A SYSTEMATIC EVALUATION OF SEMANTIC REPRESENTATIONS IN NATURAL LANGUAGE PROCESSING

A Thesis Submitted to
the Graduate School of Engineering and Sciences of
İzmir Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

in Computer Engineering

by
Özge SEVGİLİ ERGÜVEN

July 2018
İZMİR

We approve the thesis of **Özge SEVGİLİ ERGÜVEN**

Examining Committee Members:

Dr. Mutlu BEYAZIT
Department of Computer Engineering, Yaşar University

Dr. Nesli ERDOĞMUŞ
Department of Computer Engineering, İzmir Institute of Technology

---

Dr. Selma TEKİR
Department of Computer Engineering, İzmir Institute of Technology

5 July 2018

Dr. Selma TEKİR
Department of Computer Engineering
İzmir Institute of Technology

Assoc. Prof. Dr. Yusuf Murat ERTEN
Head of the Department of
Computer Engineering

Prof. Dr. Aysun SOFUOĞLU
Dean of the Graduate School of
Engineering and Sciences

ACKNOWLEDGMENTS

I would like to express sincere appreciation to my supervisor, Selma TEKİR who has encouraged me since my undergraduate studies. I am grateful for her respect for my individual opinions, her motivation, continuous support, and patience during this thesis.

I would like to express my infinite gratitude to my parents and brother for their unconditional love, unlimited patience, understanding and support without any expectations during this thesis and all my life.

Finally, I would like to state my special thanks to my husband for his endless support mentally and technically, love, motivation, patience, and especially for his understanding.

I dedicated this thesis work to my beloved family and husband.

ABSTRACT

A SYSTEMATIC EVALUATION OF SEMANTIC REPRESENTATIONS IN NATURAL LANGUAGE PROCESSING

In the studies of semantics, the main aim is to address meaning. In a computational manner, this goal is accomplished through the encoding of language constructs. These encodings are in the form of information-theoretic measures and vector representations. We have focused on the representation of words. In word representations, the earlier approaches depend on counting the statistics between word and its accompanied words, whereas the current methods are based on learning approaches. At this point, we have investigated the relation between these two approaches. We have realized that both approaches use context as the normalization factor. We support our idea by evaluating word representations on some Natural Language Processing (NLP) tasks. Furthermore, we have studied the polysemous words which carry more than one meaning. The word representation of the polysemous word provides a representation that covers more than one meaning. To overcome this issue, we provide a method to create a representation for each sense of polysemous word.

ÖZET

DOĞAL DİL İŞLEMEDE SEMANTİK GÖSTERİMLERİN SİSTEMATİK DEĞERLENDİRİLMESİ

Semantik çalışmalarında, temel amaç anlamı ele almaktır. Hesaplamalı yöntemlerde, bu hedef dil yapılarının kodlanması ile gerçekleştirilir. Bu kodlamalar istatistiksel ölçütler ve vektör gösterimleri şeklindedir. Çalışmada kelime gösterimleri üzerine odaklanılmıştır. Kelime gösterimlerinde, önceki çalışmalar kelime ve onun eşlik ettiği kelimeler arasındaki istatistiklerin sayılmasına dayanırken, mevcut yöntemler öğrenme tabanlıdır. Tezde bu iki yaklaşım arasındaki ilişki araştırılmıştır. Her iki yaklaşımın da bağlamı normalleştirme faktörü olarak kullandığı görülmüştür. Bu fikir, kelime gösterimlerinin bazı Doğal Dil İşleme problemlerinde değerlendirilmesi ile desteklenmiştir. Ayrıca, birden fazla anlam taşıyan çok-anlamlı kelimeler üzerine çalışılmıştır. Çok-anlamlı bir kelimenin kelime gösterimi, birden fazla anlamını kapsamaktadır. Bu sorunu aşmak için, çok-anlamlı kelimenin her bir anlamı için ayrı bir gösterim sağlayan bir yöntem geliştirilmiştir.

TABLE OF CONTENTS

LIST OF FIGURES

Figure **Page**

LIST OF TABLES

Table **Page**

LIST OF ABBREVIATIONS

NLP . . . . . Natural Language Processing
PMI . . . . . Pointwise Mutual Information
SVD . . . . . Singular Value Decomposition
CBOW . . . . . Continuous Bag-of-Words Model
SGNS . . . . . Continuous Skip-gram Model with Negative Sampling
DM . . . . . Distributed Memory Model of Paragraph Vectors
DBOW . . . . . Distributed Bag-of-Words Model of Paragraph Vectors
WSD . . . . . Word Sense Disambiguation
POS . . . . . Part of Speech Tag
CNN . . . . . Convolutional Neural Network
RNN . . . . . Recurrent Neural Network
LSTM . . . . . Long Short Term Memory

CHAPTER 1

INTRODUCTION

There have always been studies based on handling semantics in different fields, such as linguistics, psychology. The representations of language constructs are required to address the semantics in a computational manner. In this perspective, the main focus is on the computational representations of language constructs, specifically word and sense representations, in this dissertation.

In word representations, the main goal is to model the behavior of the word. Contexts, where the words occur, are informative to address this behavior. With this idea, earlier work depends on counting the statistics of the word and accompanied words, e.g. co-occurrence counts. However, the increasing number of data causes an obstacle in utilizing those approaches. Neural network approaches appear to tackle with the huge amount of data. As a result of this development, the underlying difference between these approaches becomes a question. In this direction, our work aims to find out the main factor governing word representations, which we propose as context. We are motivated from the study of Huang et al. (2012) in which they propose a generative model based on discourse (context) vectors and show that this model performs similarly to a count based approach, Pointwise Mutual Information (PMI). Distinctively, we point out that models involving context show the same behavior.

In sense representations, modeling the behavior of sense is more complicated as senses are embedded into words. The crucial influence of sense representations is that word representations for polysemous words refer to more than one meaning although the main aim of the word representation is to address the semantics of the word. Unfortunately, one representation is not enough to accomplish this aim for the polysemous word. Therefore, sense representations have the potential to solve this problem by creating a distinctive representation for each sense of the polysemous word. A disambiguation of words' sense is required to create a sense representation to extract the embedded meanings of the polysemous words. In Word Sense Disambiguation (WSD) studies, which sense of the polysemous word is utilized in a given context is attempted to be found. The solution approaches of WSD contain supervised, unsupervised, and knowledge/dictionary

based methodologies. While creating sense representations, the WSD methodologies are used to be able to disambiguate the polysemous words. Here, we propose an approach which brings into together the knowledge-based method of WSD and neural networks. Polysemous words are sense-labeled using the knowledge-based method and then, representations are created by neural networks using the sense-labeled data as the corpus. A similar approach is implemented by Iacobacci et al. (2015), distinctively they use another method to disambiguate and label senses of polysemous words.

Word representations are the most important component among semantic representations (e.g. phrase, paragraph, context, etc.), because other representations have been created utilizing word representations/information. In this perspective, determining the crucial factor of word representations has a direct influence on other semantic representations/semantics. Sense representations improve the word representations in which the representation of the polysemous words problem is handled. Thus, its impact on semantics is undeniable, also.

1.1. Contributions of Thesis

The contributions of our study is underlined in this section. The contributions of our work based on word embeddings can be listed as below:

- Context is the main factor governing word embeddings.
- The computational models based on context perform better on association tasks.
- The vector representations of words are improved on association tasks whenever the word-word relation is evaluated by conditional probability rather than cosine similarity.

The contributions of our study related to sense representations are explained in the following items:

- A sense embedding' method depending on the knowledge-based algorithm and neural networks is proposed.
- The sense-labeling algorithm has a crucial effect on the resultant sense representations. If the WSD algorithm can label the senses, correctly, the sense embeddings will perform at least as good as word embeddings.

1.2. Organization of Thesis

The dissertation is outlined as follows. Chapter 2 represents the background information about semantic representations. Chapter 3 covers related work in word representations and sense representations. Chapters 4 and 5 represent our approaches in detail. Chapter 6 contains the conclusion information about our studies with future work.

CHAPTER 2

SEMANTIC REPRESENTATIONS

Basically, semantics is the study of meaning in natural language. It is difficult to model semantics as language has ambiguity in its natural form. The challenge increases when considering incompleteness, imprecision, and temporal characteristics of a language. On the other hand, the encouraging idea is meaning is embedded in the language constructs. Therefore, semantics can be extracted from the usage of language (For more detail, please see (Sahlgren, 2006)).

The relation between various language elements (words-concepts-percepts-actions) builds up the semantic knowledge (Griffiths et al., 2007). Words-words relation analysis is popular among computational models as it is supported by the distributional hypothesis: "words with the similar distributions have similar meanings" (Harris, 1954) where the distribution means the neighbourhood of a word which is specified as a fixed size surrounding window.

The main idea in the distributional hypothesis is that similar words tend to have similar distributional properties. To illustrate, if two words, w_1 and w_2, frequently co-occur with the same word, w_3, then, the word, w_1, and the other word, w_2, carry a similar meaning, at least in one sense. Practically, this co-occurrence value is calculated using a sliding window in order to represent a context. The sliding window is assumed to capture the statistical properties of words. Then, the distribution of each word is extracted from the statistics of the neighbouring words.

Addressing the semantics in a computational manner requires the encoding of language constructs which can be called as a representation. Although there are various methodologies in order to extract representations, they are mainly derived from the use of language by applying the distributional hypothesis. These representations can refer to any level of the meaning. In natural language processing, these levels can be document, sentence, phrase, word, sense, etc. In this dissertation, the focus is on word representations and sense representations. We have taken into account a document or sentence information, while deriving word representations, only to be able to observe its effect on word representations.

2.1. Word Representations

Word representations can be categorized into two classes: Information-Theoretic Approaches and Word Embeddings. Although both categorizations inherently use statistical information, the interpretations of them make an essential difference.

2.1.1. Information-Theoretic Approaches

The critical issue on the information-theoretic approaches is a calculated statistical value corresponds to the two or more relevant words, e.g. a word and its neighbouring words in a context. Therefore, the resultant representations imply the association of the corresponding words. The representations may be interpreted as the representations of word-word relations/associations instead of a word representation. Co-occurrence counts and PMI values are such common metrics in computing associations between words.

2.1.1.1. Co-occurrence values

As the name suggests, the statistical information of co-occurrence values is the occurrence count of the word with the other. To find out the co-occurrence value between w_1 and w_2, the computation starts with the trace of the language corpus until w_1 is encountered. Then, w_1 is centralized in the sliding window according to window size and whether w_2 is in the window is investigated. If it is, the co-occurrence value is increased by 1. This process continues until there is no word remaining in the corpus. A computational load is dependent on the corpus size. The problematic issue is when the corpus size is increased, although the semantics met is increased, the computational load is increased, also.

Consider the following example (sentence from Samuel Beckett, (Turney and Pantel, 2010)):

”ever tried ever failed no matter try again fail again fail better”

window size: 2 and w_1 = again, w_2 = fail

[ever tried ever**]** failed no *matter try again fail again fail better*

[ever tried ever failed**]** no *matter try again fail again fail better*

[ever tried ever failed no**]** *matter try again fail again fail better*

ever **[**tried ever failed no matter**]** *try again fail again fail better*

ever tried **[**ever failed no matter try**]** *again fail again fail better*

ever tried ever **[**failed no matter try again**]** *fail again fail better*

ever tried ever failed **[**no matter try again fail**]** *again fail better*

ever tried ever failed no **[**matter try **again** $fail$ again**]** *fail better*

ever tried ever failed no matter **[**try again fail again fail**]** *better*

ever tried ever failed no matter try **[**again $fail$ **again** $fail$ better**]**

ever tried ever failed no matter try again **[**$fail$ **again** $fail$ better**]**

ever tried ever failed no matter try again $fail$ **[again** $fail$ better**]**

(again, fail) = 3

In this example, the window size is 2 and the co-occurrence values of the two words, $again$ and $fail$ are counted. The process starts with the first word in the corpus, $ever$, by taking its accompanied words in a given window size. Then, whether the taken word is the searched word is decided. If it is not, the window slides by centralizing the next word until the wanted word is found. At the eighth step, the word is found, $again$, and the second word, $fail$, is in the accompanied words, so the count is increased by 1 (at first, the count is initialized by 0). Additionally, at the tenth step, the word is encountered. In this time, there are two times of the second word, so the count is increased by 2. At the end, the count becomes 3.

2.1.1.2. Pointwise Mutual Information (PMI) values

The co-occurrence values of the frequent words tend to be high regardless of the company words. Undoubtedly, the two most frequent words co-occurrence value is higher although their semantic association could be lower. Therefore, frequency has a critical role in influencing semantic association/relation between words.

PMI (Church and Hanks, 1990) is a metric to quantify an association between words. PMI adjusts the effect of frequency in relating words. The general formula of PMI is as below:

$$PMI(x, y) = log \frac{P(x, y)}{P(x)P(y)} \tag{2.1}$$

where, $P(x, y)$ is the joint probability, $P(x)$ and $P(y)$ are the marginal probabilities. The joint probability measures x and y happen at the same time while marginal probabilities

measure the individual occurrence. Therefore, the probability of co-occurrences is normalized by the individual probabilities of occurrences. In word representations, the counts are utilized rather than the probability values, as in formula 2.2, where $\#(w_1, w_2)$ means a co-occurrence counts of w_1 and w_2, $\#(w_1)$ or $\#(w_2)$ represents a frequency counts of the relevant word and $|D|$ is the vocabulary size.

$$PMI(w_1, w_2) = log\frac{\#(w_1, w_2).|D|}{\#(w_1).\#(w_2)} \tag{2.2}$$

Therefore, the domination of the frequency can be faded away by normalizing co-occurrence values by the frequencies of the words.

2.1.2. Word Embeddings

Word embedding or word vector is, self-evidently, a computational model that represents a word as a vector. The underlying idea behind it depends on the geometrical interpretation of semantics. In this interpretation, each word is represented as a point in a semantic space. In this semantic space, similarities are evaluated based on the spatial proximity between words. This has an influence on conceptualizing similarities. In a well constructed semantic space, closer words correspond to similar words (Sahlgren, 2006).

2.1.2.1. One Hot Vectors

One hot word vectors are a kind of discrete representation. This vector model is one of the first attempts at a vector representation of a word in history. The derivation of vectors starts with sorting out all words in a vocabulary. Then each word is assigned a vector consisting of all $0s$ except for a 1 at its individual index. Therefore, vector size, $\mathbb{R}^{|V|*1}$, depends on the vocabulary size.

To illustrate: If the first word is *a* and the last word is *zebra* in the English vocabulary containing the word *language* in the middle, the representations should be as below.

Although there is no computational load to have one-hot vectors, the vectors are high-dimensional and sparse preventing it to be used in various NLP tasks. This is not enough to be able to use them in the semantic tasks, obviously, vectors do not carry any semantic knowledge.

$$w^{a} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ . \\ . \\ 0 \end{bmatrix}, \ldots, w^{language} = \begin{bmatrix} 0 \\ . \\ 1 \\ . \\ 0 \\ 0 \end{bmatrix}, \ldots, w^{zebra} = \begin{bmatrix} 0 \\ 0 \\ . \\ . \\ 0 \\ 1 \end{bmatrix}$$

2.1.2.2. Co-occurrence Vectors

The co-occurrence vector model concentrates directly on word-word co-occurrences. In a specified window, the co-occurrence values between a center word and the neighbouring words are calculated and then stored in a global matrix where rows denote vocabulary words, columns represent context words, and cells are their co-occurrence values. In this system, row vectors are interpreted as word vectors.

