Contents

List of Tables

List of Figures

Abstract

Word2vec Word Sense Disambiguation with Clustering

Word embedding is a method of mapping words to vectors to make words understandable for machines. There are various methods to map words to vectors but nowadays word2vec became one of the best and light solutions for word embedding instead of other heavy solutions.

Word Sense Disambiguation(WSD) is one of the main problems in word embedding since it hinders understanding of the information for the machines. WSD happens because basic concepts of word embedding of word2vec allow only one vector per one word. For a better understanding of the human language for machines, this problem is needed to be solved.

Several manual solutions were suggested to solve this problem but require either heavy computing power or other information more than word embedding. In this paper, the weighting method and other clustering methods are suggested to solve the WSD problem

only using word embedding of word2vec for a more simple and lighter setting for users.

Keywords: Word Sense Disambiguation(WSD), word2vec, Skip-Gram Model(SG), MultiSense Skip-Gram, Clustering

Chapter 1

Introduction

1.1 Introduction

Despite the rising tide of videos as a major data form, the text is still one of the most important types of data. The text has been useful and widespread data form since its appearance. Analyzing text data is to extract important information underneath the format. To extract information, carrying out transformation text into a vector is commonly adopted practice. The various method can be applied to the transformation process of English-data to numeric vectors.

However, the vector-space models possess a common problem called Word Sense Disambiguation(WSD). WSD is the problem of polysemy, which means one-word representation with several meanings. For example, term *bank* has different meanings. While *bank* can have the meaning of *"the land alongside or sloping down to a river or lake"*, it can also have the meaning of *"a financial establishment that invests money deposited by customers, pays it out when required, etc"*.

A variety of meanings, more practically 'sense', is disambiguated based on the context where they are used. If the term *bank* is used with the term 'money', it can be easily decided that *bank* has a sense of latter from the previous example. Using this concept, this paper introduces the method to disambiguate multiple-sense words using simple clustering methods.

The outline of this paper is as follows. In Section 2, the word embedding and transformation process will be briefly introduced as the frameworks are regarded as preliminaries. Section 3 will describe the proposal that slightly improves the concept of the previous study. Lastly, in sections 4 and 5, simulation and the analysis of the results of it will be introduced.

Chapter 2

Literature Review

2.1 Word Embedding

Word embedding is a way of mapping vectors to words. To be more precise, the definition for word embedding from the Wikipedia is *Collective name for a set of language modeling and feature learning techniques in natural language processing (NLP) where words or phrases from the vocabulary are mapped to vectors of real numbers.*

Originally, word embedding is narrowly defined as dense, continuous, and low-dimensional vector representations of words(Guo et al., 2014) or in a similar sense in many other papers. It did not contain the concept of one-hot encoding or sparse representation generally. But as many other methods and concepts are being integrated into the NLP framework, now there exist implicit agreement of definition of word embedding as Wikipedia. This can be found in the change in table of contents in other books such as "Natural Language Processing in Python(Thanaki, 2017)".

Sparse representation and Dense Representation are two categories of word embedding.

There can be several methods to carry out the conversion process, but the result will be interpreted as either sparse or dense. In either type, the polysemy problem is still likely to happen.

2.1.1 Sparse Representation

Sparse representation is a traditional way of implementing text data. The "Sparse" means that most elements of a vector are mostly zero, while only a few elements have values. One-hot encoding vector is one of the most recognized methods of sparse representation. One-hot encoding vector expresses every possible case with Independent dimensions.

One-hot encoding vector is the most simple way to convert words into vectors. First, score every word in a corpus. Then the element of a vector which stands for word becomes 1. Elements elsewhere will have a value of zero. So if there are N-word in the corpus, a vector will be N-dimensional with only one element with a value of 1. The following example shows the sample one-hot encoding vector.

$$\text{cat} \Rightarrow \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \text{dog} \Rightarrow \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \text{kitty} \Rightarrow \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

The drawback of the one-hot encoding vector is that it can not disambiguate the semantic relationship between synonyms. For example, given the words 'cat' and 'kitty', one-hot encoding vector can not differentiate the relationship between 'cat and kitty' and 'cat and dog' because of orthogonality of the representation even if the existence of the

relationship is obvious. It can be concluded that this method can not reflect the semantic relationship between words.

2.1.2 Dense Representation

Dense representation is a vector that is no more sparse. Most of the elements have values other than 0. The 'dense' is used since it is the opposite of 'sparse'. Sometimes it is called 'Distributed representation' in a sense that a word is represented in several semantic dimensions.

Different from sparse representation, elements of dense representation are combined to represent the properties of the designated word. In other words, elements are not necessarily independent. Also, a word can be represented with much fewer dimensions that sparse representation requires which can avoid the curse of dimensionality. This means that dense representation can have more generalization power than the sparse representation does. The following examples are a sample of dense representation.

$$\text{cat} \Rightarrow \begin{bmatrix} 0.7 \\ -0.3 \\ -0.5 \\ 0.42 \\ 0.73 \end{bmatrix} \quad \text{dog} \Rightarrow \begin{bmatrix} 0.65 \\ 0.4 \\ 0.5 \\ 0.3 \\ 0.71 \end{bmatrix} \quad \text{kitty} \Rightarrow \begin{bmatrix} 0.7 \\ -0.34 \\ -0.45 \\ 0.4 \\ 0.34 \end{bmatrix}$$

By using dense representation, the representations are no more orthogonal that it can be possible to express relationship.

The drawback of dense representation is that what exact attributes a dimension stands for can not be known in most cases. It is a common problem that every deep-learning models possess, but it can be fatal if any experiment is heavily concerned about interpretability.

Despite these facts, the methods with dense representation are widely adopted and proven to be effective in many cases.

