punctuation, and stopwords, and applied lemmatization. Next, I selected only nouns and verbs then used them as a query into Wordnet to find out the word senses and their glosses for built the Word2vec model with their glosses. The 14 different models were created according to category.

The Word2vec parameters consisted of the number of dimensions to 100, the window size to 5, Skip-gram as the training algorithm, and Hierarchical softmax as the model training. I used the word embedding from the Word2vec model to measure the sense similarity score for WMD which was a method for comparing dissimilarity scores between sentences and then I rated the score with MRW which was used to determine the importance of senses. Eventually, the predominant senses per each category are obtained. The training data was used to estimate $K\%$ words (senses) according to rank score, and test data was used to test the method using the estimated value of K. I manually evaluated a sense-domain list. As a result, I set K to 10%.

Results

TABLE 3.5. The results of sense assignments

Category	Sense	DSS	SFC	Correct	F-Score	IRS	P_IRS
Law	577	57	57	41	0.719	3.607	4.628
Finance	53	5	5	5	**1.000**	2.283	2.283
Industry	195	19	19	18	0.947	3.489	3.548
Publishing	178	17	17	15	0.882	3.262	3.439
Admin.	302	30	30	16	**0.533**	3.192	3.994
Economy	531	53	53	34	0.642	3.981	4.555
Art	236	23	23	14	0.609	3.634	3.734
Fashion	289	28	28	23	0.821	3.703	3.927
Politics	522	52	52	24	**0.462**	3.348	4.536
Religion	501	50	50	38	0.760	3.129	4.497
Sports	306	30	30	19	0.633	3.382	3.994
Tourism	176	17	17	14	0.824	3.071	3.439
Military	528	52	52	37	0.712	4.037	4.536
Meteorology	94	9	9	8	0.889	2.718	2.829
Average	321	31.571	31.571	21.86	**0.745**	3.345	3.853

TABLE 3.5 shows the results obtained by the topmost 10% senses according to rank score. “Sense” indicates the total number of senses which should be assigned to each category. “DSS” and “SFC” refer to the number of senses obtained by my method and appeared in the SFC resource, respectively. “Correct” shows the number of senses tagged with the best domain ranking appearing in both of my method and SFC. “F-score” indicates an F-measure of one domain tagging. “IRS” refers to Inverse Rank Score and the higher the IRS value, the better the system performance. “IRS” refers to IRS of one domain tagging. “P_IRS” indicates the perfect correct value of IRS. From TABLE 3.5 that the overall performance depends on the categories. The best performance of one domain is “Finance”. In contrast, the results of “Politics” and “Administrator(Admin.)” are 0.462$\sim$0.533.

TABLE 3.6 illustrates some examples obtained by my method but that does not appear in the SFC. TABLE 3.6 gives an example for each domain. For example, the military sense of the word "Redoubt" and the act of meting out the justice of "Administration" are correctly obtained by my method but does not occur in the SFC resource. This clearly supports the usefulness of my automated method.

TABLE 3.6. Sense example identified by my method

Category	**Word**	**Sense**
Law	Administration	The act of meting out justice according to the law.
Economy	Spending	Money paid out; an amount spent.
Politics	Labour party	A political party formed in Great Britain in 1900; characterized by the promotion of labor's interests and formerly the socialization of key industries.
Sports	Jerk	Raising a weight from shoulder height to above the head by straightening the arms.
Military	Redoubt	(Military) A temporary or supplementary fortification; typically square or polygonal without flanking defenses.

Qualitative Analysis of Errors

I perform this stage to reflect the error of my results obtained from TABLE 3.5 and use it to improve the further method. I found three main types of error.

1. The Semantic similarity measure with WMD
   FIGURE 3.3 presents the F-score against the percent of the topmost ranking of predominant senses. This graph shows that WMD performs well approximately topmost 10%$\sim$20% of senses. However, when topmost senses increase more than 30%, the F-score declines below 0.5. From the observation, I need to investigate the semantic similarity measures for improving further performance.

2. The number of domains per word
   The larger the number of domains per word, the harder it is to identify the correct domain. Consider, the word "Apprehend" and "Arrest" both words have the same sense, that is, "(take into custody), the police nabbed the suspected criminals". The word "Apprehend" is found in only two domains while the word "Arrest" is used more often for 14 domains. From this example, Identifying a correct domain for the word "Arrest" is difficult than "Apprehend". To solve this problem, I need to extend my model to assign more than one domain which is similar to the multi-label text classification task.

3. The closeness sense of the domains
   The worst results are "Politics" and "Administration" because they are semantically close with each other, that is, the word "section", I found 14 domains in all and it is tagged as "Politics" with the best-ranking order, 136, while the second-ranking as "Administration" was 143 and it was correctly assigned. As a result, the evaluation of the word "section" is incorrect, even though both the order of ranking are closer

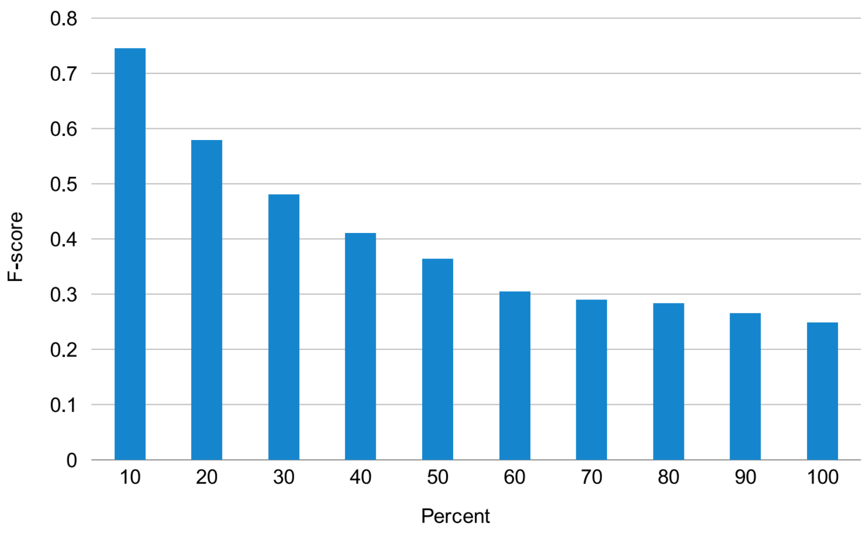

FIGURE 3.3. F-score against the percent of topmost of senses

with each other compared to other domains. From this observation, I should identify a sense of both domains correctly for further improvement.

3.3 Conclusion

In this work, I present an unsupervised method for detecting the DSS for the problem that how to choose an appropriate sense of a context, based on word embedding learning which leverages distributed representations of words and thus does not require manual annotation of sense-tagged data. I used the WMD to calculate the similarity between senses by using gloss text which is a short length of sentence. The computational cost of WMD is $O(p^3 logp)$, where p denotes the number of distinct words in the documents (Pele and Werman 2009) and that of PageRank is $O(N^2 \times k)$ where N indicates the matrix dimension and k refers to the number of repetition.

The results using Reuters 1996 corpus and WordNet 3.1 showed that embedding learning is effective for detecting Domain-Specific Senses. I have performed both the preliminary experiments and quantitative experiments. I obtained the Macro F-score at the topmost 20% is 0.737 that exceeds the Cosine method 0.182 in the preliminary experiments. Whereas in the quantitative experiments, I obtained the Macro F-score at the topmost 10% is 0.745. However, if a fair comparison is performed at the topmost 20%, the F-score of the first experiment (0.737) is better than the second experiment (0.685) because of the first experiment, the number of categories, and the number of POS are less than the second experiment. Therefore, it is easier to identify the correct sense.

There are several directions for future work. I am going to compare with other statistical methods as a baseline and compare it with sophistical methods. I am going to apply my DSS method on other datasets or different languages other than English and examine the performance to demonstrate the robustness of my method. I focused on the gloss texts of

WordNet in this work. I utilize another thesaurus, for example, Roget's by using corpus statistics. This is a rich space for further exploration. I should extend the method by using other part-of-speech other than nouns and verbs. I also need to investigate methods to improve run-time efficiencies. I should determine the same percent of the topmost in the experiment to make a fair comparison.

CHAPTER 4

Detecting Domain-Specific Senses with a Neural Random Walk Model

Although the approach I proposed in CHAPTER 3 can resolve the inefficient traditional methods and inadequate annotated resources, the method is unsupervised manner that still have room to improve the overall performance. In this Chapter, I address the problem and propose an automated method to detect DSS based on the deep learning technique.

4.1 Predict Sense Labels with a Neural Model

Nowadays, increasing numbers of data are too numerous and complicated. To deal with this issue, one approach is using graphs. The graph includes nodes and edges. Each node has node features (the data of nodes) and an edge is used to link the relationship between nodes (the structure of the graph). A graph can be used to describe complicated relationships between data. For instance, the relationships between users and products that they bought in a recommendation system. In the transportation network, warehouses are linked to each other with distance relationships. However, Extracting useful information from the graph is not trivial. Feature engineering techniques i.e. node degree, PageRank score is utilized to obtain node features. To achieve better performance, node features and the structures should be used as input into neural network learning to detect what information is useful and this is called graph representation learning. One attention technique is Graph Convolutional Network (GCN) that can solve the problem of classifying nodes in a graph. The main idea of GCN is to obtain the feature information from neighbors of each node along with itself. All feature information is computed by aggregate function such as average.

The Convolutional is derived from image processing and then applied to Graphs. However, each image has a fixed structure, Graphs are much more complicated.

