Evaluation of High-Dimensional Word Embeddings using Cluster and Semantic Similarity Analysis

by

Shahin Atakishiyev

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Software Engineering and Intelligent Systems

Department of Electrical and Computer Engineering

University of Alberta

ABSTRACT

Vector representations of words, also known as a distributed representation of words or word embeddings, have recently attracted a lot of attention in computational semantics and natural language processing. They have been used in a variety of tasks such as named entity recognition, part-of-speech tagging, spoken language understanding, and several word similarity tasks. The numerical representations of words are mainly acquired either through co-occurrence of words (count-based) or neural networks (predictive). These two techniques represent words with different numerical distributions, and therefore, several studies have been led to estimate the quality of word embeddings for several downstream tasks. Our research sheds light on the evaluation of predictive and count-based word vectors using cluster and semantic similarity analysis. In particular, we have analyzed two crisp clustering algorithms- k-means, k-medoids and two fuzzy clustering algorithms – fuzzy C-means and fuzzy Gustafson-Kessel on several dimensional word vectors from the perspective of quality of word clustering. Moreover, we also measure the semantic similarity of word vectors in regard to a gold standard dataset to observe how different dimensional word vectors express a similarity to human-based judgment. The empirical results show that fuzzy C-means algorithm with adjusted parameter settings group words properly until around hundred dimensions but fails in higher dimensions. Also, fuzzy Gustafson-Kessel clustering was proved to be completely unstable even in around fifty dimensions. Crisp clustering methods, on the other hand, show impressive performance, and even surprisingly their performance becomes better as the dimensionality increases. Furthermore, the results indicated that higher dimensions represent words better in word similarity tasks based on the human judgment. Finally, we conclude that one word embedding method cannot be unanimously said to beat another one in all tasks, and different word vector architectures may produce different experimental results depending on a specific problem.

Preface

This thesis is submitted in fulfillment of the requirements for the degree of Master of Science in Software Engineering and Intelligent Systems. Several parts of the thesis will be used to prepare a journal paper that will be submitted to *Data Mining and Knowledge Discovery* journal by Springer, and a conference paper to be submitted to *The* 7*th* *World Conference on Soft Computing*.

Acknowledgements

There are many people helping, inspiring, and motivating me in my master's study. First and foremost, I would like to thank my parents, brother, sister, and their families for everything they have done for me. They have always supported and helped me to pursue ambitions that I would not even have dreamed of.

I would especially like to express my sincere gratitude to my supervisor- Professor Marek Reformat for his guidance within these two years. His friendly attitude and constructive feedbacks motivated me to conduct research, and study systematically.

I would like to thank all of my teachers – from elementary school to university. I feel so lucky to have good teachers helping me to pursue education and become a useful person to the universe.

I gained a lot of friends in these two years in my country and Canada. I would like to thank all of them for their moral support, motivation, and making me feel at home in the difficult times.

Special thanks to Professor Lotfi Zadeh, who, although no longer with us, introduced fuzzy logic and showed how fair and tolerant decisions can be made in this imperfect world.

I would like to offer thanks to Professor Rafik Aliev whose personality and contributions to fuzzy logic, information sciences, and control engineering have been a great inspiration and motivation for me, and many other people. Knowing him indeed became a milestone in my life and motivated me to dedicate my further life to science and the promotion of education. I am so proud that my country has such a nice person and academic.

Finally, I would like to thank Ministry of Education of Azerbaijan for funding my graduate studies within the "State Program on Education of Azerbaijani Youth Abroad in the Years of 2007-2015" program.

Table of Contents

List of Figures

List of Tables

List of Acronyms

NLP- Natural Language Processing

DSM- Distributional Semantic Models

ASR- Automatic Speech Recognition

PMI- Pointwise Mutual Information

SVD- Singular Value Decomposition

NER- Named Entity Recognition

POS- Part-of-Speech

CBOW- Continuous Bag of Words

CHK-Chunking

MENT- Mention Detection

ANN-Artificial Neural Networks

PAM- Partitions Around Medoids

FCM- Fuzzy C-means

FGK- Fuzzy Gustafson-Kessel

XB- Xie-Beni

t-SNE- t-Distributed Stochastic Neighbor Embedding

RBO- Rank- Biased Overlap

CHAPTER 1

Introduction

1.1 Setting

Recent studies in the distributional semantics have shown interesting linguistic and semantic properties of words. Representing words as distributed vectors has disclosed many surprising outcomes. To be more precise, distributional semantic models (DSM) embed words as distributed vectors in a way that semantically similar words are located near each other in vector space. In addition, simple vector calculations of words seem to enable the discovery of nearby objects in the distributed space. Eventually, word embeddings have attracted many researchers' attention in the field of artificial intelligence, particularly, in natural language processing and computational linguistics. They have been applied to several NLP tasks such as part-of-speech tagging, named entity recognition, syntactic parsing, chunking [1] [43] [44], and linguistic tasks such as word similarity and analogical reasoning tasks. Moreover, word embeddings, due to recent improvements in their quality, have also been applied to error detection in spoken language understanding [45], and the calibration of speech confidence in automatic speech recognition (ASR) systems [46].

The idea of representing words as distributed vectors gained popularity after the introduction of neural language models by Bengio et al. [11]. The diverse approaches using word vector space concept can be categorized in two ways: predictive and count-based methods [12]. Predictive models leverage a neural probabilistic language that uses artificial neural networks to capture and learn the distributed representations of words. Count-based methods, on the other hand, rely on the latent sentiment analysis (LSA) technique to inspect relationships between terms and set of documents by calculating the statistics of how often specific words appear with their surrounding words in text corpora. Specifically, Word2Vec, introduced by Google Research, and GloVe, proposed by Stanford University researchers are two successful examples of word representations indicating the features of these two distinctive methods, respectively. The word embeddings produced by these models significantly outperform other techniques, i.e., traditional models, such as pointwise mutual information (PMI) and Singular Value Decomposition (SVD).

Several hundreds of thousands most frequent word embeddings yielded by these models have been made publicly available for further natural language studies: GloVe vectors in 50, 100, 200,300, and Word2Vec vectors in 300 dimensions are available online.

1.2 Thesis goal

Our research focuses on comparative analysis of two word embeddings: Word2Vec and GloVe, in terms of their ability to embody a semantic similarity of words. We investigate the influence of embeddings' dimensionality on the similarity of words using two different approaches. Firstly, we use four data clustering algorithms, K-means, K-medoids, Fuzzy C-means and Fuzzy Gustafson-Kessel, to inspect how well words represented by embeddings of different dimensionality are grouped by these techniques. Secondly, we use the WordSim-353 dataset to create a *gold standard* of semantic similarity of words and investigate how well semantic similarity of words represented by Word2Vec and GloVe matches similarities obtained with embeddings of different dimensionality. In particular, our study answers the following questions:

• <u>For cluster-based analysis</u>: How different data clustering algorithms are able to group words and how this grouping depending on the embeddings' dimensionality? Is any of the embedding methods better than others?

• <u>For ranking-based similarity evaluation</u>: How does semantic similarity of words determined with different embeddings change depending on the embedding dimensionality in reference to the *gold standard*? Which word embedding technique provides better similarity values when the same dimension of embeddings is used?

1.3 Thesis outline

The thesis structure is organized in the following way: Chapter 2 presents works related to estimation of the quality of word embeddings. Chapter 3 includes a brief overview of artificial neural networks used for constructing embeddings. The theoretical backgrounds behind the two embeddings: GloVe and Word2Vec are covered in Chapter 4. In Chapter 5, we describe the used clustering techniques and visualization method for the word embeddings. Chapter 6 describes evaluation criteria used for assessing and comparison of word embeddings. Finally, we show our empirical results in Chapter 7 and conclude with the contributions of our research in Chapter 8.

CHAPTER 2

Related Works

As the vector representations of words carry many interesting linguistic and semantic regularities, several studies have been performed to estimate their performance in several application domains. Schnabel et al. [1] have performed a solid study of evaluating word embeddings from different perspectives. They have shown a comprehensive overview of existing methodologies (i.e., intrinsic and extrinsic evaluation techniques) for estimating words in vector spaces and also propose new techniques for the reliability of these evaluations. They propose that except pre-collected offline data, word embeddings may be evaluated directly with the online query and real users. In addition to this, instead of taking words in ad hoc trend, they suggest that word queries should be selected in regard to their part-of-speech, abstractness and frequency. The research led by Zhai et al. [2] sheds light on the intrinsic and extrinsic estimation of word embeddings. For the intrinsic evaluation, the results of k-means, g-means, hierarchical-g-means and agglomerative clustering algorithms have been compared to the collection of hundred most frequent synonyms and hyponyms of verbs and nouns of WordNet [3] lexical database. The authors report that k-means algorithm substantially outperforms other clustering algorithms in terms of the purity score (i.e., the percentage of correctly classified data points) and degree of agreement with WordNet synonyms collection. For instance, the purity scores for several cluster numbers in terms of k-means and fuzzy c-means are as presented in Table 2.1.

Table 2.1: Purity scores yielded by k-means and fuzzy c-means. Reproduced from Zhai et al.'s clustering plots.

Number of clusters	20	25	30	35	40	45	50
K-means	0.23	0.27	0.31	0.33	0.37	0.40	0.43
Fuzzy C-means	0.15	0.18	0.19	0.22	0.24	0.23	0.24

For the extrinsic evaluation, word representation clusters have been utilized as features for two natural language processing tasks: Named Entity Recognition (NER) and Part-of-Speech tagging (POS). As text corpora, the English version of OntoNotes 5 has been used and Continuous-Bag-of Words (CBOW), Skip-Gram with negative sampling, and GloVe models have been employed to produce word embeddings (These architectures will be discussed in Chapter 4 in detail). Brown, agglomerative, k-means, g-means have been taken as clustering algorithms. For evaluation purposes, F-1 score and accuracy have been used. According to authors' findings, the highest F-1 score (86.19) has been produced by skip-gram with negative sampling model using agglomerative clustering. Conversely, the best accuracy (97.51) has been yielded by Brown clustering. The authors conclude that increasing number of clusters does not significantly enhance the quality of clustering for the NER and POS tasks.

Ghannay et al. [4] have explored the performance of four word representations (GloVe, Word2Vec, CSLM, and Word2vecf) on four different NLP tasks (POS, Chunking (CHK), NER, and Mention detection (MENT)), and two linguistic tasks – analogical reasoning and word similarity tasks. They have used 4 billion words from Gigaword corpus and eliminated words occurring less than hundred times. Experiments have been led in 200 dimensions and a window size of 5. According to results, w2vf-deps outperforms remaining word embeddings across all of these NLP tasks with 96,66% accuracy in POS, 92.02% F1 score in CHK, 79.37 F1% score in NER, and 58.06% F1 score in MENT task. For the analogical tasks, five types of semantic questions such as capital- country relationships (Warsaw: Poland → Prague: ?) and family (father: mother → brother: ?) , and nine types of syntactic questions (nice: nicely→ loud: ?) have been evaluated. In these experiments, GloVe with 65.5% accuracy has surpassed Skip-Gram (62.3%) and CBOW (57.2%) models. Regarding word similarity, WordSim-353(Finkelstein et al., 2001), Rare Words (Luong et al., 2013) and MEN (Bruni et al., 2012) gold standard datasets have been employed, and while CBOW with 59.0 percent beats other models in reference to WordSim-353, Skip-Gram has achieved better results in the remaining two gold standards with 50.2% Rare Words and 66.2% MEN similarities. Except these evaluations criteria, authors have also shown that the combination of word embeddings using concatenation, Principal Component Analysis (PCA) and autoencoder achieves a good performance across many tasks such as 81.06%, 79.66% and 80.43% F1 scores in concatenation, PCA and autoencoder, respectively in NER task, and 71.4%, 70.07% analogical reasoning task using autoencoder and PCA. They also prove that the combination of word embedding significantly improves automatic speech recognition (ASR) error detection.

The study led by Nayak et al. [5] examines the evaluations of word representations using a benchmark suite of practical tasks. They have tested the two syntactic (part-of-speech tagging and chunking) and the four semantic properties (named entity recognition, sentiment classification, question classification, phrase-level natural language inference) of word embeddings. Then these results have been compared to singular values decomposition (SVD) – a baseline method to obtain word vectors. The experimental results indicate that except NER, all the remaining tasks outperform baseline method. The authors note that this study is mainly concerned with providing a benchmark suite to estimate words in vector spaces efficiently and will enable extrinsic evaluations to be differentiated in a more interpretable manner.