As an example;

”ever tried ever failed no matter try again fail again fail better”

Window Size: 2

	ever	tried	failed	no	matter	try	again	fail	better
ever	1	2	1	1	0	0	0	0	0
tried	2	0	1	0	0	0	0	0	0
failed	1	1	0	1	1	0	0	0	0
no	1	0	1	0	1	1	0	0	0
matter	0	0	1	1	0	1	1	0	0
try	0	0	0	1	1	0	1	1	0
again	0	0	0	0	1	1	2	3	1
fail	0	0	0	0	0	1	3	2	1
better	0	0	0	0	0	0	1	1	0

”ever” = [1 2 1 1 0 0 0 0 0]

”no” = [1 0 1 0 1 1 0 0 0]

In this example, for each word, the accompanied words are counted and stored in a matrix whose rows and columns are the vocabulary words. The accompanied words are limited by the window size, so the co-occurrence are calculated with this size, i.e. the

window size before and after words are taken into account as accompanied/neigbouring words. After all words' co-occurrence values are calculated, the row is taken as the word vector corresponding to a word.

While working with the real data, data size become an issue. A better model meeting semantics is the one that requires more data to construct. Furthermore, the construction and storage of a global matrix from real data add an extra computational load.

Even so, the actual problem with the model is the sparsity of the resultant matrix as one-hot vectors. Words tend to highly co-occur in some contexts, with some specific neighbouring words. Also, the resultant word vectors are so high dimensional that it influences their utility in NLP tasks, as one-hot vectors.

2.1.2.3. PMI Vectors

The PMI vectors are the same as co-occurrence vectors except that their cells contain PMI values rather than co-occurrence values. In the calculation of PMI matrix cells, as many word pairs' co-occurrence value is 0, taking the logarithm yields $-\infty$, (formula 2.2):

$$PMI(w_1, w_2) = log0 = -\infty$$

An alternative solution to this problem is to take 0 when $\#(w_1, w_2) = 0$. However, this results in a conflict with negative PMI values where some pairs of words could have small co-occurrence values although they have high frequency in a corpus (Levy and Goldberg, 2014). A Positive PMI values are used to solve this issue:

$$PPMI(w_1, w_2) = max(PMI(w_1, w_2), 0) \tag{2.3}$$

PPMI usage is stated intuitive in that humans can discriminate a positive association more easily than a negative one (Levy and Goldberg, 2014). Therefore, concentrating on only positive associations is in accordance with the human perception.

2.1.2.4. Singular Value Decomposition Based Vectors

Singular Value Decomposition (SVD) is a dimensionality reduction technique. SVD is used to solve the sparsity problem of embeddings derived from a global matrix.

Additionally, the resultant vectors will not be high-dimensional, anymore.

SVD factorizes a matrix M into three matrices: $M = UDV$ where U and V are orthonormal matrices (orthonormal matrix is a matrix whose rows are unit length and orthogonal to each other (Goldberg, 2017)) and D is a diagonal matrix where diagonal elements contain singular values in a decreasing order. To reduce dimensionality, the first d diagonal values of the matrix D are preserved and other diagonal values are replaced by 0. Similarly, all values of matrix U and V are replaced by 0 except for the first d rows of matrix U and d columns of the matrix V. Multiplying these reformed three matrices results in a reduced version of M (Goldberg, 2017), (Levy and Goldberg, 2014).

2.1.2.5. Learning Based Approaches

Although applying dimensionality reduction diminishes the problems related to the high-dimensionality of word embeddings, calculating and storing a huge global matrix stay as a concern. The underlying idea of using learning techniques in word embeddings is to eliminate the global matrix and learn the representations iteratively.

The real challenge of learning word representations comes from the difficulty of catching various features of each word. When using a global matrix, these features are represented by the context words and feature weights are calculated based on co-occurrence or PMI values. However, the majority of machine learning techniques is designed to make a prediction with the given features as input. Features are generally over-specified and incomplete so the specification of all features is not easy (Socher, 2014).

Deep learning, a branch of machine learning, has a crucial impact on this point as it can tackle to learn good features and give a final prediction. It is a kind of neural network whose computation is motivated by the human brain functionality. In general, the system attempts to manage both prediction and learning. A prediction is achieved through the transformation from input to the output. The neural network consists of nodes and edges building up layers which transform the input to the output. Learning is accomplished on edges, called as weights. In each iteration, the system learns more by taking into account the difference between the prediction and actual output.

The only difference of deep learning from a neural network is it has more layers. Conceptually, it is assumed that layers are able to meet complexity and abstraction, with its non-linear operations. Therefore, deep learning is more convenient than neural

networks for the NLP tasks as language is naturally complex and abstract.

Many deep learning architectures have been developed by modifying input-output transformation logic. In this thesis, Word2Vec and Doc2Vec architectures are analyzed, respectively.

Word2Vec (Mikolov et al., 2013), In this work, the main aim is to feed a huge dataset to the system, efficiently. Before this work, developed systems were able to work with a big data using deep learning however their complexity is arguable. Word2Vec depends on the observation that computational complexity results from the hidden layer in the model and proposes two architectures where the hidden layer is removed. In this time, the models may not be capable to present data as precisely as before, whereas they can be trained using much more data, efficiently.

- **CBOW:** The system takes the context words as an input and attempts to predict the middle word in the given context as an output, as seen in Figure 2.1. Here, all the context words share the projection/hidden layer, so all words in the context are projected into the same position by taking the average of context words' vectors. As the words' order does not affect the projection, architecture name was given continuous bag-of-words. The resultant word vectors are derived from the trained weights.

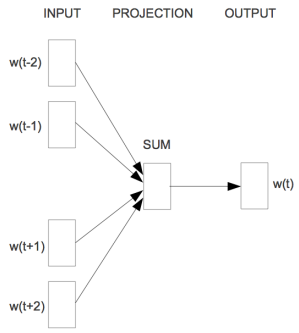

Figure 2.1. CBOW architecture

In Figure 2.1, each $w(t \pm n)$ denotes a context word and $w(t)$ denotes the middle word in a context. The input words and the output word are represented as one-hot vectors.

More detailed architecture is represented by Richard Socher (The detailed architecture is summarized from the Course Lecture (CS 224D: Deep Learning for NLP)

of Richard Socher, in Spring 2015.) in Figure 2.2. Before the training starts, the weight vectors are created, $W^1 \in \mathbb{R}^{n \times |V|}$ and $W^2 \in \mathbb{R}^{|V| \times n}$, where n is the dimensionality of produced word vectors and $|V|$ is the vocabulary size.

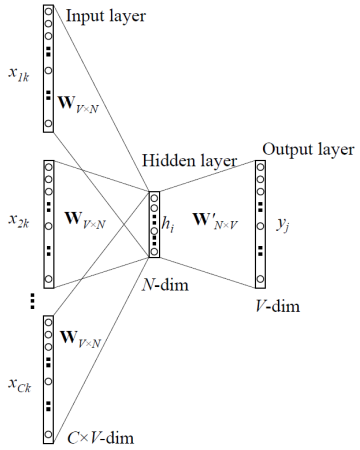

Figure 2.2. CBOW architecture in detail

In Figure 2.2, each x_s denotes a context word and y denotes the middle word in a context. The input words and the output word are represented as one-hot vectors.

The main goal is to learn the two matrices and they are used as word embeddings, at the end. Each column in W^1 represents one word as each row in W^2 with the dimensionality $n \times 1$. Then, the architecture pursues the following steps while training.

- First, one-hot vectors are derived for the words in a context and multiplied by the weight matrix, $u_{k-C} = W^1 w_{k-C}, .., u_k = W^1 w_k, .., u_{k+C} = W^1 w_{k+C}$.
- Average of these one-hot vectors are calculated in projection/hidden layer, $h = \frac{\sum_{k-C}^{k+C}}{2C} u_k$.
- Then, the score vector is created by computing $z = W^2 h$.
- The score should be turned into probability, $y' = softmax(z)$.

Softmax function is used in categorical distribution. Its formula is described as $softmax(z) = \frac{e^z}{\sum_{k=1}^{K} e_k^z}$.

The system prediction is y'. To be able to train the system, the difference between the prediction and actual output, named as a loss function, is necessary. Cross-

entropy is used as the loss function in this architecture. The formula of the cross-entropy is as follows:

$$H(y', y) = -\sum_{j=1}^{|V|} y_j log(y'_j) \tag{2.4}$$

When taking one instance into consideration, as $-y_j log(y'_j)$, y_j is 1 coming from one-hot vectors and if the system's prediction is completely true, then y'_j becomes 1. So, the loss becomes 0, $-1log(1) = 0$. If the prediction is not perfect, say 0.4, then the loss becomes 0.3979, $-1log(0.4) = -1(-0.3979) = 0.3979$. Therefore, it is appropriate to evaluate the loss.

Lastly, for training, log-likelihood is used as an objective function.

$$= -logP(w_i|w_{i-C}, ..., w_{i-1}, w_{i+1}, ..., w_{i+C})$$

$$= -logP(v_i|h)$$

$$= -log\frac{exp(v_i^T h)}{\sum_{j=1}^{|V|} exp(v_i^T u_j)}$$

$$= -v_i^T h + log\sum_{j=1}^{|V|} exp(v_i^T u_j)$$

- **Skip-gram:** The system is similar to CBOW except this one takes the middle word as an input and tries to predict or create context words as an output. Again, the resultant word embeddings are the weights of the architecture, Figure 2.3.

  In Figure 2.3, each $w(t)$ denotes the middle word and $w(t \pm n)$ denote the context words in a sentence. The input word and output words are represented as one-hot vectors.

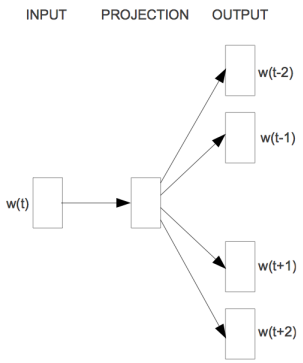

Figure 2.3. Skip-Gram Architecture

Again, the more detailed figure is represented by Richard Socher, Figure 2.4. The weights creation step is the same as CBOW.

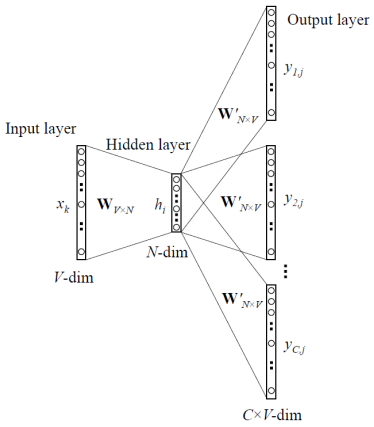

Figure 2.4. Skip-Gram Architecture in detail

In Figure 2.4, each x denotes the middle word and y_s denotes the context words in a sentence. The input word and output words are represented as one-hot vectors.

The architecture accomplishes the following steps while training.

- First, the one-hot vector is derived for the middle word in a context and the input word vector is multiplied by weight matrix, $u = W^1 w$.
- In this time, no averaging process in projection/hidden layer so the output is given as the input, $h = u$.
- Then, $2C$ items score vectors are created by computing $z = W^2 h$
- Next, the score should be turned into probability, $y' = softmax(z)$.

The system prediction is y', however, in this time, the prediction holds $2C$ items y' in terms of each context word. The loss function is cross-entropy, for this system, also. Again, log-likelihood is used as an objective function.

$$= -logP(w_{i-C}, ..., w_{i-1}, w_{i+1}, ..., w_{i+C}|w_i)$$

$$= -log \prod_{j=0, j\neq C}^{2C} P(w_{i-C+j}|w_i)$$

$$= -log \prod_{j=0, j\neq C}^{2C} P(v_{i-C+j}|u_i)$$

$$= -log \prod_{j=0, j \neq C}^{2C} \frac{exp(v_{i-C+j}^T h)}{\sum_{j=1}^{|V|} exp(v_k^T h)}$$

$$= - \sum_{j=0, j \neq C}^{2C} v_{i-C+j}^T h + 2C \sum_{k=1}^{|V|} exp(v_k^T h)$$

- **Skip-gram with Negative Sampling (SGNS):** (Mikolov et al., 2013) This model is the same as Skip-gram except for the objective function. The main aim of this model is to optimize the objective function because the summarization over $|V|$ takes a lot of computational time.

  The new objective function depends on maximizing $P(D = 1|w, c)$ where w, c is the word-context pair. This probability denotes the probability of (w, c) came from the corpus data. Additionally, $P(D = 0|w, c)$ should be maximized if (w,c) pair is not included in the corpus data. In this condition, (w, c) pair is randomly sampled, as the name suggests negative sampled.

  Firstly, the distribution is accomplished by the sigmoid function:

  $P(D = 1|w, c) = \frac{1}{1+e^{v_c^T v_w}}$

  The objective function is modeled as:

  $-log\sigma(v_{i-C+j} h) + \sum_{k=1}^{K} -log\sigma(v'_k h)$

Doc2Vec (Le and Mikolov, 2014) Their main goal is to create a representation for different levels: sentences, paragraphs or documents as well as words. Their architecture is quite similar to the Word2Vec except for the extension with a paragraph vector. They propose two architectures that are known as Distributed Memory Model of Paragraph Vectors (DM) and Distributed Bag-of-Words Model of Paragraph Vectors (DBOW).

- **DM - Paragraph Vectors** In this architecture, inputs are the words in a context except for the last word and a paragraph (sentence or document) containing this context, the output is the last word of the context. The word vectors and paragraph vector are concatenated while they are fed into the system. The paragraph vector can be interpreted as another word in a context.

  In Figure 2.5, a paragraph vector and the context word vectors are fed into the system. The last word is predicted by the architecture. Here, instead of the paragraph another level of context can be used, e.g. document. In this case, the context words are words in the document.

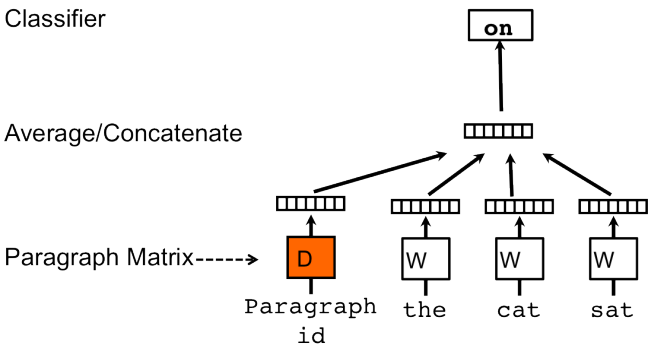

Figure 2.5. DM Architecture

- **DBOW - Paragraph Vectors** The input of the architecture is a paragraph vector. The model predicts the words randomly sampled from the given paragraph.

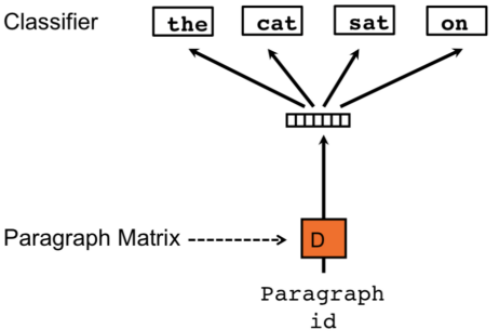

Figure 2.6. DBOW Achitecture

In Figure 2.6, a paragraph vector is fed into the system. The random context words are predicted by the architecture. The order of the predicted words are not important in this architecture. Again, instead of the paragraph another level of context can be used.