The FIGURE 4.1 illustrates a simple sense network for binary classification. Each node represents a gloss text (node features), while the edge is a co-occurrence relationship between two senses in any documents. Consider the white node obtain the neighbors' features from two categories (red node denotes politics and blue nodes denote sports) along with itself, then take aggregate function. The result is used as input through a neural network to return a resulting vector. Finally, the softmax function is used to calculate the probability of the node label. GCN generally has two layers that are shown in FIGURE 4.2. The result of the first layer is the input of the second layer and the neural network is a fully connected layer. For my experiment, I have classified nodes with 14 categories within graphs. The number of layers denotes the longest distance that node features can propagate. In the case of 1 layer, each node

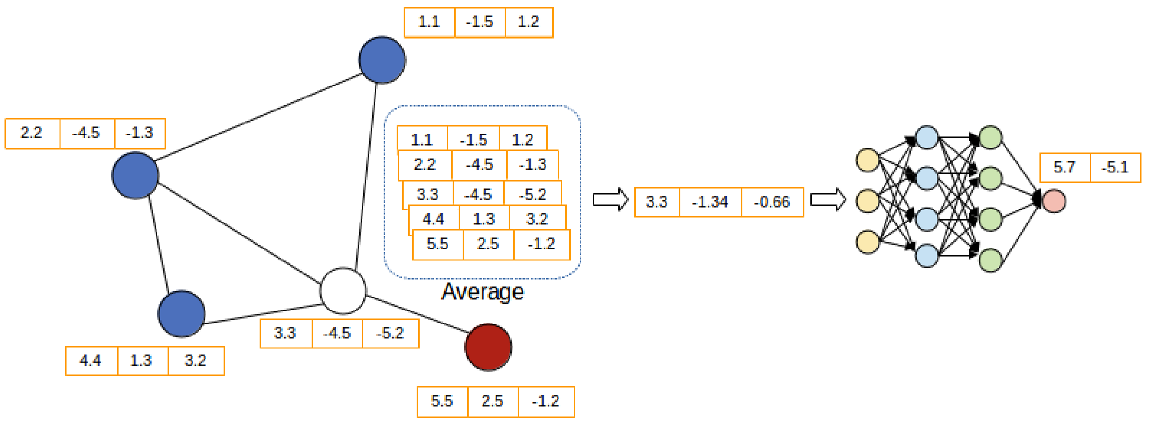

FIGURE 4.1. The idea of GCN

gathers the information from the adjacent node, in which this process occurs independently and simultaneously. For 2 GCN layers, the information gathering takes place one more time. The number of layers indicates the maximum number of hops that each node can propagate. From Kipf (Kipf and Welling 2016) experimental results demonstrate the best accuracy is obtained with a 2 or 3 layer model. Moreover, the number of layers is stacked more than 7, it tends to obtain poor performances.

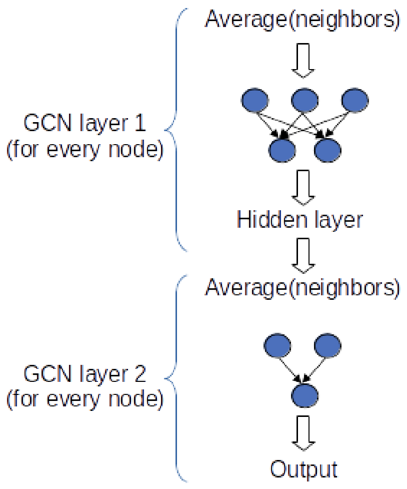

FIGURE 4.2. GCN 2 layers

However, the propagation of neural networks with message passing loses its focus on the local neighborhood (Li et al. 2018) when many layers are applied. Klicpera et al. (Klicpera et al. 2018) addressed the problem and proposed the PPR as a propagation scheme instead to solve the problem.

From 4.3 illustrate a benefit of PPNP model. The prediction (h_i) decouple from propagation that the neural network of a node can be independently designed i.e. structure (MLP, CNN, RNN), and neural network depth from propagation. Whereas Message Passing Neural

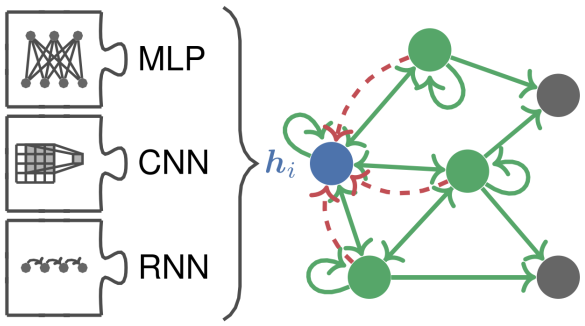

FIGURE 4.3. Neural network prediction (Klicpera et al. 2018)

Network (MPNN), neural network depth (prediction) associates with neighborhood size (propagation).

4.2 Framework of the System

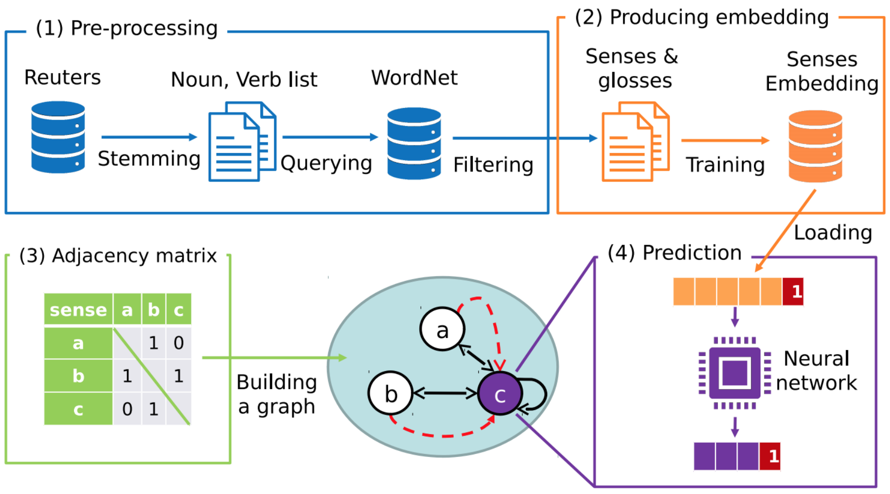

FIGURE 4.4. System framework

The objective of my work is to identify the predominant sense for each domain by learning the features of each sense through the relationship of senses in the graph structure. The method consists of four steps, (1) Pre-processing, (2) Producing embedding, (3) Building a graph with an adjacency matrix, and (4) Predicting categories and propagation. FIGURE 4.4 illustrates an overview of my framework.

4.2.1 Pre-processing

The objective of the pre-processing is to extract senses and glosses in the WordNet from given categories. I initially gather the documents from RCV1. Each word is annotated for Part Of

Speech (POS) and is lemmatized using Stanford CoreNLP (Manning et al. 2014). Noun and verb words are chosen and used to find their senses and gloss texts from the WordNet. For each category, noun and verb words are extracted.

4.2.2 Producing Embedding

The sense embedding is learned by using bert-as-service (Xiao 2018). BERT is a type of neural network model for pre-training language embeddings developed by Devlin et al. (Devlin et al. 2018). I apply BERT to learns the feature representation of gloss texts $S_i = \{w_1, \cdots, w_m\}$ where each $w_i, (1 \leq i \leq m)$ denotes a word in gloss texts to build sense embeddings as an input for the prediction stage. In this work, I use The pre-trained BERT model as BERT-Base, Uncased (L=12, H=768, A=12) which represents a model consisting of 12 layers, 768 hidden units, and 12 attention heads. BERT's input formatting has two important special tokens consisting of [SEP] that is used to indicate the end of a sentence or used as a separator between two sentences. [CLS] is used to indicate the beginning of a sentence. Both tokens are inserted into a sentence after that I sum token embedding, the segment embedding, and the position embedding to build input embeddings. I use input embeddings which have four dimensions including the number of layers, the number of batches, the number of tokens, and the number of features for creating sense embeddings. I select the pooling strategy as the average pooling on the second-to-last layer to obtained sense embeddings.

4.2.3 Building a graph with an adjacency matrix

I begin to create a co-occurrence matrix between senses, each of which meets three criteria: firstly, target senses and their POS are found in RCV1 documents. Secondly, they have the same category. Thirdly, each document contains more than one sense. By utilizing sense relationships, I create an adjacency matrix by converting the non-zero value in a co-occurrence matrix equal to one.

4.2.4 Predicting categories and propagation

The final step for determining Domain-Specific Senses which are illustrated in FIGURE 4.4 is to predict each sense on a graph using a neural network model. I utilize the APPNP model because it is based on the GCN model (Kipf and Welling 2016) that is a very powerful neural network even 2 layers of GCN can generate useful feature representation of nodes on graphs and it also solves the lost focus issue with PPR. I use sense embedding which is the result from the second step as an input and then training with the APPNP model that predicts a proper category for each sense.

I build a graph. $G_d = (V, E)$ is a graph that represents the relationships between senses in all domains by adjacency matrix $A \in \mathbb{R}^{n \times n}$ where n is a node set. $\tilde{A} = A + I_n$ represents the adjacency matrix with added self-loops. V is the vertices set consisting vertex v_i that is a sense gloss texts. E is a edge set. Each edge e_{ij} denotes co-occurrence relationship of v_i and v_j in documents. The sense gloss texts v_n are represented by the feature matrix $X \in \mathbb{R}^{n \times f}$ where f denotes the number of features, and the category matrix $Y \in \mathbb{R}^{n \times c}$ where c denotes the number of categories. GCN model with two layers is defined by

$$Z_{GCN} = \text{softmax}\Big(\hat{\tilde{A}}\ \text{ReLu}\ \big(\hat{\tilde{A}}XW^{(0)}\big)\ W^{(1)}\Big), \tag{4.1}$$

where $Z_{GCN} \in \mathbb{R}^{n \times c}$ is the prediction of each node, $\hat{\tilde{A}} = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$ is a normalized adjacency matrix with self-loops, $\tilde{D}_{ij} = \Sigma_k \tilde{A}_{ik}\delta_{ij}$ is the diagonal matrix, and $W^{(0)}$ and $W^{(1)}$ are weight matrix.