Tsvetkov et al. [6] propose QVEC – an intrinsic measure of the word vectors based on alignment to feature matrix obtained from manual lexical resources. As a performance measure, QVEC uses *recall*, based on the intuition that incoherent dimensions in word vector space are less detrimental than main dimensions we are missing. The key point behind the QVEC is to measure the correlation between distributed vector representations of words and linguistic resource obtained by human judgment. For the evaluation of the semantic word vectors, SemCor (a corpus of syntactically and semantically tagged words) is taken, and a set of linguistic word vectors are constructed. Then the dimensions of the word vector spaces are aligned to dimensions in the linguistic word vectors. This alignment enables to obtain a reasonable annotation of the dimensions in the vector space. Thus, the primary conjecture of QVEC is that dimensions in the distributed word vectors correspond to the linguistic properties. Apparently, the dimensions of the linguistic matrix may not capture all properties correctly, and low correlations are usually due to the missing information in those linguistic matrices. Therefore, QVEC is a more recall-oriented quantification than accuracy. The authors have used a variety of word embeddings model (Word2Vec, GloVe, LSA) for word similarity, text classification, and metaphor detection tasks and calculate the correlation between QVEC and these benchmark tasks. They report the Pearson's correlation $r = 0.87$ between sentiment analysis (a binary classification task between positive and negative movie reviews) and QVEC, and $r = 0.75$ between metaphor detection and QVEC. So, they conclude that the proposed model is an efficient intrinsic estimation method for word vectors and indicates a strong correlation with the rankings yielded by the downstream tasks.

One interesting study led by Baroni et al. [7] compares the performance of context-counting and context-predicting word vectors. They have estimated these vectors in a variety of benchmark tasks. The authors have performed their studies under different parameter settings for both word embedding methods. More precisely, they firstly construct a corpus of 2.8 billion tokens by concatenating the English Wikipedia[1], the British National Corpus, and ukWAC[2] and uses first 300,000 most occurring words in the corpus for both word vector models. Count-based models are extracted using DİSSECT toolkit[3]. Using different parameter settings and dimensionality (from 200 to 500), they evaluate 36 models. Predictive models, on the other hand, are acquired using Word2Vec toolkit[4]. With the same dimensions, as in the count-based model, they evaluate 48 predictive models totally. Then the following benchmark tasks are implemented:

• Semantic relatedness – Correlation between word vectors and gold standard datasets (RG, WordSim353, WordNet, and MEN) are measured

• Synonym detection – classic TOEFL dataset (80 multiple questions aiming the most appropriate target term with four candidates)

• Selectional preference – selecting most associated noun out of several nouns for a given verb

• Analogy – several word analogy tasks to find an appropriate word (i.e., *quick→quickly*, *careful* →?) where the correct answer should exactly be *carefully*.

The empirical results show the impressive overall performance of predictive vectors on count-based vectors in most benchmark tasks. For example, for the predictive and count based vectors, Spearman's correlation on RG data is 74 %, 84%, for WordSim-353 data is 62%, 75%, and for MEN data is 72%, 80%, respectively. In synonym detection, the predictive model also outperforms count vectors in accuracy (91%,76%). Only in the selectional preference task count vectors perform almost same as the predictive ones (41% Spearman's correlation for the both of

[1] http://en.wikipedia.org
[2] http://wacky.sslmit.unibo.it
[3] http://clic.cimec.unitn.it/composes/toolkit/
[4] https://code.google.com/archive/p/word2vec/

models). Thus, they extrapolate that the predictive distributed vector space models are generally better than the count-based vectors in terms of quality across many downstream tasks when performing a systematic comparison.

CHAPTER 3

A Brief Review of Artificial Neural Networks

3.1 Introduction

Artificial Neural Networks (ANN), inspired by the biological neural networks are computational models simulating working principle of the human brain [8]. The brain is a nonlinear, an extremely complex, and a massive information processing system. It consists of more than a hundred billions of interconnected neurons to perform several calculations (i.e., perception, recognition, reaction, control) many times faster than the existing fastest computers do today. ANN is a calculation model that can be applied to real-world problems such as providing input-output mapping, evidential response, contextual information, and approximation [8,9].We have described the structure and learning algorithms for the single and multiple layer neural networks in the subsequent sections.

3.2 Single Layer Neural Networks

Single layer neural network also called a *single layer perceptron* is the simplest artificial neural network to classify linearly separable objects (also known as Rosenblatt's perceptron). The target output is either 0 or 1:

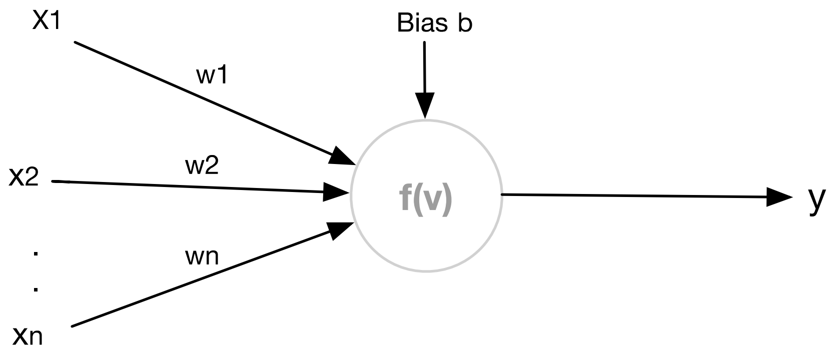


Figure 3.1: Single layer perceptron

Here x are input neurons, w appropriate weight, and b bias term to shift the activation function if necessary.This perceptron model works in the following way:

$$y = f(v) \tag{3.1}$$

The model calculates the sum of the linear combination of the input values x applied to the synaptic weights w. If the result is negative or zero, then output y becomes zero and if positive then the output becomes one:

$$y = \begin{cases} 1, \ if \ \sum_1^n w_i x_i > 0 \\ 0, \ otherwise \end{cases} \tag{3.2}$$

After input values are given the perceptron if the predicted output equals the desired output, then the performance is good, and no any updates are made to the synaptic weights. Nevertheless, if the actual output does not match the desired output, then the following weight adjustment is applied:

$$\Delta w = \eta \cdot d \cdot x \tag{3.3}$$

Here η is a *learning rate* parameter (usually between 0 and 1), d is the difference between the predicted and actual output and x is the input value given to the perceptron.

3.3 Multiple Layer Neural Networks

Above we have shown that the applications of a Rosenblatt's perceptron are limited to linear separation of the objects. However, in many real-world problems, data often become more complicated, and the single layer perceptron model cannot be applied to such kind of problems. To overcome the practical restrictions of the Rosenblatt's perceptron, we use an artificial neural network structure known as a *multilayer perceptron* (also known as the multilayer feedforward network).Multilayer perceptron differs from the single-layer perceptron by containing one or more hidden layers. We have depicted the structure of the multilayer perceptron in Figure 3.2:

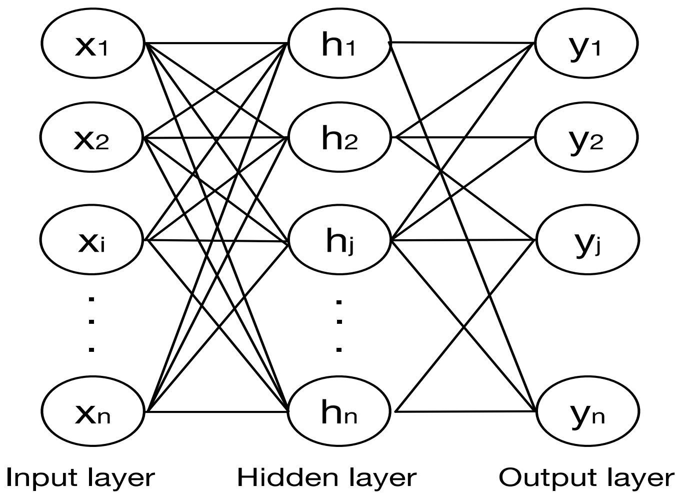


Figure 3.2: Structure of the multilayer perceptron

An effective method to train the multilayer perceptron is called *backpropagation* algorithm. It consists of two stages: the *forward* phase and the *back* phase propagation. In the forward phase propagation, the weights remain fixed, and the input signals are propagated through the network:

$$v = \sum w \cdot x \tag{3.4}$$

$$f(v) = \frac{1}{1+e^{-v}} \tag{3.5}$$

In the backward propagation, for the output neuron, the error signal is yielded by comparing the output of the network with the desired output in the following way:

$$d_o = y \cdot (1-y) \cdot (t-y) \tag{3.6}$$

For the hidden layer, the error is produced as follows:

$$d_i = y_i \cdot (1-y_i) \cdot (w_i \cdot d_o) \tag{3.7}$$

Finally, the updates to the weights are made in this way:

$$\Delta w = \eta \cdot d \cdot x \tag{3.8}$$

From (3.6) and (3.7), we see that the computation of weight adjustments for the output layer is pretty simple, however, fairly challenging for the neurons of hidden layer.

CHAPTER 4

Predictive and Count-based Vector Space Models

4.1 Word2Vec

Word2Vec, proposed by Mikolov et al. [13] is an efficient predictive model that learns distributed word representations from a text corpus. It can use two models of architecture to yield word embeddings: the Continuous Bag of Words (CBOW) and the Skip-Gram. CBOW predicts target words from a window of surrounding context. Utilizing a bag-of-words presumption, the order of the context words does not affect the prediction. In the Skip-Gram model, the inverse operation is performed, and the source words are predicted from the given target words.According to the authors, CBOW is several times faster than the Skip-Gram and slightly shows a better precision for the frequent words. On the contrary, Skip-Gram represents rare words well and works better with a small amount of data. In the next sections, we have indicated the neurocomputational model for both of the architectures.

4.1.1 Continuous Bag-of-Words Architecture

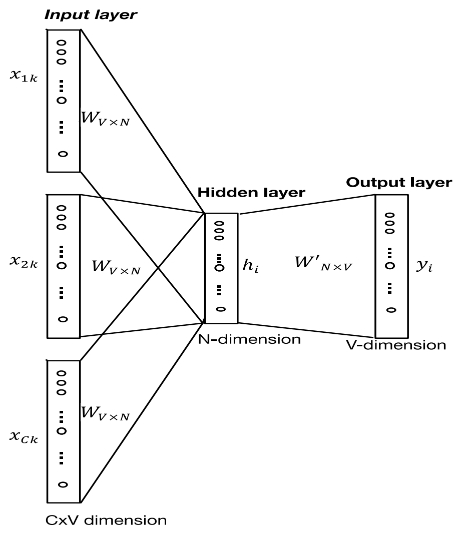
Assuming we have a corpus of {“The”, “children”, “on”, “the” “ground”} as a text context and we would like to predict the center word “played”. First, let us set up the known parameters. We represent the known parameters as one hot encoded vectors: Indicate each of words as $\mathbb{R}^{|V|\times 1}$ vector with all 0s and only 1 at the index of that specific word [14]. Then we define our input context as $x^{(c)}$ and the output content as $y^{(c)}$. Since we will have only one output in the CBOW model, we can call the output just y so that there will be only one hot vector for the predicted center word. We denote two matrices, $A \in \mathbb{R}^{n\times|V|}$ and $M \in \mathbb{R}^{|V|\times n}$, where n is any size that defines the size of word-vector space. A is the input matrix such that the i-th column of A is an n-dimensional vector for the word w_i when it is given as an input to the model. We call this vector as v_i. Likewise, M is the output matrix and the j-th row of M is an n-dimensional vector for word w_j when it gets an output of this model. We indicate this row of M as u_j. So, in fact, in this model we get two embedded word representation for each of word w_i: one input vector - v_i, and one output vector - u_j. For convenience, we have tabulated above notations in Table 4.1:

Table 4.1: CBOW parameter notations

$x^{(c)}$	one hot encoded input word vector
y	one hot encoded output word vectors
w_i	an i-th word from vocabulary V
$A \in \mathbb{R}^{n\times\|V\|}$	Word matrix of input
v_i	an i-th column of A, input vector space model of word w_i
$M \in \mathbb{R}^{\|V\|\times n}$	Word matrix of output
u_j	*an* i-th row of M, output vector space model of word w_i

Once we have all of the parameters ready, we can send them to a neural network to train the model. Figure 4.1 demonstrates the structure and artificial neural network model for the CBOW architecture:

Figure 4.1: Working principle of CBOW model. Algorithm aims to learn $W_{V\times N}$ and $W'_{N\times V}$



The flow of the model is as follows:

1. Having the input context with the size m, we encode it with one hot vectors ($x^{(c-m)}, \dots x^{(c-1)}, x^{(c+1)}, \dots x^{(c+m)}$)

2. We have two sets of synaptic weights: the first one ($W_{V\times N}$) is between input and hidden layer, and the second one ($W'_{N\times V}$) between hidden and output layer. Here *N* is a hyper parameter for the network and can be any number. It also gets the dimension of the embedding space.

3. Obtain vector representations of words from the content (v_{c-m}=$Ax^{(c-m)}, \dots, v_{c+m}$=$Ax^{(c+m)}$)

4. Find average of these vectors to get $\mathcal{v} = \frac{v_{c-m}+v_{c-m+1}+\dots+v_{c+m}}{2m}$

5. Produce a score vector of $z = M\mathcal{v}$

6. Generate the probabilities of these scores γ=$softmax(z)$

7. We want the obtained probabilities, γ, to correspond to the true probabilities, y, that is expected to be one hot vector of the actual output word.