2.2 word2vec

The word2vec model and application suggedsted by Tomas Mikolov (2013) produce vector representation that has been shown to carry semantic meanings. The results are proven to be effective in various NLP tasks. Word2vec model adopts autoencoder to produce word embedding. Word embedding can learn compressed information in a lower dimension by using an undercomplete autoencoder. Fact that a one-hot encoding vector is used in the input layer makes possible to specify a column of the hidden layer as the word embedding.

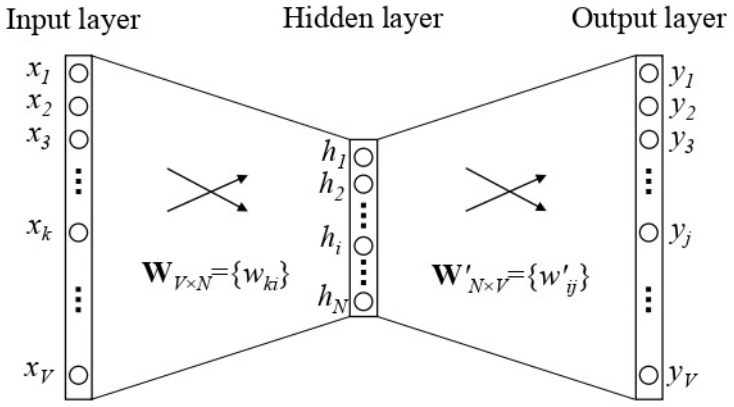

Figure 2.1: Basic Autoencoder

(Rong, 2014)

The learning process of word2vec is analytically explained at word2vec parameter learning expalined(Rong, 2014). The parameter learning process of the word2vec model is using a simple neural network with advanced optimization techniques. Computation time is minimized without harming the quality of the word embedding by using hierarchical

softmax and negative sampling. The word2vec introduces 2 models to process the word embedding, which is Continuous Bag-of-Word Model and Skip-gram Model. Word2vec takes a sigmoid function for its computation efficiency.

The basic computation process is as follows. Assume a network model with a simple context where Vocabulary size is V and hidden layer size N. Vocabulary means the dictionary of the words at the corpus. The units near the layers are fully connected. One-hot encoding vector will be input, making only one out of V units, $\{x_1, \cdots x_V\}$, 1. Row of matrix W from Figure 2.1 is N-dimension vector representation v_w of the associated word of the input layer. Then v_w^T is row i of W.

$$h := W^T x = W^T_{(k,)} = v^T_{w_I} \tag{2.1}$$

given a context assuming $x_k = 1$ and $x_{k'} = 0$ for $k' \neq k$ that equals copying the k-th row of W to h. $v^T_{w_I}$ becomes the vector representation of the input. This means that link function between input and hidden layer units is simple linear.

From hidden to output layer, different weight matrix $W'_{N\times V} = \{w'_{ij}\}$ is defined. Score u_j for each word in the vocabulary can be computed as

$$u_j = {v'_{w_j}}^T h, \tag{2.2}$$

where v'_{w_j} is the j-th column of the weight matrix W'. To obtain the posterior distribution of words, apply softmax on the equation. Then the equation becomes

$$p(w_j|w_I) = y_j = \frac{exp(u_j)}{\sum_{j'=1}^{V} exp(u_{j'})} \tag{2.3}$$

where y_j is the output of the j-th unit in the output. By putting (2.1), (2.2) and (2.3), following equation

$$p(w_j|w_I) = \frac{exp({v'_{w_j}}^T v_{W_I})}{\sum_{j'=1}^{V} exp({v'_{w_j}}^T v_{W_I})} \tag{2.4}$$

can be obtained.

The training objective is to maximize the conditional probability of observing the actual word w_O given input context word w_I, then the equation is

$$maximize\ p(w_O|w_I) = y_j = \frac{exp({v'_{w_j}}^T v_{W_I})}{\sum_{j'=1}^{V} exp({v'_{w_j}}^T v_{W_I})} \tag{2.5}$$

where j^* is the index of the actual output word in output layer. Then the loss function will be

$$max\ p(w_O|w_I) = max\ y_{j^*} = max\ log\ y_{j^*} \tag{2.6}$$

$$= u_{j^*} - log \sum_{j'=1}^{V} exp(u'_j) := -E \tag{2.7}$$

$$\iff E = -log\ p(w_O|w_I) \tag{2.8}$$

$$\therefore max\ p(w_O|w_I) = min\ E. \tag{2.9}$$

This loss function is one of special cases of the cross-entropy measurement between tow probability distributions. The rest of the process will follow the procedure of the back propagation.

2.2.1 Continuous Bag-Of-Words(CBOW)

CBOW model of Figure 2.2 shows the multi-context setting of the learning model. The target word is set at the output layer and context words are set to the input layer. This model takes the average of the vectors of the input context words to train the model. Through this process, the model can be trained better than a one-word context model.

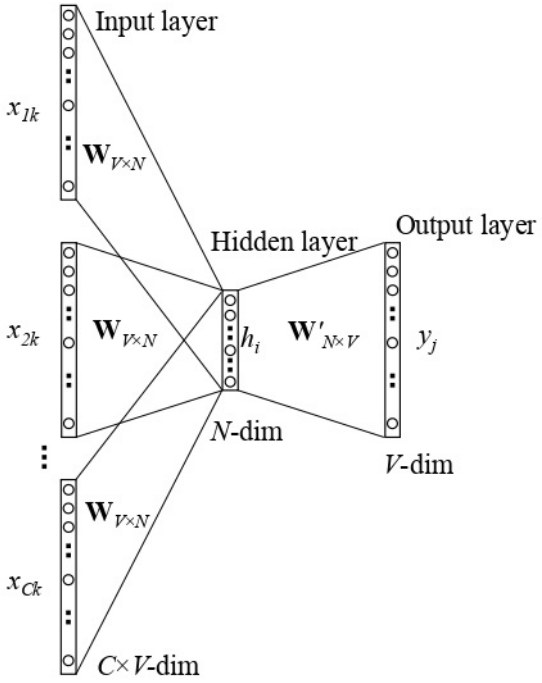

Figure 2.2: CBOW Model

(Rong, 2014)