The propagation in this graph model derived from PageRank (Brin and Page 1998) that is defined by $\pi_{pr} = A_{rw}\pi_{pr}$ with $A_{rw} = AD^{-1}$ where D denotes the diagonal matrix. The Personalized PageRank adapts from PageRank for recognizing the connection between nodes and is defined by $\pi_{ppr}(i_x) = \alpha\big(I_n - (1-\alpha)\hat{\tilde{A}}\big)^{-1} i_x$ where x is root node and i_x denotes teleport vector with teleport probability $\alpha \in [0, 1]$.

APPNP applies the above ideas and produces the first prediction for each node and then propagates it with PPR to produce the final prediction. It is defined by

$$\begin{aligned} Z^{(0)} &= H, \quad H = f_\theta(X), \\ Z^{(k+1)} &= (1-\alpha)\hat{\tilde{A}}Z^{(k)} + \alpha H, \\ Z^{(K)} &= \text{softmax}\Big((1-\alpha)\hat{\tilde{A}}Z^{(K-1)} + \alpha H\Big), \end{aligned} \tag{4.2}$$

where $Z \in \mathbb{R}^{n \times c}$ is the prediction of each node. H is the prediction for each node and represents both the starting vector and teleport set, f_θ is a neural network with the number of parameters θ. K denotes the number of power iteration steps for approximate topic-sensitive PageRank and $k \in [0, K-2]$.

4.3 Experiments

4.3.1 Corpus and sense inventory

I evaluate my method using 6-months RCV1 corpus and WordNet 3.0 with SFC resource (Magnini and Cavaglia 2000) that is a gold standard domain of word senses. I choose 14 categories of documents out of 126 categories that appear in the SFC. I obtain nouns and verbs list appearing 14 categories of SFC and their corresponding sense from the WordNet.

TABLE 4.1 shows data statistics. “#doc" refers to the number of documents and “#sense" shows the number of senses. The total number of senses is 6,082 senses. Of these, I remove senses which do not have any relationship with other senses. As a result, I use 4,567 senses and create a graph. In the prediction step, the total senses in a graph are divided into a visible and a test set. The 3,000 nodes are sampling for the visible set and then I split a training set size per category vary from 5 to 35. In the second part of the visible set, I split 500 nodes for an early stopping set and the rest nodes are the number of the validation set. A test set of 1,567 nodes were sampled. For the parameter settings in my model, I set the number of hidden units as 64 and the dropout rate as 0.5. I also set patience to 300 for early stopping,

TABLE 4.1. The Subject Field Codes (SFC) and Reuters category correspondences.

SFC/Reuters	# doc	# sense
Admin/Management	5,830	401
Art/Arts	1,906	317
Economy/Economics	59,888	741
Fashion/Fashion	194	420
Finance/Funding	20,760	75
Industry/Production	12,156	394
Law/Legal	6,607	785
Meteorology/Weather	2,164	108
Military/War	15,864	696
Politics/Politics	28,668	624
Publishing/Advertising	1,230	259
Religion/Religion	1,478	627
Sports/Sports	18,410	371
Tourism/Travel	291	264

the maximum epochs for training as 10,000. I select Adam optimizer with a learning rate of 0.01 and the teleport probability (α) as 0.2.

4.3.2 Performance comparison

For the performance comparison of DSS, I compare an unsupervised approach (WMD-DSS) and a semi-supervised approach (APPNP-DSS). For the acquisition of WMD-DSS, I firstly pre-process for RCV1 corpus including POS tagging and lemmatization. I select the only noun and verb words to fetch senses and glosses from WordNet and then create word embedding with Word2Vec by applying the skip-gram model as a training algorithm and using hierarchical softmax as model training and dimension of word vector of 100 and window size of 5. I measure the similarity of senses with Word Mover's Distance (WMD) algorithm (Kusner et al. 2015) that requires word embedding for computation. If both senses are similar, the WMD value is small whereas senses are different, the WMD value is larger. Finally, the similarity results are applied with a simple Markov Random Walk (MRW) to ranking senses within a graph order by the most predominant senses with the high WMD value.

4.3.3 Results

TABLE 4.2 shows the test dataset results obtained by my method when using training size equals to 35. “WMD-DSS F-score" denotes the results obtained by the topmost 20 % senses from an unsupervised approach that computed sense similarity with that relies on Word2Vec and ranked with PageRank.

It demonstrates the overall performance obtained by my model attains at the macro average F-score at 0.647. The best F-score is “Religion" while the worst result is “Finance". One of the reasons that “Finance" is less effective than other categories because the number of neighbor

TABLE 4.2. The results of sense assignments

SFC	WMD-DSS F-score	APPNP-DSS F-score
Admin.	0.367	0.486(+.119)
Art	0.489	0.573(+.084)
Economy	0.500	0.535(+.035)
Fashion	0.719	0.783(+.064)
Finance	0.600	**0.268(-.332)**
Industry	0.744	0.699(-.045)
Law	0.470	0.687(+.217)
Meteorology	0.889	0.741(-.148)
Military	0.581	0.760(+.179)
Politics	0.356	0.566(+.210)
Publishing	0.543	0.662(+.119)
Religion	0.630	**0.813(+.183)**
Sports	0.508	0.797(+.227)
Tourism	0.714	0.661(-.053)
Macro F-score	0.579	**0.647(+0.068)**

nodes of the “Finance" category is lower than other categories. Another reason is the number of senses in the “Finance" category is too small compared to other categories. The number of training sets is the same size as the other categories at 35 nodes. However, the number of validation sets of the “Finance" category is only 2 nodes. Consequently, it affects the tuning hyperparameter of the model in this category which deteriorates the overall performance. This could be solved by collecting senses from subcategories related to “Finance" category in order to increasing the number of nodes in the experiment.

I also have an experiment to examine how the number of training labeled data affect the overall performance. The results are illustrated in FIGURE 4.5. The number of training labeled data of 35 gain the best F-score at 0.647 whereas the smallest of training labeled data of 5 attained at 0.463 F-score. I note that when the number of training data increases, the F-score increases accordingly. For acquiring DSS with WMD, it is an unsupervised method, thus I can obtain the predominant senses per category with only the topmost 20 %. At the topmost 30 %, I obtain the F-score of 0.480, and the F-score drops gradually until the topmost 100 % which obtains a 0.248 F-score.

4.4 Conclusion

I proposed a semi-supervised method for acquiring the DSS based on BERT embedding and deep learning techniques. I also compare the results of my method with the unsupervised method which works well at the topmost 20%. My method can reach an F-score of 0.647 for 1,567 senses, whereas the unsupervised method at the topmost 20% obtains an F-score of 0.579 for 892 senses.

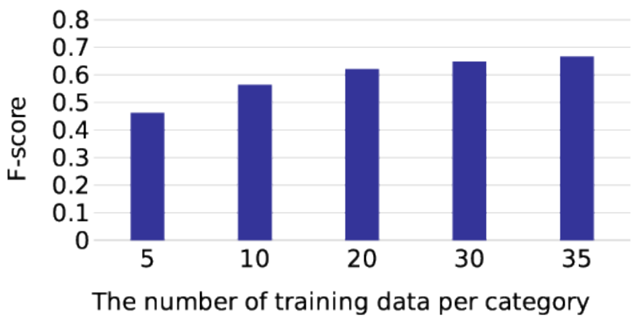

FIGURE 4.5. The number of training labeled data against F-score

For future work, there are several approaches that can be further study. I am going to apply my method to other part-of-speech e.g. adjectives and adverbs as well as other datasets and thesaurus for quantitative evaluation of my method. Comparison to the state-of-art WSD technique (Bevilacqua and Navigli 2020) by using the same datasets, SemEval Check whether this dataset is correct or not is also necessary to examine the effectiveness of the method.

CHAPTER 5

Extrinsic Evaluation through Text Categorization

This chapter demonstrates the influence of DSS on text categorization. The comparison consists of two parts: 1) Evaluation between unsupervised learning and WSD 2) Evaluation between semi-supervised learning and unsupervised learning.

5.1 Text Categorization with CNN

The process of annotating labels or categories to text according to its content is called "Text categorization". Recently, many applications leverage on text categorization following as:

(1) Spam mail detection: the email has become an indispensable part of a modern office, therefore spammer leverages this fact to generate emails to reach a large number of users. This issue is increasing day by day. This require text categorization because it can prevent users from spam. Many methods work well, however given DSS, can improve categorization efficiently since classifier is semantic oriented.
(2) Article paper categorization: the number of articles currently being submitted to editors increasingly. This increases the workload for editors to categorize articles and match articles more appropriate for the reviewer. Text categorization is essential to solving the problem that it can discriminate articles faster. When DSS is combined, it makes the article title more clear.
(3) Sentiment analysis: customers can express their opinions and feelings about products or services more freely than before. Therefore, it is crucial to monitor and understand their opinions and then respond to them faster to make the satisfaction of the customer as much as possible. This also needs DSS, because the sentiment words are highly ambiguous; if DSS is applied with text categorization, it improves a better understanding of opinions.

Many authors have attempted to apply deep learning techniques including LSTM technique (Ghosh et al. 2016; Huang et al. 2015; Kågebäck and Salomonsson 2016; Yao and Huang 2016, CNN Wang et al. 2018), the attention based CNN (Yang et al. 2016), bag-of-words based CNN (Johnson and Zhang 2015), Simple Graph Convolution (SGC) (Wu et al. 2019), the combination of CNN, Gated Recurrent Units (GRU) and attention mechanism (Abreu et al. 2019), and the combination of CNN and recurrent neural network (Zhang et al. 2016) to text categorization. In the real world, Luz de Araujo et al. (Araujo et al. 2020) built a dataset from Brazilian legal documents and tested document categorization with different techniques to reduce sorting cases by humans. The F1 score results demonstrate the CNN and the Bidirectional Long Short-Term Memory (BiLSTM) outperform than other techniques.