Now that we have a clear understanding of the model, how the input matrix A and the output matrix M can be learned? For this, objective function is needed to be defined. As a reliable choice to find a probability from the true probability, we use cross-entropy as it is a preferred method to calculate the distance between two distributions. It can be shown in the following way for our model:

$$H(\gamma, y) = -\sum_{j=1}^{|V|} y_j \cdot log\,(\gamma_j) \tag{4.1}$$

Knowing that y is a one-hot encoded vector, we can simplify above formula as follows:

$$H(\gamma, y) = y_i \cdot \log(\gamma_i) \tag{4.2}$$

In the fourth step, the parameter *c* stands for the index where encoded one hot vector for the correct word equals 1. For the perfect prediction, γ_c=1. So, applying this in Eq. (4.2) we get -*1·log (1)* =0 which means we do not have any loss. However, if we suppose the pessimistic case, such that, γ_c is very close to 0 (i.e., 0.001) then calculating it in Eq. (4.2) gives -1·log (0.001) ≈6.91. Thus, we can conclude that cross entropy gives a good result and should be a preferred technique. Eventually, we try to minimize the following objective function:

$$J = -\log\ P(w_c\ |w_{c-m}\,, \dots, w_{c+m}\ ) = -\log P(u_c|v)$$

$$= -\log\frac{\exp(u_c^T v)}{\sum_{j=1}^{|V|}\exp(u_j^T v)} = -u_j^T v + \log\sum_{j=1}^{|V|}\exp(u_j^T v) \qquad (4.3)$$

Then all of the pertinent word vectors - u_c and v_j can be adjusted by using stochastic gradient descent. The synaptic weights between hidden layer and output layer will be vector representations of the related words. Mikolov et al. report that CBOW works pretty well with frequent words, but also has some issues regarding the quality of the vectors. The model by default generates only one vector for each unique term. For instance, in cases words having the same spelling but different meanings (i.e., *apple* as fruit and *Apple* as company), the model may predict other meaning of word (i.e., predicts *fruit* apple instead of the *company* Apple) depending on the context.

4.1.2 Skip-Gram Architecture

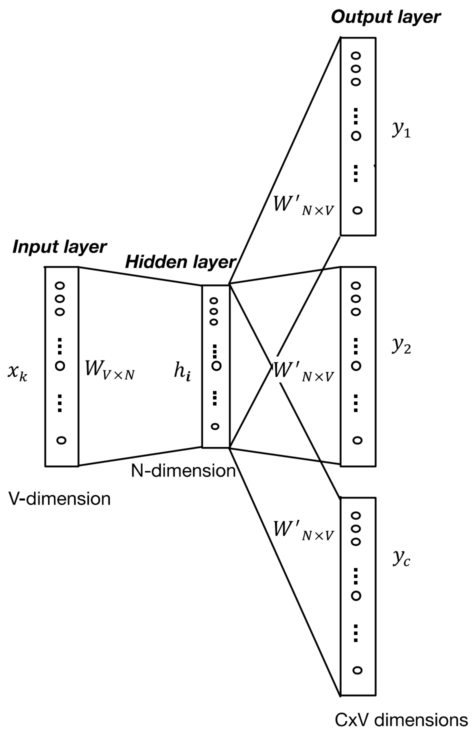
The Skip-Gram model intends to predict surrounding words given a context word. Referring to the previous example, given the word "played," the model tries to predict the surrounding words {"The", "children", "on", "the", "ground"}. So, its parameter setup is very similar to CBOW architecture except the input and output values are swapped here (i.e., *x* becomes *y,* and *y* becomes *x*). Since there is only one word as an input, we represent it as one hot encoded vector *x,* and the output vectors as $y^{(c)}$. Thus, the parameter settings for the Skip-Gram model can be summarized as given in the Table 4.2:

Table 4.2: Skip-Gram parameter notations

x	one hot encoded input word vector
$y^{(c)}$	one hot encoded output word vectors
w_i	an i-th word from vocabulary V
$A \in \mathbb{R}^{n\times\|V\|}$	Word matrix of input
v_i	an i-th column of A, input vector space model of word w_i
$M \in \mathbb{R}^{\|V\|\times n}$	Word matrix of output
u_j	an i-th row of M, output vector space model of word w_i

Having the parameters, we can describe the artificial neural network for this architecture. Figure 4.2 summarizes the Skip-Gram model:

Figure 4.2: Working principle of the Skip-Gram. Algorithm aims to learn $W_{V\times N}$ and $W'_{N\times V}$



The working flow of the model goes through these steps:

1. We produce one hot encoded vector x for the input word

2. We obtain vector for the context $v_c = Ax$

3. As there is no any averaging of vectors, we can assign $v = v_c$

4. We create $2m$ score vectors u_{c-m},... $u_{c-1,}$, ... u_{c+m} from $u = Mv_c$

5. Generate the scores into the probabilities using $y = softmax(u)$

6. We expect that the generated probability vector will match true probabilities that is $y^{(c-m)}, \ldots, y^{(c-1)}, \ldots, y^{(c+m)}$, real one hot vector of the output.

Thus, we see that the learning method of the Skip-Gram model is very similar to the CBOW architecture in terms of a process flow. Now we need to define our cost function to validate the model efficiently. Unlike the bag-of-words assumption, it is a little bit difficult to define the cost function for the Skip-Gram model as the obtained surrounding word contexts are independent words. So, using naïve assumption, we try to minimize:

$$\begin{aligned} \mathrm{J} &= -\log P\,(w_{c-m}, \ldots, w_{c+1}, \ldots, w_{c+m} | w_c) = -\log \prod_{j=0, j\neq m}^{2m} P\big(w_{c-m+j} \big| w_c\big) \\ &= -\log \prod_{j=0, j\neq m}^{2m} P\big(u_{c-m+j} \big| v_c\big) = -\log \prod_{j=0, j\neq m}^{2m} \frac{\exp\big(u_{c-m+j}^T v_c\big)}{\sum_{k=1}^{|V|} \exp(u_k^T v_c)} \\ &= -\sum_{j=0, j\neq m}^{2m} u_{c-m+j}^T\, v_c + 2m \log \sum_{k=1}^{|V|} \exp(\,u_k^T v_c) \end{aligned} \tag{4.4}$$

Utilizing this cost function, we are able to calculate the gradients with respect to the unknown parameters and adjust them consistently in each iteration by mean of Stochastic Gradient Descent.

4.1.3 Training Word2Vec

Word2Vec model is trained using two ways: *hierarchical softmax* and *negative sampling*. In the following subsections, we have indicated the overview of both approximation methods in detail.

4.1.3.1 Hierarchical Softmax

First, let us recall the softmax model. Assuming our word context comprises a sequence of T words $w_1, w_2, \ldots, w_T$ that pertain to some vocabulary V. Each of the context word is assigned an input embedding v_w with dimensions d and an output embedding v'_w. Having h as an output vector of the next layer, softmax function estimates the probability of word w in a context of c as follows[15]:

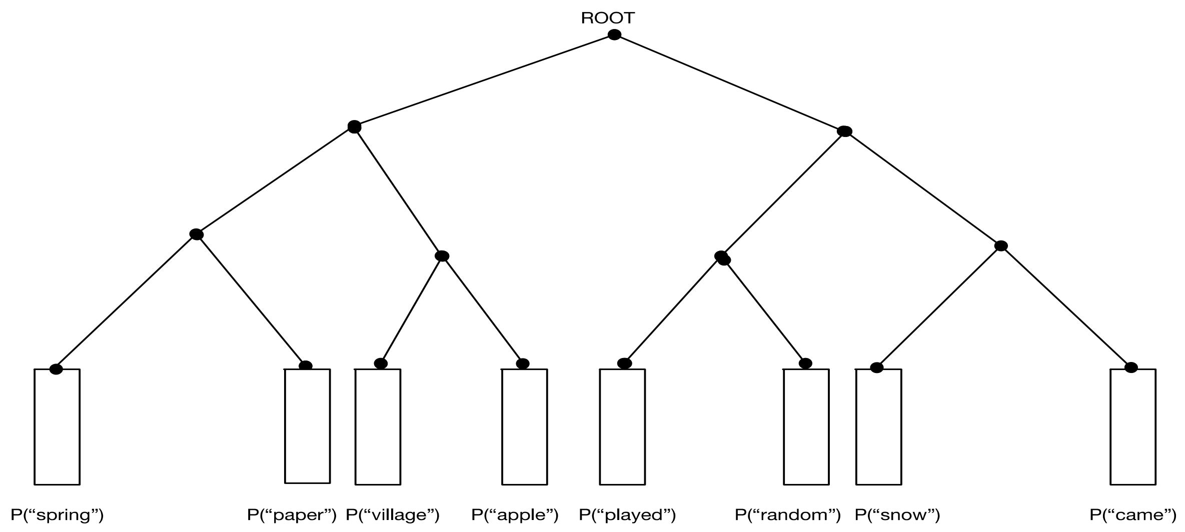
$$\mathrm{p(w|c)} = \frac{\exp(h^T v'_w)}{\sum_{w=1}^{V} \exp(h^T v'_{w_i})} \tag{4.5}$$

We see that this approximation is computationally expensive because our model train very large number of words (usually more than 10^5). Thus, due to this issue, a more robust formulation is needed. Such function was firstly introduced by Morin et al. in the context of neurocomputing and is called *a hierarchical softmax* [16]. The main superiority of hierarchical softmax over regular softmax is that instead of computing W output nodes to get the probability distribution, we actually need to calculate approximately $\log_2 W$ nodes. It replaces the standard softmax layer with a hierarchical layer and utilize a binary tree representation of the output layer of W words as leaves in a tree and assigns probability distributions for each of the child nodes. To be more precise, we know that there is a path to any word ω from the root of the tree in the binary tree model. Let the j-th node on the path from the root of tree to ω become $n(\omega, j)$ and $L(\omega)$ become the length of this path, therefore, $n(\omega, L(\omega)) = \omega$. Moreover, let $child(n)$ be any fixed child of n and we say $[\![x]\!]$ equals 1 when x is true and -1 in all other cases. Then the hierarchical softmax function can be formulated in the following way:

$$p(\omega|w_I) = \prod_{j=1}^{L(\omega)-1} \sigma([\![n(\omega, j+1) = child(n(\omega, j))]\!] \cdot {v'}^{T}_{n(\omega,j)} v_{\omega I}) \qquad (4.6)$$

where $\sigma(x) = (\frac{1}{1+exp^{-x}})$. Hierarchical softmax, in terms of a binary tree, can be described as follows:

Figure 4.3: Binary tree representation of hierarchical softmax



Recalling the previous example {“The”, “children”, “?”, “on”, “the”, “ground”}, from the fixed vocabulary V that consists of words in the tree, we expect that the predicted word “played” will have a higher probability among other words, and CBOW will predict this word for our missing input context. As this a balanced binary tree, the maximal depth is $\log_2(|V|)$, therefore, we only need to compute at most $\log_2(|V|)$ nodes to get the final probability of any word. The sum of the probabilities of all leaves will be 1 as this is a probability distribution. Thus, we see that hierarchical softmax approximation substantially outperforms the standard softmax approach.

4.1.3.2 Negative Sampling

Another objective function to hierarchical softmax is Noise Contrastive Estimation (NCE), proposed by Gutmann et al. [17] and applied to neural probabilistic language modeling by Mnih et al. [18]. The NCE function assumes that a robust model should separate data from the noisy terms. As the goal of the Skip-Gram model is to learn high-quality vector space models, we can simplify the NCE approach for our model until the embedded vectors keep their quality. The Negative sampling (NEG) for our model is defined by the objective of

$$\log \sigma({v'}^{T}_{\omega O}\, v_{\omega I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(\omega)}\left[\log \sigma(-{v'}^{T}_{w_i} v_{\omega I})\right] \tag{4.7}$$

to substitute every $\log P(\omega_O|\omega_I)$ term in the Skip-Gram. Eventually, our goal becomes to differentiate the target word ω_O from the noisy distribution $P_n(\omega)$ employing logistic regression, where there are k samples for each of data sample. Mikolov et al. report that for smaller datasets, the optimal range for k is 5-20 whereas 2-5 range is more preferable to train larger datasets [13]. The main advantage of NEG on NCE is that NEG only uses samples while NCE calculates both samples and the noisy distributions.