The equations are as follow.

$$h = \frac{1}{M} W^T (x_1 + x_2 + \cdots x_M) \tag{2.10}$$

$$= \frac{1}{M} W^T (v_{w_1} + v_{w_1} + \cdots v_{w_M})^T \tag{2.11}$$

where M is the number of words in the context, $w_1, \cdots, w_M$ are context words. The loss function is updated to

$$E = -log\ p(w_O | w_{I,1}, \cdots, w_{I,M}) \tag{2.12}$$

$$= -u_{j^*} + log \sum_{j'=1}^{V} exp(u'_j) \tag{2.13}$$

$$= -{v'_{w_O}}^T \cdot h + log \sum_{j'=1}^{V} exp({v'_{w_O}}^T \cdot h) \tag{2.14}$$

which is same as (2.9) only that h is different.

2.2.2 Skip-Gram(SG)

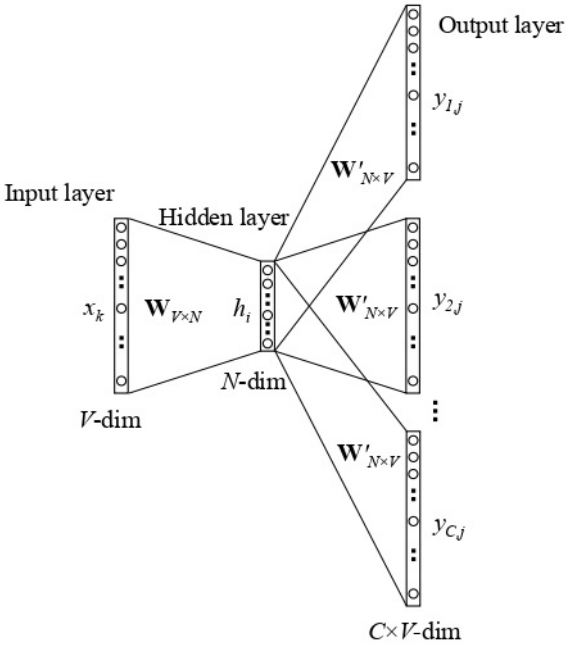

Figure 2.3: Skip-Gram Model

(Rong, 2014)

SG model of Figure 2.3 is the opposite of the CBOW model. The target word is set at the input layer. SG model generally shows better results at training the word embedding. Intuitively, letting target words to learn the context words will be better and composite than the opposite way. For SG, h is same as the (2.9). But on the output layer, instead of outputting one multinomial distribution, SG produces M multinomial distributions.

$$p(w_{M,j} = w_{O,M}|w_I) = y_{M,j} = \frac{exp(u_{c,j})}{\sum_{j'=1}^{V} exp(u_{j'})} \tag{2.15}$$

where $w_{M,j}$ is the j-th word on the c-th panel of the output layer; $w_{O,m}$ is the actual c-th word in the output context words; w_I is the only input word; $y_{m,j}$ is the output of the

j-th unit on the c-th panel of the output layer; $u_{m,j}$ is the net input of the j-th unit on the c-th panel of the output layer. Since the output layer panels share same weight,

$$u_{M,j} = u_j = {v'_{w_j}}^T \cdot h \text{ for } M = 1, 2, \cdots, M \tag{2.16}$$

where v'_{w_j} is output vector of the j-th word in vocabulary, w_j. Then the parameter update equation is slightly changed from the basic autoencdoer. The loss function is modified to

$$E = -log\ p(w_{O,1}, w_{O,2}, \cdots, w_{O,M}|w_I) \tag{2.17}$$

$$= -log \prod_{m=1}^{M} \frac{exp(u_{m,j^*_m})}{\sum_{j'=1}^{V} exp(u_{j'})} \tag{2.18}$$

$$= -\sum_{m=1}^{M} u_{m,j^*_m} + M \cdot log \sum_{j'=1}^{V} exp(u_{j'}) \tag{2.19}$$

where j^*_m is the index of the actual m-th output context word in the vocabulary.

However, the results of word2vec obtain the problem of polysemy, since it can not differentiate the senses of a word. The concept of word embedding itself only allows only one vector representation for one letter representation, thus there must be a problem of polysemy. Polysemy is the problem of one word with multiple meanings or multiple senses. Trying to solve this problem, the following solution is suggested.

2.3 MultiSense Skip-Gram(MSSG)

The MultiSense Skip Gram(MSSG) (Neelakantan et al., 2015) is more developed version of the SG. At MSSG, terms have SG vectors as their dense representation and also cluster sense from the context vectors. Sense vectors can be gained by averaging the global vectors of the context words of the target word. The basic idea of generating MSSG is to carry

out CBOW based process with the SG generated vectors. The equation is as follows.

$$\frac{1}{2 \times R_t} \sum_{c \in c_t} V_g(c) \text{ where } C_t = \{W_{t-R_t}, ..., W_{t-1}, W_{t+1}, ..., W_{t+R_t}\}, \tag{2.20}$$

where R_t is the size of window, t is the position of target word in sentence and c is the target word. Also W_i will be redefined as i-th word in a sentence.