A celebrated technique was proposed by Kim (Kim 2014) that applied CNN technique which was commonly used in computer vision into sentence categorization. He reported simple CNN which was tuned with little hyperparameters that could obtain impressive results on multiple benchmarks. Zhang and Wallace (Zhang and Wallace 2015) reported tuning for the number of feature maps and filter region size are the important factors. Furthermore, They proposed grouped weight sharing for external resource such as Brown clusters, WordNet, and so forth, into text categorization. They used the two-channel model as input. The first channel was word embedding from external resources and the second channel was weight-sharing embedding among resources. The result showed two-channel perform better than single-channel.

Most of them demonstrated that neural network models were powerful for learning features from texts, while they focused on single-label or a few labels problem. Several efforts had been made to multi-labels (Johnson and Zhang 2014). Liu et al. explored a family of new CNN models which were tailored for extreme multi-label categorization (Liu et al. 2017). They used a dynamic max pooling scheme, a binary cross-entropy loss, and a hidden bottleneck layer to improve the overall performance. The results by using six benchmark datasets where the label-set sizes were up to 670 K showed that their method attained at the best or second best in comparison with seven state-of-the-art methods including FastText (Joulin et al. 2017) based CNN. However, all of these attempts aimed at utilizing a large volume of data. Nooralahzadeh et al. proposed Domain-specific Word embeddings using oil and gas corpus and evaluate them with the CNN model and obtained effective results (Nooralahzadeh et al. 2018). Wang et al. Wang et al. 2017 proposed an approach for short text categorization that merged explicit representation and implicit representation together. They mapped a short text into semantic concepts and then constructed word-concept embedding after that was supplied into a CNN to learning explicit knowledge. Furthermore, they concatenated output from a separate CNN with character embedding (Kim et al. 2016) as the input in fully-connected layer of main network, so with this technique they could obtain morphemes level. Wang et al. attempt was similar to my work, while their method used fine-grained and large-scale semantic knowledge that needed to tune the word sense heuristic depending on the domain in which the word is used.

5.2 Application to Text Categorization

My hypothesis about text categorization is that the document assigned to a specific category includes predominant word sense related to the category. I combined the knowledge of Domain-Specific Senses with the embedding of documents. In FIGURE 5.1, the original document is "the court look strict" and word "court" is replaced with sense that is assigned from DSS method. It is the combination between documents and sense embeddings that is, each word in the document is disambiguated and is replaced to its DSS sense obtained from the WordNet. I used it as the input of the Convolutional Neural Network (CNN). With this model, it can learn rich features from both the word level and the sense level, simultaneously.

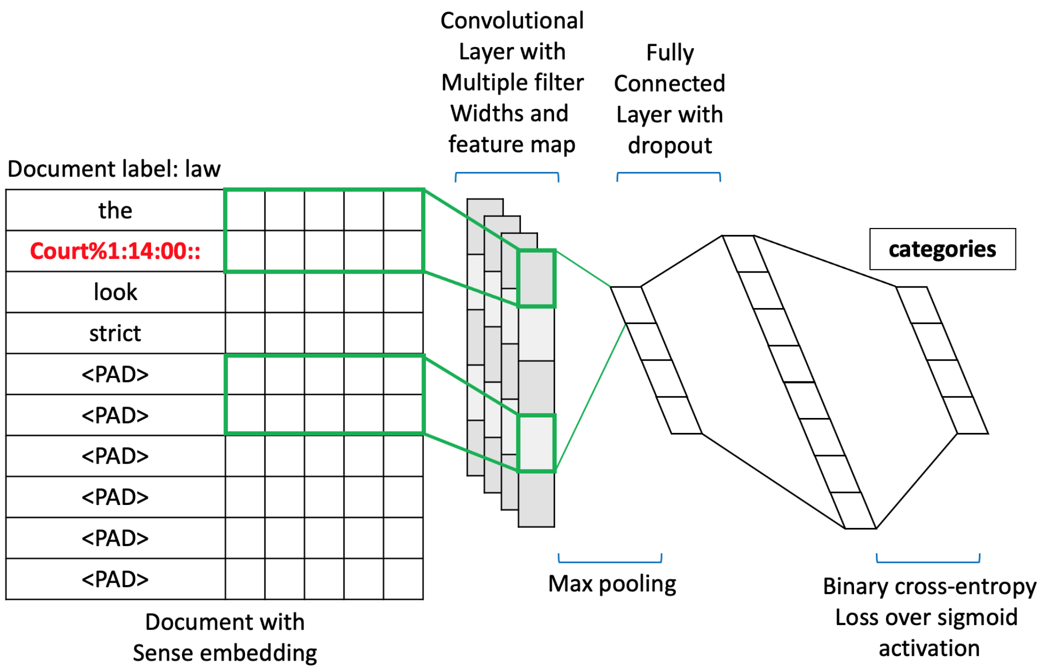
Convolutional Layer with Multiple filter Widths and feature map
Fully Connected Layer with dropout
Document label: law
the
Court%1:14:00::
look
strict
<PAD>
<PAD>
<PAD>
<PAD>
<PAD>
<PAD>
categories
Max pooling
Binary cross-entropy Loss over sigmoid activation
Document with Sense embedding

FIGURE 5.1. Convolutional Neural Network (CNN) model for text categorization.

Similar to other CNN models (Johnson and Zhang 2015; Liu et al. 2017), my model, which is shown in FIGURE 5.1, is based on (Kim 2014). Kim's model applied two channels as input to the model consisting of a static channel and dynamic channel whereas my model only applied a single channel as document embedding integrating with DSS. Another difference is that Kim's model can classify binary and multi-class classification for a sentence. However, my model can classify a document with a multi-label classification which is more complex than his model. Let $\mathbf{x}_i \in \mathbb{R}^k$ be the k-dimensional word vector with the i-th word in the input of CNN obtained by applying the Skip-gram model provided in Word2Vec. The input with length n is represented as $\mathbf{x}_{1:n} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n] \in \mathbb{R}^{nk}$. A convolution filter $\mathbf{w} \in \mathbb{R}^{hk}$ is applied to a window size of h words to produce a new feature, $c_i = f(\mathbf{w} \cdot \mathbf{x}_{i:i+h-1} + b)$ where $b \in \mathbb{R}$ indicates a bias term and f refers to a non-linear activation function. I applied this convolution filter to each possible window size in the input and obtained a feature map, $m \in \mathbb{R}^{n-h+1}$. As shown in FIGURE 5.1, I then apply a max-pooling operation over the feature map and obtain the maximum value $\hat{m}$ as a feature of this filter. I obtained multiple filters by varying window sizes and multiple features. These features form a pooling layer and are passed to a fully connected layer. In the fully connected layer, I applied dropout (Hinton et al. 2012). The dropout randomly sets values in the layer to 0. Finally, I obtained the probability distribution over categories. The network is trained with the objective that minimizes the binary cross-entropy (BCE) of the predicted distributions and the actual distributions by performing stochastic gradient descent.

5.3 Experiments

I have divided experiments into two sections. The first section is to apply DSS from unsupervised learning to the text categorization task to examine how well the automatically acquired senses contribute to categorization accuracy by comparing one of the WSD techniques, Context2vec. Whereas the second section is also to apply DSS from semi-supervised learning to the text categorization task. However, I investigate the performance of unsupervised learning and semi-supervised learning.

5.3.1 Text categorization with unsupervised learning

For comparison of the performance between WSD and my DSS models, I choose the Context2vec model as a WSD method. The 3-month Reuters Training dataset is fed to the Context2vec model and it creates context embeddings and target word embeddings. Both embeddings are used as input to Supervised WSD tasks along with Senseval-3 English lexical samples for training and use a 6-month Reuters dataset for testing to acquire the Context2vec's predicted senses in each context.

The DSS's predicted senses are applied for text categorization to examine how the results obtained by my method affect categorization performance. For each category, I divide a 6-month Reuters dataset into two folds—80% for training and 20% for test data. I further divided the training data into two folds—80% for training data and 20% for validation data. My model setting for CNN is demonstrated in TABLE 5.1, and 5.2. The validation data is used for tuning these settings with the Optuna framework (Akiba et al. 2019) Dropout rate1 in TABLE 5.1, 5.2 shows dropout immediately after the embedding layer, and Dropout rate2 denotes dropout in a fully connected layer. I chose MSF (Multiplicative Scoring Function) with a threshold value of 0.5 to distinguish multi-label categorization (Shimura et al. 2018).

TABLE 5.1. CNN model settings

Description	**Values**
Input size	Maximum length of text $\times$ 100
Input word vectors	Word2Vec
Stride size	1
Filters	32 $\times$ 3
Pooling	1-max pooling
Dropout rate1	0.25
Hidden layers	2048
Learning rate	Predicted by Adam
Loss function	BCE loss over sigmoid activation

The categorization using CNN is as follows—for the target category, I replace each word in the test document with its sense. If the category assign to the test document by the CNN model and the target category match, the test document is judged to classify into the target category. The procedure is applied to each test document and the target category. The results are summarized in TABLE 5.3.

TABLE 5.2. CNN model settings (cont.)