4.1.4 Training Corpora

The authors have used one billion Google news words as a text corpus. The words existing less than five times have been discarded in order to boost the quality of word embeddings, and eventually, 692KB data with context size of 5 and 300 dimensions have been trained with the

Skip Gram model. From this corpus, 3,000,000 word vectors have been made publicly available by the authors.[5]

4.1.5 Specific Features of Word2Vec Vectors

The vectors produced by Word2Vec model have yielded some interesting outcomes. These vector spaces embed words in a way that words semantically alike are mapped close to each other. Consequently, clustering these vectors produce very sensible results. In addition, Word2Vec vectors can capture many linguistic and semantic regularities. For example, if "V" is denoted as a vector representation of words, and word vectors such as "king," "male," and" female" are chosen, the following vector – *V (king) - V(male) + V(female)* – will be close to the vector for "queen." Word embeddings can also capture plurality and other aspects of meaning for linguistic research such as *verb tense* and *capital/country* relationships between words. In two examples, *V(walking) -V(walked) ≈ V(swimming) - V(swam)* and *V(Portugal) - V (Lisbon) ≈ V(Germany) - V (Berlin)*. Producing semantics from simple vector calculations is indeed noteworthy. Consequently, vector representations of words seem to have broad perspectives in further computational linguistics and natural language processing research.

4.2 GloVe Vectors

After the introduction of Word2Vec model by Mikolov et al., several studies have been led by researchers to get meaningful word representations. Except the neural probabilistic model, the statistics of word occurrences in text corpora seems to be an efficient way to produce distributed model for words. GloVe, proposed by Pennington et al., is a model to produce word embeddings based on this hypothesis [19]. The term GloVe stands for Global Vectors as it is able to capture global corpora statistics. The authors propose a particular weighted least squares model that counts word-word co-occurrence and makes the frequency statistics useful for word representation purposes.

First of all, we create some parameter notations. Let us define word-word co-occurrence counts matrix as *X*, where X_{ij} shows the number of times the word j exists in the context of the word *i*. Let use denote $X_i = \sum_k X_{ik}$ the number of times any word appears in the context of word *i*.

[5] Pre-trained Word2Vec vectors available in: https://code.google.com/archive/p/word2vec/

Lastly, let $P_{ij} = P(j|i) = X_{ij}/X_i$ become the probability in which word j exists in the context of word *i.* Using simple approach, we demonstrate how aspects of meaning can be extracted from word-word co-occurrences statistics. Pennington et al. shows with good examples. Assuming we have text corpora related to thermodynamics and we may take words $i = ice$ and $j = steam$. We investigate the relationship of these words by learning from the co-occurrence probabilities with other sample words. For instance, if take word ice and word *t*, we can expect that P_{it}/P_{jt} will be large. Likewise, if we select words *t* that are related to *steam* not *ice* such that *t*=*gas,* then we expect that the value of ratio should be small. The authors have reported the co-occurrence probabilities for target words *ice* and *steam* with context words from six billion token corpora. The obtained ratio values prove the proposed hypothesis to be valid: The pertinent words *solid* and *gas* have been distinguished from irrelevant terms *water* and *fashion* such that the same concept words have higher probabilities of co-occurrences. Below table demonstrates the probabilities and ratios for the given sample words:

Table 4.3: Co-occurrence probabilities of sample words *ice* and *steam* with chosen context words from a six billion token corpus. Reproduced from Pennington et al. (2014)

Probability and ratio	**t=*solid***	**t=*gas***	**t=*water***	**t=*fashion***
P(t|*ice*)	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
P(t|*steam*)	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
P(t|*ice*)/P(t|*steam*)	8.9	8.5×10^{-2}	1.36	0.96

Thus, we see that word-word co-occurrence seems to be a good starting point to get distributed representation from frequency statistics. However, one question can naturally be raised here: how word vectors can be obtained from this method? We know that unlike Word2Vec which is based on neural networks, Glove should yield vectors without neuro-computing. We leverage Singular Value Decomposition (SVD) technique to produce the vectors for this approach. For a canonical example, let us assume we have three sentences and the window size is one:

1. I play piano.

2. I like soccer.

3. I like computer science.

Then our word-word co-occurrence matrix can be illustrated as follows:

	I	like	computer	science	soccer	play	piano	.
I	0	2	0	0	0	1	0	0
like	2	0	1	0	0	1	0	0
computer	0	1	0	0	0	0	0	0
science	0	0	1	0	0	0	0	1
soccer	0	1	0	0	0	1	1	1
play	1	0	0	0	1	0	1	0
piano	0	0	0	0	0	1	0	1
.	0	0	0	1	1	0	1	0

We apply SVD on this co-occurrence matrix, observe the singular values, and cut them in some specified index *h*. The SVD for X is defined as follows:

$$X = USV^T \tag{4.8}$$

After this, we use the submatrix of $U_{1:|V|,1:h}$ to become our matrix of word embeddings. As a result, we get an h-dimensional vector representation for each of word in our vocabulary. However, the obtained context word vectors $\tilde{w}$, and word vectors w have some drawbacks to be used as actual representation of words [14]. First, the matrix becomes very sparse as the majority of words are not prone to co-occur. Also, the dimensions of matrix become moderately high and it gets computationally expensive to perform SVD. In the section 4.3.1, we will show how appropriate word representations are achieved.

The approach showed that learning vector representations by capturing frequency statistics should rely on finding ratios of co-occurrence probabilities rather than utilizing those numerical probabilities. Global vectors try to leverage a series of functions called *F* that represents those ratios [19] [20]. These F functions for the ratio of P_{it}/P_{jt} depend on words i, j, t to reflect the vector space models with linear structures:

$$F\left(w_i, w_j, w_t^{\tilde{}}\right) = \frac{P_{it}}{P_{jt}} \tag{4.9}$$

where $w \in R$ are real word vectors and $w_t^{\tilde{}} \in R$ are separate context word vectors. In order to attain the symmetry, we require F to be a homomorphism and eventually express Eq. (4.9) as:

$$\frac{F\left(w_i^T w_t^{\tilde{}}\right)}{F\left(w_j^T w_t^{\tilde{}}\right)} = \frac{P_{it}}{P_{jt}} \tag{4.10}$$

Adding bias terms for the b_i and $b_t^{\tilde{}}$ for the vectors w_i and $w_t^{\tilde{}}$ and expressing F=*exp*,

$$w_i^T\, w_t^{\tilde{}} + b_i + b_t^{\tilde{}} = \log\,(X_{it}) \tag{4.11}$$

One disadvantage of the Eq. (4.11) is that the logarithm diverges when its argument becomes 0. Optimal solution to deal with this problem is to represent the right side as $\log(1 + X_{it})$ where it preserves the sparsity of *X* and avoid the divergence.

Based on the above method, the objective function for Glove which combines a least squares regression model with the weight function $f(X_{ij})$ is defined in the following way:

$$J = \sum_{i,j=1}^{V} f(X_{ij})\,(w_i^T\, w_j^{\tilde{}} + b_i + b_j^{\tilde{}} - log\ X_{ij})^2 \tag{4.12}$$

Here V is the size of the vocabulary and X_{ij} shows the number of times the word j exists in the context of the word *i*. The weighting function should conform to the following properties:

i. $f(0) = 0$. If we see the *f* as a continuous function, then when x→0, f should approach zero.

ii. $f(x)$ has to be a non-decreasing function such that infrequent co-occurrences are not overweighted.

iii. When the argument x gets large values, $f(x)$ should get smaller values in order to avoid the overweighting of frequent co-occurrences.

We can find out many functions that obey these properties. Optimal one proposed by the authors is as follows:

$$f(x) = \begin{cases} (x/x_{max})^{\alpha}, & if\ x < x_{max} \\ 1 & otherwise \end{cases} \quad (4.13)$$

After several experiments, the optimal choice of α found to be 3/4, according to the authors. In addition, x_{max} has been set to 100 in the training.

4.2.1 Training GloVe and Corpora

The objective to train Glove model is to find appropriate vectors that minimize the objective function in Eq. (4.12). As standard gradient descent algorithm heavily depends on the same learning rate, it does not become helpful to find errors and update them properly. Adaptive gradient algorithm (AdaGrad) has been proposed to solve the problem which adaptively assigns different learning rates to each of parameters [19] [20]. After training, the model produces two sets of vectors - W and $W^{\sim}$. When X is symmetric, the generated word vectors intrinsically perform equally and can become different only owing to random initializations. The authors show the best way to handle with these two vectors is to sum and assign the sum vector as a unique representation for our word:

$$W_{final} = W + W^{\sim} \quad (4.14)$$

That is it! Summing two sets of vectors into one effectively reflects words in the embedded space. The model has been trained on five different text corpora. Below table summarizes the data source and size for each one:

Table 4.4: Training corpora of Glove

Data source	Size
2010 Wikipedia dump	1 billion tokens
2014 Wikipedia dump	1.6 billion tokens
Gigaword 5	4.3 billion tokens
Gigaword 5 + Wikipedia 2014 dump	6 billion tokens
Common Crawl web data	42 billion tokens

Each corpus has been lowercased and tokenized by Stanford tokenizer. The authors have built the vocabulary of most frequent 400,000 words, and have made them publicly available online with 50,100, 200 and 300 dimensions, under Public Domain Dedication and License[6].

Like Word2Vec, Glove also proves to capture syntactic, linguistic and semantic features of words. For instance, *V(walking) - V(walked) ≈ V(swimming) - V(swam).* In addition, in the provided example [51] if word vectors such as "paris," "france," and "germany" are chosen, the generated probabilities/words for the following vector – *V (paris) - V(france) + V(germany)* are as follows:
{'*berlin*'- 0.8015437, '*paris*'- 0.7623165, '*munich*' - 0.7013252, '*leipzig*' -0.6616945 '*germany*'-0.6540700}. Thus, we see that model correctly captured the expected word '*berlin*' with the highest probability among other context words.

[6]Pre-trained 400,000 GloVe vectors available in: https://nlp.stanford.edu/projects/glove/

CHAPTER 5

High-Dimensional Data Clustering and Visualization

5.1 Introduction

Data clustering is the task of grouping objects in a way that similarity between data points of the same groups (clusters) becomes as high as possible and similarity between different groups gets as small as possible. It is an important task in data mining and has successful applications in pattern recognition [21] [22], image segmentation [23], fault diagnosis and search engines [24]. Clustering is helpful to understand hidden structure of set of objects or discover knowledge from data. There are several well-known clustering algorithms in machine learning, and the performance of these data clustering techniques can change depending on training corpora, parameter settings, dimensionality and so on. Generally, in terms of data point and cluster membership principle, clustering methods can be divided into two types- *crisp* and *fuzzy* clustering algorithms. Crisp clustering techniques assign data points to exactly one cluster whereas in fuzzy clustering data may belong to several groups with varying membership degrees. Specifically, high dimensional spaces often have a devastating effect on data in terms of performance, where this issue is regarded as the *curse of dimensionality*. From the perspective of clustering word embeddings, we have employed two crisp clustering algorithms – k-means and k-medoids, and two fuzzy clustering algorithms- fuzzy C-means and fuzzy Gustafson-Kessel clustering to analyze how these algorithms behave in high-dimensional spaces. In the following sections, we have described an overview of these algorithms in detail.

5.2 Crisp Clustering Algorithms

5.2.1 K-means Clustering

K-means is a data-partitioning algorithm that assigns *n* observations into *k* groups K= $\{k_1, k_2 ... k_k\}$ by minimizing within cluster sum of squares (i.e., variance) using iterative refinement [25]: To be more precise, the algorithm aims to find:

$$\arg\min \sum_{i=1}^{k} \sum_{x \in K_i} ||x_i - \mu_i||^2 \tag{5.1}$$

where μ_i is the mean of data points in K_i and x_i are data points. The algorithm proceeds in the following way:

1. Choose *k* initial cluster centroids or randomly initialize them

2. Calculate point to cluster centroid distances of all data points for each cluster

3. Assign each data point to the cluster with the nearest distance

4. Calculate the average of the data points in each cluster to get new locations of centroids

5. Repeat steps 2,3,4 until observations-cluster assignments do not change.

5.2.2 K-medoids clustering

K-medoids clustering [26] is very similar to the k-means algorithm: Both algorithms try to partition data to the specified number of groups and minimize the distance between observations labeled to a group and point identified as the center of that group. The main difference between k-medoids and k-means is that k-medoids utilizes a generalized form of Manhattan distance (i.e., uses pairwise dissimilarities) rather than standard Euclidean distance, and always chooses its data points to be cluster centers. Due to this, k-medoids is usually solider to outliers and noise [26]. This method is also known as the Partition Around Medoids (PAM) algorithm. The algorithm generally proceeds as follows:

1. Randomly select *k* of the *n* observations as medoids

2. Assign each data point to the closest medoid

3. For each of medoid *m* and each non-medoid observation *o* associated to *m,* swap *m* and *o,* and re-calculate cost.

5.2.3 Validity Indices for Crisp Clustering

Selecting an optimal number of clusters for both - k-means and k-medoids is an important step to efficiently separate data into clusters. There are several methods to determine the number of clusters for these algorithms that are called *cluster validity indices.* We have chosen two well-known indices, *Dunn* index, and *PBM* index to determine k. The details of these indices have been described in the upcoming sections.