2.2. Sense Representations

In this section, the detailed information about sense representations is introduced. First of all, Word Sense Disambiguation is represented, and then, Sense Representations are analyzed.

2.2.1. Word Sense Disambiguation

There are words carrying more than one meaning/sense, known as polysemous words, in natural language. To illustrate, a word, $bank$, has various meanings as a noun or a verb, and some of them can be seen in the following sample sentences (from Miller (1995)):

"He cashed a check at the bank."

"He sat on the bank of the river."

"The pilot had to bank the aircraft."

For three sentences above, the meanings of $bank$ are "a financial institution", "a sloping land", and "tip laterally", respectively. It is impossible to distinguish the senses without the accompanied words of the polysemous word. When the first sentence is taken into consideration, without the accompanied word, especially, $cashed$ or $check$, a human being cannot give a prediction about a meaning of a word, $bank$, because the bank is a multi-sense word. However, the situation is not the same for monosemic words in which human being can easily predict the meaning of a word without any further information. Therefore, there is a strong relation between the sentence/context and senses of the polysemous words.

Word Sense Disambiguation (WSD) is the study of the senses of polysemous words, in a computational manner. By the definition, word sense disambiguation is a computational ability to identify the sense of a word in a context. As explained before, while fitting a model to word representations, the computational algorithms get help from accompanied words of the corresponding words, similar to senses. However, the challenging issue is that the senses are embedded into word although the word is apparent in a context. Therefore, the relation of senses cannot be seen as clearly as the relation of words.

As explained before, in section 2, the utilization of the word in a context is focused to address the semantics. On the other hand, when polysemous words are taken into account, one word can frequently occur in different contexts in terms of each sense. In this perspective, to be able to catch the context semantics, clearly, the identification of the senses of polysemous words has a crucial influence. Therefore, it can be said that the context and sense have a dual influence, in which the disambiguation of a word is necessary to address contextual semantics as the context is required to detect senses of a word.

2.2.1.1. Formalization of WSD

WSD can be interpreted as a classification task where the senses are the classes. The classical classification task contains classes which can be pre-determined and constant for every instance. For example, Part-of-Speech (POS) tagging can be considered as a classification task. POS tagging mainly attempts to discriminate the words in a context in terms of their part-of-speech, i.e. noun, verb, adjective, etc. The word's POS tag is defined in a consideration of its context, like WSD. However, the part-of-speeches are definite, not changing according to the word. However, in WSD, the classes (senses) change according to an instance (word) where for every word there are a different set of senses. Therefore, it cannot be interpreted as a classical classification task although ir is similar.

Another issue related to the WSD is that every time the disambiguation scenario can be changed due to the change of the usage of a specific word. Some words may start to carry more senses, whereas others may lose some senses or carry the same senses as before. The reason for this is because language lives with humans and its usage are changed, accordingly. As Wittgenstein said, "Language derives its meaning through use". Therefore, the senses of a word cannot always stay the same and cannot be determined to be used for every time.

Relating with the argument introduced above, the senses of a word cannot always be easily discretized in which each sense holds different meaning (Navigli, 2009). There may not be a clear edge where the usage of one sense is finished whereas the new sense usage is started. Also, the granularity of the meaning which the sense should hold can be changed according to the interpretation of a language. To illustrate (from (Navigli, 2009)),

"She chopped the vegetables with a chef's $knife$"

"A man was beaten and cut with a $knife$"

Both of the sentences above utilize the word, $knife$. In the first sentence the meaning of the word, $knife$, is $a\ tool$ whereas, in the second sentence, the meaning refers to $a\ weapon$ for the word, $knife$. Although they point to the same object, the usage of the object may cause the different senses. Whether these two meanings of the word, $knife$, should be discretized or not is arguable. The problem has not a clear answer. Actually, this problem is related to an ambiguity of a language. This ambiguity prevents the automatization of the discretization of the senses.

2.2.1.2. Solution Approaches

The various solutions have been developed to address the WSD problem. There is no dominated solution scheme, like word2vec, instead, all solution categories have some drawbacks and benefits. Mainly, the solution approaches can be categorized into three pieces, supervised, unsupervised, and dictionary/knowledge-based algorithm. Although a majority of studies fall into one of these approaches, some work combined them in a different way.

Supervised approaches need to have a training set, as always. The training set consists of labeled data which can change according to the problem. In WSD, the labeled data contains the polysemous word with its context where the polysemous word's intended sense is labeled. Here, various machine learning methodologies are tried. Their results are more successful than other types of solutions. However, the main problem with this approach is that the creation and maintenance of such a data creates a crucial obstacle regarding disambiguation scenario changes. Therefore, the data labeling is challenging in WSD, known as knowledge bottleneck.

In this point, unsupervised methods have a potentiality, because it only needs a raw data. In general, the solutions of these approaches are based on the clustering of words' contexts (cite). Although they can solve the data labeling problem, unsupervised methods cause to fall into a different problem, called $Word\ Sense\ Discrimination$.

In this manner, Word Sense Discrimination is the ability to identify word sense clusters. The main problem related to it is that the resultant clusters are required to be associated with the senses. Therefore, it can be interpreted as a subproblem in Word Sense Disambiguation like labeling the word senses.

There are some knowledge resources containing the information related with the senses, known as knowledge/dictionary resources. The most common resources are WordNet and BabelNet (Navigli and Ponzetto, 2012). WordNet consists of sets of synonyms, known as synsets, for each word. These synsets are interpreted as the set of word senses which hold the same meaning. For each synset, WordNet provides a gloss which contains the definition of the synset and sample sentences. Furthermore, WordNet provides a POS tag for each synset. For example, for a word, $bank$, some of the synsets are as in Table 2.1.

BabelNet utilizes some knowledge bases (WordNet, OmegaWiki, the English Wiktionary, Wikidata, FrameNet, VerbNet) and Wikipedia data and automatically link the

Wikipedia data with the knowledge resources. At the end, it provides sense-annotated data.

Table 2.1. Some synsets of a word $bank$ from WordNet

Sense	Definition and Samples
$bank^1$	(n) sloping land (especially the slope beside a body of water) ”they pulled the canoe up on the bank”; ”he sat on the bank of the river and watched the currents”
$bank^2$	(n) depository financial institution, banking concern, banking company (a financial institution that accepts deposits and channels the money into lending activities) ”he cashed a check at the bank”; ”that bank holds the mortgage on my home”
...	...
$bank^{11}$	(v) bank (tip laterally) ”the pilot had to bank the aircraft”

2.2.2. Sense Embeddings

Various word embedding models have been developed to represent words in a computational manner. In these representations, the main assumption is that a word vector should refer to a word's meaning. However, the condition changes when polysemous words are taken into account where a word has more than one meaning. In this condition, the polysemous word vectors contain the representations of multiple word senses. Therefore, there exist studies based on sense embeddings to have one vector per meaning/sense.

When compared with other semantic representations (e.g. word, phrase, context), sense representations have a distinctive challenge in which sense is not apparently observable in a text instead it should be derived from the word usage. Other semantic representations can easily be taken from the text, i.e. paragraph can be taken from the corpus without any extra effort.

The studies on sense embeddings have been increases after the word embeddings successfully captured word meanings with the huge amount of data using deep learning, i.e. word2vec. Because the senses are embedded into words, extracting sense vectors are challenging. Therefore, the tendency for the solution is to combine WSD and Deep Learning. However, some studies analyze the relation between word and senses where sense embeddings are derived from the word. The approaches are summarized in a detail in the following chapter.

Undoubtedly, polysemous word vectors cannot represent the meaning of a word properly as the most used sense is dominated no matter which methodology is used to derive the word embeddings. To address the semantics, different representation for each meaning is required. The sense embeddings have a crucial effect in this manner.

CHAPTER 3

RELATED WORK

This section covers the related work about our studies related to word representations and sense representations, respectively.

3.1. Semantic Interpretations of Word Representations

It has always been a controversial issue whether word representations are able to meet semantics or not. Increasing number of developed word representations leads to a new question: what is the difference between various representations in a semantic manner. In other words, which semantic features can be fulfilled by a specific representation. There has been some work in this direction, which is analyzed in the following paragraphs.

Some studies depend on the comparisons word representations on the association tasks (described below). First, Griffiths et al. (2007) propose a generative approach to find the probability distribution of word given context (they called context as a gist). In this method, they give importance on disambiguation and context of a word. First of all, topic distributions with a given document are found $P(z|g)$. Word distributions are extracted from these topic distributions $P(w|z)$, this step can be considered as a disambiguation step, as seen in Figure 3.1.

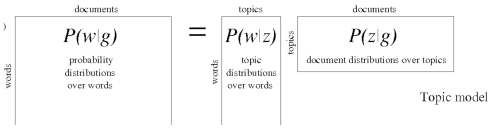

Figure 3.1. The Generative Approach Model

$$P(w|g) = P(w|z)P(z|g)$$

In Figure 3.1, g in the figure represents a gist/context, z denotes the topic distribution, and w is a word.

Then, they utilize their word representation to compare it with the vector representations in association tasks. Vector representations use cosine similarity, defined with the Formula 4.1, to estimate the association between two words when proposed method already provides two words' association score with the conditional probability value (i.e. $p(w|g)$). The main point underlined in the paper is that vector representations fall into geometric constraints when cosine similarity is used as a metric. This statement is grounded on two phenomena; triangle inequality and asymmetry:

- Triangle inequality is a metric axiom evaluating whether three points, x, y, z, in a space can create a triangle or not: for a triangle, any metric, d $d(x, z) \leq d(x, y) + d(y, z)$ should hold. The relation of this metric with NLP is whether three words create a triangle or not. Considering x, y, z as words, if x is similar to y and y is similar to z, then x must be similar to z. However, when words are taken into consideration, triangle inequality does not hold for words, where, *asteroid* is associated with *belt* and *belt* is associated with *buckle*, but *asteroid* is not associated with *buckle*. However, vector representations obey the triangle inequality by nature, where the association between two words are calculated by a cosine similarity between words, so this implies a high similarity value between *asteroid* and *buckle*.

- Asymmetry is another metric axiom and according to the definition (Tversky and Gati, 1982), any metric should be symmetric. However, again, it is not true for words, e.g. North Korea is more similar to China than China is to North Korea. Cosine similarity satisfies the symmetry, although it should not.

This paper is written in 2007 when the idea of neural network utilization appeared newly. Therefore, the only evaluation of the word vectors depends on cosine similarity.

Similarly, Nematzadeh et al. (2017) focus on the comparisons of word representations on association tasks, however, they include also word2vec and Glove (Pennington et al., 2014) word vectors to assess whether they fall into the geometric constraint or not (Glove is a neural network approach to represent words where the optimization function depends on the conditional probability of a word given context). They evaluate vectors on the association, triangle inequality, and asymmetry tasks. As a result, word2vec and GLove suffer from the difficulty of capturing the human judgments, like others. The only important observation is that GLove performs better compared to other models on asymmetry. The reason can be that GLove uses probability as an objective function.

The important point in this work is that asymmetry is evaluated using $p(w_1|w_2)$ and $p(w_2|w_1)$. For the calculation of these probabilities for vector representations, they use a reformulation of probability defined by Pennington et al. (2014) as:

$$p(w_2|w_1) = \frac{exp(w_2.w_1)}{\sum_{w_j} exp(w_j.w_1)}$$

Unlike the previous studies, the work Marco Baroni, Georgiana Dinu (2014) investigates word representations by separating them into two classes: count models (co-occurrence matrix, PMI matrix, SVD reduced matrices, etc.) and predictive models (learning based approaches). The literature consists of little work on the experimental comparison between count and predictive models. Therefore, their main aim is to provide many comparisons of models on various linguistic tasks.

They evaluate various models (PMI matrix, co-occurrence matrix, CBOW, SGNS, etc.) in different linguistic tasks (similarity, synonym, analogy, etc.) and report the best model for each task. Although they provide extensive experimental work, the theoretical reasoning behind the results is not comprehensive.

Other studies directly target the explanation of the relation between PMI and the neural approaches, especially SGNS. Firstly, Levy and Goldberg (2014) mainly advocate that SGNS is implicitly factorizing a global matrix in which each cell contains the PMI value. As seen before, SGNS trains the weights based on whether the word-context pair, (w, c), is included in a corpus or not. The weights of SGNS architecture hold two kinds of matrix information, words and contexts, W and C. In general, word matrix is used, however, when both matrices are taken into consideration, it could be seen, $W.C^T = M$ where each row represents a word $w \in V_w$, each column corresponds to a context $c \in V_c$ and each cell consists of $f(w, c)$ value. In this perspective, it is plausible to say that SGNS is able to factorize a matrix M as $|V_w|$ and $|V_c|$.

In addition to factorization, they present that if the objective function of SGNS is optimized, a matrix M will be the same as a global matrix whose cells contain a PMI value shifted by a constant $log(k)$, $f(w, c) = PMI(w, c) - log(k)$. They have two different experiments, first, they evaluate how well each method optimizes the objective. Second, they evaluate the performance of each algorithm on two linguistic tasks (similarity and analogy, described in the Section 4).

Although the experiment part cannot clearly support their ideas, the relation between SGNS and PMI matrix has a crucial effect on the analysis of word representations.

The huge data can be fed to the SGNS and it can converge in a relatively small computational time whereas creating a PMI matrix requires a high computational load. Therefore, in this perspective, the benefits of PMI matrix can be met by SGNS with a smaller computational load.

Melamud et al. (2017) are based on the previous study. They propose a new neural model depending on PMI matrix factorization. The proposed model is trained with the same way as SGNS except the prediction of the architecture is the PMI value rather than word-context presence. Their motivation comes from Levy and Goldberg (2014) where they formulate (w, c) as:

$$f(w,c) = pmi(w,c) - log(k) = log\frac{p(w|c)}{p(w)} - log(k) = log\frac{p(w|c)}{kp(w)}$$

They reformulate this as $p(w|c) \propto exp(\overrightarrow{w}.\overrightarrow{c}).p(w)$ and use this function as an objective function. In the adaptation of this formula to SGNS, c is taken as the context and w is the predicted word.

They do not directly analyze the word representations instead they combine two different techniques.

Arora et al. (2016a) mainly attempt to find the reason of the formula below:

$$f(w,c) \approx PMI(w,c) - log(k)$$

They found some missing points in the Levy and Goldberg (2014) explanation in which they use high-dimensional word embeddings besides that SGNS is a discriminative model rather than a generative one. However, in this paper, their explanation depends on the low dimensionality of word vectors.

They propose a generative model to explain the formula. The model consists of discourse vectors and word vectors (time-invariant). The model depends on the random walk over a discourse vector (c_t) as:

$$Pr[w\ emitted\ time\ t|c_t] \propto exp(c_t, v_w)$$

The crucial issue in this probability function is to take a discourse vector as a prior. They relate this prior to a Bayesian prior, by taking the partition function as $Z = \sum_w exp(v_w, c)$. This function is interpreted as the normalization.

In this manner, our motivation comes from the last paper in which they present the importance of discourse vectors. These vectors can be considered as the context vectors. We specifically concentrate on whether context can be the main factor on every model.

Particularly, in neural approaches, including the context as a vector can create the same effect as the model presented in the last paper.