Description	**Values**
A number of output categories	14
Filter region size	(4,5,6)
Feature maps (m)	32
Activation function	ReLu
Dropout	Randomly selected
Dropout rate2	0.5
Batch sizes	128
Epoch	40 with early stopping
Threshold value for MSF	0.5

TABLE 5.3. Categorization performance (Topmost 10%)

Category	**CNN**	**WSD**	**DSS**	**SFC**
Law	0.846	0.853(+.007)	0.899(+.046)	0.908(+.009)
Finance	0.904	0.906(+.002)	0.939(+.033)	0.923(−.016)
Industry	0.798	0.796(−.002)	0.893(+.097)	0.873(−.020)
Publishing	0.738	0.736(−.002)	0.753(+.017)	0.739(−.014)
Admin.	0.875	0.875(.000)	0.918(+.043)	0.933(+.015)
Economy	0.924	0.930(+.006)	0.968(+.038)	0.972(+.004)
Art	0.734	0.741(+.007)	**0.753(+.012)**	0.817(+.064)
Fashion	0.608	0.609(+.001)	0.628(+.019)	**0.775(+.147)**
Politics	0.817	0.813(−.004)	0.900(+.087)	0.964(+.064)
Religion	0.710	0.639(−.071)	**0.855(+.216)**	**0.804(-0.051)**
Sports	0.987	0.987(.000)	0.993(+.006)	0.995(+.002)
Tourism	0.342	0.298(−.044)	0.348(+.050)	0.469(+.121)
Military	0.873	0.873(.000)	0.917(+.044)	0.943(+.026)
Meteorology	0.853	0.848(−.005)	0.875(+.027)	0.863(−.012)
Micro F-score	0.887	0.889(+.002)	0.937(+.048)	0.948(+.011)
Macro F-score	0.786	0.779(−.007)	**0.832(+.053)**	**0.855(+.023)**

TABLE 5.3 demonstrate categories, categorization performance (F-score) with and without Domain-Specific Senses. “CNN” refers to the result without Domain-Specific Senses and “DSS” shows the results obtained by my method. “SFC” shows the results from gold-standard SFC codes. “WSD” refers to the results by Context2vec. DSS shows that the results obtained by DSS are statistically significant compared to those obtained by CNN. Similarly, SFC indicates that the results by SFC are statistically significant compared to those by CNN. I used a t-test, p -value < 0.05. In WSD result, I only used 18 words with 22 senses in all, each of which appears in SFC. In contrast, in my DSS, I used the topmost 10% of the target words that have 442 senses in all.

Overall, the results show that Domain-Specific Senses improved text categorization performance. The best improvement is “Religion” (+0.216), and the worst is “Art” (+0.012).

In contrast, SFC is the best improvement (+0.147) for “Fashion” and the worst “Religion” (-0.051). One reason is that “Religion” is frequently associated with “Politics” and “Military” and its document size is smaller than the other two categories according to TABLE 3.4, however, it betters WSD significantly at 0.165.

The text categorization used here is very simple, that is, CNN with a single channel. There are lots of text categorization techniques applicable to the small number of training documents (Wu et al. 2012; Wang et al. 2017), and it will be worthwhile examining these with my model.

5.3.2 Text categorization with semi-supervised learning

I apply the results of my method (APPNP-DSS) as an input to text categorization and compare the performance with CNN and WMD-DSS approach. The dataset used for text categorization, 6-month RCV1 corpus is divided into two folds consisting of 80 % for the training sets, and 20 % for the test sets. I then divide the training sets once, 80 % for the training sets, and 20 % for the validation sets. All three methods have the same CNN model configurations as shown in TABLE 5.4 and 5.5. I use the Optuna framework (Akiba et al. 2019) for optimizing the best settings of the CNN model.

TABLE 5.4. CNN model configurations

Description	**Values**
Input size	Maximum length of text $\times$ 100
Input word vectors	Word2Vec
Stride size	1
Filters	32 $\times$ 3
Pooling	1-max pooling
Dropout rate1	0.25
Hidden layers	2048
Learning rate	Predicted by Adam
Loss function	BCE loss over sigmoid activation

TABLE 5.5. CNN model configurations (cont.)

Description	**Values**
A number of output categories	14
Filter region size	(4,5,6)
Feature maps (m)	32
Activation function	ReLu
Dropout	Randomly selected
Dropout rate2	0.5
Batch sizes	128
Epoch	40 with early stopping
Threshold value for MSF	0.5

For categorizing using the CNN model, I replace the target words in the document with glosses from my prediction method. The target word is replaced only if the category of sense and category of document match. Another condition is POS of sense and POS of the target word match.

TABLE 5.6. Categorization performance

Category	CNN	WMD-DSS 20%	APPNP-DSS	APPNP-SFC
Law	0.843	0.911(+.068)	0.957(+.046)	0.965(+.008)
Finance	0.904	0.945(+.041)	0.950(+.005)	0.936(-.014)
Industry	0.793	0.899(+.106)	**0.894(-.005)**	0.893(-.001)
Publishing	0.723	0.819(+.096)	0.861(+.042)	**0.826(-.035)**
Administration	0.864	0.913(+.049)	0.970(+.057)	0.969(-.001)
Economy	0.927	0.973(+.046)	0.978(+.005)	0.983(+.005)
Art	0.730	0.773(+.043)	0.914(+.141)	0.897(-.017)
Fashion	0.666	0.775(+.109)	**0.978(+.203)**	0.978(.000)
Politics	0.818	0.926(+.108)	0.960(+.034)	0.978(+.018)
Religion	0.655	0.855(+.200)	0.921(+.066)	0.925(+.004)
Sports	0.988	0.992(+.004)	0.997(+.005)	0.996(-.001)
Tourism	0.246	0.493(+.247)	0.578(+.085)	**0.721(+.143)**
Military	0.871	0.933(+.062)	0.976(+.043)	0.968(-.008)
Meteorology	0.842	0.885(+.043)	0.922(+.037)	0.930(+.008)
Micro F-score	0.886	0.945(+.059)	0.964(+.019)	0.966(+.002)
Macro F-score	0.776	0.864(+.088)	**0.918(+.054)**	**0.926(+.008)**

TABLE 5.6 is a comparison of the F-score performance between CNN, an unsupervised method (WMD-DSS), a supervised method (APPNP-DSS), and a gold-standard of a supervised method (APPNP-SFC). Overall, I notice that the application of DSS in text categorization provides better performance than a normal CNN model. APPNP-DSS performs better than WMD-DSS at the topmost 20 %. The best improvement is “Fashion" (+.203) and the worst is “Industry" (-.005). APPNP-DSS is better than WMD-DSS as the Macro F1-score is 0.918. In contrast, APPNP-SFC is the best improvement (+.143) for “Tourism" and the worst “Publishing" (-0.035). One possible reason why some categories APPNP-DSS have a higher F-score than APPNP-SFC is that the number of times the target word is replaced with the gloss text of APPNP-DSS is larger than the gloss text of APPNP-SFC. Also, if the number of words in gloss text is large, it affects categorization as well. However, APPNP-SFC is still better than APPNP-DSS as the Macro F1-score is 0.926.

5.4 Conclusions

In this chapter, I present the influence of DSS on text categorization The comparison consists of two parts: 1) Evaluation between unsupervised learning and WSD 2) Evaluation between semi-supervised learning and unsupervised learning.

The text categorization task with unsupervised learning has empirically proven that DSS is able to achieve a better performance than the WSD method. The DSS results attained at F-score 0.745 for 442 senses, furthermore when applying them to text categorization, I obtained the categorization accuracy as the Macro F-score is 0.832 that exceeds the WSD method 0.053.

The text categorization task with semi-supervised learning has demonstrated that this approach performs better than unsupervised learning. The results of this experiment showed that my method can improve text categorization performance as it achieved a 0.918 macro F-score and 0.142 improvements compared with the CNN baseline model.

For future work, there are several approaches that can be further study. I am going to apply DSS to other NLP applications such as machine translation, question answering, and sentiment analysis. I also apply my method to other part-of-speech e.g. adjectives and adverbs as well as other datasets and thesaurus for quantitative evaluation of my method.

CHAPTER 6

Conclusion

6.1 Conclusion

In this thesis, I focused on Domain-specific senses and presents methods for detecting predominant senses for each domain. My thesis proposal includes: (1) to identify DSS using an unsupervised technique based on word embedding and (2) to demonstrate DSS performance to text classification and (3) to identify DSS using a semi-supervised technique and to evaluate DSS performance with an unsupervised technique to text classification.

In Chapter 3, I proposed an unsupervised method for detecting predominant senses based on word embedding. I used the RCV1 corpus as a dataset and WordNet 3.1 as a lexicon. I first extracted only noun words that had word frequency more than five from RCV1 and then I used those words to retrieve all sense of each word from WordNet. Stop word list is removed from gloss text before training Word2Vec that is used to generate word embedding. Sense similarity between gloss text is calculated using WMD. It measures the distance that the embedded words of sentence travel to the embedded words of another sentence. The final step is to detect the predominant senses for each domain. I used the MRW model to choose the importance of sense within a graph. The results from 6 categories showed that the overall performance of word embedding learning using the WMD technique was better than the Cosine technique. When examining the topmost ratio of ranking, I found at the topmost 20% obtain the F-score 0.737, and at the topmost more than 30%, the F-score gradually decreases to the topmost 50% that the F-score is approximately 0.55. The results from 14 categories showed that the overall performance of word embedding learning using the WMD technique still worked well. I found at the topmost 10% obtain the F-score 0.745.

In chapter 4, I proposed a semi-supervised method based on a neural random walk model because an unsupervised method is not effective. I still focused on noun senses and verb senses similar in chapter 4. along with 14 domains. However, to obtain DSS, I utilized a graph-based model consisting of GCN and Pagerank. Furthermore, I choose BERT to generate sense embedding since it obtains a better quality of embedding than the traditional method. This method attained a macro F-score t 0.647.

In chapter 5, I examined the influence of DSS on text classification that covers both unsupervised learning and semi-supervised learning. The unsupervised learning results showed DSS obtain a macro F-score at 0.832 which was better than a CNN baseline at 0.046. Whereas I investigated the efficacy of DSS to text classification with semi-supervised learning. It reached a macro F1-score at 0.918 while the CNN baseline was 0.776.