5.2.3.1 Dunn index

The diameter of a cluster can be determined in several ways. For example, we can estimate it as a distance between two farthest data points in a cluster or we may determine it as a mean of all the pairwise distances between observations in a cluster. Let us denote C_i as a cluster of vectors. Assuming x and y is arbitrary two n dimensional feature vector associated with that cluster C_i. Then we define largest distance between x and y as follows:

$$\Delta_i = max_{x,y\in C_i} d(x,y) \tag{5.2}$$

Then let $\delta(C_i, C_i)$ become intercluster distance between the two clusters C_i and C_j. Within these definitions, if we have t clusters, then the Dunn index [27] is calculated as follows:

$$D = \frac{min_{1\le i<j\le t}\delta(C_i, C_j)}{max_{1\le k\le t}\Delta_k} \tag{5.3}$$

Bigger D corresponds to a better clustering. Thus the number of clusters that maximizing this index becomes the optimal number of clusters for given data.

5.2.3.2 PBM index

The PBM index (the acronym shows the names of authors) is another effective way to determine an optimal number of clusters [28]. It is computed using distances between observations and their cluster center, and the distances between individual cluster centers and the overall gravity center of the dataset. Let us define D_c as the largest distance between two cluster centers and M_i as data points:

$$D_c = max_{k<k'} d\left(G^{\{k\}}, G^{\{k'\}}\right) \tag{5.4}$$

Then let us define the sum of the distances of the observations of each cluster to their center as E_W and the sum of the distances of all observations to the gravity center of the whole dataset as E_T:

$$E_W = \sum_{k=1}^{K} \sum_{i \in I_k} d\left(M_i, G^{\{k\}}\right) \tag{5.5}$$

$$E_T = \sum_{i=0}^{N} d(M_i, G) \tag{5.6}$$

Having these parameters, the PBM index is calculated as follows:

$$PBM = \left(\frac{1}{K} \times \frac{E_T}{E_W} \times D_c\right)^2 \tag{5.7}$$

The maximum value of PBM indicates best partition. Thus, if take some range of K values to be number of the clusters, and calculate PBM index, the K_i where we acquire the maximum value of this validity index will be optimal number of clusters for the dataset.

5.3 Fuzzy Clustering

After Zadeh's fuzzy sets theory [29], a lot of studies have been led by researchers to apply theoretical and empirical concepts of fuzzy logic-based clustering. In contrast to hard clustering techniques where one point is exactly assigned to only one cluster, fuzzy clustering allows data points to belong to several numbers of clusters with different membership grades. We have specifically analyzed the performance of two most known fuzzy clustering methods to see how they behave in clustering high dimensional data. The details of these clustering techniques have been depicted in the following sections.

5.3.1 Fuzzy C-means Clustering

Fuzzy C-means (FCM) was introduced by Bezdek [30] in 1974. It allows the observation to belong to multiple clusters with varying grades of membership. Having *D* as the number of data points, *N* as the number of clusters, *m* as the fuzzifier parameter, x_i as the *i*th data point, c_i as the center of the *j*th cluster, μ_{ij} as the membership degree of x_i for the *j*th cluster, FCM aims to minimize

$$J = \sum_{i=1}^{D}\sum_{j=1}^{N} \mu_{ij}^{m} \,||x_i - c_j||^2 \tag{5.8}$$

The FCM clustering proceeds in the following way:

1. Cluster membership values μ_{ij} and initial cluster centers are initialized randomly.

2. Cluster centers are computed according to this formula:

$$c_j = \frac{\sum_{i=1}^{D} \mu_{ij}^{m} x_i}{\sum_{i=1}^{D} \mu_{ij}^{m}} \tag{5.9}$$

3. Membership grades μ_{ij} are updated in the following way:

$$\mu_{ij} = \frac{1}{\sum_{1}^{N} \left(\frac{||x_i - c_j||}{||x_i - c_k||}\right)^{\frac{2}{m-1}}} \, , u_{ij} \in [0,1] \ \ and \ \sum_{i=1}^{c} u_{ij} = 1 \tag{5.10}$$

4. The objective function J is calculated

5. The steps 2,3,4 are repeated until the objective function gets less than a specified threshold.

Fuzzy C-means has many useful applications in medical image analysis, pattern recognition, software quality prediction [30,31] and so on. The most important factors affecting the performance of this algorithm is the fuzzifier parameter m, the size of training data, and the dimensionality of data. The performance analysis of the algorithm for high dimensional clustering will be discussed in Chapter 7 in detail.

5.3.2 Fuzzy Gustafson-Kessel Clustering

Fuzzy Gustafson-Kessel (FGK) extends fuzzy C-means by introducing an adaptive distance norm that allows the algorithm to identify clusters with different geometrical shapes [32]. The distance metric is defined in the following way:

$$D_{GK}^2 = (x_k - \mathrm{v}_i)^T A_i (x_k - \mathrm{v}_i) \tag{5.11}$$

where A_i itself is computed from fuzzy covariance matrix of each cluster:

$$A_i = (\rho_i |C_i)^{1/d} C_i^{-1}, \quad C_i = \frac{\sum_{k=1}^{N} \mu_{ik}^m (x_k - \mathrm{v}_i)^T (x_k - \mathrm{v}_i)}{\sum_{k=1}^{N} \mu_{ik}^m} \tag{5.12}$$

Here the parameter ρ_i is the constrained form of the determinant of A_i:

$$|A_i| = \rho_i\,, \ \rho_i > 0, \forall\, i \tag{5.13}$$

Enabling the matrix A_i to change with fixed determinant serves to optimize the shape of clusters by keeping the cluster's volume constant [32]. Having this parameter setting, Gustafson-Kessel clustering minimizes the following criterion:

$$J = \sum_{i=1}^{c} \sum_{k=1}^{N} \mu_{ik}^m D_{GK}^2 = \sum_{i=1}^{c} \sum_{i=1}^{N} \mu_{ik}^m (x_k - \mathrm{v}_i)^T A_i (x_k - \mathrm{v}_i) \tag{5.14}$$

Like FCM, this optimization is also subject to the following constraints:

$$u_{ik} \in [0,1] , \forall\, i,k \text{ and } \sum_{i=1}^{c} u_{ik} = 1, \forall k \tag{5.15}$$

We see that the computation of the FGK algorithm is more convoluted than FCM clustering.

5.3.3 Validity Indices for Fuzzy Clustering

There are several validity indices to analyze the performance of the fuzzy clustering algorithms. One of them was proposed by Bezdek in 1974 [30] and called a *fuzzy partition coefficient (FPC)*. This index is calculated as follows:

$$FPC = \frac{1}{N}\sum_{k=1}^{N}\sum_{i=1}^{c} {u_{ik}}^{2} \tag{5.16}$$

FPC changes between [0,1] range and the maximum value indicates best clustering quality. Another popular index to measure fuzzy clustering quality was proposed by Xie and Beni (XB) in 1991 [33] which focuses on two properties: cluster compactness and separation:

$$XB = \sum_{i=1}^{c} \frac{\sum_{k=1}^{N} (\mu_{ik})^{m} ||x_k - \text{v}_i||^2}{N min_{ik} ||x_k - \text{v}_i||^2} \tag{5.17}$$

The numerator part shows the strength of the compactness of fuzzy grouping, and the denominator shows the strength of separation between those fuzzy groups. If a range of clusters $\{ k_1, k_2 \ldots k_i\}$ are taken, the k_i minimizing this index will be the optimal number of clusters for the dataset.

5.4 Visualization of High-Dimensional Word Embeddings

Visualizing high-dimensional data is a significant problem in a variety of machine learning applications. High-dimensional data can be observed in many areas such as chemistry, biology, health science, applied mathematics and so on. To get a clear understanding of such data, firstly they should be transformed from high dimensions into a lower one. One of the good dimensionality reduction methods was proposed by Maaten and Hinton in 2008 [34] and called *t-distributed stochastic neighbor embedding* (t-SNE). The t-SNE is a non-linear dimensionality-reduction method that transforms high-dimensional data into lower dimensions (two or three) by constructing probability distribution on the pairs of high-dimensional objects. First, high-dimensional Euclidean distances between observations are converted to conditional probabilities. Having N-dimensional objects $x_1, x_2, \dots x_n$, the probabilities p_{ij} indicating the similarity of the objects x_i and x_j are calculated in the following way:

$$p_{j|i} = \frac{\exp(-||x_i - x_j||^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-||x_i - x_k||^2 / 2\sigma_i^2)} \tag{5.18}$$

Here σ_i^2 is the Gaussian variance. Maaten and Hinton posit that the similarity between x_i and x_j is the conditional probability in which x_i takes x_j as its neighboring point in correspondence with their probability density under the Gaussian distribution. Then the final conditional probability can be indicated as follows:

$$p_{ij} = \frac{p_{ij} + p_{ji}}{2N} \tag{5.19}$$

The t-SNE intends to learn a d-dimensional map $d_1, d_2, \dots d_N$ reflecting the similarities as much as possible. Eventually, using a similar approach, the similarities between any two points in the map d_i and d_j is measured in the following way:

$$q_{ij} = \frac{(1 + ||d_i - d_j||^2)^{-1}}{\sum_{k \neq i}(1 + ||d_i - d_j||^2)^{-1}} \tag{5.20}$$

If the similarity between high-dimensional observations x_i and x_j are mapped correctly by the map points d_i and d_j, then the probabilities p_{ij} and q_{ij} will be same. Inspired by this outcome, stochastic neighbor embedding intends to find out a low-dimensional space that minimize the discrepancy between p_{ij} and q_{ij}. A good and reliable measure for this purpose is the *Kullback-Leibler divergence* [35]. This measure for the distribution Q and the distribution P can be computed as follows:

$$KL_{P||Q} = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{5.21}$$

The sum of all Kullback-Leibler divergence on data points is minimized by the stochastic gradient descent method. The outcome of this optimization forms a map that indicates the similarities between high-dimensional data points properly. The transformed space then can easily be visualized using a scatter plot.

CHAPTER 6

Evaluation Criteria

6.1 Introduction

Our evaluation of word embeddings is based on two criteria: the rank correlation and the clustering analysis. For correlation analysis, we use WordSim353 [36] – a semantic similarity dataset of word pairs, as a gold standard. This dataset contains semantic similarity scores of 353 word pairs from 437 different words. These pairs have been merged from 153 pairs scored by 13 humans and 200 pairs scored by 16 humans. The semantic similarity scores for the pairs vary in the range of [0-10]. For example, the similarity measures for the words *journey* and *voyage*, *computer* and *internet*, and *media* and *gain* are 9.29, 7.58 and 2.88, respectively. Many researchers have referred to WordSim353 as a gold standard for different word similarity tasks [19,37,38,39]. We firstly sort these 353 pairs based on their similarity ranks and take this as a *golden ranking.* Then 437 words and corresponding vectors are extracted from the GloVe and Word2Vec data for all available dimensions. Finally, cosine similarities between the vectors are calculated. Given two word vectors A and B, the cosine similarity between these vectors is computed as follows:

$$\cos(\theta) = \frac{A \cdot B}{||A|| \cdot ||B||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{6.1}$$

where A_i and B_i are the corresponding vector components. We sort the cosine similarities for those dimensions based on the golden ranking of WordSim353, and then correlation ranks are measured. We have employed three well-known rank correlation methods: Spearman's rank correlation, Kendall's rank correlation, and Rank-Biased Overlap. The overview of these correlation techniques will be provided in the following sections. Regarding the clustering analysis, we compare word clustering quality between different dimensions of GloVe vectors with each other, and 300-dimensional GloVe vectors with 300 dimensional Word2Vec vectors based on cluster validity indices and WordSim pair counts in clusters. The golden 437 words of

WordSim353 have been used for this purpose as well. The exact details of the cluster analysis will be indicated in the Empirical results section.

6.2 Spearman's Rank Correlation Coefficient

Spearman's rank correlation is a nonparametric measure used to determine the strength of association between two ranked variables [40]. This correlation can assess both linear and non-linear (i.e., monotonic) relationships. Depending on the rankings, Spearman's correlation can be calculated in two ways: 1) Data do not have any tied ranks and 2) Data have one or more tied ranks. If there are no any tied ranks, then the correlation is calculated as follows:

$$\rho = 1 - \frac{6\sum d_i^2}{n(n^2-1)} \tag{6.2}$$

where d_i indicates the difference in the paired ranks and *n* indicates the number of observations. If there are any tied ranks between the corresponding pairs, the correlation is formulated in the following way:

$$\rho = \frac{\sum_i (x_i - \tilde{x})(y_i - \tilde{y})}{\sqrt{\sum_i (x_i - \tilde{x})^2 \sum_i (y_i - \tilde{y})^2}} \tag{6.3}$$

where the numerator shows the covariance of the ranked variables and the denominator shows the standard deviations of these variables. Spearman's correlation varies between -1 and 1, in which -1 indicates a perfect negative correlation and 1 indicates a perfect positive correlation. For example, we have two ranked variables: X- the independent variable, and Y- the dependent variable. If X increases and Y also tends to increase, the Spearman's rank correlation becomes positive. If Y tends to increase when X decreases, the rank correlation becomes negative. The correlation is expected to be zero when any tendency for Y is not observed when increasing or decreasing X.