3.2. Sense Embeddings

Word representations are utilized to provide semantic representation for words. In these representations, the polysemous words lead to an obstacle; although they carry more than one meaning, each polysemous word is represented by one vector. Sense embeddings attempt to tackle this problem by providing distinct vector representation for each sense of polysemous word. The methodologies to derive sense embeddings are discussed in this section.

Some studies depend on the clustering of the polysemous words' contexts to derive sense embeddings. The work of Reisinger and Mooney (2010) is one of the first attempts in this fashion. Their motivation is based on the idea that sense representations have a potential to solve the triangle inequality problem, introduced in Section 3.1, where words' similarities violate the triangle inequality whereas senses' similarities can obey as it includes the context information. Their method pursues the following steps:

- Occurrences of the polysemous word in a corpus are collected.
- For each occurrence, feature vectors (e.g. vectors from tf-idf matrix (Tf-idf matrix is the same as co-occurrence matrix except for the columns of tf-idf matrix represent context words and rows contain vocabulary words.)) are extracted.
- These feature vectors are clustered.

The cluster centroids are represented by prototype vectors (the average of the word vectors in the cluster). Clustering is accomplished according to similarities between these prototype vectors and feature vectors. To evaluate this similarity, they propose two context-included similarity measurements.

$$AvgSimC(w, w') = \frac{1}{K^2} \sum_{j=1}^{K} \sum_{k=1}^{K} d_{c,w,k} d_{c',w'} d(\pi_k(w), \pi_j(w')) \tag{3.1}$$

$$MaxSimC(w, w') = d(\pi'(w), \pi'(w')) \tag{3.2}$$

Both measurements are used for cluster assignments. In the equation 3.1, $d_{c,w,k} = d(v(c), \pi_k(w))$ is distance between feature vector of context, $v(c)$ and cluster centroid

$\pi_k(w)$; this distance evaluates the likelihood of context c belonging to cluster $\pi_k(w)$. Additionally, K denotes the number of cluster. In equation 3.2, $\pi'(w) = \pi_{argmax_{1 \leq k \leq K} d_{c,w,k}}(w)$ is the maximum likelihood cluster for w in context c. Here, each distance is evaluated by the cosine similarity between vectors.

Instead of using the feature vector, Huang et al. (2012) propose a neural network architecture to produce word vectors and context is represented according to these word vectors. In this architecture, local and global information is combined when the idea of word2vec newly appeared. The main goal of the model in Figure 3.2, is to predict the next word from the given sequence (referring local information) and document (referring global information) in which the sequence exists.

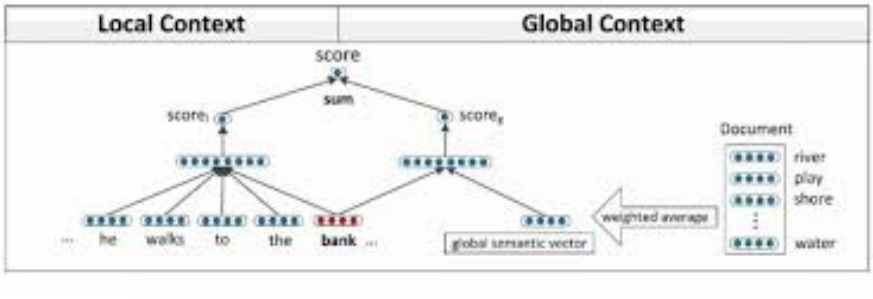

Figure 3.2. The model utilizes both local and global information

The objective function is in 3.3 where $g(s, d)$ and $g(s^w, d)$ are scoring functions in which s^w is a sequence s with the last word is replaced by the predicted word w. The main goal of the objective function is to minimize the difference between the actual sequence and the sequence whose last word is predicted by the model.

$$C_{s,d} = \sum_{w \in V} max(0, 1 - g(s, d) + g(s^w, d)) \tag{3.3}$$

The sense embeddings are captured utilizing the learned word representations like the method of Reisinger and Mooney (2010). This approach depends on the clustering of context-dependent feature vectors. As Reisinger and Mooney (2010), all occurrences of the polysemous words are collected and other steps are followed, however, in this time, word vectors from neural network are used, rather than feature vectors. For clustering, the spherical k-means is used. Furthermore, they propose a new dataset, SCWS, where two words are assessed in a sentential context by a human.

Chen et al. (2015) utilize the clustering approach in a different perspective. They propose an approach based on Convolutional Neural Network (CNN) (CNN is a kind

of Deep Neural Networks used for mostly classification tasks) architecture in which the input is the gloss sentence from the WordNet and the output is the predicted sense vector. The gloss sentence is represented by the concatenated word vectors which are trained by CBOW. The objective function of CNN is as below:

$$G_s = \sum_{s \in P} max\{0, 1 - f(s) + f(s')\}$$

Here, s denotes the gloss sentence and s' represents the constructed negative training sample sentence where some words in the gloss sentence s are randomly replaced. $f(.)$ is the scoring function representing the neural network without the last layer (softmax layer). They use the resultant sense embeddings to initialize the clustering algorithm. As a clustering method, they improved the model proposed by Neelakantan et al. (2015) in which the Skip-Gram model is extended by clustering word embeddings according to the contexts of the words. In this algorithm, the sense embeddings are initialized randomly and for all words, there are k senses/clusters. Instead, the current method uses the sense embeddings derived from CNN and the number of clusters, k, changes according to the word.

It can be said that in clustering based methods, sense embeddings are derived from the words. Arora et al. (2016b) utilize words again to have sense embeddings. However, in this time, the relation between words and senses is focused and they represent word by senses to lie in a linear superposition of the word embeddings. They are inspired from the analogy task in which the linear algebra can solve the problem. Firstly, they carry out an experiment to check whether the linear superposition idea works or not. In this experiment, they choose two random words (here, w_1 is more frequent than w_2), w_1 and w_2, and they attempt to create an artificial word w_{new} from the randomly selected words. Every occurrence of w_1 or w_2 are counted as the occurrence of the w_{new}. The embedding for w_{new} is derived using the same embedding method for w_1 and w_2. The result of repeating this experiment with many randomly selected word pairs is that the new embedding w_{new} is close to the subspace spanned by w_1 and w_2. After this experiment, they observe that the senses of the polysemous word reside in a linear superposition of word embeddings. For this reason, they attempt to solve the linear superposition problem in which the coefficients and sense vectors are unknown:

$$v_w = \sum_{j=1}^{m} \alpha_{w,j} A_j + n_w$$

where $\alpha_{w,1}$ ...$\alpha_{w,m}$ are coefficients, v_w is a polysemous word vector and A_j is the sense vectors while n_w is used as a noise vector. In this equation, there are two unknowns, coefficients and sense embeddings, $\alpha_{w,j}$ and A_j. k-SVD algorithm (Damnjanovic, Davies, and Plumbley, Damnjanovic et al.) is used to solve this optimization. The word vectors derived from the model based random-walk of Arora et al. (2016a) are used.

Sun et al. (2017) extract sense embeddings based on the idea of Arora et al. (2016b) in which the word embedding can be represented as a linear combination of its contexts. Here, the coefficient of the contexts is calculated by the normalized co-occurrence value as below:

$$W_{ij} = \frac{\#(w_i, w_j)}{\#(w_i)\#(w_j)}$$

The sense vector is calculated by the following formula:

$$u = \frac{1}{|S|} \sum_{w_j \in S} W_{ij} Cj$$

Here, S is the context and C_j is the word embedding (from word2vec) of the context word. For example, a sentence, S, is $I\ must\ stop\ by\ the\ bank\ for\ a\ quick\ withdrawal$. The sense vector of the word $bank$, u, is calculated by taking the weighted sum of the word embeddings of the context words.

Some other approaches depend on automatically creating sense-labeled data in which the words are disambiguated and the intended sense is placed into a corpus, then, the sense embeddings are extracted from the newly created corpus. Iacobacci et al. (2015) use a Babelfy algorithm (the algorithm takes the input text and constructs a subgraph of a semantic network representing the input text. Babelfy searches the intended sense using this subgraph) to label senses. First of all, the senses are disambiguated using BabelNet (Navigli and Ponzetto, 2012) as an underlying sense inventory and then Babelfy algorithm is used, which is known as an Entity Linking algorithm. Then the disambiguated/sense-tagged data are used as the raw data for CBOW. Therefore, the resultant vectors of CBOW are the sense vectors rather than word vectors. Furthermore, Trask et al. (2015) utilize CBOW with the labeled data. The distinctive property is that the data is not sense-labeled. Instead, it is labeled by the Part-of-Speech tagger or Named-entity-Recognition tagger (it is a word classification task where words are classified according to name-entities, e.g. person, location, organization, etc.). In this case, at least, words are disambiguated according to different features. Similarly, Chen et al. (2014) change the objective function

of the Skip-gram in which the main aim is to predict the context words from the middle word. Instead, the proposed objective function attempts to predict the context words' senses. To evaluate whether the predicted senses are true, they disambiguate the polysemous words, before the data are given to the system. In the disambiguation part, the most similar sense to the context is chosen as the intended sense. Here, the sense is represented as a vector using the gloss words of WordNet. All the words in the gloss are taken and the similarity values of these words with the polysemous word is calculated. The average of word vectors (the word vectors are initialized by the Skip-gram) which is higher than the threshold is assigned as the sense vector. Similarly, context vectors are calculated by the average of all context word vectors. Then, the most similar sense vector to the current context vector is chosen as the intended sense. Therefore, the Skip-gram with the new objective function is trained with a loss between the predicted sense and the previously disambiguated sense.

Alternatively, sense embeddings can be learned while training the system. Li et al. (2016) use neural network architecture. Specifically, they use a Recurrent Neural Network (RNN) (RNN is a kind of Deep Neural Networks used mostly in NLP as RNN is able to catch the word sequences). They trained their system utilizing sense definitions from WordNet. The main goal of the architecture is to predict the last word in the definition and the objective function is a cosine similarity between the predicted embedding and the actual word/sense embedding. In this comparison, the actual word/sense embeddings are derived from the initialization of them. They initialize the monosemous words with their word embedding extracted from Skip-Gram (trained on Google News dataset), explained in section 2.1. In the initialization of polysemous words, firstly, the synonym word which consists of only one sense is taken from WordNet and the word embedding of this one-sense synonym word is used as the sense embedding. However, every time the situation cannot hold because of the lack of one-sense synonym words. In this case, the most similar word in the sense definition of the polysemous word is taken and the sense embedding is initialized by this word' embedding. While selecting the most similar word, the highest similarity value should be greater than 0.2, otherwise, they initialize the sense embedding with the word embedding of the polysemous word. In another approach, Melamud et al. (2016) propose an architecture to learn context embeddings. However, their main goal is to address context embeddings and analyze them on WSD task rather than capturing sense embeddings, directly. The neural network is the same as CBOW except this architecture utilizes the bidirectional Long Short Term Memory (LSTM) (LSTM is a kind of Neural

Networks and capable of holding information for more time) rather than the average of word vectors. The extracted context embeddings are used to find out to correct sense in a supervised WSD scenario. Here, the sense whose context embedding is the most similar to the context embedding of the test instance is selected as the predicted sense.

A quite different approach is to find the analogy between WSD and a well-known problem and use the solution to this problem on WSD. Here, Li and Jurafsky (2015) define WSD as new sense vector should appear when evidence in a context suggests the current sense is distinctively different from others. In this perspective, the problem is similar to the Chinese Restaurant Process where a customer could either sit at one of the existing tables or choose a new table. Here, the customer is mapped to the current word while tables are interpreted as the senses. The decision of which table is chosen is based on the customers/words already sitting at the table. Actually, this algorithm can be interpreted as a kind of clustering in which a set of senses for each polysemous word w is denoted by $Z_w = z_w^1, z_w^2, ..., z_w^{|Z_w|}$ and each sense embedding is represented by e_w^z. The current word can be included in the existing cluster/context (according to semantic relatedness) or a new cluster is created based on the distribution below:

$$Pr(z_w = z) \propto \begin{cases} N_z^w P(e_w^z|context) & if\ z\ already\ exists, \\ \gamma P(e_w^z|z_{new}) & if\ z\ is\ new \end{cases} \tag{3.4}$$

where N_z^w is the number of times of w which is already assigned to z (the popularity of word senses) and γ is the probability value of evaluating the amount of difference.

The main goal of Panchenko (2016) is to make a link between sense embeddings and their senses in a dictionary, automatically. The method is interpreted as the bridge between the Word Sense Discrimination and Disambiguation (their difference is explained before). As a dictionary/sense-inventory, they used BabelNet and as a sense embedding algorithm they used Adagram (Bartunov et al., 2015) (Adagram can learn senses using a Bayesian nonparametric extension of the Skip-gram model). Their algorithm takes sense embeddings and a set of synsets as the input and gives an output of mapping between these two inputs. To be able to represent the synsets, they use the words in the synset words, glosses and even captions of images representing synset. Each word, in this set, is weighted according to frequency values. If the word has n synsets and m sense vectors, then $n * m$ similarities are calculated and a vector-synset pair is matched if the similarity value is higher than the threshold, t. The algorithm is summarized in the equation 3.5, below.

$$match(v_i, s_j) = \begin{cases} 1 & if\ sim(v_i, s_j) \geq t \\ 0 & otherwise \end{cases} \tag{3.5}$$

where $sim(v_i, s_j)$ is a similarity score (the similarity is either cosine or overlap) between a word, v_i, which is among the 200 most similar words of a sense vector and a word, s_j, from the set of synset words. They present one more matching algorithm in which the synset is disambiguated by Adagram method. The input to the algorithm is the set of words from synset interpreted as the context of the word. The assigned sense is taken as the matched sense if the similarity value is higher than the threshold, t.

CHAPTER 4

EVALUATIONS OF WORD REPRESENTATIONS

In this section, our theoretical explanation related to word representations and case study are explained.

4.1. Theoretical Explanation

Pointwise Mutual Information (PMI) is an information-theoretic metric, evaluating the association between words. The critical point in PMI is that it analyzes two words' relation by taking into the individual frequencies of words besides the co-occurrence values. It prevents the frequent words from having high association values. This prevention can be interpreted as the normalization.

$$PMI(w_1, w_2) = log\frac{\#(w_1, w_2).|D|}{\#(w_1).\#(w_2)}$$

The original formula of PMI is defined as:

$$PMI(w_1, w_2) = log\frac{p(w_1, w_2)}{p(w_1).p(w_2)}$$

Here, the probability of w_1 (or w_2) can be expanded as:

$$p(w_1) = \sum_{i}^{|V|} p(w_1, v_i)$$

Each word is assumed as a context word in the PMI. Therefore, we can consider w_1 or w_2 as a context word. In this case, $p(w_1) = p(c)$, the normalization in PMI is accomplished by a context. However, this context is limited by the word, w_1 (or w_2).

There has been some work to find out the relation between the vector representations and PMI values. When word vectors derived from neural approaches are taken into account, there is no normalization explicitly. Instead, the objective functions contain some normalization terms in training, in section 2.1. Therefore, it is wise to say the word vectors are normalized, implicitly. When taking an objective function of one neural approach, e.g. CBOW, we can recognize the normalization is accomplished depending on

the words which are accompanied with the middle word, as also explained by Arora et al. (2016a).

$$objective\ function = -logP(w_i|w_{i-C}, ..., w_{i-1}, w_{i+1}, ..., w_{i+C})$$
$$= -logP(w_i|h)$$
$$= -log\frac{exp(w_i h)}{\sum_{j=1}^{|V|} exp(w_j h)}$$

When objective function is analyzed, the denominator, $\sum_{j=1}^{|V|} exp(w_j h)$, can be considered as a context, $p(h)$. However, like PMI, this context is limited by the middle words, say the context of the middle word.