6.2 Future Work

There are several directions for future work. In the first work, I going to apply the DSS method to larger datasets and test a larger number of domains, i.e., Wikipedia. My method should be covered to other part-of-speech e.g. adjectives and adverbs for quantitative evaluation. It would be interesting to see the efficacy of the embedding methods by switching Word2Vec to other methods i.e. FastText, GloVe. The computational cost of WMD is $O(p^3 logp)$, where p denotes the number of distinct words in the documents (Pele and Werman 2009). The computational cost is expensive as it is based on a linear optimization problem. I also need to investigate methods to improve run-time efficiencies and this is a rich space for further exploration. Throughout this study, I focus on an English-based corpus. I should experiment with different languages to investigate the performance of my method. The WordNet thesaurus is utilized in this work, therefore I need to utilize another thesaurus, for example, Roget's by using corpus statistics.

In the second work, I should test on different datasets to examine the performance of my method. It would be interesting to explore the variant of the BERT method i.e. Sentence-BERT (Reimers and Gurevych 2019), tBERT (Peinelt et al. 2020) to see the significance of them. A novel graph model would be investigated to see the performance i.e. NENN (Yang and Li 2020) and this is exciting for further exploration. Comparison to the state-of-art WSD technique (Bevilacqua and Navigli 2020) by using the same datasets, SemEval Check whether this dataset is correct or not is also necessary to examine the effectiveness of the method.

In the third work, For DSS evaluation on NLP task, I need to compare my method to the SOTA text classification techniques (Wang et al. 2017) and WSD techniques to examine the effectiveness of both methods. I should conduct DSS to other NLP applications such as Question-Answering, Machine-translation, etc. Moreover, I further need to detect DSS with the supervised learning method and then I can compare DSS from three types of learning methods.

Bibliography

Abreu, J., L. Fred; D. Macêdo and C. Zanchettin (2019). ‘Hierarchical Attentional Hybrid Neural Networks for Document Classification’. In: *CoRR* abs/1901.06610.

Agirre, E. and A. Soroa (2009). ‘Personalizing Pagerank for Word Sense Disambiguation’. In: *Proc. of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 33–41.

Akiba, T. et al. (2019). ‘Optuna: A Next-generation Hyperparameter Optimization Framework’. In: *CoRR* abs/1907.10902.

Araujo, P.H. Luz de et al. (2020). ‘VICTOR: A dataset for Brazilian legal documents classification’. In: *Proceedings of the 12th Language Resources and Evaluation Conference*, pp. 1449–1458.

Beltagy, I., A. Cohan and K. Lo (2019). ‘SciBERT: Pretrained Contextualized Embeddings for Scientific Text’. In: *CoRR* abs/1903.10676.

Bentivogli, L. et al. (2004). ‘Revising the WordNet Domains Hierarchy: Semantics, Coverage and Balancing’. In: *Proc. of the Workshop on Multilingual Linguistic Resources*, pp. 101–108.

Bevilacqua, M. and R. Navigli (2020). ‘Breaking Through the 80% Glass Ceiling: Raising the State of the Art in Word Sense Disambiguation by Incorporating Knowledge Graph Information’. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 2854–2864.

Blei, D. M., A. Y. Ng and M. I. Jordan (2003). ‘Latent Dirichlet Allocation’. In: *Journal of Machine Learning Research* 3, pp. 993–1022.

Bojanowski, P. et al. (2016). ‘Enriching Word Vectors with Subword Information’. In: *CoRR* abs/1607.04606.

Bremaud, P. (1999). *Markov Chains: Gibbs Fields, Monte Carlo Simulation, and Queues*. Springer-Verlag.

Brin, S. and L. Page (1998). ‘The Anatomy of a Large-scale Hypertextual Web Search Engine’. In: *Computer Networks and ISDN Systems* 30, pp. 107–117.

Buitelaar, P. and B. Sacaleanu (2001). ‘Ranking and Selecting Synsets by Domain Relevance’. In: *Proc. of the WordNet and Other Lexical Resources: Applications, Extensions and Customizations. NAACL Workshop*, pp. 119–124.

Cristani, M. et al. (2018). ‘Future paradigms of automated processing of business documents’. In: *Int. J. Inf. Manag. Sci.*, pp. 67–75.

Deerwester, S. et al. (1990). ‘Indexing by Latent Semantic Analysis’. In: *Journal of the American Society of Information Science* 41.6, pp. 391–407.

Devlin, J. et al. (2018). ‘BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding’. In: *CoRR* abs/1810.04805.

Dongsuk, O. et al. (2018). ‘Word Sense Disambiguation Based on Word Similarity Calculation Using Word Vector Representation from a Knowledge-based Graph’. In: *Proc. of the 27th COLING*, pp. 2704–2714.

Finkel, J. R., T. Grenager and C. Manning (2005). ‘Incorporating Non-Local Information into Information Extraction Systems by Gibbs Sampling’. In: *Proc. of the 43rd Annual Meeting on Association for Computational Linguistics*, pp. 363–370.

Firth, J. R. (1957). ‘A synopsis of linguistic theory’. In: *Studies in Linguistic Analysis (special volume of the Philological Society)* 1952-59.

Fukumoto, F. and Y. Suzuki (2010). ‘Identifying Domain-specific Senses and Its Application to Text Classification’. In: *Proceedings of the International Conference on Knowledge Engineering and Ontology Development*, pp. 263–268.

Gale, W. A., K. W. Church and D. Yarowsky (1992). ‘One Sense Per Discourse’. In: *Proc. of Speech and Natural Language Workshop 1992*, pp. 233–237.

Ghosh, S. et al. (2016). ‘Contextual LSTM(CLSTM) models for large scale NLP tasks’. In: *CoRR* abs/1907.10902.

Greene, D. and P. Cunningham (2006). ‘Practical Solutions to the Problem of Diagonal Dominance in Kernel Document Clustering’. In: *Proc. of the 23rd International Conference on Machine Learning*, pp. 377–384.

Hinton, G. E. et al. (2012). ‘Improving Neural Networks by Preventing Co-Adaptation of Feature Detectors’. In: *CoRR* abs/1207.0580.

Hitchcock, F. L. (1941). ‘The Distribution of a Product from Several Sources to Numerous Localities’. In: *Journal of Mathematics and Physics* 20, pp. 224–230.

Huang, Z., W. Xu and K. Yu (2015). ‘Bidirectional LSTM-CRF models for sequence tagging’. In: *CoRR* abs/1508.01991.

Johnson, R. and T. Zhang (2014). ‘Effective Use of Word Order for Text Categorization with Convolutional Neural Networks’. In: *CoRR* abs/1412.1058.

— (2015). ‘Semi-Supervised Convolutional Neural Networks for Text Categorization via Region Embedding’. In: *Adv Neural Inf Process Syst* 28, pp. 919–927.

Joulin, A. et al. (2017). ‘Bag of Tricks for Efficient Text Classification’. In: *Proc. of the 15th Conference of the EACL*, pp. 427–431.

Kågebäck, M. and H. Salomonsson (2016). ‘Word Sense Disambiguation using a Bidirectional LSTM’. In: *Proc. of 5th Workshop on Cognitive Aspects of the Lexicon (CogALex)*, pp. 51–56.

Kim, Y. (2014). ‘Convolutional Neural Networks for Sentence Classification’. In: *Proc. of the 2014 Conf. on EMNLP*, pp. 1746–1751.

Kim, Y. et al. (2016). ‘Character-aware neural language models’. In: *Proc. of the 30th AAAI*, pp. 2741–2749.

Kipf, T. N. and M. Welling (2016). ‘Semi-Supervised Classification with Graph Convolutional Networks’. In: *CoRR* abs/1609.02907.

Klicpera, J., A. Bojchevski and S. Günnemann (2018). ‘Personalized Embedding Propagation: Combining Neural Networks on Graphs with Personalized PageRank’. In: *CoRR* abs/1810.05997.

Kusner, M. et al. (2015). ‘From Word Embeddings to Document Distances’. In: *Proc. of the 32nd International Conference on Machine Learning*, pp. 957–966.

Kutuzov, A. et al. (2018). 'Learning Graph Embeddings from WordNet-based Similarity Measures'. In: *CoRR* abs/1808.05611.

Lee, J. et al. (2019). 'BioBERT: a pre-trained biomedical language representation model for biomedical text mining'. In: *CoRR* abs/1901.08746.

Li, Q., Z. Han and X-M Wu (2018). 'Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning'. In: *CoRR* abs/1801.07606.

Lin, D. (1998). 'Automatic Retrieval and Clustering of Similar Words'. In: *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 2*, pp. 768–774.

Liu, J. et al. (2017). 'Deep Learning for Extreme Multi-Label Text Classification'. In: *Proc. of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Lu, J. et al. (2019). 'ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks'. In: *CoRR* abs/1908.02265.

Magnini, B. and G. Cavaglia (2000). 'Integrating Subject Field Codes into WordNet'. In: *Proc. of the International Conference on Language Resources and Evaluation 2000*, pp. 1413–1418.

Magnini, B. et al. (2002). 'The Role of Domain Information in Word Sense Disambiguation'. In: *Natural Language Engineering* 8, pp. 359–373.

Manning, C. et al. (2014). 'The Stanford CoreNLP Natural Language Processing Toolkit'. In: *Proc. of 52nd Annual Meeting of the ACL: System Demonstrations*, pp. 553–590.

McCarthy, D. et al. (2004). 'Finding Predominant Word Senses in Untagged Text'. In: *Proc. of the 42nd Annual Meeting on Association for Computational Linguistics*, pp. 279–286.

— (2007). 'Unsupervised Acquisition of Predominant Word Senses'. In: *Comput. Linguist.*, pp. 553–590.

Melamud, O., J. Goldberger and I. Dagan (2016). 'Context2vec: Learning Generic Context Embedding wuth Bidirectional LSTM'. In: *Proc. of the 20th SIGNLL*, pp. 51–61.