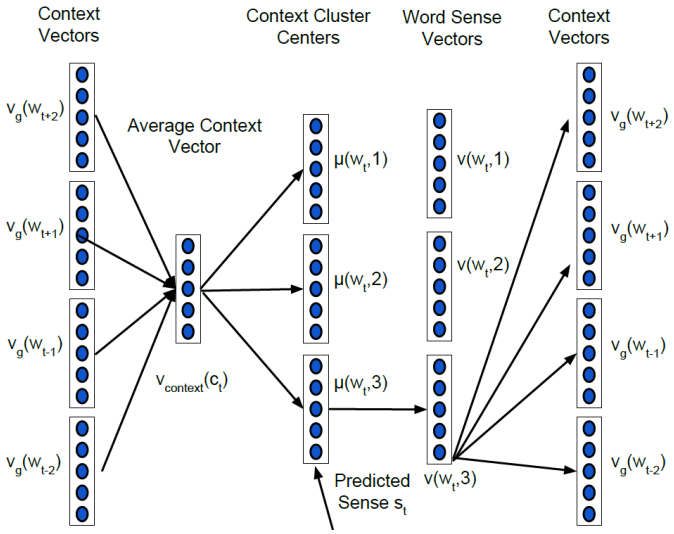

Figure 2.4: Multi Sense Skip-Gram Model

(Neelakantan et al., 2015)

Then by clustering the context vectors, the senses of the target word can be known. The paper proposes a parametric version and a non-parametric version of the process, this paper will focus on the parametric version for more statistical sense. The parametric version is simply carrying out k-means clustering on the sense vectors with cosine distance. It can be understood as carrying out CBOW procedure on SG generated vectors to produce sense vector and cluster sense. So by giving the resulting sense as one attribute with the global vector, the user will be informed which sense the word belongs to.

2.3.1 Sliding Window

Originally from MSSG, the sense vector is defined as

$$\frac{1}{2 \times R_t} \sum_{c \in c_t} V_g(c) \text{ where } C_t = \{W_{t-R_t}, ..., W_{t-1}, W_{t+1}, ..., W_{t+R_t}\}.$$

It is practical to adopt as many context as a word can get for better word2vec result. If window of both sides are not equal, the context will not be arbitrarily truncated to match window size, but use every context disregard of different window size. So the same procedure is needed to be applied on building context vector. Originally suggested equation does not reflect this characteristic, so the modified equation is as follows.

$$\frac{1}{\text{Cardinality}(C_t)} \sum_{c \in c_t} V_g(c) \text{ where } C_t = \{W_{t-R_t}, ..., W_{t-1}, W_{t+1}, ..., W_{t+R_t}\}. \tag{2.21}$$

2.3.2 Clustering

MSSG process k-means with cosine distance, more precisely based on cosine similarity. This clustering method is usually called spherical k-means. The Cosine similarity is the measure usually adopted for high-dimensional text data rather than the Euclidean distance. This implies that the direction of a vector is more important than the magnitude.

Standard k-means clustering is to minimize the mean-squared error.

$$E = \frac{1}{N} \sum_{x} ||x - \mu_{k(x)}||^2, \tag{2.22}$$

where N is the total number of vectors and $k(x) = \underset{k \in \{1, \cdots, k\}}{argmin} ||x - \mu_{k(x)}||$ is the index of the closest cluster centroid to x. It is known that standard k-means algorithm is based on Gaussian. On the other hand, k-means with cosine similarity, also called k-means on a hypersphere normalize each vectors to have unit length. The objective function to

maximize is

$$L = \sum_{x} x^T \mu_{k(x)} \tag{2.23}$$

where $\{\mu_1, \cdots \mu_k\}$ is a set of unit-length centroid vectors and $k(x) = \underset{k}{argmax}\, x^T \mu_{k(x)}$.

Chapter 3

Proposed Method

3.1 Motivation

Word2vec is a very common and easy method to carry out word-embedding. Then, to solve the problem of WSD, several methods are proposed such as using POS. MSSG is one of the proposed methods and has good performances. The basic concept is excellent but there seems a slight chance of supplementation. Also, there had not been a lot of explanatory research about this problem.

3.2 Weighting Context

3.2.1 Weight by Position

While constructing Skip-Gram from the corpus, a vector of a term is weighted by the context terms near it. In other words, if the same context term keeps making its appearance in many other cases, the vector of a term will get placed close to the context term. It can be said that the vector of a term is weighted by frequency.

However, every element of context has the same weight at MSSG while constructing a sense vector. The concept of weight by frequency is deteriorated since this sense vector mainly focuses on the single specific term. Point is that building a context vector for a specific term in a document should be different from building a global vector for a term in the corpus.

For example, assume two partial sentences "...power plant near the forest" and "...plant habitat near nuclear facility". The first sentence will have both forest and power as context with each frequency 1. The context vector will not be clear because of the same weight. The second sentence has the term plant as 1 tree context and 2 power context with the meaning of tree. But since there are more power contexts than tree context, it will be classified as power even though it is a tree context.

Giving weight by position can be a breakthrough to this kind of problem. This concept is based on the assumption that more related word appears closer in sentences just like examples above. The bigger weight will be assigned to the word that is closer to the target word.

Various combinations of weighting method can be applied, but the applied weight by position at this paper will be simply described as

$$WeightVector = \{1, 2, 3, 4, 5, 5, 4, 3, 2, 1\}. \tag{3.1}$$

But this form does not fully express the weight if there exist truncation on the context window. More precise form of equation for this vector will be

$$weight_j = d + 1 - |j| \text{ where } j = -R_t, -R_t + 1, \cdots, -1, 1, \cdots, R_t - 1, R_t \tag{3.2}$$

where $c \in c_t$. In this way, the truncation can be fully expressed in equation. This equation is just made to describe the above vector, so if to implement different weight, equation

can be arbitrarily modified.

From subsection of Sliding Window, recall equation

$$\frac{1}{\text{Cardinality}(C_t)} \sum_{c \in c_t} V_g(c) \text{ where } C_t = \{W_{t-R_t}, ..., W_{t-1}, W_{t+1}, ..., W_{t+R_t}\}.$$

To Implement the weight by position, the equation needs slight modification to be exact. The new implemented method will be as following. First multiplication of weight to the sense vector is needed. Then denominator of the flex context equation need to be changed from Count(C_t) to $\sum_{c \in c_t} weight_c$. So the final equation will be

$$\frac{1}{\sum_{c \in c_t} weight_c} \sum_{c \in c_t} weight_j \times V_g(c) \text{ where } C_t = \{W_{t-R_t}, ..., W_{t-1}, W_{t+1}, ..., W_{t+R_t}\}. \quad (3.3)$$

.