The impact of context in word2vec approach is not so differentiable because the context size (window size) is not allowed to be high due to computational efficiency. In order to be able to underline the effect of context in neural approaches, doc2vec is more preferable. doc2vec has a similar architecture with word2vec except it has one more term, document vector, to be fed into the system. The context, document vector, is not limited by a word in this architecture. Therefore, the context impact is more clearly differentiable. Also, we can analyze the effect of context with this architecture by constructing an analogy between a document vector and a context prior in the Bayesian formula like the generative model of Arora et al. (2016a).

In this perspective, we specifically attempt to interpret a context as a controlling factor. In other words, playing context size has a distinctive effect on word representations while we know the context is used for a normalization term.

We support our idea with our case study in which word representations including doc2vec with different document sizes (paragraph, document, etc.) are implemented.

4.2. A Case Study

This section presents our case study about word representations.

4.2.1. Introduction

There have been a number of tasks to evaluate semantics in natural language processing. In general, these tasks depend on the evaluation of the relation between two

words. In order to be able to evaluate two words, the numerical valuation denoting the relation between words is necessary. For information-theoretic approaches of word representations, it is not difficult since their representations already depend on two words. However, vector embeddings represent only one word so a metric to evaluate the relation of two words is required.

The cosine of the angle between the two word vectors/embeddings in a semantic space has been used in order to evaluate the relation of these two words, described by Landauer and Dumais (1997). It is called as cosine similarity and defined by the following formula:

$$cos(w_1, w_2) = \frac{w_1^T w_2}{||w_1||||w_2||} \tag{4.1}$$

Here, the numerator, $w_1^T w_2$, is the inner product of the word vectors and $||w1||$ is the norm, length of a vector. The norm is calculated by the formula of $\sqrt{w_T w}$. The similarity values are between -1 and 1, $[-1, 1]$. The similarity becomes 1 when the angle between the vectors is $0°$, and the similarity becomes -1 when the angle is $180°$. Therefore, it can be said that the cosine similarity measures the orientation. The relation between word similarity and the cosine similarity is defined by Landauer and Dumais (1997).

In addition to cosine similarity, conditional probability has started to be used to evaluate the relation of two words, after GLove (Pennington et al., 2014) is represented where the objective function is based on the conditional probability. The conditional probability calculation over vectors is accomplished using cosine similarity. However, in this case, the rate of the two words similarity over the total similarity of given word (conditioned word) with all other words in a vocabulary is taken as described by the following formula.

$$p(w_2|w_1) = \frac{cos(w_1, w_2)}{\sum_i cos(w_1, w_i)} \tag{4.2}$$

4.2.2. Experimental Setup

We have implemented various types of word representations. A text corpus is required to create a word representation. This section consists of a detailed explanation of the data that we have utilized to create word representations and our implementation details of the word representation models.

4.2.2.1. Data

We have used two datasets for all word representations to also evaluate whether dataset influences the representations or not, Wikipedia 2017 dump (the dump is taken in 2017 May) and Reuters data (Rose et al., 2002).

Wikipedia data are the compressed form of xml formatted file. To extract the text corpus from such data, gensim library (Řehůřek and Sojka, 2010) is used. The extracted file contains the title and the text for each title. Here, the title means each page in the Wikipedia.

Reuters data contain daily news information in a hierarchy. Each day folder is in a compressed form. For each day, there are folders of news in an xml formatted file.

The details of each dataset are given in the table 4.2.2.1 (the size of the datasets is the size of compressed form of them):

Table 4.1. The details of the datasets

Dataset	Size
Wikipedia	14.64 GB
Reuters	1,03 GB

Wikipedia data are used as the main data to create word representations whereas Reuters data are used due to its hierarchical structure. The hierarchy is required to evaluate the context impact in the form of context size to prove our idea introduced in the Section 4.1 (as a hierarchy, we have used a day, document, and paragraph information as the contexts). The hierarchical structure of Wikipedia is quite complicated to be able to utilize.

4.2.2.2. Implementation of Word Representations

We have implemented both information-theoretic approaches and vector representations. For the information-theoretic approaches, we have implemented Positive Pointwise Mutual Information (PPMI), a kind of PMI metric and explained in a detail in Section 2.1.2.3. For vector representations, we have implemented PPMI vector, SVD on the PPMI matrix, CBOW, SGNS, and Doc2Vec. The details of each model implementation are in the following paragraphs.

Positive Pointwise Mutual Information (PPMI) implementation: The formula of PPMI is introduced before, in the section 2.1.2.3. It can be interpreted as the value of PMI shifted by a constant of $log(k)$, where k is 1 as in the formula below:

$$ppmi(w_1, w_2) = max(pmi(w_1, w_2) - log(k), 0) \tag{4.3}$$

Here, the pmi values are shifted by $log(k)$ to improve the association metric as used by Levy and Goldberg (2014). The PPMI global matrix is implemented using gensim library. Here, we have played with the constant k, as 1, 5, and 15. When k is taken as 1, $log(k)$ becomes 0, as $log(1) = 0$. In this condition, we only take the Positive PMI values.

As an implementation detail, the vocabulary size is taken 80000 for the Wikipedia dataset. The window/context size is taken as 10 for both datasets. For each (w_1, w_2) pair in the vocabulary, the number of their cooccurrences is calculated, referred to $\#(w_1, w_2)$ in the formula of PMI 2.2. Then, each word's individual frequency is calculated, $\#(w_1)$. $|D|$ is 80000 for Wikipedia dataset in our experiment. The index words are stored for the creation of 3 PPMI global matrices for each dataset.

After all the implementations are completed, we have 6 PPMI global matrices; PPMI global matrix for $k = 1$ shift, PPMI global matrix for $k = 5$ shift, and one for $k = 15$ shift.

PPMI vector representations: For PPMI vector representations, the PPMI global matrix is used directly. In this matrix, each row corresponding a word is taken as the vector of that word. Therefore, no other computation is required except for deriving the row vectors as word vectors.

Vector representations based on SVD over PPMI global matrix: Singular Value Decomposition (SVD) is used to reduce the dimensionality of a global matrix. Here, as a global matrix, PPMI matrix is used. In our implementation, the gensim library is used, again. We have applied SVD to have word vectors having dimensions 100, 500, and 1000.

CBOW, SGNS: We have used the same vocabulary and window size as PPMI global matrix construction. The gensim library is used to calculate the CBOW and SGNS word vectors. The dimensionality of word vectors is as 100, 500, and 1000 for each model. Additionally, they are run with 8 threads to train the model. For SGNS, the negative values parameter is taken as 10.

Doc2vec: We have used the same vocabulary. The window size is taken as 8. The gensim library is used. The dimensionality of word vectors are, 100, 500, and 1000. For Reuters data, day, document, and paragraph information are taken as contexts given

to the system, separately. Therefore, 3 different models are created. For Wikipedia data document is used as the context. Here, the document refers to a page in the Wikipedia. For all contexts, the DBOW model is used as DBOW works more efficiently than the DM model.

4.2.3. Tasks

There has been a number of tasks to evaluate semantics in natural language processing. Word representations are used for the evaluation.

4.2.3.1. Similarity and Relatedness Task

There are some benchmarks that have been created by the human assessment of how much two words are similar to each other or how much they are related. Human subjects deliver a numerical rate denoting the degree of similarity/relatedness between two words. To evaluate two words' similarity with vector representations, cosine similarity is used.

- Datasets: We have used three different datasets, WordSim353 (2002), Rubenstein and Goodenough (rg) (1965), and Hassan and Mihalcea (mc) (2011). The format of the datasets is word, word, and the amount of similarity/relatedness assessed by human, as (the first few lines of wordsim353 similarity dataset) in Table 4.2:

Table 4.2. The first few lines of wordsim353 similarity dataset

tiger	cat	7.35
plane	car	5.77
train	car	6.31
television	radio	6.77
media	radio	7.42
bread	butter	6.19

rg and mc contain only similarity assessments while wordsim353 consists of both similarity and relatedness assessments in separate files. Wordsim353 has 203 word-pairs from the similarity task and 252 word-pairs for the relatedness task while rg contains 65 and mc holds 30 word-pairs.

- Evaluation: The similarity assessment performance of the computational model is evaluated by the correlation between human judgment and computational result. For each word-pair in the dataset, in information-theoretic approaches, e.g. PMI values, the values between two words are taken directly as a similarity/relatedness evaluation whereas for vector representations cosine similarity values between two words are taken. Then, the Spearman correlation between model predictions and human evaluations is calculated. Spearman correlation is a metric evaluating the strength and direction of the association between two lists of variables. In general, we have used it to measure the association between model predictions and human assessments. The results denote how close a computational model approximates human assessments. In this task, this approximation shows whether the computational model catch the similarity of two words or not.

4.2.3.2. Most Similars Task

It is said that a good computational model should predict the neighbouring words of a given word. Here, neighbouring words mean the words which are semantically similar to the given word.

- Dataset: There is no benchmark to evaluate this task. We randomly select one word among the most frequent words in a vocabulary and evaluate whether the most similar words represent meaningful neighbouring words or not. Here, our randomly selected word is $good$. Besides the neighbouring words, the similarity value of the neighbouring words to the selected word is evaluated.

- Evaluation: For each computational model, the most similar n words are derived for a selected word. In information-theoretic approaches, the row of a given word is derived from the global matrix and the words with the highest n values are taken as the neighbouring words. For vector representations, all cosine similarity values between a given word and all other words in the vocabulary are calculated and the words with the highest n values are taken. Then, the neighbouring words and the similarity values are analyzed. While evaluating, we determine n as 3. Therefore, the most similar 3 words and their similarity values are taken into account in our evaluations.

4.2.3.3. Analogy Task

The analogy task is based on assessing whether a computational model can catch semantic or syntactic analogies between words. There are four words in the analogy tasks and two pairs contain the same semantic/syntactic relation. The challenge is to find out the last word given the other three words. The model should first find the relation between the given word-pair, and then find a word having the same relation with the third word. To illustrate, the two word-pairs are "Brazil-real USA-dollar". "brazil real USA" are given to a computational model and the last word should be predicted as "dollar" by the model. Here, the semantic relation is currency. In addition to semantic relations, the syntactic relations are also included, e.g. "bird birds horse horses". The computational model is not required to find the relation, explicitly. To be succeeded in the task, the correct prediction of the last word is sufficient.

- Dataset: We have used a dataset developed by Mikolov et al. (2013). There are 14 categories in which 5 categories; common countries-capital, all countries-capital, currency, state-city, and family, are related with semantics and others, adjective-adverb, opposite, comparative, superlative, present-participle, nationality-adjective, past-tense, and plural, are syntactic categories. In the file, each line contains a relation separated by a space and each category is separated by a colon. There are 19558 lines in the file.

Table 4.3. A few lines from the dataset

: capital-common-countries			
Athens	Greece	Baghdad	Iraq
Athens	Greece	Bangkok	Thailand
Athens	Greece	Beijing	China
...			
: capital-world			
Abuja	Nigeria	Accra	Ghana
Abuja	Nigeria	Algiers	Algeria
Abuja	Nigeria	Amman	Jordan
...			

- Evaluation: Three words out of four are given to the computational model, as explained before. In vector representations, the last word is predicted as the formula defined by Mikolov et al. (2013):

$$w_4 = w_2 - w_1 + w_3$$

For example, with the given three words "brazil real USA", the vector, $real - brazil + USA$, is calculated. The most similar word of this vector is taken as a prediction. For the information-theoretic approaches, there is no described formula.

4.2.3.4. Association Task

This task relates the association to human memory. In the task, the human is asked to provide the first word that comes to his/her mind when one word is given. The given word is called as the cue and the response is the target. The computational model's performance is measured by the degree to which the prediction matches with human responses.

- Dataset: Nelson Norms Association (Nelson et al., 1998) is used as a dataset. In this dataset, there are 72176 cue-target pairs. There is more information than just association values. However, we have used cue-target word pair and their association value that is human assessed association value. Each line contains a cue-target pair and their evaluations separated by a comma.

- Evaluation: Spearman correlation is used to evaluate whether the responses of a computational model approximate human responses or not. For information-theoretic approaches, the values are association values in nature. For vector representations, we have used two types of evaluation. The first one is taking the cosine similarity between two words as an association value. The other one is to calculate the conditional probability between two words. The Spearman correlation value between model predictions and human evaluations denotes how well the model performs in this task.

4.2.3.5. Pun Task

Actually, this task is not a common NLP task. However, it is quite appropriate to test the word associations. A pun is a form of wordplay in which a word suggests two or

more meanings in a sentence. To illustrate, the sentence ”I used to be a banker, but I lost interest”, contains a pun which is located as the last word in the sentence, ”interest”. In this experiment, we have attempted to find the location of puns, known as the pun location task. As a solution scheme, we have used the solution proposed by N-Hance (Sevgili et al., 2017), where the solution depends on the distinctive word associated with the pun word. For example, in the given sample sentence the pun word ”interest” is associated with the ”banker”, with a distinctive meaning. Pun task consists of homographic and heterographic puns. If the two meanings of puns share the same pronunciation, puns are called as homographic puns, whereas if puns have similar but not the same sounding, they are named as heterographic puns. However, the solution scheme works for both types of pun. The reason to take the pun task into evaluation is that the selected solution is based on catching the distinctive word associations. Word association is an important task to evaluate both representations and the effect of the context over them.

- Dataset: The dataset, SemEval2017-Task 7, is delivered by Miller et al. (2017). There are 2250 sentences containing homographic puns, 1780 sentences containing heterographic puns in the dataset. The format of the file is xml consisting of tokenized words and punctuations in a sentence for both homographic puns and heterographic puns. Each word and sentence are given unique ids as an attribute. The id of predicted word is delivered as the location of the pun.

- Evaluation: The evaluation is based on whether a computational model can predict the pun word id correctly or not. For prediction, a model should find out the distinctive word association, according to the solution scheme, explained above. For information-theoretic approaches, the pairwise values are taken and these values are ranked. The word pair with the highest value is selected and the second word in the selected pair is predicted as a pun. The reason to choose the second word as a pun is the fact that the pun is located towards the end of the sentence, in general. For the word vectors, the cosine similarity values and conditional probabilities are used to find out the distinctive word associations. Again, all pairwise cosine similarity/conditional probability values are calculated and ranked. The word pair with the highest value is selected and the second word in the selected pair is predicted as the pun as before. The results from the cosine similarity and conditional probability are evaluated, separately. Evaluation is based on the accuracy value of predicted pun locations.

4.2.3.6. Asymmetry Task

As explained before, words are asymmetric to each other according to (Nematzadeh et al., 2017), e.g. China-North Korea relation in which North Korea is more similar to China than China is to North Korea. Whether a computational model can catch the asymmetry between words is evaluated using conditional probability. The asymmetry between $w1$ and $w2$ is defined as (Nematzadeh et al., 2017):

$$\frac{p(w_2|w_1)}{p(w_1|w_2)} \tag{4.4}$$

Here, the ratio of conditional probabilities calculated by the formula 4.2 is used as the asymmetry measure between w_1 and w_2.

- Dataset: As a dataset, Nelson Norms Association is utilized, again. However, in this case, we select those cue-target pairs for which target-cue pairs exist in order to be able to calculate the above ratio. In this case, there are 16772 cue-target pairs that have their reverses. The filtering is required in order to be able to apply the Formula 4.4.