Mihalcea, R. (2005). 'Language Independent Extractive Summarization'. In: *Proc. of the ACL Interactive Poster and Demonstration Sessions*, pp. 49–52.

Mikolov, T. et al. (2013). 'Efficient Estimation of Word Representations in Vector Space'. In: *Computing Research Repository* abs/1301.3781.

Miller, G. A. (1995). 'WordNet: A Lexical Database for English'. In: *Communications of the ACM* 38, pp. 39–41.

Netlib (2007). 'http://www.netlib.org/scalapack/index.html'. In: *Netlib Repository at UTK and ORNL.*

Nooralahzadeh, F., L. Ovrelid and J. T. Lønning (2018). 'Evaluation of Domain-specific Word Embeddings using Knowledge Resources'. In: *Proc. of the Eleventh International Conf. on LREC 2018.*

Pasini, T. and R. Navigli (2017). 'Train-O-Matic: Large-Scale Supervised Word Sense Disambiguation in Multiple Languages without Manual Training Data'. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 78–88.

— (2019). 'Train-O-Matic: Supervised Word Sense Disambiguation with no (manual) effort'. In: *Artif. Intell.* 279.

Peinelt, N., D. Nguyen and M. Liakata (2020). ‘tBERT: Topic Models and BERT Joining Forces for Semantic Similarity Detection’. In: *58th annual meeting of the association for computational linguistics*, pp. 7047–7055.

Pele, O. and M. Werman (2009). ‘Fast and Robust Earth Mover’s Distances.’ In: *Proc of the 12th International Conference on Computer Vision*, pp. 460–467.

Peng, Y., S. Yan and Z. Lu (2019). ‘Transfer Learning in Biomedical Natural Language Processing: An Evaluation of BERT and ELMo on Ten Benchmarking Datasets’. In: *CoRR* abs/1906.05474.

Pennington, J., R. Socher and C. D. Manning (2014). ‘GloVe: Global Vectors for Word Representation’. In: *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543.

Perozzi, B., R. AI-Rfou and S. Skiena (2014). ‘DeepWalk: Online Learning of Social Representations’. In: *Proc. of the 20th ACM SIGKDD*, pp. 701–710.

Ramage, D., A. N. Rafferty and C. D. Manning (2009). ‘Random Walks for Text Semantic Similarity’. In: *Proc. of the 4th TextGraphs Workshop on Graph-based Algorithms in NLP*, pp. 23–31.

Reddy, S. et al. (2010). ‘IIITH: Domain Specific Word Sense Disambiguation’. In: *Proc. of the 5th International Workshop on Semantic Evaluation*, pp. 387–391.

Reimers, N. and I. Gurevych (2019). ‘Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks’. In: *2019 Conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing*, pp. 3982–3992.

Rumelhart, D. E., G. E. Hinton and R. J. Williams (1986). ‘Learning Representation by Back-Propagating Errors’. In: *Nature* 323.6088, pp. 533–536.

Scarlini, B., T. Pasini and R. Navigli (2019). ‘Just “OneSeC" for Producing Multilingual Sense-Annotated Data’. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Lisnguistics*, pp. 699–709.

Schmid, H. (1994). ‘Probabilistic Part-of-Speech Tagging Using Decision Trees’. In: *Proc. of the International Conference on New Methods in Language Processing*, pp. 44–49.

Shimura, K., J. Li and F. Fukumoto (2018). ‘HFT-CNN:Learning Hierarchical Category Structure for Multi-label Short Text Categorization’. In: *Proc. of the 2018 EMNLP*, pp. 811–816.

Sinha, R. and R. Mihalcea (2007). ‘Unsupervised Graph-based Word Sense Disambiguation using Measures of Word Semantic Similarity’. In: *Proc. of the International Conference on Semantic Computing*, pp. 363–369.

Sun, Z. et al. (2020). ‘MobileBERT: a Compact Task-Agnostic BERT for Resource-Limited Devices’. In: *ACL (2020).*

Taghipour, K. and H. T. Ng (2015). ‘One Million Sense-Tagged Instances for Word Sense Disambiguation and Induction’. In: *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pp. 338–344.

Vaswani, A. et al. (2017). ‘Attention Is All You Need’. In: *CoRR* abs/1706.03762.

Walker, D. E. and R. A. Amsler (1986). ‘The use of machine-readable dictionaries in sublanguage analysis. Analyzing Language in Restricted Domains’. In:

Wang, J. et al. (2017). ‘Combining Knowledge with Deep Convolutional Neural Networks for Short Text Classification’. In: *Proc. of the 26th IJCAI*, pp. 2915–2921.

Wang, Y. et al. (2018). 'Dual Transfer Learning for Neural Machine Translation with Marginal Distribution Regularization'. In: *Proc. of the 32nd AAAI Conference on Artificial Intelligence*.

Wu, F. et al. (2019). 'Simplifying Graph Convolutional Networks'. In: *CoRR* abs/1902.07153.

Wu, W. et al. (2012). 'A Probabilistic Taxonomy for Text Understanding'. In: *Proc. of the 2012 ACM SIGMOD International Conference on Management of Data*, pp. 481–492.

Wu, Y. et al. (2016). 'Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation'. In: *CoRR* abs/1609.08144.

Xiao, H. (2018). *bert-as-service*. `https://github.com/hanxiao/bert-as-service`.

Yang, Y. and D. Li (2020). 'NENN: Incorporate Node and Edge Features in Graph Neural Networks'. In: *12th Asian Conference on Machine Learning*, pp. 593–608.

Yang, Z. et al. (2016). 'Hierarchical Attention Networks for Document Classification'. In: *Proc. of the 2016 Conf. of the NAACL-HLT*, pp. 1480–1489.

Yao, Y. and Z. Huang (2016). 'Bi-directional LSTM recurrent neural network for Chinese word segmentation'. In: *CoRR* abs/1602.04874.

Yarowsky, D. (1993). 'One Sense per Collocation'. In: *Proceedings of a Workshop on Human Language Technology*, pp. 266–271.

Zhang, R., H. Lee and D. Radev (2016). 'Dependency Sensitive Convolutional Neural Networks for Modeling Sentences and Documents'. In: *Proc. of the 2016 Conf. of the NAACL-HLT*, pp. 1512–1521.

Zhang, Y. and B. C. Wallace (2015). 'A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification'. In: *CoRR* abs/1510.03820.

Zhong, Z. and H. T. Ng (2010). 'It Makes Sense: A Wide-Coverage Word Sense Disambiguation System for Free Text'. In: *Proceedings of the ACL 2010 System Demonstrations*, pp. 78–83.

Zonghan, W. et al. (2019). 'A Comprehensive Survey on Graph Neural Networks'. In: *CoRR* abs/1901.00596.

APPENDIX A

Appendix

A1 : An example Reuters Corpus Volume 1 document

This is an example file of RCV1 file that I extracted news content and its categories.

TABLE A.1. The example of RCV1 document

```xml
<?xml version="1.0" encoding="iso-8859-1" ?>
<newsitem itemid="679352" id="root" date="1997-06-23" xml:lang="en">
<title>USA: Capital Growth Holdings sets first div.</title>
<headline>Capital Growth Holdings sets first div.</headline>
<dateline>GREENWICH, Conn. 1997-06-23</dateline>
<text>
<p>Capital Growth Holdings Ltd said Monday its board of directors declared a $0.225 annual per share dividend on shares
of its common stock for each of 1997 and 1998.</p>
<p>The first dividend payment is slated for July 2 for shareholders of record June 30. After that, the dividend is
anticipated to be paid quarterly.</p>
<p>Capital Growth Holdings Ltd develops diversified financial services pursuing world-wide business opportunities.</p>
</text>
<copyright>(c) Reuters Limited 1997</copyright>
<metadata>
<codes class="bip:countries:1.0">
  <code code="USA">
    <editdetail attribution="Reuters BIP Coding Group" action="confirmed" date="1997-06-23"/>
  </code>
</codes>
<codes class="bip:topics:1.0">
  <code code="C15">
    <editdetail attribution="Reuters BIP Coding Group" action="confirmed" date="1997-06-23"/>
  </code>
  <code code="C151">
    <editdetail attribution="Reuters BIP Coding Group" action="confirmed" date="1997-06-23"/>
  </code>
  <code code="CCAT">
    <editdetail attribution="Reuters BIP Coding Group" action="confirmed" date="1997-06-23"/>
  </code>
</codes>
<dc element="dc.date.created" value="1997-06-23"/>
<dc element="dc.publisher" value="Reuters Holdings Plc"/>
<dc element="dc.date.published" value="1997-06-23"/>
<dc element="dc.source" value="Reuters"/>
<dc element="dc.creator.location" value="GREENWICH, Conn."/>
<dc element="dc.creator.location.country.name" value="USA"/>
<dc element="dc.source" value="Reuters"/>
</metadata>
</newsitem>
```

A2 : Stopword list

This table illustrates an NLTK(Natural Language Toolkit) stopword list including 179 words that we applied in our experiment.