3.2.2 Clustering

On the parametric version of MSSG, it clusters the sense vector with the method of k-means. The difference from the original k-means is only that it uses Cosine distance instead of Euclidean distance. It is practical to use Cosine distance at Information Retrieval and many other fields involving information.

Using the hierarchical clustering method is also a practical way in Information Retrieval since information is not something discrete. Hierarchical Clustering Method well reflects this characteristic since it makes clusters continuously. It can be assumed that sense has similar continuity property as information so that the Hierarchical Clustering Method can be applied.

Hierarchical Clustering requires distance or similarity between observations. Then according to the preset linkage method, forms the clustering dendrogram. Then by cutting the tree of a dendrogram, the clustering results can be acquired. A and B are two vectors

of elements where A_i and B_i are each component of vector A and B. Then Euclidean distance is defined as follows.

$$d(A, B) = ||A - B||_2 = \sqrt{\sum_i (A_i - B_i)^2}. \tag{3.4}$$

At same condition, Cosine similarity is defined as follows.

$$d(A, B) = \frac{A \cdot B}{||A||\,||B||} = \frac{\sum_i A_i B_i}{\sqrt{\sum_i A_i^2}\sqrt{\sum_i B_i^2}}. \tag{3.5}$$

The linkage are will be used as follow.

Names	Formula
Single	$min\{d(a, b) : a \in A, b \in B\}$
Average	$\frac{1}{\lvert A\rvert \cdot \lvert B\rvert} \sum_{a \in A} \sum_{b \in B} d(a, b)$
Complete	$max\{d(a, b) : a \in A, b \in B\}$

Table 3.1: Linkage for Hierarchical Method

From k-means to Hierarchical clustering will be compared on this paper. For the k-means, both Euclidean and Cosine will be compared. For Hierarchical, many linkage methods can be adopted with also two different distance matrix. Performance needs to be re-examined to find out which clustering method is better for the clustering contexts.

Chapter 4

Simulation Study

4.1 Data Generation

In this section, the simulation will be conducted to compare the performance of clustering methods on both weighted and non-weighted context vectors. The term 'plant' is set up as the target word for the analysis. The plant is a term that has two unrelated meanings. The two meanings are *"a living thing that grows in earth, in water, or on other plants, usually has a stem, leaves, roots, and flowers, and produces seeds"* and *"a factory in which a particular product is made or power is produced"* according to the Cambridge Dictionary. To simplify, the former will be labeled 'tree' and the latter will be 'power'.

4.1.1 Corpus

Corpus will be generated from the abstracts of the *Journal of Statistical Software*(JSS) since it provides code to extract information through simple tasks on `R`. The code can be found on 4th chapter from "topicmodels: An `R` Package for Fitting Topic Models"(Hornik and Grün, 2011). Using the code provided by JSS, data of JSS papers can be downloaded

through R. This JSS paper data is updated after every publication or conference. For reproducible research, the sample of JSS paper is limited by publication data under 12/31/2018. Also, the whole process had taken place under the C locale for reproducibility according to the original code. The provided code also carry out simple HTML_markup removal task.

Under this condition, a total of 884 documents can be collected from JSS. Check for the target word existence is needed at this process. The term 'plant' is found in 4 documents from the corpus and marked whether the sense belongs to tree or power. At this experiment, 4 papers all belong to the sense of the tree.

4.1.2 Target Word Injection

The target word will be injected into the corpus. Since the corpus does not have enough terms of the plant in it, injection of the selected abstracts will be taken place. The abstracts are selected according to whether they contain the term plant and designated sense. 10 abstracts which contain plant with a sense of tree and 10 abstracts which contains plants with a sense of power are injected into the corpus. So there are 14 tree-related abstracts including 4 abstracts already existed and 10 power-related abstracts out of 904 documents.

Category	Titles of the Abstracts
jss tree	Estimating and Analyzing Demographic Models Using the popbio Package in R
jss tree	The ecodist Package for Dissimilarity-based Analysis of Ecological Data
jss tree	solaR: Solar Radiation and Photovoltaic Systems with R
jss tree	POPS: A Software for Prediction of Population Genetic Structure Using Latent Regression Models
tree	A kiwellin disarms the metabolic activity of a secreted fungal virulence factor
tree	A male-expressed rice embryogenic trigger redirected for asexual propagation through seeds
tree	A small peptide modulates stomatal control via abscisic acid in long-distance signalling
tree	Bacterial cGAS-like enzymes synthesize diverse nucleotide signals
tree	Enhanced plant growth by siderophores produced by plant growth-promoting rhizobacteria
tree	Farming with crops and rocks to address global climate, food and soil security
tree	Long-term effects of species loss on community properties across contrasting ecosystems
tree	Mobile PEAR transcription factors integrate positional cues to prime cambial growth
tree	Modulating plant growth metabolism coordination for sustainable agriculture
tree	Plant behaviour from human imprints and the cultivation of wild cereals in Holocene Sahara
power	Application of Fuzzy Comprehensive Evaluation on COGAG Power Plant of Performance
power	Extension Study on Electricity Market of Power Plant Investment Environment
power	Feasibility Analysis of Constructing Solar Power Plant by Combining Large Scale Wind Farm
power	Human Reliability Analysis for Digitized Nuclear Power Plants Case Study on the LingAo II-Nuclear Power Plant
power	Innovative Nuclear Power Building Arrangement in Consideration of Decommissioning
power	Performance of Regenerative Gas Turbine Power Plant
power	Performance of a 270 MW Gas Power Plant Using Exergy and Heat Rate
power	Sequential Probability Ratio Test for Nuclear Plant Component Surveillance
power	The Cognitive and Economic Value of a Nuclear Power Plant in Korea
power	The Optimal Steam Pressure of Thermal Power Plant in a Given Load

Table 4.1: Titles of Abstracts with the word ‘plant’.