- Evaluation: Firstly, for each computational model the conditional probabilities are calculated, $p(w_2|w_1)$. The information-theoretic approaches deliver symmetric words. Therefore, a conditional probability calculation is necessary. In order to calculate the conditional probability value for the information-theoretic approaches, the row of a word is taken, w_1, from the global PMI matrix. Then, the ratio of pmi value of the words ,$w_1 and w_2$, over the sum of all the pmi values in this row is taken as the conditional probability. To illustrate, for pmi values, the calculation is as below:

  $$p(w_2|w_1) = \frac{pmi(w_1, w_2)}{\sum_i pmi(w_1, w_i)}$$

  The conditional probability for vector representations is described before as in Formula 4.2. After the conditional probability calculations, the asymmetry values are derived using the ratio, as described in Formula 4.4. The same asymmetry calculation is accomplished for the cue-target pairs in Nelson Norms Association. Then, the Spearman correlation values between these asymmetry values and the values predicted by the computational model are calculated. This correlation gives the asymmetry performance of the computational model.

4.2.4. Results

Context has the normalization impact on word representation for both the information-theoretic approach, PMI, and neural approaches, word2vec or doc2vec, as explained in section 4.1. Here, we have analyzed this influence on different tasks with various word representations where we play with the document sizes, also. We have two datasets, Wikipedia and Reuters, to derive word representations. Wikipedia data are used as the main data while Reuters data are used to evaluate the document granularity where the hierarchy exists (the documents of a day are combined in a file and in the documents, there are paragraphs. Therefore, we have three context levels; paragraph, document, and day). In Wikipedia, such a hierarchy is complicated to be utilized.

4.2.4.1. Similarity and Relatedness Task Results

The Spearman Correlation values between the human assessments' similarity values and word representations' cosine similarity values are evaluated.

Table 4.4. Wikipedia-Similarity and Relatedness Spearman Correlation Results on WordSim353 dataset

	d=100	d=500	d=1000	d=100	d=500	d=1000
wikipedia	wordsim353-r			wordsim353-s		
CBOW	**0.629**	0.614	0.585	**0.751**	**0.755**	0.753
SGNS	0.557	0.558	0.556	0.622	0.599	0.576
doc2vec document	0.551	0.571	0.596	0.713	0.752	0.747
SPPMI (k=1)-value	0.570			0.702		
SPPMI (k=5)-value	0.126 (cov = 83/252)			0.396 (cov = 84/203)		
SPPMI (k=15)-value	0.049 (cov = 31/252)			0.237 (cov = 43/203)		
SPPMI (k=1)-vector	0.524			0.641		
SPPMI (k=5)-vector	0.625			0.709		
SPPMI (k=15)-vector	0.563			0.703		
SVD-PPMI (k=1)	0.574	**0.667**	**0.677**	0.677	0.748	**0.759**
SVD-PPMI (k=5)	0.557	0.625	0.575	0.662	0.716	0.727
SVD-PPMI (k=15)	0.564	0.549	0.418	0.625	0.634	0.618

In Table 4.4 (and in all other tables), k denotes the shift parameter in the Formula 4.3 and cov denotes the coverage. The table displays the correlation values. Here, wordsim353-r represents WordSim353-relatedness where human evaluate two words in terms of their relatedness and wordsim352-s represents WordSim353-similarity where human evaluate word-pairs in terms of their similarity. CBOW and SVD-PPMI shifted by 1 get best results in this dataset. Also, when we compare the results of SPPMI values shifted by 1 and SPPMI values shifted by 5, there is a gap. This results from the shifting. The SPPMI values hold small values and when they are shifted, they lose the competitiveness.

Table 4.5. Wikipedia-Similarity and Relatedness Spearman Correlation Results on rg and mc datasets

	d=100	d=500	d=1000	d=100	d=500	d=1000
wikipedia	rg			mc		
CBOW	0.736	0.785	0.812	0.723	0.797	0.752
SGNS	0.625	0.709	0.679	0.577	0.626	0.645
doc2vec document	0.697	0.754	0.769	0.699	0.755	0.769
SPPMI (k=1)-value	**0.796**			**0.785**		
SPPMI (k=5)-value	0.541 (cov = 28/65)			0.440 (cov = 14/30)		
SPPMI (k=15)-value	0.517 (cov = 12/65)			0.048 (cov = 8/30)		
SPPMI (k=1)-vector	0.524			0.592		
SPPMI (k=5)-vector	0.742			0.726		
SPPMI (k=15)-vector	0.718			0.724		
SVD-PPMI (k=1)	0.746	**0.822**	**0.841**	0.748	**0.864**	**0.865**
SVD-PPMI (k=5)	0.686	0.758	0.797	0.694	0.822	0.847
SVD-PPMI (k=15)	0.638	0.771	0.684	0.639	0.773	0.731

Table 4.5 represents correlation results rg and mc, and word representations. The CBOW values, again, are among the highest values although the SPPMI value shifted by $k = 1$ and SVD-PPMI shifted by $k = 1$ give the best results.

In the word representations derived from Reuters data, CBOW dominates again for all datasets, with the correlation values between 0.68 and 0.75. The SPPMI values and vector values, also, follow CBOW. In the analysis of the doc2vec values in Tables 4.6

Table 4.6. Reuters-Similarity and Relatedness Tasks Spearman Correlation Results on WordSim353 dataset

	d=100	d=500	d=1000	d=100	d=500	d=1000
reuters	wordsim353-r			wordsim353-s		
doc2vec paragraph	0.4515	0.4921	0.4865	0.5645	0.6248	0.6240
doc2vec document	0.4525	0.4833	0.4640	0.5750	0.6238	0.6229
doc2vec day	0.2558	0.2012	0.1945	0.3033	0.2876	0.2851

Table 4.7. Reuters-Similarity and Relatedness Tasks Spearman Correlation Results on rg and mc datasets

	d=100	d=500	d=1000	d=100	d=500	d=1000
reuters	rg			mc		
doc2vec paragraph	0.4500	0.4591	0.4481	0.4628	0.4682	0.4575
doc2vec document	0.4104	0.4886	0.4549	0.5029	0.5296	.4384
doc2vec day	0.1837	0.1605	0.1784	0.4170	0.4173	0.4333

and 4.7, when the context size is smaller, the results are increased, in general. CBOW can be interpreted as the kind of the doc2vec because the context is fed into the system in CBOW. In this consideration, while the context size decreased, the results are better with the context granularity order doc2vec-day, doc2vec-document, doc2vec-paragraph, and CBOW.

4.2.4.2. Pun Task Results

In this task, the evaluation is to determine the pun location in a sentence. The accuracy values are calculated between the word representations' predictions and actual values. Homographic and heterographic types of puns are evaluated, separately, as in the given dataset.

Table 4.8 shows the accuracy values of pun location where the word representations attempt to predict the pun word in a pun containing sentence. This prediction depends on the cosine values between word pairs in a sentence, as in Formula 4.1. The best values come from the SPPMI vectors shifted by $k = 15$ for both homographic and

heterographic puns. However, we cannot observe the effect of context in either CBOW or doc2vec.

Table 4.8. Wikipedia-The Accuracy Values of Pun Location

	d=100	d=500	d=1000	d=100	d=500	d=1000
wikipedia	homographic			heterographic		
CBOW	0.3043	0.3120	0.3251	0.2723	0.2905	0.2851
SGNS	0.2304	0.2689	0.2797	0.2140	0.2386	0.2386
doc2vec document	0.2234	0.2165	0.2080	0.2022	0.2131	0.1840
SPPMI (k=1)-value	0.3413			0.2942		
SPPMI (k=5)-value	0.2989			0.2550		
SPPMI (k=15)-value	0.2034			0.2122		
SPPMI (k=1)-vector	0.2219			0.1530		
SPPMI (k=5)-vector	0.3975			0.3033		
SPPMI (k=15)-vector	**0.4276**			**0.3242**		
SVD-PPMI (k=1)	0.2773	0.2866	0.2928	0.2404	0.2368	0.2395
SVD-PPMI (k=5)	0.3259	0.3197	0.3320	0.2577	0.2668	0.2668
SVD-PPMI (k=15)	0.3621	0.3744	0.3528	0.2851	0.2969	0.2969

In Reuters data, Table 4.9, the best results come from representations of doc2vec model, however, this can be related to the data size. While the context size is increased, the values are increased for both homographic and heterographic puns. Also, when dimensionality is increased, the values become greater, except for representations from doc2vec - document.

Table 4.9. Reuters-The Accuracy Values of Pun Location

	d=100	d=500	d=1000	d=100	d=500	d=1000
reuters	homographic			heterographic		
doc2vec paragraph	0.2180	0.2126	0.2350	0.1757	0.2149	0.2596
doc2vec document	0.2350	0.2095	0.2173	0.1821	0.2350	0.2641
doc2vec day	0.2927	0.3436	0.3413	0.1985	0.2322	0.2304

Table 4.10. Wikipedia-The Accuracy Values of Pun Location V2

	d=100	d=500	d=1000	d=100	d=500	d=1000
wikipedia	homographic			heterographic		
CBOW	0.5686	0.5686	0.5678	0.4335	0.4335	0.4335
SGNS	0.5686	0.5686	0.5678	0.4326	0.4326	0.4335
doc2vec document	**0.5948**	**0.5932**	**0.5948**	**0.5401**	**0.5392**	**0.5401**
SPPMI (k=1)-value	0.3413			0.2942		
SPPMI (k=5)-value	0.2989			0.2550		
SPPMI (k=15)-value	0.2034			0.2122		
SPPMI (k=1)-vector	0.5686			0.4335		
SPPMI (k=5)-vector	0.5693			0.4335		
SPPMI (k=15)-vector	0.5678			0.4335		
SVD-PPMI (k=1)	0.5686	0.5693	0.5678	0.4326	0.4326	0.4335
SVD-PPMI (k=5)	0.5693	0.5701	0.5693	0.4326	0.4335	0.4326
SVD-PPMI (k=15)	0.5678	0.5693	0.5686	0.4326	0.4335	0.4335

In Table 4.10, the representation is evaluated according to their conditional probability values, as in Formula 4.2, rather than cosine similarity. As the solution of pun location is focused on the words' association values, the accuracy values are better than the previous evaluation for both homographic and heterographic puns. Here, we can distinctively observe the context effect as the representations based on context give the highest scores.

Table 4.11. Reuters-The Accuracy Values of Pun Location V2

	d=100	d=500	d=1000	d=100	d=500	d=1000
reuters	homographic			heterographic		
doc2vec paragraph	0.5817	0.5810	0.5817	0.4481	0.4472	0.4472
doc2vec document	0.5801	0.5801	0.5809	0.4481	0.4472	0.4472
doc2vec day	0.5847	0.5847	0.5847	0.4472	0.4481	0.4481

The best values, again, are from doc2vec in Reuters data. The values are nearly the same between different context sized vectors in table 4.11.

4.2.4.3. Most Similars Task Results

We mainly focus on the most similarity values of a randomly selected word, $good$, among the most frequent words instead of the most similar words.

Table 4.12. Wikipedia-Most Similar Words and Values

wikipedia	most similars of $good$: similarity rates
CBOW	d = 100 - 'excellent': 0.6502; 'perfect', 0.6399; 'decent': 0.6228 d = 500 - 'bad': 0.4478; 'excellent', 0.4419; 'decent': 0.4252 d = 1000 - 'bad': 0.3964; 'excellent': 0.3425; 'decent': 0.3296
SGNS	d = 100 - 'st': 0.6335; 'sorrows': 0.5856; 'faithful', 0.5825 d = 500 - 'excellent': 0.3487; 'healy': 0.3201; 'christ': 0.3164 d = 1000 - 'hope': 0.2779; 'galway': 0.2447; 'mary': 0.2423
doc2vec document	d = 100 - 'always': **0.8354**; 'too': 0.7967; 'so': 0.7901 d = 500 - 'bad': 0.6325; 'better': 0.6096; 'what': 0.5899 d = 1000 - 'bad': 0.5482; 'better': 0.5470; 'excellent': 0.5042
SPPMI (k=1)-value	'Good': 0.0567; 'natured': 0.0481; 't9cnicos': 0.0462
SPPMI (k=5)-value	'Good': 0.2740; 'natured': 0.2167; 't9cnicos': 0.2037
SPPMI (k=15)-value	'Good': 0.5443; 'natured': 0.3895; 't9cnicos': 0.3543
SPPMI (k=1)-vector	'something': 0.5775; 'you': 0.5711; 'things': 0.5703
SPPMI (k=5)-vector	'bad': 0.1867; 'Good': 0.1849;'guys': 0.1293
SPPMI (k=15)-vector	'followed': 0.2285; 'guys': 0.1841; 'bad', 0.1566
SVD-PPMI (k=1)	d = 100 - 'proud': 0.8250; 'honest': 0.8004; 'sure': 0.7950 d = 500 - 'your': 0.5422; 'you': 0.5214; 'thing': 0.5001 d = 1000 - 'excellent': 0.4893; 'thing': 0.4158; 'things': 0.4073
SVD-PPMI (k=5)	d = 100 - 'kind', 0.7713; 'think', 0.7684; 'thing', 0.7564 d = 500 - 'guys': 0.5703; 'you': 0.5699; 'thing': 0.5637 d = 1000 - 'thing': 0.4828; 'pretty': 0.4736; 'kind': 0.4551
SVD-PPMI (k=15)	d = 100 - 'guys': 0.6338; 'wouldn': 0.5225; 'fool': 0.5146 d= 500 - 'luck': 0.4726; 'cnicos': 0.4089; 'guys': 0.3991 d = 1000 - 'cnicos': 0.4743; 'rudos': 0.4579; 'guys': 0.4530

Table 4.12 displays the most similar 3 words and the similarity values of the word $good$. The highest similarity values are delivered by doc2vec. For each word embedding, when dimensionality is increased, word vectors become distant to each other. Thus, the similarity values between them are decreased. The similarity values are cosine similarity.

The highest similarity value from the word representations of the Reuters data comes from the SVD-PPMI shifted by $k = 5$ with the dimensionality is 100 (giving a similarity value of 0.9906). When the granularity is increased, the values are increased.

Table 4.13. Reuters-Most Similar Words and Values

reuters	most similars of 'good': similarity rates
doc2vec paragraph	d=100 - 'strong': 0.8431; 'better': 0.8381; 'solid': 0.8334 d=500 - 'excellent': 0.6751; 'strong': 0.6362; 'better', 0.6358 d=1000 - 'excellent': 0.5203; 'better': 0.5091; 'strong': 0.4921
doc2vec document	d=100 - 'very': 0.8006; 'great': 0.7761; 'better': 0.7621 d=500 - 'excellent': 0.6831; 'better': 0.6541; 'strong': 0.6313 d=1000 - 'excellent': 0.5707; 'better': 0.5510; 'very': 0.5163
doc2vec day	d=100 - 'great': 0.9227; 'very': 0.9099; 'tremendous': 0.8955 d=500 -'great': 0.8434; 'strange': 0.8178; 'fantastic': 0.8100 d=1000 -'great': 0.8491; 'strange': 0.8157; 'fantastic': 0.8119

The result is the same as in Table 4.12, when CBOW and doc2vec are compared.

4.2.4.4. Analogy Task Results

The accuracy values between the predicted word from word representations and the actual word are evaluated. There is no accepted solution scheme for SPPMI values.