TABLE A.2. NLTK stopword list

i	me	my	myself	we
our	ours	ourselves	you	you're
you've	you'll	you'd	your	yours
yourself	yourselves	he	him	his
himself	she	she's	her	hers
herself	it	it's	its	itself
they	them	their	theirs	themselves
what	which	who	whom	this
that	that'll	these	those	am
is	are	was	were	be
been	being	have	has	had
having	do	does	did	doing
a	an	the	and	but
if	or	because	as	until
while	of	at	by	for
with	about	against	between	into
through	during	before	after	above
below	to	from	up	down
in	out	on	off	over
under	again	further	then	once
here	there	when	where	why
how	all	any	both	each
few	more	most	other	some
such	no	nor	not	only
own	same	so	than	too
very	s	t	can	will
just	don	don't	should	should've
now	d	ll	m	o
re	ve	y	ain	aren
aren't	couldn	couldn't	didn	didn't
doesn	doesn't	hadn	hadn't	hasn
hasn't	haven	haven't	isn	isn't
ma	mightn	mightn't	mustn	mustn't
needn	needn't	shan	shan't	shouldn
shouldn't	wasn	wasn't	weren	weren't
won	won't	wouldn	wouldn't	

A3 : list of POS tags used in the Penn Treebank Project

TABLE A.3. POS list

Number	Tag	Description
1.	CC	Coordinating conjunction
2.	CD	Cardinal number
3.	DT	Determiner
4.	EX	Existential there
5.	FW	Foreign word
6.	IN	Preposition or subordinating conjunction
7.	JJ	Adjective
8.	JJR	Adjective, comparative
9.	JJS	Adjective, superlative
10.	LS	List item marker
11.	MD	Modal
12.	NN	Noun, singular or mass
13.	NNS	Noun, plural
14.	NNP	Proper noun, singular
15.	NNPS	Proper noun, plural
16.	PDT	Predeterminer
17.	POS	Possessive ending
18.	PRP	Personal pronoun
19.	PRP$	Possessive pronoun
20.	RB	Adverb
21.	RBR	Adverb, comparative
22.	RBS	Adverb, superlative
23.	RP	Particle
24.	SYM	Symbol
25.	TO	to
26.	UH	Interjection
27.	VB	Verb, base form
28.	VBD	Verb, past tense
29.	VBG	Verb, gerund or present participle
30.	VBN	Verb, past participle
31.	VBP	Verb, non-3rd person singular present
32.	VBZ	Verb, 3rd person singular present
33.	WDT	Wh-determiner
34.	WP	Wh-pronoun
35.	WP$	Possessive wh-pronoun
36.	WRB	Wh-adverb

A4 : Reuters topic codes

TABLE A.4. Reuters topic codes

Topic codes	Description
1POL	CURRENT NEWS - POLITICS
2ECO	CURRENT NEWS - ECONOMICS
3SPO	CURRENT NEWS - SPORT
4GEN	CURRENT NEWS - GENERAL
6INS	CURRENT NEWS - INSURANCE
7RSK	CURRENT NEWS - RISK NEWS
8YDB	TEMPORARY
9BNX	TEMPORARY
ADS10	CURRENT NEWS - ADVERTISING
BNW14	CURRENT NEWS - BUSINESS NEWS
BRP11	CURRENT NEWS - BRANDS
C11	STRATEGY/PLANS
C12	LEGAL/JUDICIAL
C13	REGULATION/POLICY
C14	SHARE LISTINGS
C15	PERFORMANCE
C151	ACCOUNTS/EARNINGS
C1511	ANNUAL RESULTS
C152	COMMENT/FORECASTS
C16	INSOLVENCY/LIQUIDITY
C17	FUNDING/CAPITAL
C171	SHARE CAPITAL
C172	BONDS/DEBT ISSUES
C173	LOANS/CREDITS
C174	CREDIT RATINGS
C18	OWNERSHIP CHANGES
C181	MERGERS/ACQUISITIONS
C182	ASSET TRANSFERS
C183	PRIVATISATIONS
C21	PRODUCTION/SERVICES
C22	NEW PRODUCTS/SERVICES
C23	RESEARCH/DEVELOPMENT
C24	CAPACITY/FACILITIES
C31	MARKETS/MARKETING
C311	DOMESTIC MARKETS
C312	EXTERNAL MARKETS
C313	MARKET SHARE
C32	ADVERTISING/PROMOTION

TABLE A.4. Reuters topic codes (cont.)

Topic codes	Description
C33	CONTRACTS/ORDERS
C331	DEFENCE CONTRACTS
C34	MONOPOLIES/COMPETITION
C41	MANAGEMENT
C411	MANAGEMENT MOVES
C42	LABOUR
CCAT	CORPORATE/INDUSTRIAL
E11	ECONOMIC PERFORMANCE
E12	MONETARY/ECONOMIC
E121	MONEY SUPPLY
E13	INFLATION/PRICES
E131	CONSUMER PRICES
E132	WHOLESALE PRICES
E14	CONSUMER FINANCE
E141	PERSONAL INCOME
E142	CONSUMER CREDIT
E143	RETAIL SALES
E21	GOVERNMENT FINANCE
E211	EXPENDITURE/REVENUE
E212	GOVERNMENT BORROWING
E31	OUTPUT/CAPACITY
E311	INDUSTRIAL PRODUCTION
E312	CAPACITY UTILIZATION
E313	INVENTORIES
E41	EMPLOYMENT/LABOUR
E411	UNEMPLOYMENT
E51	TRADE/RESERVES
E511	BALANCE OF PAYMENTS
E512	MERCHANDISE TRADE
E513	RESERVES
E61	HOUSING STARTS
E71	LEADING INDICATORS
ECAT	ECONOMICS
ENT12	CURRENT NEWS - ENTERTAINMENT
G11	SOCIAL AFFAIRS
G111	HEALTH/SAFETY
G112	SOCIAL SECURITY
G113	EDUCATION/RESEARCH
G12	INTERNAL POLITICS

TABLE A.4. Reuters topic codes (cont.)

Topic codes	Description
G13	INTERNATIONAL RELATIONS
G131	DEFENCE
G14	ENVIRONMENT
G15	EUROPEAN COMMUNITY
G151	EC INTERNAL MARKET
G152	EC CORPORATE POLICY
G153	EC AGRICULTURE POLICY
G154	EC MONETARY/ECONOMIC
G155	EC INSTITUTIONS
G156	EC ENVIRONMENT ISSUES
G157	EC COMPETITION/SUBSIDY
G158	EC EXTERNAL RELATIONS
G159	EC GENERAL
GCAT	GOVERNMENT/SOCIAL
GCRIM	CRIME, LAW ENFORCEMENT
GDEF	DEFENCE
GDIP	INTERNATIONAL RELATIONS
GDIS	DISASTERS AND ACCIDENTS
GEDU	EDUCATION
GENT	ARTS, CULTURE, ENTERTAINMENT
GENV	ENVIRONMENT AND NATURAL WORLD
GFAS	FASHION
GHEA	HEALTH
GJOB	LABOUR ISSUES
GMIL	MILLENNIUM ISSUES
GOBIT	OBITUARIES
GODD	HUMAN INTEREST
GPOL	DOMESTIC POLITICS
GPRO	BIOGRAPHIES, PERSONALITIES, PEOPLE
GREL	RELIGION
GSCI	SCIENCE AND TECHNOLOGY
GSPO	SPORTS
GTOUR	TRAVEL AND TOURISM
GVIO	WAR, CIVIL WAR
GVOTE	ELECTIONS
GWEA	WEATHER
GWELF	WELFARE, SOCIAL SERVICES
M11	EQUITY MARKETS
M12	BOND MARKETS

TABLE A.4. Reuters topic codes (cont.)

Topic codes	Description
M13	MONEY MARKETS
M131	INTERBANK MARKETS
M132	FOREX MARKETS
M14	COMMODITY MARKETS
M141	SOFT COMMODITIES
M142	METALS TRADING
M143	ENERGY MARKETS
MCAT	MARKETS
MEUR	EURO CURRENCY
PRB13	CURRENT NEWS - PRESS RELEASE WIRES

A5 : English sample lexical task

Lists of words in the data set, and corresponding size of training/test data

TABLE A.5. English Lexical Sample

No.	Lexical unit	Training size	Test size
1.	activate.v	228	114
2.	add.v	263	132
3.	appear.v	265	133
4.	argument.n	221	111
5.	arm.n	266	133
6.	ask.v	261	131
7.	atmosphere.n	161	81
8.	audience.n	200	100
9.	bank.n	262	132
10.	begin.v	181	79
11.	climb.v	133	67
12.	decide.v	122	62
13.	degree.n	256	128
14.	difference.n	226	114
15.	different.a	98	50
16.	difficulty.n	46	23
17.	disc.n	200	100
18.	eat.v	181	87
19.	encounter.v	130	65
20.	expect.v	156	78
21.	express.v	110	55
22.	hear.v	63	32
23.	hot.a	86	43
24.	image.n	146	74
25.	important.a	36	19
26.	interest.n	185	93
27.	judgment.n	62	32
28.	lose.v	71	36
29.	mean.v	80	40
30.	miss.v	58	30
31.	note.v	132	67
32.	operate.v	35	18
33.	organization.n	112	56
34.	paper.n	232	117
35.	party.n	230	116

TABLE A.5. English Lexical Sample (cont.)

No.	Lexical unit	Training size	Test size
36.	performance.n	172	87
37.	plan.n	166	84
38.	play.v	104	52
39.	produce.v	186	94
40.	provide.v	136	69
41.	receive.v	52	27
42.	remain.v	139	70
43.	rule.v	59	30
44.	shelter.n	196	98
45.	simple.a	36	18
46.	smell.v	108	55
47.	solid.a	58	29
48.	sort.n	190	96
49.	source.n	64	32
50.	suspend.v	128	64
51.	talk.v	146	73
52.	treat.v	112	57
53.	use.v	26	14
54.	wash.v	66	34
55.	watch.v	100	51
56.	win.v	78	39
57.	write.v	44	23

A6 : Published Paper

(1) Wangpoonsarp Attaporn, Fukumoto Fumiyo, *Identification of Domain-Specific Senses based on Word Embedding Learning*, In Proc. of 8th Language and Technology Conference (LTC'17), November 2017, Poznań, Poland.

(2) Wangpoonsarp Attaporn, Shimura Kazuya, Fukumoto Fumiyo, *Unsupervised Predominant Sense Detection and its Application to Text Classification*, Appl. Sci. 2020, 10, 6052.

(3) Wangpoonsarp Attaporn, Fukumoto Fumiyo, *Predominant Sense Acquisition with a neural random walk model*, Applied Intelligence, 2020.(Under revising as suggested by reviewer.)