6.3 Kendall's Rank Correlation Coefficient

Kendall's rank correlation, also known as Kendall Tau (τ) determines the probability of the two items appearing in the same order in two different ranked lists [41]. It measures the difference between the probability that data in two lists have same ordering versus the probability that data of the two ranked lists do not have the same ordering. Let us define (x_1, y_1), (x_2, y_2), …, (x_n, y_n) to be a set of data points of the random variables X and Y correspondingly, and all the values for the (x_i) and (y_i) are distinct. The pairs (x_i, y_i) and (x_j, y_j) with $i \neq j$ preserving ranks of both elements (i.e., if both $x_i > x_j$ and $y_i > y_j$ or if both $x_i < x_j$ and $y_i < y_j$) are called the *concordant pairs.* The pairs are called *discordant* if $x_i > x_j$ and $y_i < y_j$ or $x_i < x_j$ and $y_i > y_j$. Having these definitions, Kendall's Tau is mathematically expressed in the following way:

$$\tau = P(C) - P(D) = \frac{C-D}{n(n-1)/2} \tag{6.4}$$

The parameter *C* and *D* indicates the number of the concordant and discordant pairs, respectively. The denominator indicates the total number of pairs in which *n* is the number of items in a list. Like Spearman's correlation, Kendall's Tau also changes in the [-1;1] range. If the two ranked lists have a perfect agreement, the rank correlation becomes 1. On the other hand, if a perfect disagreement is observed between the lists, this coefficient becomes -1. Finally, if the variables are totally independent, the correlation may be expected to be nearly zero.

6.4 Rank-Biased Overlap

There are several common challenges Spearman's correlation and Kendall's Tau do not meet effectively. The first issue is *dis-jointness* of lists, where an item is observed in only one of the links. The second one is a *top-weightedness* problem where the top of the list is considered more important than the tail. And the third, they are *indefinite*: For instance, in an example of search engines, if the user searches for query "2017 new books", the search engine may return thousands of results related to this query. However, the user is not likely to browse all of these links and probably will not look beyond first few query results. Eventually, these three features form *an indefinite ranking*: As the ranking is top-weighted, the value decreases with depth and thus urges the shortening of the list in some arbitrary rank, due to this decay. That's why this cut-off makes ranking indefinite.

Rank-Biased Overlap (RBO), proposed by Webber et al. [42] is a measure meeting all the criteria identified as similarity measure for indefinite ranking. This approach is based on a simple model where the user makes a comparison of the overlap between two ranks at growing depths.

The user imposes some level of patience and specifies a probability of stopping which can be principled as a Bernoulli random variable. Based on this, RBO is calculated as the expected average overlap in which the user continues comparing the two lists. The measure specifies a parameter, indicating the probability that the user knowing overlap at some ranking will continue to examine the overlap at the next rank. The product of these probabilities determines the weight of the overlap at the rank where the user will come. The most important point behind RBO is to regulate the corresponding overlap at each point by a convergent series of these weights (i.e., the sum of the series is bounded). Let us show the steps behind this intuition. We start by giving some notations. Assuming *M* and *N* are two infinite ranks, and M_i is the element at ranking *i* in the list *M*. We indicate the elements from the place *c* to *d* as $M_{c:d}$ in the list *M*. At each depth *d*, the intersection of M and D is shown as follows:

$$I_{M,N,d} = M_{:d} \cap N_{:d} \tag{6.5}$$

The size of the intersection shows the overlap of these two lists at depth *d*:

$$L_{M,N,d} = |I_{M,N,d}| \tag{6.6}$$

The overlapping agreement of these lists at depth *d* is then calculated as

$$A_{M,N,d} = \frac{L_{M,N,d}}{d} \tag{6.7}$$

For the estimation depth *k*, the average overlap is determined as follows:

$$AO(M,N,k) = \frac{1}{k}\sum_{d=1}^{k} A_{M,N,d} \tag{6.8}$$

Considering overlap-based rank similarity families form

$$SIM(M,N,w) = \sum_{d=1}^{\infty} w_d \cdot A_{M,N,d} \tag{6.9}$$

where w shows vector of weights and w_d shows the weight at position *d*. $\mathrm{SIM(M,N,w)}$ changes in the interval $[0, \sum_d w_d]$. If w converges, each $\mathrm{A_{M,N,d}}$ obtains a fixed distribution $w_d / \sum_d w_d$ [42]. A solid example to such a convergent series is a geometric sequence in which the value for the *d*th term equals p^{d-1}, and can be formulated as

$$\sum_{d=1}^{\infty} p^{d-1} = \frac{1}{1-p} \tag{6.10}$$

If we set w_d to $(1-p)p^{d-1}$ where $\sum_d w_d = 1$, we can finally formulate RBO as follows:

$$RBO(M,N,p) = (1-p)\sum_{d=1}^{\infty} p^{d-1} A_{M,N,d} \tag{6.11}$$

RBO changes in the range [0,1] where 0 shows disjoint and 1 means an identical ranking. Here the parameter p indicates the steepness of the decline [42]. The metric becomes more top-weighted when p becomes smaller. Conversely, if p gets close to 1, then the estimation of measure becomes deeper. Therefore, for the evaluation of rankings with many elements, p is recommended to be taken between 0.9 and 1.

CHAPTER 7

Empirical Results

7.1 Data Sets

The experiments and their results presented in this section use words from the WordSim-353 test collection [36]. The set contains a total of 353 pairs of words. Each pair's relatedness was evaluated by 13 or 16 individuals that 'possessed a near-native command of English' [36]. Hereafter, we call this set of pairs *the gold standard*. We treat it as a reference in determining how well word embeddings represent semantic similarity. The list of pairs is included in this link[7].

The words from the gold standard are used in the clustering experiments. There are 437 unique words among all pairs from the standard. Further, to evaluate quality – from the point of view of semantic similarity – of obtained clusters, we used a subset of pairs from the gold standard. In this case, we select of pairs of words with a value of similarity above 0.75 (in the scale from 0 to 1). Such a selected subset allows us to determine if creating clusters contain pairs of words from the gold standard and how many of them. Such an evaluation of clusters is motivated by the fact that a clustering process works based on similarity of words; therefore, clusters should contain pairs of similar words.

7.2 Ranking-based Analysis of Embeddings

7.2.1 Description

The approach presented here is based on a comparison of the similarity of word pairs determined using word embeddings with the ranking determined by the gold standard, i.e., by the ranking expressing human-based semantic similarity of words.

Each word embedding, i.e., GloVe with the representation of words by vectors of size 50, 100, 200, 300 and Word2Vec, is used to determine the similarity of word pairs. We use cosine

[7] http://www.cs.technion.ac.il/~gabr/resources/data/wordsim353/

similarity measure for that purpose. The word pairs are sorted based on the obtained similarity values. As a result, we have five different rankings. Each of them represents similarity of words determined using different word embeddings.

We compare these rankings with three different correlation measures (Sections 6.2, 6.3, 6.4):

- Spearman
- Kendall tau
- Rank Biased Overlap

7.2.2 Results and Their Analysis

The values of correlation measures obtained via comparing the ranking of *the gold standard* with rankings obtained using different embeddings are shown in Table 7.1.

Table 7.1 Embeddings vs Golden Standard: Correlation of Rankings

embedding	**GloVe50**	**GloVe100**	**GloVe200**	**GloVe300**	**Word2Vec**
Spearman	0.49	0.52	0.57	0.60	0.70
Kendall tau	0.35	0.38	0.41	0.44	0.50
Rank Biased Overlap	0.41	0.43	0.46	0.48	0.59

As we can see, the presented values lead to the following conclusions:

- Increase in the dimensionality of vectors used for word embedding showed an increase in the ability of a given embedding to represent/express human-based semantic similarity.
- Word2Vec seems to represent the semantic similarity of words better than GloVe300 (even with a version with 300-dimensional vectors).

Please note that these observations are consistent across all three correlation measures.

7.3 Clustering-based Analysis of Embeddings

7.3.1 Description

One of the important parameters of a clustering process is the number of clusters. The nature of unsupervised learning means that number needs to be set a priori. In our experiments, we have determined the boundaries for the number of clusters we consider.

Lower Boundary. We use a simple visualization of words based on t-SNE (Section 5.4). The two-dimensional representation is shown in Figure 7.1

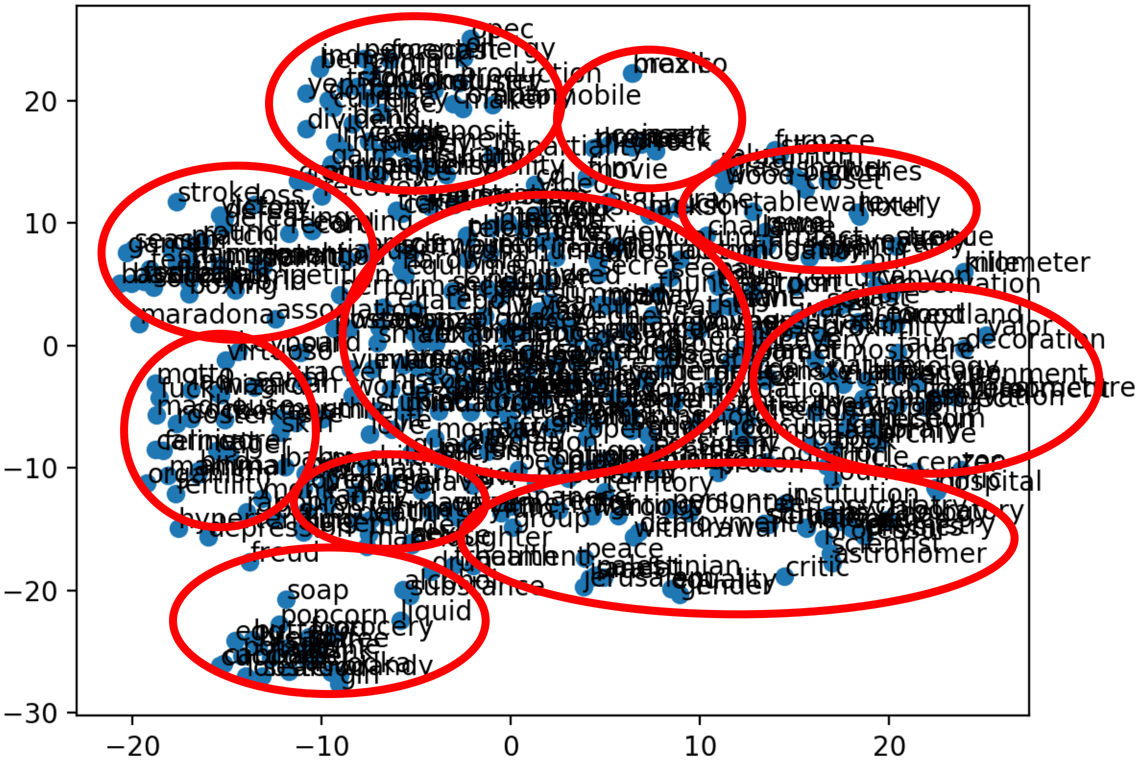


Figure 7.1 Two-dimensional representation of 437 words used in the clustering processes.

Based on a visual inspection, we have identified the most obvious groups of words. As you can see, there are ten locations characterized by a higher concentration of words. Therefore, we use ten as our lower boundary for the number of clusters.

Upper Boundary. There are 437 words in the dataset we use in clustering experiments. We have anticipated that a larger number of clusters would provide better performance in the sense of clustering performance measures. However, we would like to avoid creating too small clusters – smaller cluster would be counterintuitive to our need for observing pairs of words in clusters. Therefore, we have established the acceptable smaller size, on average, of a cluster to around 10. That would lead to a maximum of 50 cluster – and this becomes our upper boundary for the number of clusters.

7.3.2 The Quantitative Analysis of the Results

The first set of the experiments focuses entirely on the clustering. Here, we use four different clustering techniques: two fuzzy ones, and two crisp ones. Each type of clustering has its own performance measures: Xie-Beni and FPC (Section 5.3.3) for fuzzy clustering, and Dunn and PBM indexes (Section 5.2.3) for crisp clustering.

Once the lower and upper boundaries have been set to 10 and 50, respectively, we have decided on the following numbers of clusters: 10, 15, 20, 25, 30, 40, and 50. All experiments presented here are done using four different clustering techniques for these number of clusters, and they are evaluated by two different performance indexes.

Fuzzy Clustering. The results for obtained using fuzzy clustering are shown in Tables 7.2, 7.3, and 7.4. It is well-known that fuzzy clustering shows some problems in the case of clustering high dimensional data [47]. Winkler et al. have shown that performance of fuzzy clustering dramatically changes with the fuzzifier parameter: when using m=2, the majority of prototypes go into the center of the gravity of the whole dataset; therefore, we neither acquire the expected number of clusters nor sensible clustering results. Adjusting the *fuzzifier* around 1 such as 1.1 substantially improves the performance of the clustering and we get high-quality groupings of observations until some dimensions. Based on this conclusion, we have set the *fuzzifier* to 1.1 for our experiments as well. As we can see in the tables,clustering of words with 200dim embedding (Table 7.4) results with the performance measures values that start to look quite unreasonable (Xie-Beni index), while values of the index FPC become quite small. Due to such a situation, we consider for further analysis (next subsection) clusters obtained using FCM and FGK with 50dim GloVe embedding, and only clusters obtained using FCM for 100dim embedding.