Table 4.14. Wikipedia-The Accuracy Values of Analogy Task

	d=100	d=500	d=1000
wikipedia	analogy tasks		
CBOW	**0.5602**	0.6093	0.5387
SGNS	0.2283	0.3472	0.3558
doc2vec document	0.5053	**0.6552**	**0.6589**
SPPMI (k=1)-value	*		
SPPMI (k=5)-value	*		
SPPMI (k=15)-value	*		
SPPMI (k=1)-vector	0.4776		
SPPMI (k=5)-vector	0.405		
SPPMI (k=15)-vector	0.3093		
SVD-PPMI (k=1)	0.3150	0.4810	0.5388
SVD-PPMI (k=5)	0.1644	0.2698	0.3040
SVD-PPMI (k=15)	0.0989	0.1665	*

In Table 4.14, the representations form doc2vec and CBOW dominate with their results. The vectors from SVD-PPMI shifted by $k = 1$ model follow them.

Table 4.15. Reuters-The Accuracy Values of Analogy Task

	d=100	d=500	d=1000
reuters	analogy tasks		
doc2vec paragraph	0.1312	0.1456	0.1250
doc2vec document	0.1120	0.1498	0.1295
doc2vec day	0.0138	0.0141	0.0139

In Reuters data, CBOW gets the best results with the values between 0.5 and 0.5937. When context size decreases, accuracy values increase, in Table 4.15. The CBOW confirms this while considering it as a kind of doc2vec with the smaller context size.

4.2.4.5. Association Task Results

The Spearman Correlation values between the cosine similarity/conditional probability values from word representations and human assessments are taken as the evaluation criteria in this task.

Table 4.16. Wikipedia-The Spearman Correlation Results of Association Task

	d=100	d=500	d=1000
wikipedia	association task		
CBOW	0.2031	0.2173	0.2149
SGNS	0.1558	0.1783	0.1833
doc2vec document	0.1755	0.2037	0.2120
SPPMI (k=1)-value	**0.2148**		
SPPMI (k=5)-value	0.1369		
SPPMI (k=15)-value	0.0980		
SPPMI (k=1)-vector	0.1657		
SPPMI (k=5)-vector	0.1935		
SPPMI (k=15)-vector	0.1880		
SVD-PPMI (k=1)	0.1764	**0.2251**	**0.2486**
SVD-PPMI (k=5)	0.1535	0.2020	0.2207
SVD-PPMI (k=15)	0.1344	0.1756	0.1888

SPPMI shifted by $k = 1$ value and its vectors get the best results in the association tasks, as shown in Table 4.16. The word-pairs' scores are the cosine similarity values in word vectors.

Table 4.17. Reuters-The Spearman Correlation Results of Association Task

	d=100	d=500	d=1000
reuters	association task		
doc2vec paragraph	0.1421	0.1762	0.1803
doc2vec document	0.1396	0.1597	0.1642
doc2vec day	0.0341	0.0297	0.0298

The best results in Reuters data are delivered by CBOW with the correlation values from 0.2031 to 2074. Table 4.17 represents that the smaller context sized word vectors get better results.

Table 4.18. Wikipedia-The Spearman Correlation Results of Association Task V2

	d=100	d=500	d=1000
wikipedia	association task		
CBOW	**0.2427**	0.2330	0.2222
SGNS	0.2136	0.2086	0.2024
doc2vec document	0.2269	0.2304	0.2279
SPPMI (k=1)-value	0.2148		
SPPMI (k=5)-value	0.1369		
SPPMI (k=15)-value	0.0980		
SPPMI (k=1)-vector	0.2154		
SPPMI (k=5)-vector	0.2101		
SPPMI (k=15)-vector	0.2015		
SVD-PPMI (k=1)	0.2326	**0.2403**	**0.2389**
SVD-PPMI (k=5)	0.2193	0.2330	0.2348
SVD-PPMI (k=15)	0.2076	0.2212	0.2240

When the word-pairs are evaluated according to conditional probability value rather than cosine similarity, the correlation values of CBOW and SVD-PPMI shifted by $k = 1$ are the highest ones, as seen in Table 4.18.

Table 4.19. Reuters-The Spearman Correlation Results of Association Task V2

	d=100	d=500	d=1000
reuters	association task		
doc2vec paragraph	0.2170	0.2158	0.2071
doc2vec document	0.2160	0.2139	0.2075
doc2vec day	0.1779	0.1763	0.1765

When the context size is smaller, the results are better in this version, Table 4.19 the same as in the previous version Table 4.17. The doc2vec paragraph gets the best results in this version.

4.2.4.6. Asymmetry Task Results

In this task, the evaluation is based on Spearman Correlation between word representations and human evaluation based on the ratio between (w_1, w_2) and (w_2, w_1) for all pairs.

Table 4.20. Wikipedia-The Spearman Correlation Values of Asymmetry Task

	d=100	d=500	d=1000
wikipedia			
CBOW	0.5226	0.5227	0.5238
SGNS	0.5257	0.5256	0.5261
doc2vec document	**0.5336**	**0.5341**	**0.5347**
SPPMI (k=1)-value	0.5257		
SPPMI (k=5)-value	0.4646		
SPPMI (k=15)-value	0.3930		
SPPMI (k=1)-vector	**0.5346**		
SPPMI (k=5)-vector	0.5252		
SPPMI (k=15)-vector	0.5221		
SVD-PPMI (k=1)	0.5221	0.5244	0.5265
SVD-PPMI (k=5)	0.5119	0.5161	0.5186
SVD-PPMI (k=15)	0.5056	0.5097	0.5120

In Table 4.20, the best values are given by the doc2vec and SPPMI vector shifted by $k = 1$. However, they are not so distinctive as other word embeddings' scores are very close to the best results.

Table 4.21. Reuters-The Spearman Correlation Values of Asymmetry Task

	d=100	d=500	d=1000
reuters			
doc2vec paragraph	0.5291	0.5272	0.5251
doc2vec document	0.5288	0.5274	0.5256
doc2vec day	0.5277	0.5255	0.5256

For Reuters data, again the results are close to each other where the best results come from the doc2vec paragraph. Here, again, the context size is decreased, the values are increased. However, the values, again, are close to each other.

4.2.5. Conclusion

We have evaluated word representations on six different tasks. Word representations are implemented by various word representation approaches. Our main goal is to see whether our theoretical explanation is confirmed by the word representations on NLP tasks.

In general, word vectors from CBOW, doc2vec, and PPMI based (value or vectors) methods perform better. Actually, in CBOW, the context is given as an input and the middle word is predicted. Therefore, the context is directly fed into the system similar to doc2vec architecture (for both versions). In this perspective, we consider CBOW is kind of doc2vec having smaller context size. Then, the order of granularity in terms of the context size is doc2vec-day, document, paragraph, and CBOW in Reuters data while it is doc2vec, CBOW in Wikipedia. Thus, we can interpret the results as the success of these three models come from their consideration of the context while deriving word representations.

Distinctively, the context-based models, doc2vec and PPMI based methods, are better in the association tasks, e.g. Pun Task, Association Task. It can be said that association evaluation is sensitive to normalization where the word representations from

context-based models are normalized by context words. Furthermore, when the word vectors are evaluated with conditional probability rather than cosine similarity, the results become better.

By taking all of the evaluations into consideration, the distinctive effect of the context on word representations cannot be denied. Additionally, we recognize this impact can be more observable on association tasks. The vector representations' scores can be enhanced by taking the conditional probability as the metric on association tasks.

CHAPTER 5

A CASE STUDY: A KNOWLEDGE BASED DEEP LEARNING APPROACH TO SENSE EMBEDDINGS

This section covers information related to our proposed method to create sense representations.

5.1. Introduction

Sense embeddings address the polysemous word vector representation problem providing vector representation for each sense of polysemous words. We represent an approach to sense embeddings where the disambiguated senses are taken as text corpus for the learning algorithm, e.g CBOW, SGNS, etc.

The first step of the proposed method is to disambiguate the polysemous words' senses. The lesk algorithm (Lesk, 1986) is used for disambiguation. The algorithm achieves 50-70%. It is knowledge-based or dictionary-based algorithm which uses WordNet as the knowledge resource. As explained before WordNet contains the description sentence and samples for each sense. Lesk algorithm depends on the calculation of the word overlap between the context of the polysemous word in the text and the dictionary definition and samples of the senses. The sense is selected with the highest word overlap between sense description and context containing a polysemous word.

$$score_{Lesk}(S_i) = |context(w) \cap gloss(S_i)|$$

In this formula, S_i is each sense of the word, $context(w)$ is a bag-of-words in a context except the polysemous word, w, and $gloss(S_i)$ is, again, bag-of-words in a dictionary definition and samples of sense, S_i. The context similarity is evaluated as the number of word overlap in this algorithm.

Table 5.1 displays the first three senses of a polysemous word, key. The example sentence, from Navigli (2009), is $I\ inserted\ the\ key\ and\ locked\ the\ door$. When the lesk algorithm is run, key^1 has 3 overlap words ($inserted$, $lock$, and $lock's$), as others

Table 5.1. The first three senses of a word key from WordNet

Sense	Definition and Samples
key^1	Metal device shaped in such a way that when it is inserted into the appropriate lock the lock's mechanism can be rotated
key^2	Something crucial for explaining, "the key to development is economic integration"
key^3	Pitch of the voice, "he spoke in a low key"

have no overlap words. Therefore, the selected sense is key^1.

After all words are disambiguated by the Lesk algorithm, the text containing disambiguated words is taken as an input text to the word2vec. As known, word2vec takes the text data as input and in the sliding window context, it either predicts the middle word from context words (CBOW) or predicts the context from the middle word (Skip-gram). In both conditions, the words fed to the system is one-hot vectors and the trained weights of the architecture are taken as the resultant word vectors. In the disambiguated words, again, the words/senses fed to the system are represented as one-hot vectors. However, in this condition, the vocabulary size increases as it does not hold only words instead it holds the senses.

For example, the first few lines of the disambiguated text in Wikipedia are as follows:

anarchism political.a.03 philosophy.n.03 that recommend.v.01 self.n.02 regulate.v.02 society.n.04 free-base.v.01 volunteer.n.01 initiation.n.02 these exist.v.01 often.r.03 report.v.01 stateless society.n.04 although several.s.03 generator.n.03 take.v.35 specify.v.03 them more.r.02 specifically mental-hospital.n.01 free-base.v.01 non hierarchical unblock.v.03 association.n.08 anarchism retain.v.03...,

where, the disambiguated words are represented with their POS tags (a for an adjective, n for a noun, v for a verb, etc.) and sense numbers. The vocabulary is constructed by these sense-tagged data. For each sense in the vocabulary, one-hot vectors are created and fed to the system. Therefore, the resultant vectors, now, represent senses rather than words.

5.2. Experimental Setup

We have developed a method to create a sense representation. The method consists of knowledge-based disambiguation algorithm and neural networks. This section contains

a detailed explanation of the data utilized and the implementation of the proposed method to create sense representation.

5.2.1. Data

We have used the dataset, Wikipedia 2017 dump, the same as in section 4.

5.2.2. Implementation of Sense Embeddings

Lesk algorithm is implemented using $pywsd.lesk$ (Tan, 2014) library in which the $simple\_lesk$ algorithm is used. The algorithm takes a word and a sentence/context as an input and returns a list of candidate senses. We select the first candidate sense in the list as the predicted sense. As a sentence/context, we use the sliding window with the size 10 for the Wikipedia data. The middle word in this sliding window is given to the algorithm as the input word and the sliding window is given to the algorithm as the sentence/context.

The vocabulary is taken as 80000, the same as the word embeddings. However, as for each word, senses are replaced, vocabulary of senses becomes greater than the vocabulary of words.

CBOW is implemented as the word2vec algorithm with the disambiguated corpus. The gensim library (Řehůřek and Sojka, 2010) is used for CBOW implementation, again. In this condition, the input is sense-labeled data rather than raw data. We have implemented sense embeddings in various dimensionalities.

5.2.3. Tasks

There are two common tasks to evaluate the sense embeddings. One of them is the similarity/relatedness task. There are two words and their similarity or relatedness score evaluated by a human in each line, as explained before, in section 4. Four benchmarks have been used, WordSim353, rg, and mc, MEN. The first three datasets are explained in the Section 4. MEN dataset is developed by Bruni et al. (2014). The format is similar to other datasets; there are two words and their similarity rate evaluated by a human in every line.

While evaluating word embeddings, the cosine similarities of two words' vectors are taken and the Spearman correlation value between human evaluations and cosine similarities is based to determine whether the word embeddings predict as human or not. The evaluation of sense embeddings rather than word embeddings should be accomplished at the sense level rather than word level. However, the benchmarks consist of the words rather than senses. The common solution to this problem is to evaluate the senses of the polysemous word. Each sense of the first word and second word are taken from WordNet, then the cosine similarities between sense pairs (one from the first word, the other from the second word) are calculated. The maximum cosine similarity value is taken as the predicted similarity/relatedness value. To illustrate, take word pair, 'opera' and 'performance':

The senses of the word 'opera' are 'opera.n.01', 'opera.n.02', and 'opera.n.03'. The senses of the word 'performance' are 'performance.n.01', 'performance.n.02', 'performance.n.03', 'performance.n.04', and 'operation.n.08'. The similarity values between the senses of the first word and the senses of the second word are as follows:

'opera.n.01' - 'performance.n.01' = 0.2321
'opera.n.01' - 'performance.n.02' = 0.3327
'opera.n.01' - 'performance.n.03' = 0.0841
'opera.n.01' - 'performance.n.04' = 0.1949
'opera.n.01' - 'operation.n.08' = 0.0154
'opera.n.02' - 'performance.n.01' = 0.1830
'opera.n.02' - 'performance.n.02' = 0.1639
'opera.n.02' - 'performance.n.03' = 0.0902
'opera.n.02' - 'performance.n.04' = 0.1749
'opera.n.02' - 'operation.n.08' = 0.1452
'opera.n.03' - 'performance.n.01' = 0.3064
'opera.n.03' - 'performance.n.02' = 0.3096
'opera.n.03' - 'performance.n.03' = 0.1221
'opera.n.03' - 'performance.n.04' = 0.2685
'opera.n.03' - 'operation.n.08' = 0.0089

The most similar value, 0.3327, is taken as the predicted value.

For every word pair in the benchmark, the prediction value from the sense embeddings is calculated. Then, the Spearman correlation value between the predicted values and human evaluation values is calculated. The results display how much close the pre-

dictions of the sense embeddings to the predictions of human evaluators.

The other task is developed by Huang et al. (2012), named as SCWS test. In this task, there are, again, two words and their similarity values are evaluated. However, in this task, each word contains its context. Therefore, the words are evaluated according to their contexts. It enables the evaluation of the senses by discriminating which sense is used in the context. Furthermore, the benchmark contains 10 human evaluator and their evaluations are provided by average evaluation values and separate evaluation values as seen in Table 5.2.

Table 5.2. The first few lines of SCWS benchmark

Word 1	POS 1	Word 2	POS 2	Context 1	Context 2	Human	1-Human	..
Brazil	n	nut	n	gap in income between...	of the neck , bridge , and , pickups ,...	1.1	0.0	...
Brazil	n	triple	n	being elected , has not used...	Pythagorean triple . A ...	0.6	2.0	...
CD	n	aglow	a	other official languages , except for	. In some places the shock...	1.8	0.0	...