4.2 Preprocess Text Data

Before the implementation of word2Vec on the corpus, the data needed to be properly preprocessed. Preprocessing and word2Vec implementation tasks will be done under the `Python` environment.

First of all, sentences are tokenized using `nltk` package from the `Python`. `Gensim` also provide similar tokenization function, but tokenizer from `nltk` is much easier to handle. On the other hand, the function of split sentences from `Gensim` is adopted since it provides a better result than the sentence splitter from `nltk`. Punctuation is removed from the corpus using `Python`.

Stemming is a process that morphs back the words to their stem, base or root form. Before using word2vec, stemming must be done. For example, if stemming is not done before implementing word2vec, the model will differentiate such words, 'do' and 'did' since they have different expressions. This procedure is done using `Porter` stemmer for its wide use.

4.3 Word2vec Implementation

The word2vec result can be generated manually through `tensorflow` or `keras`, but much faster `Python` implementation is already set for open source. The word2vec model from the `Gensim` is used to produce the results. To be more precise on the comparison, the word2vec dimension is set from 100 to 500 at intervals of 50. The window size of 5 is used since it is generally adopted.

Also, the removal of words that are under minimum number is set to compare the performance since the removal is common practice on carrying out word2vec. The randomness

of words that appears only a few at the corpus can heavily affect the model negatively. Then word2vec is generated by `Gensim` from the `python`. Using created word2vec, generate sense vectors to carry out clustering comparison.

	Corpus			Vocabulary		
Minword	Total	Retain	Ratio	Total	Retain	Ratio
Min 1	118816	118816	100%	6803	6803	100%
Min 2	118816	115762	97%	6803	3749	55%
Min 3	118816	113784	95%	6803	2760	40%
Min 5	118816	110835	93%	6803	1900	27%

Table 4.2: Corpus Vocabulary comparison with Minword

The table shows the characteristics of the Corpus consisted of Abstracts. The Minword means that the condition of minimum word frequency required to remain in the model. When the minimum word frequency is set to 2, 45% of the vocabularies are gone though the total word frequency decreased by 3%. This states that abstracts use lots of unique words so proper consideration of this property is needed at analyzing the results. The setting of a minimum word frequency of 5 is stated for comparison purposes since many analyses usually adopt the designated setting for big size corpus. The setting is not used in this paper since it deletes too many vocabularies in this experiment.

	0	1	2	3	4	5	6	7	8	9	...	190
the	-0.005274	-0.004881	0.008978	0.016738	-0.196532	0.039860	-0.008283	0.057658	-0.013314	0.038370	...	-0.056950
of	-0.046892	0.078688	0.009536	0.001345	-0.150550	0.070457	0.033300	-0.055566	0.050849	-0.038141	...	0.091164
and	-0.013463	-0.099499	0.057102	0.047343	-0.056065	0.034342	-0.037420	-0.044739	-0.029994	0.022349	...	-0.111621
a	0.088975	-0.026309	-0.010853	-0.052011	0.015842	0.121120	-0.012029	-0.061442	0.029275	0.061125	...	-0.051282
to	-0.001451	-0.096425	-0.050938	0.123212	-0.034227	0.015846	0.039572	-0.032654	0.065712	0.057349	...	0.020379
in	-0.010377	-0.155409	0.080253	-0.034712	-0.024648	0.038298	0.070546	0.062922	0.076265	0.131673	...	-0.002128
for	-0.014069	0.072710	-0.002113	0.022438	-0.108717	0.017320	-0.026740	-0.092279	-0.001959	0.007090	...	-0.031778
model	0.059081	-0.002837	0.087326	-0.148797	0.010424	0.093209	0.036014	0.058492	0.017576	-0.011537	...	-0.053278
is	0.032415	0.009137	0.043433	-0.025170	-0.133684	0.002041	0.062189	-0.048085	0.012151	-0.076218	...	-0.024664
packag	-0.011378	0.042784	-0.015387	0.050004	-0.100221	-0.027605	-0.023230	0.132628	0.022325	0.110128	...	-0.016424
use	-0.003510	-0.051181	-0.003584	-0.038074	0.051430	-0.022697	-0.086584	0.010690	0.005365	0.065578	...	-0.098157
data	-0.062955	-0.016199	0.081558	-0.051680	-0.094418	0.056436	0.189726	0.036928	0.093235	0.057148	...	-0.073167
are	0.027333	0.062075	0.013693	-0.072706	-0.026380	0.030791	0.026584	0.014936	-0.112236	-0.132439	...	0.001078
r	0.041546	0.049575	0.045000	-0.110312	-0.037867	-0.004644	-0.033628	0.128619	0.076034	0.028486	...	-0.036084
thi	0.026243	0.016053	-0.007376	-0.104401	-0.066868	-0.063286	0.082070	-0.009953	0.043020	0.083790	...	-0.034547
with	-0.014180	0.031629	0.002237	0.018051	-0.054588	0.161907	0.020293	-0.024866	0.027719	0.140517	...	-0.001839
as	0.025814	0.055456	-0.024011	-0.091078	-0.028687	0.026749	-0.116749	-0.094028	0.105277	0.074279	...	-0.150292
that	-0.027353	0.024537	-0.016085	-0.046037	-0.092417	-0.011636	-0.014884	-0.007575	0.047328	0.003559	...	-0.000196
be	-0.022672	-0.043258	-0.018999	-0.070277	0.031477	0.011959	0.043153	-0.062431	0.070694	0.014364	...	0.029787
we	-0.024113	0.105714	-0.049783	0.054866	0.026242	-0.036484	-0.077901	-0.047014	0.039281	0.000504	...	-0.028556
on	-0.114511	-0.080015	-0.007271	0.042869	-0.021789	0.075222	0.048453	-0.074878	0.097548	-0.045232	...	-0.046449
method	0.035439	0.085446	0.031535	0.065097	-0.137242	0.034622	0.024615	0.044898	0.030258	0.057919	...	-0.003838
an	0.073803	-0.031584	0.087269	-0.025697	-0.098544	-0.081930	-0.012254	0.011796	0.027309	0.066877	...	0.014081

Figure 4.1: Sample SG Results

4.4 Clustering Result

Clustering results are generated by package `factoextra` from `R`. The distance matrix can be manually calculated, but package `proxy` provides several options to produce various types of the distance matrix. Using `factoextra` and `proxy`, the clustering results are generated. Since the sense of the words is known, the classification results could be evaluated. The results are as follows.