Table 7.2 GloVe Embedding: 50dim (fuzzy clustering)

# of clusters	10	15	20	25	30	40	50
Fuzzy C-Means							
Xie-Beni index	0.0054	0.0053	0.0053	0.0056	0.0053	0.0047	**0.0041**
FPC	0.7246	0.7242	0.7476	0.8021	0.8227	0.8683	**0.8878**
Fuzzy Gustafson-Kessel							
Xie-Beni index	16.48	13.43	10.12	13.43	11.58	9.53	**8.00**
FPC	0.9999	0.9878	0.9863	0.9875	0.9874	0.9879	**0.9892**

Table 7.3 GloVe Embedding: 100dim (fuzzy clustering)

# of clusters	10	15	20	25	30	40	50
Fuzzy C-Means							
Xie-Beni index	0.0096	0.0122	0.0092	0.0081	0.0108	0.0080	**0.0076**
FPC	0.5917	0.5987	0.6471	0.6980	0.7210	0.7817	**0.8322**
Fuzzy Gustafson-Kessel							
Xie-Beni index*	30.03	25.40	21.56	20.58	15.11	12.65	**10.87**
FPC	**0.9817**	0.9783	0.9708	0.9747	0.9713	0.9648	0.9701

*numbers in gray represent unacceptable values

Table 7.4 GloVe Embedding: 200dim (fuzzy clustering)

# of clusters	10	15	20	25	30	40	50
Fuzzy C-Means							
Xie-Beni index*	30997.7	8685.2	10542791	156571.2	641690.1	572974.6	30602.8
FPC	0.2612	0.2896	0.3470	0.3638	0.4935	0.5714	**0.5907**
Fuzzy Gustafson-Kessel							
Xie-Beni index*	696.62	731.98	706.72	728.19	782.20	769.02	749.63
FPC	**0.4154**	0.2978	0.2276	0.1842	0.1568	0.1194	0.0951

*numbers in gray represent unacceptable values

Fuzzy clustering means that it is possible that some words do not fully belong to a single cluster. Some words can belong to a few clusters to a different degree. Among all words we have clustered, there are 25 words that have at the maximum membership value of 0.75 to a single cluster for 50dim embedding, and 52 such words for 100dim embedding.

Crisp Clustering. In the case of crisp clustering, we have been able to obtain clusters of acceptable quality for all dimensions. The results are shown in Tables 7.5 – 7.9. A few interesting facts can be observed:

- the values of the performance measures indicate a different number of clusters as the highest performance situation: 50 clusters are indicted when we consider Dunn index, and 10 clusters when PBM;
- if we compare the results across dimensionality of vectors – the higher dimensionality leads to clusters that provide better values of performance measures;
- the k-means clustering method seems to be a technique that provides better values of performance measures.

Table 7.5 GloVe Embedding: 50dim (crisp clustering)

# of clusters	10	15	20	25	30	40	50
K-Means							
Dunn index	0.2685	0.2551	0.2766	0.2836	0.2835	0.2987	**0.3183**
PBM	**0.3288**	0.1896	0.1699	0.1210	0.0731	0.0516	0.0409
K-Medoids							
Dunn index	0.2295	0.2355	0.2537	0.2537	0.2537	0.2636	**0.3116**
PBM	**0.3486**	0.1913	0.1259	0.1249	0.0915	0.0679	0.0467

Table 7.6 GloVe Embedding: 100dim (crisp clustering)

# of clusters	10	15	20	25	30	40	50
K-Means							
Dunn index	0.2659	0.3740	0.3752	0.3751	0.3241	0.3121	**0.4059**
PBM	**0.2638**	0.1978	0.1180	0.0891	0.0850	0.0603	0.0376
K-Medoids							
Dunn index	0.2582	0.2582	0.3544	0.3544	0.2685	0.3676	**0.3637**
PBM	**0.4029**	0.2698	0.1699	0.1134	0.0827	0.0568	0.0523

Table 7.7 GloVe Embedding: 200dim (crisp clustering)

# of clusters	10	15	20	25	30	40	50
K-Means							
Dunn index	0.3615	0.2853	**0.4373**	0.3938	0.4091	0.3634	0.3669
PBM	**0.3092**	0.2658	0.1211	0.1013	0.0861	0.0625	0.0632
K-Medoids							
Dunn index	0.3292	0.2945	0.3471	0.3633	0.3657	0.3657	**0.3674**
PBM	**0.6030**	0.3110	0.1824	0.1529	0.1412	0.0851	0.0798

Table 7.8 GloVe Embedding: 300dim (crisp clustering)

# of clusters	10	15	20	25	30	40	50
K-Means							
Dunn index	0.3613	0.3029	0.3029	0.3821	0.3029	0.3211	**0.3821**
PBM	**0.3003**	0.1988	0.1636	0.0900	0.1137	0.0684	0.0665
K-Medoids							
Dunn index	0.2838	0.3006	0.3393	0.3597	0.3597	0.3606	**0.3624**
PBM	**0.8492**	0.4094	0.2735	0.2105	0.1513	0.1252	0.0852

Table 7.9 Word2Vec Embedding: 300dim (crisp clustering)

# of clusters	10	15	20	25	30	40	50
K-Means							
Dunn index	0.3137	0.3507	0.2979	0.3560	0.3720	0.4106	**0.4331**
PBM	0.0713	0.0645	0.0405	**0.1069**	0.0771	0.0460	0.0312
K-Medoids							
Dunn index	**0.3171**	0.2182	0.2305	0.2641	0.2641	0.2757	0.2982
PBM	**0.1291**	0.0681	0.1057	0.0699	0.0742	0.0448	0.0307

7.3.3 The Qualitative Analysis of the Results

The results presented in the previous subsection describe clusters from the point of view of their quality as measured by the performance indexes. However, these indexes do not show how well the clusters' and clustering techniques' group semantically similar words. For this purpose, we propose another way of determining the quality of clusters from the point of view of grouping similar – according to humans – words.

The first step in the proposed approach has been to identify a set of pairs of words that are similar. Here, we use *the gold standard*, i.e., a set of 353 pairs of words. We have assumed that the similarity value of 0.75 could be considered as a reasonable and practical level of treating words as similar. As a result, we obtain 93 pairs of words.

The second step of the approach is to determine the number of pairs that are present in the same cluster. Additionally, we look at the distribution of pairs among clusters, i.e., we have identified clusters with zero, one, two and so on number of pairs.

Let us start with fuzzy clustering. As mentioned earlier, we have only clusters for low-dimensionality embedding. The results are presented in Table 7.10. As we can see, the Fuzzy C-Means is the best performing technique. Also, increase in dimensionality leads to better results.

Table 7.10 GloVe Embedding: number of word pairs found in clusters:
a clustering process with 50 clusters

GloVe dimensionality:	50	100
Fuzzy C-Means	44	48
Fuzzy Gustafson-Kessel	9	-

For the case of crisp clustering, we have more clustering results thus we can perform a better analysis of obtained clusters. At the beginning, let us take a look at the GloVe embedding and analysis an influence of dimensionality in the Table 7.11. We can draw two conclusions: K-means leads to better results, and better results are associated with the higher dimensionality of embedding.

Table 7.11 GloVe Embedding: number of word pairs found in clusters 50 clusters

GloVe dimensionality:	50	100	200	300
K-Means	44	43	47	45
K-Medoids	26	40	30	42

It seems more intriguing in the comparison of two different embeddings: GloVe and Word2Vec. Here we provide results for K-means, in Table 7.12, and K-medoids, in Table 7.14. Each of these tables contains a number of pairs found in clusters when clustering process has been repeated 10 times. This allows us to perform statistical analysis that leads to a more formal indication which embedding is better in reflecting human similarity.

K-means clustering. Table 7.12 contains numbers of word pairs found in clusters found in each result of clustering for GloVe 300-dim and Word2Vec which also has 300-dim vector representation. The question we have been interested in is related to the quality of embedding. The assumption is that higher number of word pairs found in the clusters – better the embedding.

Table 7.12: Number of word pairs in K-means clustering: 50 clusters, 300dim embeddings

experiment #	1	2	3	4	5	6	7	8	9	10
GloVe	45	39	38	42	41	43	43	38	36	37
Word2Vec	43	41	52	54	46	48	48	38	40	53

Table 7.13 provides the outcomes of statistical analysis of obtained clustering results. If we assume a normal distribution of the number of found word pairs, we can rely on a t-test. The obtained value of p = 0.007658 indicates the statement that Word2Vec is a better embedding from the semantic similarity point of view which is statistically significant at $p < 0.01$. However, if we remove the assumption of the normal distribution, we perform Mann-Whitney (Wilcoxon) test for unpaired data. With this test, the same statement is significant at $p < 0.02$.

Table 7.13 K-means clustering: 50 clusters, 300dim embeddings

	mean	std. dev.	Level of significance	
			t-test	Mann-Whitney (Wilcoxon) test for unpaired data
GloVe	40.2	3.01	p=.007658	p =.018661
Word2Vec	46.3	5.68		

Figure 7.2 represents the distribution of pairs among clusters with GloVe embedding. As we can see, the majority of clusters with the word pairs contain just a single pair. It seems that in almost all clustering experiments we have clusters with one, two and three word pairs. The clusters with a larger number of pairs do not appear consistently in all clustering runs.

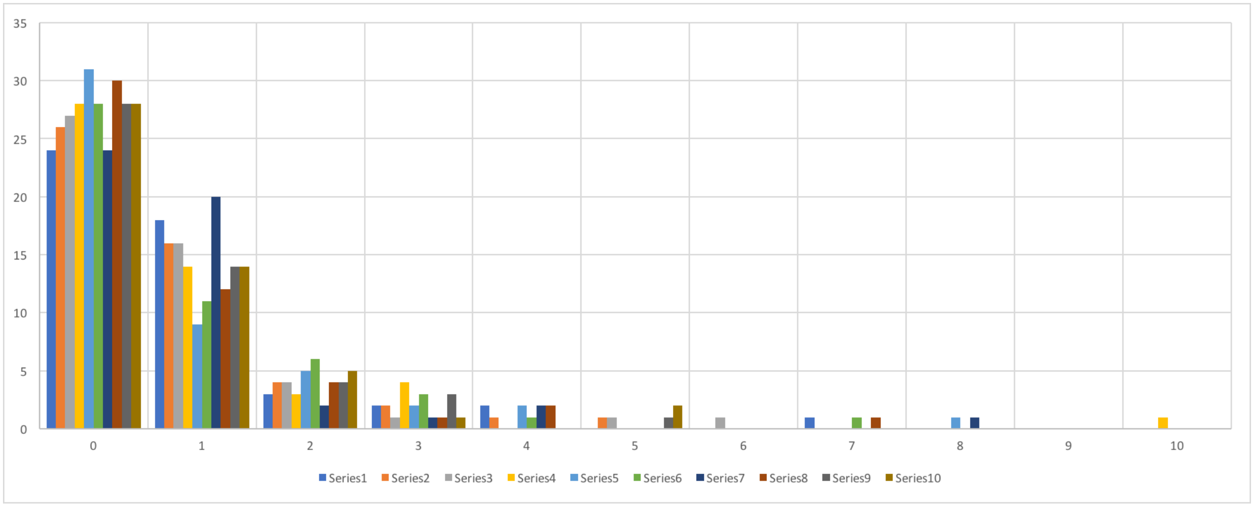


Figure 7.2. The distribution of the number of word pairs among clusters for GloVe embedding (x – the number of pairs per clusters, y – the number of clusters with a specific number of pairs).

Figure 7.3 represents the distribution of pairs among clusters with Word2Vec embedding. This time, all clustering experiments we have clusters with one, two and three and even four word pairs. When distributions are compared, we can say that in the case of Word2Vec there are more clusters with larger numbers of word pairs.