In Table 5.2, $Human$ denotes the average value of 10 human evaluators, and ... denotes the evaluations of the individual of human until $10 - Human$.

To determine which sense is used in the context, again we have used the Lesk algorithm. The cosine similarity between the predicted senses is taken as the predicted similarity between words in the given contexts.

5.3. Results

We have developed the system to provide the sense embeddings. The developed sense embeddings are evaluated in two different tasks; the similarity/relatedness tasks and similarity based on context task (SCWS). In this section, the results of these tasks are displayed.

First, the similarity/relatedness tasks' results are shown, then SCWS results are presented. In the similarity/relatedness tasks, we compare the results of sense embeddings with the word embeddings to assess whether the sense embeddings improve the results or not. Additionally, we compare our sense embedding results with results of other sense embeddings developed by Huang et al. (2012) and Iacobacci et al. (2015). The results

of Huang et al. (2012) are generally compared as they construct the dataset, SCWS. The results of Iacobacci et al. (2015) are compared as their proposed method is similar to our method. Thus, we attempt to find out the effects of the differences on the results.

Table 5.3. Similarity and Relatedness Spearman Correlation Results on WordSim353 dataset

	d=100	d=500	d=1000	d=100	d=500	d=1000
wikipedia	wordsim353-r			wordsim353-s		
Word Embeddings	0.629	0.614	0.585	0.751	0.755	0.753
Sense Embeddings	0.562	0.539	0.543	0.665	0.691	0.697

Table 5.3 displays the correlation values between the predictions of the sense/word embeddings and human predictions. Here, wordsim353-r represents WordSim353-relatedness where human evaluate two words in terms of their relatedness and wordsim352-s represents WordSim353-similarity where human evaluate word-pairs in terms of their similarity. CBOW architecture is used for both types of embeddings. In this benchmark, the results of word embeddings are better than the sense embeddings. However, when dimensions are increased, the results of sense embeddings become better.

Table 5.4. Similarity and Relatedness Spearman Correlation Results on rg and mc datasets

	d=100	d=500	d=1000	d=100	d=500	d=1000
wikipedia	rg			mc		
Word Embeddings	0.736	0.785	0.812	0.723	0.797	0.752
Sense Embeddings	0.814	0.848	0.858	0.735	0.828	0.851

Table 5.4 displays the correlation values, again. Here, sense embeddings improve the results, in every dimension.

Table 5.5. Similarity and Relatedness Spearman Correlation Results on MEN dataset

	d=100	d=500	d=1000
wikipedia	MEN		
Word Embeddings	0.719	0.743	0.740
Sense Embeddings	0.688	0.708	0.709

Table 5.5 shows the correlation values, again. Here, the results of word embeddings dominate in every dimension. However, the difference between the results from word embeddings and sense embeddings is not so high.

Table 5.6. Similarity and Relatedness Spearman Correlation Results on WordSim353 and MEN dataset

	d=400	d=400	d=400
wikipedia	wordsim353-r	wordsim353-s	MEN
Iacobacci et al. (2015)	0.645	0.894	0.779
Our Sense Embeddings	0.534	0.680	0.708

Table 5.6 presents the correlation values of the sense embeddings. Here, the dimension is restricted by the 400, because Iacobacci et al. (2015) provide the results only in this dimension and their results are better than ours.

Table 5.7. Similarity and Relatedness Spearman Correlation Results on WordSim353

	d=50
wikipedia	wordsim353
Huang et al. (2012)	0.713
Iacobacci et al. (2015)	0.714
Our Sense Embeddings	0.605

Table 5.7 displays the correlation values of the sense embeddings. Here, the dimension is restricted by the 50 and the dataset is only WordSim353 (both relatedness and similarity), because Huang et al. (2012) provide the results only in this dimension and this dataset. Their results are so close to each other.

Table 5.8. Similarity and Relatedness Spearman Correlation Results

	d=100	d=300	d=500	d=700	d=1000
wikipedia	Our Sense Embeddings				
WordSim353-r	0.562	0.543	0.539	0.537	0.543
WordSim353-s	0.665	0.684	0.691	0.675	0.697
rg	0.814	0.841	0.848	0.842	0.858
mc	0.735	0.802	0.828	0.824	0.851
MEN	0.688	0.708	0.708	0.711	0.709

Table 5.8 presents the correlation values of our sense embeddings. Here, various dimensions are implemented to be able to see in which dimension our embeddings work best. Actually, except dimension 100, the embeddings tend to give better results when dimensionality is increased.

Table 5.9 displays the correlation values of sense embeddings. The distinctive issue in this dataset, the context is included in the evaluation. The result of Huang et al.

Table 5.9. Similarity Spearman Correlation Results on SCWS

	d=50
wikipedia	SCWS
Huang et al. (2012)	0.657
Our Sense Embeddings	0.440

(2012) is better. This is related to the method of Huang et al. (2012) in which they use the clustering technique.

5.4. Conclusion

We have developed a method for sense representations Sense representations are implemented in various dimensions. Our main goal is to see whether sense embeddings developed by knowledge-based algorithm improve the results or not.

In general, other sense embeddings' results are better than us. The reason for this result is that the method to label the raw data has a crucial effect as Iacobacci et al. (2015) propose another method to label raw data. Distinctively, our sense embeddings give better results in mc and rg dataset where we compare our results with word embeddings.

CHAPTER 6

CONCLUSION AND FUTURE WORK

This section contains the information of the conclusion of the thesis and future work, respectively.

6.1. Conclusion

In this thesis, we have focused on the analysis of semantic representations. We have two main goals:

- There are two main approaches to word representations, information-theoretic and vectorial approaches. We attempt to find out the relation between them.
- In word representations, the main aim is to address the meaning of a word. However, there are polysemous words. In this case, one representation covers all senses of a polysemous word. We attempt to create a sense representation for each meaning of polysemous words.

For the first goal, we have analyzed both information-theoretic and vectorial approaches. We have observed that PMI values, an approach in information-theory, evaluate two words relation with a normalization term. This normalization prevents high frequent words to have high association values. And then, we start to analyze vectorial word representations from a similar normalization perspective. In CBOW, a neural approach in vectorial representations, the objective function has a normalization term. However, this term is embedded into the architecture. Our main observation is that both normalization terms depend on context. After this observation, we focus on the doc2vec architecture where the context is fed into the system, directly. Our idea is that if the context has the normalization impact on the representations, we can observe this influence in the doc2vec architecture, more clearly, because this architecture utilize the context (normalization) term explicitly whereas CBOW utilizes it implicitly in its objective function. This impact is evaluated by the various NLP tasks. After the implementation of various word

representations and their evaluation on different NLP tasks, we have four main results to underline:

- The relation between two main approaches is that they both use the context as the normalization factor. In PMI, this is accomplished explicitly whereas, in CBOW, it is done, implicitly.
- In the experimental work, the representations using context give the highest scores.
- In the experiments, we observe that when we use conditional probability rather than cosine similarity while evaluating two words' relation in vectorial representations, results become better.
- We relate this improvement with the normalization, again, where we interpret conditional probability as the normalized version of cosine similarity.

For the second goal, we have developed a method to have sense embeddings. Our idea is that if we label the raw data according to the senses, we can use these data in the word representation methods and we can derive the sense embeddings rather than word embeddings in this time. Thus, to label data, we have used the Lesk algorithm and the labeled data are fed into the CBOW. In the end, we compare our result with Iacobacci et al. (2015) where they use a different algorithm to label data and their results are better than us. Therefore, the method to label data is critical for these tasks. Additionally, we have improved the results of word embeddings in the similarity task (in two datasets; mc and rg), by using our sense embeddings.

In conclusion, semantics is a topic which is studied in various fields. We have focused on semantics in a computational manner in this dissertation and improve it in some perspectives.

6.2. Future Work

For the first study:

- In doc2vec, there are two versions, DM, and DBOW, as explained before. In the experiment part, we have implemented one of them. As a future work, both versions can be implemented. And then, their relation can be analyzed. In this study, the context impact can be observed, again.

- For Wikipedia, the hierarchical structure can be extracted. The level of the document can be analyzed in Wikipedia, also.
- Embeddings may be run on the triangle task.

For the second study:

- The impact of context may be analyzed on the sense embeddings.

REFERENCES

Arora, S., Y. Li, Y. Liang, T. Ma, and A. Risteski (2016a). A latent variable model approach to pmi-based word embeddings. *Transactions of the Association for Computational Linguistics 4*, 385–399.

Arora, S., Y. Li, Y. Liang, T. Ma, and A. Risteski (2016b). Linear algebraic structure of word senses, with applications to polysemy. *CoRR abs/1601.03764*.

Bartunov, S., D. Kondrashkin, A. Osokin, and D. P. Vetrov (2015). Breaking sticks and ambiguities with adaptive skip-gram. *CoRR abs/1502.07257*.

Bruni, E., N. K. Tran, and M. Baroni (2014, January). Multimodal distributional semantics. *J. Artif. Int. Res. 49*(1), 1–47.

Chen, T., R. Xu, Y. He, and X. Wang (2015). Improving distributed representation of word sense via wordnet gloss composition and context clustering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*, pp. 15–20.

Chen, X., Z. Liu, and M. Sun (2014). A unified model for word sense representation and disambiguation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pp. 1025–1035.

Church, K. W. and P. Hanks (1990, March). Word association norms, mutual information, and lexicography. *Comput. Linguist. 16*(1), 22–29.

Damnjanovic, I., M. E. P. Davies, and M. D. Plumbley. Smallbox - an evaluation framework for sparse representations and dictionary learning algorithms. In *in LVA/ICA 10*, pp. 418425.

Goldberg, Y. (2017). *Neural Network Methods for Natural Language Processing*, Volume 37 of *Synthesis Lectures on Human Language Technologies*. San Rafael, CA: Morgan & Claypool.

Griffiths, T. L., M. Steyvers, and J. B. Tenenbaum (2007). Topics in semantic representation. *Psychological review 114 2*, 211–44.

Harris, Z. (1954). Distributional structure. *Word 10*(23), 146–162.

Huang, E. H., R. Socher, C. D. Manning, and A. Y. Ng (2012). Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, Stroudsburg, PA, USA, pp. 873–882. Association for Computational Linguistics.

Iacobacci, I., M. T. Pilehvar, and R. Navigli (2015). Sensembed: Learning sense embeddings for word and relational similarity. In *In Proceedings of ACL*, pp. 95–105.

Landauer, T. and S. Dumais (1997). A solution to Plato's problem: The Latent Semantic Analysis theory of the acquisition, induction, and representation of knowledge. *Psychological Review 104*, 211 – 240.

Le, Q. V. and T. Mikolov (2014). Distributed representations of sentences and documents. In *ICML*, Volume 32 of *JMLR Workshop and Conference Proceedings*, pp. 1188–1196. JMLR.org.

Lesk, M. (1986). Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of the 5th Annual International Conference on Systems Documentation*, SIGDOC '86, New York, NY, USA, pp. 24–26. ACM.

Levy, O. and Y. Goldberg (2014). Neural word embedding as implicit matrix factorization. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, Cambridge, MA, USA, pp. 2177–2185. MIT Press.

Li, J. and D. Jurafsky (2015). Do multi-sense embeddings improve natural language understanding? *CoRR abs/1506.01070*.

Li, Q., T. Li, and B. Chang (2016). Multi-phase word sense embedding learning using a corpus and a lexical ontology. *CoRR abs/1606.04835*.

Marco Baroni, Georgiana Dinu, G. K. (2014). Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. *52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014 - Proceedings of the Conference 1*, 238–247.

Melamud, O., I. Dagan, and J. Goldberger (2017). A simple language model based on PMI matrix approximations. *CoRR abs/1707.05266*.

Melamud, O., J. Goldberger, and I. Dagan (2016). context2vec: Learning generic context embedding with bidirectional lstm. In Y. Goldberg and S. Riezler (Eds.), *CoNLL*, pp. 51–61. ACL.

Mikolov, T., K. Chen, G. Corrado, and J. Dean (2013). Efficient estimation of word representations in vector space. *CoRR abs/1301.3781*.

Mikolov, T., I. Sutskever, K. Chen, G. Corrado, and J. Dean (2013). Distributed representations of words and phrases and their compositionality. *CoRR abs/1310.4546*.

Miller, G. A. (1995, November). Wordnet: A lexical database for english. *Commun. ACM 38*(11), 39–41.

Miller, T., C. Hempelmann, and I. Gurevych (2017, August). Semeval-2017 task 7: Detection and interpretation of english puns. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, Vancouver, Canada, pp. 58–68. Association for Computational Linguistics.

Navigli, R. (2009, February). Word sense disambiguation: A survey. *ACM Comput. Surv. 41*(2), 10:1–10:69.

Navigli, R. and S. P. Ponzetto (2012). BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence 193*, 217–250.

Neelakantan, A., J. Shankar, A. Passos, and A. McCallum (2015). Efficient non-parametric estimation of multiple embeddings per word in vector space. *CoRR abs/1504.06654*.

Nelson, D. L., McEvoy, C. L., and T. A. Schreiber (1998). The University of South Florida word association, rhyme, and word fragment norms.

Nematzadeh, A., S. C. Meylan, and T. L. Griffiths (2017). Evaluating vector-space models of word representation, or, the unreasonable effectiveness of counting words near other words. In *CogSci*.

Panchenko, A. (2016, may). Best of both worlds: Making word sense embeddings interpretable. In N. C. C. Chair), K. Choukri, T. Declerck, S. Goggi, M. Grobelnik, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk, and S. Piperidis (Eds.), *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Pennington, J., R. Socher, and C. D. Manning (2014). Glove: Global vectors for word representation. In *EMNLP*, Volume 14, pp. 1532–1543.

Řehůřek, R. and P. Sojka (2010, May). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, Valletta, Malta, pp. 45–50. ELRA.

Reisinger, J. and R. J. Mooney (2010). Multi-prototype vector-space models of word meaning. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, Stroudsburg, PA, USA, pp. 109–117. Association for Computational Linguistics.

Rose, T., M. Stevenson, and M. Whitehead (2002). The reuters corpus volume 1-from

yesterday's news to tomorrow's language resources. *Proceedings of the Third International Conference on Language Resources and Evaluation*, 29–31.

Sahlgren, M. (2006). *The Word-Space Model: Using Distributional Analysis to Represent Syntagmatic and Paradigmatic Relations between Words in High-Dimensional Vector Spaces*. Ph. D. thesis, Stockholm University, Stockholm, Sweden.

Sevgili, Ö., N. Ghotbi, and S. Tekir (2017). N-hance at semeval-2017 task 7: A computational approach using word association for puns. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pp. 436–439. Association for Computational Linguistics.

Socher, R. (2014). Recursive deep learning for natural language processing and computer vision.

Sun, Y., N. Rao, and W. Ding (2017). A simple approach to learn polysemous word embeddings. *CoRR abs/1707.01793*.

Tan, L. (2014). Pywsd: Python implementations of word sense disambiguation (wsd) technologies [software]. https://github.com/alvations/pywsd.

Trask, A., P. Michalak, and J. Liu (2015). sense2vec - A fast and accurate method for word sense disambiguation in neural word embeddings. *CoRR abs/1511.06388*.

Turney, P. D. and P. Pantel (2010). From frequency to meaning: Vector space models of semantics. *CoRR abs/1003.1141*.

Tversky, A. and I. Gati (1982). Similarity, separability, and the triangle inequality. *Psychological review 89*(2), 123.