4.4.1 Minword 1

Minword 1 Distance	k-means Clustering Euclidean		Cosine	
Dim	Ordinary	Proposal	Ordinary	Proposal
100	75.71%	84.29%	78.57%	82.86%
150	82.86%	85.71%	80.00%	81.43%
200	82.86%	84.29%	78.57%	81.43%
250	82.86%	85.71%	80.00%	82.86%
300	84.29%	84.29%	80.00%	82.86%
350	85.71%	84.29%	80.00%	82.86%
400	82.86%	84.29%	80.00%	82.86%
450	84.29%	84.29%	80.00%	82.86%
500	84.51%	81.43%	80.00%	82.86%

Table 4.3: k-means Clustering Accuracy with Minword 1

Minword 1 Linkage	Euclidean Hierarchical Clustering Single		Average		Complete	
Dim	Ordinary	Proposal	Ordinary	Proposal	Ordinary	Proposal
100	52.86%	52.86%	61.43%	61.43%	64.29%	65.71%
150	52.86%	52.86%	54.29%	52.86%	71.43%	74.29%
200	54.29%	52.86%	54.29%	54.29%	62.86%	71.43%
250	54.29%	52.86%	54.29%	52.86%	61.43%	70.00%
300	54.29%	52.86%	54.29%	54.29%	67.14%	67.14%
350	52.86%	54.29%	54.29%	54.29%	60.00%	72.86%
400	52.86%	52.86%	54.29%	54.29%	60.00%	70.00%
450	54.29%	54.29%	54.29%	54.29%	67.14%	71.43%
500	52.86%	54.29%	54.29%	54.29%	72.46%	71.43%

Table 4.4: Euclidean Hierarchical Clustering Accuracy with Minword 1

Minword 1	Cosine Hierarchical Clustering					
Linkage	Single		Average		Complete	
Dim	Ordinary	Proposal	Ordinary	Proposal	Ordinary	Proposal
100	52.86%	52.86%	52.86%	57.14%	60.00%	61.43%
150	51.43%	51.43%	51.43%	57.14%	78.57%	62.86%
200	52.86%	52.86%	51.43%	58.57%	75.71%	87.14%
250	51.43%	51.43%	51.43%	51.43%	81.43%	54.29%
300	51.43%	51.43%	54.29%	51.43%	71.60%	87.14%
350	51.43%	52.86%	54.29%	51.43%	87.14%	71.43%
400	51.43%	51.43%	54.29%	51.43%	80.00%	90.00%
450	51.43%	52.86%	54.29%	51.43%	62.86%	88.57%
500	51.43%	52.86%	54.29%	51.43%	51.43%	90.00%

Table 4.5: Cosine Hierarchical Clustering Accuracy with Minword 1

For minword 1, the hierarchical clustering method does not perform better than the k-means clustering. Some hierarchical clustering method with cosine distance sometimes fit better, but mostly k-means methods perform better. Generally, the weighted clustering method performs better at clustering sense than the previous model.