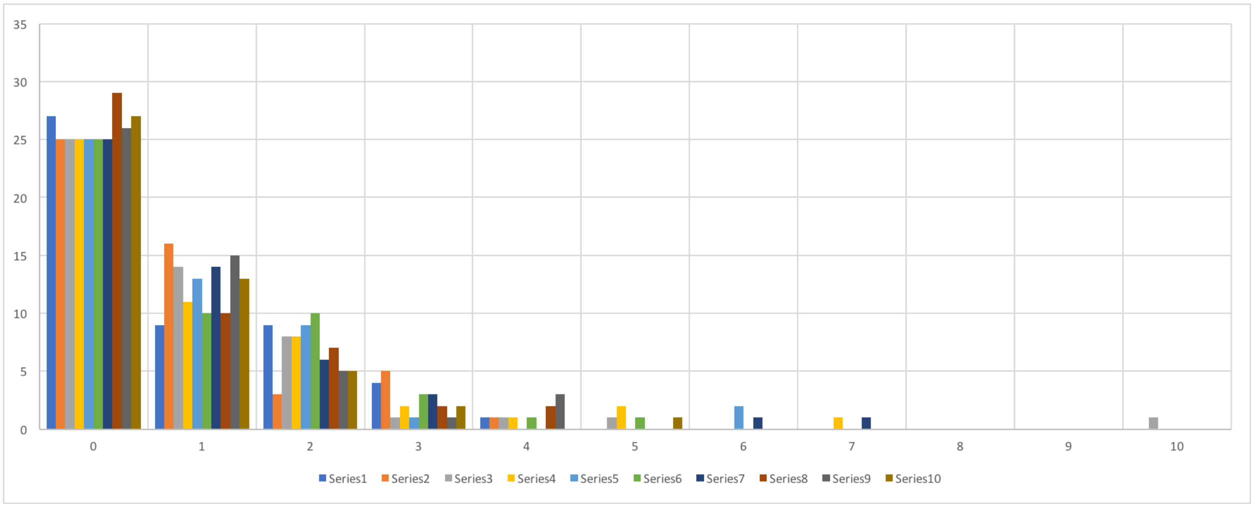


Figure 7.3.The distribution of the number of word pairs among clusters for Word2Vec embedding (x – the number of pairs per clusters, y – the number of clusters with a specific number of pairs).

K-medoids clustering. Table 7.14 includes the results of experiments with K-medoids clustering technique while Table 7.15 contains their statistical analysis. As we can see, based on the obtained p-values for the both t-test and Mann-Whitney test, we cannot conclude which of the embeddings represents semantic similarity better.

Table 7.14 GloVe Embedding: 50 clusters, 300dim, K-medoids

experiment #	1	2	3	4	5	6	7	8	9	10
GloVe	41	36	42	39	32	33	50	41	38	42
Word2Vec	37	50	43	41	39	38	39	31	40	41

Table 7.15 K-medoids clustering: 50 clusters, 300dim embeddings

	mean	std. dev.	Level of significance	
			t-test	Mann-Whitney (Wilcoxon) test for unpaired data
GloVe	39.4	2.83	p=.825065	p =.969656
Word2Vec	39.9	9.19		

When we compare the both clustering techniques – K-means and K-medoids – for Word2Vec embedding, the value of p=.007658 for the t-test, and p =.018661 for the Mann-Whitney test is observed. Based on these outcomes, we can say with significance at $p < 0.03$ that K-means is a better clustering method than K-medoids for Word2Vec word embedding. However, for GloVe embedding, it is not resolvable (p=.825065 for the t-test, and p =.969656 for the Mann-Whitney test).

CHAPTER 8

Conclusions, Contributions and Future Works

8.1 Conclusions

In this thesis, we have examined the performance of high-dimensional word vector spaces using cluster and semantic similarity-based approaches. We analyzed the performance of two well-known crisp clustering algorithms: K-means and K-medoids; and two fuzzy clustering ones: fuzzy C-means and fuzzy Gustafson-Kessel. We applied them to cluster word embeddings and examined obtained clusters. In addition, we explored the semantic similarity of words in reference to the existing gold standard dataset. Our study and related works prove that vector spaces of words obtained by the state of the art models have broad and justified applications in computational linguistics and natural language processing. They can be applied in areas, such as machine translation, sentiment analysis, information retrieval, pattern recognition, spoken language understanding, and automatic speech recognition.

8.2 Contributions

The main contributions of our research can be described as follows:

I. Among the proposed data clustering algorithms, K-means seems to be the most efficient one to cluster word embeddings. Higher quality of word clusters and more cardinality of highly similar word pairs are mostly observed in this clustering technique. Interestingly, the performance of K-means clustering improves with the dimensionality of embeddings: higher dimensionality resulted in a better performance regarding the quality of clustering. The performance of K-medoids can be said to be moderate, and its performance is lesser to K-means only by a small margin.

II. Fuzzy clustering algorithms were proved to be very sensitive to high dimensional data. Fuzzy C-means with fuzzifier parameter $m=1.1$ can provide sensible word clustering results for up to 100-dimensional word embeddings. However, in higher dimensions, it fails to produce both the expected number of clusters and plausible word clustering results. Fuzzy Gustafson-Kessel clustering technique, on the other hand, should be avoided for high dimensional data. Even for the case of fifty-dimensional data, very poor performance has been observed.

III. Increasing the dimensionality of word vector space has indicated the ability of the examined embeddings, according to all used correlation measures, to represent human-based similarity judgments better: the similarity scores consistently increased with the increased dimensionality. Moreover, each of the three correlations measures has indicated that the Word2Vec vectors express human-based similarity better than the 300-dimensional GloVe.

IV. Overall, based on our research and the related works, we can conclude that there is no certain winner between predictive and count-based word embedding methods. It is possible that one of them can be more suitable for a specific task but not so for other ones. Consequently, the comparative analysis of these word representations should be performed for considered tasks.

8.3 Future Works

As a further task, we will extend our research by comparing the word embeddings to the other gold standard datasets. These datasets include Bruni et al.'s MEN [48], Hill et al.'s SimLex [49] and HyperLex [50]. As our current study revealed the proportionality of higher dimensionality to better human-based similarity scores, it is worthwhile to take further directions to investigate whether the disclosed tendency is observed in the case of other gold standard datasets as well. We also anticipate additional work on the suitability of clustering methods for word embeddings.

References

[1] Schnabel, T., Labutov, I., Mimno, D. M., & Joachims, T. (2015, September). Evaluation methods for unsupervised word embeddings. In *EMNLP* (pp. 298-307).

[2] Zhai, M., Tan, J., & Choi, J. D. (2016, February). Intrinsic and Extrinsic Evaluations of Word Embeddings. In *AAAI* (pp. 4282-4283).

[3] Miller, G. A. (1995). WordNet: a lexical database for English. *Communications of the ACM*, *38*(11), 39-41.

[4] Ghannay, S., Favre, B., Esteve, Y., & Camelin, N. (2016, May). Word Embedding Evaluation and Combination. In *LREC*.

[5] Nayak, N., Angeli, G., & Manning, C. D. (2016). Evaluating Word Embeddings Using a Representative Suite of Practical Tasks. *ACL 2016*, 19.

[6] Tsvetkov, Y., Faruqui, M., Ling, W., Lample, G., & Dyer, C. (2015). Evaluation of word vector representations by subspace alignment.

[7] Baroni, M., Dinu, G., & Kruszewski, G. (2014, June). Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *ACL (1)* (pp. 238-247).

[8] Haykin, S. (2009). *Neural networks and learning machines* (Vol. 3). Upper Saddle River, NJ, USA:: Pearson.

[9] Cheng, B., & Titterington, D. M. (1994). Neural networks: A review from a statistical perspective. *Statistical science*, 2-30.

[10] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*.

[11] Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A neural probabilistic language model. *Journal of machine learning research*, *3*(Feb), 1137-1155.

[12] *Vector representation of Words* (2017). https://www.tensorflow.org/tutorials/word2vec

[13] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111-3119).

[14] Socher, R., & Mundra, R. S. (2016). CS 224D: Deep Learning for NLP1.

[15] Rong, X. (2014). word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*.

[16] Morin, F., & Bengio, Y. (2005, January). Hierarchical Probabilistic Neural Network Language Model. In *Aistats* (Vol. 5, pp. 246-252).

[17] Gutmann, M. U., & Hyvärinen, A. (2012). Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, *13*(Feb), 307-361.

[18] Mnih, A., & Teh, Y. W. (2012). A fast and simple algorithm for training neural probabilistic language models. *arXiv preprint arXiv:1206.6426*.

[19] Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1532-1543).

[20] Chen, Z., Huang, Y., Liang, Y., Wang, Y., Fu, X., & Fu, K. (2017). RGloVe: An Improved Approach of Global Vectors for Distributional Entity Relation Representation. *Algorithms*, *10*(2), 42.

[21] Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: a review. *ACM computing surveys (CSUR)*, *31*(3), 264-323.

[22] Duda, R. O., Hart, P. E., & Stork, D. G. (2012). *Pattern classification*. John Wiley & Sons.

[23] Dhanachandra, N., Manglem, K., & Chanu, Y. J. (2015). Image segmentation using K-means clustering algorithm and subtractive clustering algorithm. *Procedia Computer Science*, *54*, 764-771.

[24] Bijuraj, L. V. (2013). Clustering and its Applications. In *Proceedings of National Conference on New Horizons in IT-NCNHIT* (p. 169).

[25] Lloyd, S. (1982). Least squares quantization in PCM. *IEEE transactions on information theory*, *28*(2), 129-137.

[26] Kaufman, L., & Rousseeuw, P. J. (2005). Finding groups in data: An introduction to cluster analysis (Wiley series in probability and statistics). New York: Wiley-Interscience.

[27] Dunn, J. C. (1973). A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters.

[28] Pakhira, M. K., Bandyopadhyay, S., & Maulik, U. (2004). Validity index for crisp and fuzzy clusters. *Pattern recognition*, *37*(3), 487-501.

[29] Zadeh, L. (1965). Fuzzy sets. *Information and control*, *8*(3), 338-353.

[30] Bezdek, J. C. Pattern Recognition with Fuzzy Objective Function, (1981). *Algorithms*.

[31] De Oliveira, J. V., & Pedrycz, W. (Eds.). (2007). *Advances in fuzzy clustering and its applications*. John Wiley & Sons.

[32] Gustafson, D. E., & Kessel, W. C. (1979, January). Fuzzy clustering with a fuzzy covariance matrix. In *Decision and Control including the 17th Symposium on Adaptive Processes, 1978 IEEE Conference on* (pp. 761-766). IEEE.

[33] Xie, X. L., & Beni, G. (1991). A validity measure for fuzzy clustering. *IEEE Transactions on pattern analysis and machine intelligence*, *13*(8), 841-847.

[34] Maaten, L. V. D., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, *9*(Nov), 2579-2605.

[35] Kullback, S., & Leibler, R. A. (1951). On information and sufficiency. *The annals of mathematical statistics*, *22*(1), 79-86.

[36] Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., & Ruppin, E. (2001, April). Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web* (pp. 406-414). ACM.

[37] Recski, G. A., Iklódi, E., Pajkossy, K. A., & Kornai, A. (2016). Measuring semantic similarity of words using concept networks. Association for Computational Linguistics.

[38] Etcheverry, M., & Wonsever, D. (2016). Spanish Word Vectors from Wikipedia. In *LREC*.

[39] Agirre, E., Alfonseca, E., Hall, K., Kravalova, J., Paşca, M., & Soroa, A. (2009, May). A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics* (pp. 19-27). Association for Computational Linguistics.

[40] Spearman, C. (1904). The proof and measurement of association between two things. *The American journal of psychology*, *15*(1), 72-101.

[41] Kendall, M. G. (1938). A new measure of rank correlation. *Biometrika*, *30*(1/2), 81-93.

[42] Webber, W., Moffat, A., & Zobel, J. (2010). A similarity measure for indefinite rankings. *ACM Transactions on Information Systems (TOIS)*, *28*(4), 20.

[43] Bansal, M., Gimpel, K., & Livescu, K. (2014, June). Tailoring Continuous Word Representations for Dependency Parsing. In *ACL (2)* (pp. 809-815).

[44] Turian, J., Ratinov, L., & Bengio, Y. (2010, July). Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics* (pp. 384-394). Association for Computational Linguistics.

[45] Mesnil, G., Dauphin, Y., Yao, K., Bengio, Y., Deng, L., Hakkani-Tur, D., ... & Zweig, G. (2015). Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, *23*(3), 530-539.

[46] Ghannay, S., Estève, Y., Camelin, N., Dutrey, C., Santiago, F., & Adda-Decker, M. (2015, November). Combining continuous word representation and prosodic features for asr error prediction. In *International Conference on Statistical Language and Speech Processing* (pp. 84-95). Springer, Cham.

[47] Winkler, R., Klawonn, F., & Kruse, R. (2013). Fuzzy c-means in high dimensional spaces. *Int. J. Fuzzy Syst. Appl., vol*, *1*, 1-16.

[48] Bruni, E., Tran, N. K., & Baroni, M. (2014). Multimodal distributional semantics. *J. Artif. Intell. Res.(JAIR)*, *49*(2014), 1-47.

[49] Hill, F., Reichart, R., & Korhonen, A. (2015). Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, *41*(4), 665-695.

[50] Vulić, I., Gerz, D., Kiela, D., Hill, F., & Korhonen, A. (2016). HyperLex: A Large-Scale Evaluation of Graded Lexical Entailment. *arXiv preprint arXiv:1608.02117*.

[51] Selivanov D. (2017). GloVe Word Embeddings. http://text2vec.org/glove.html