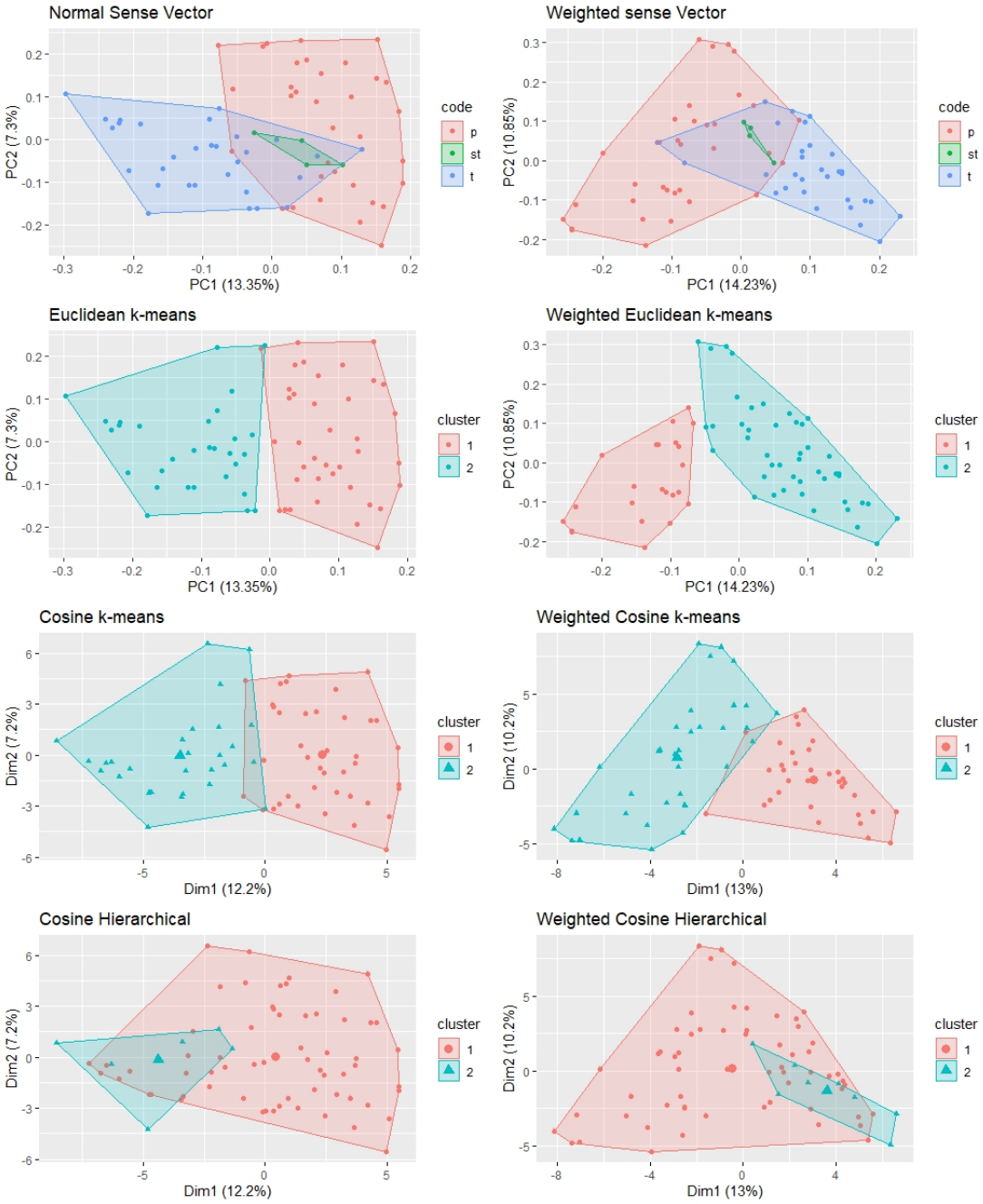

Figure 4.2: Clustering Visualization with Minword 1, Dim 100

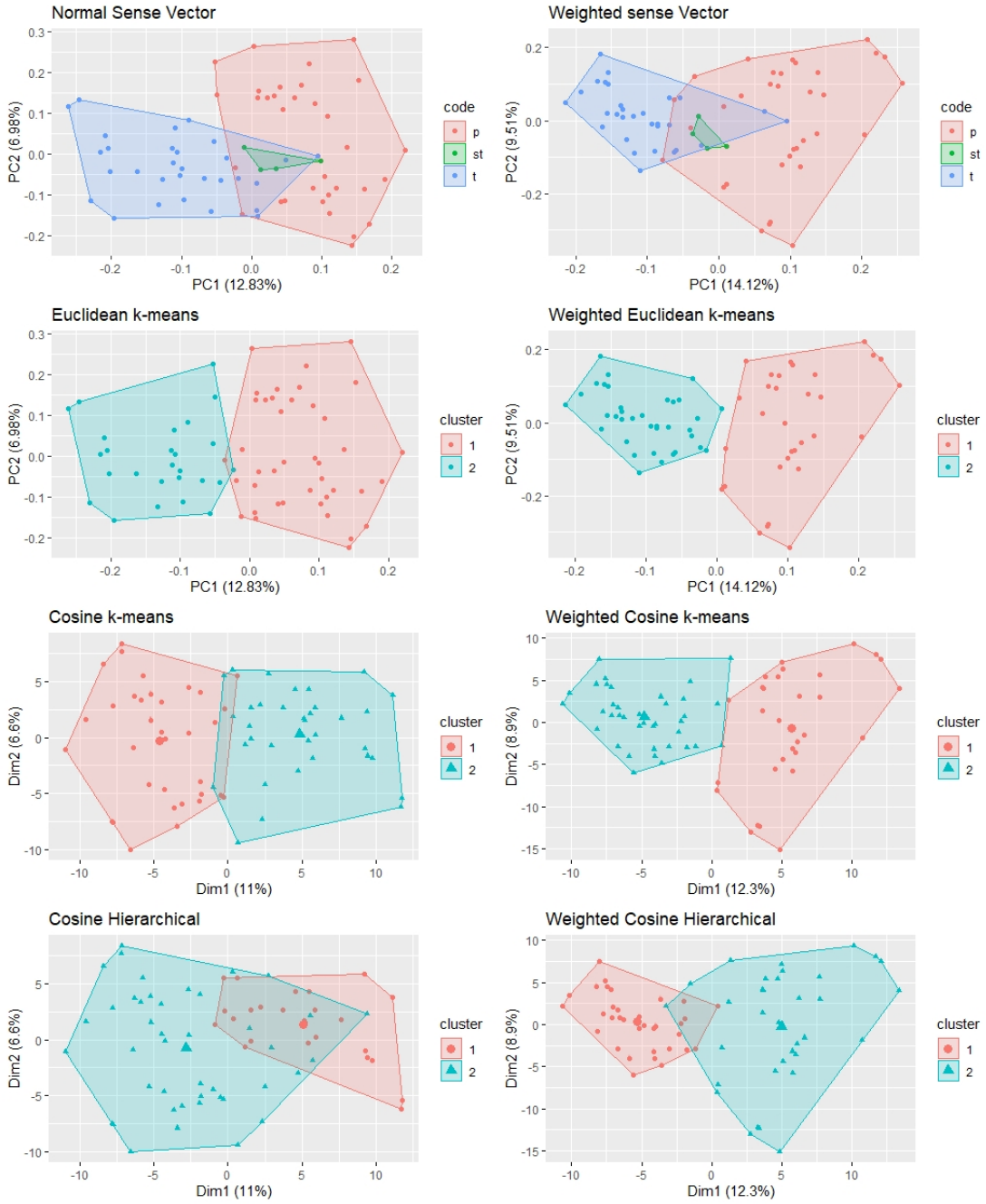

Figure 4.3: Clustering Visualization with Minword 1, Dim 300

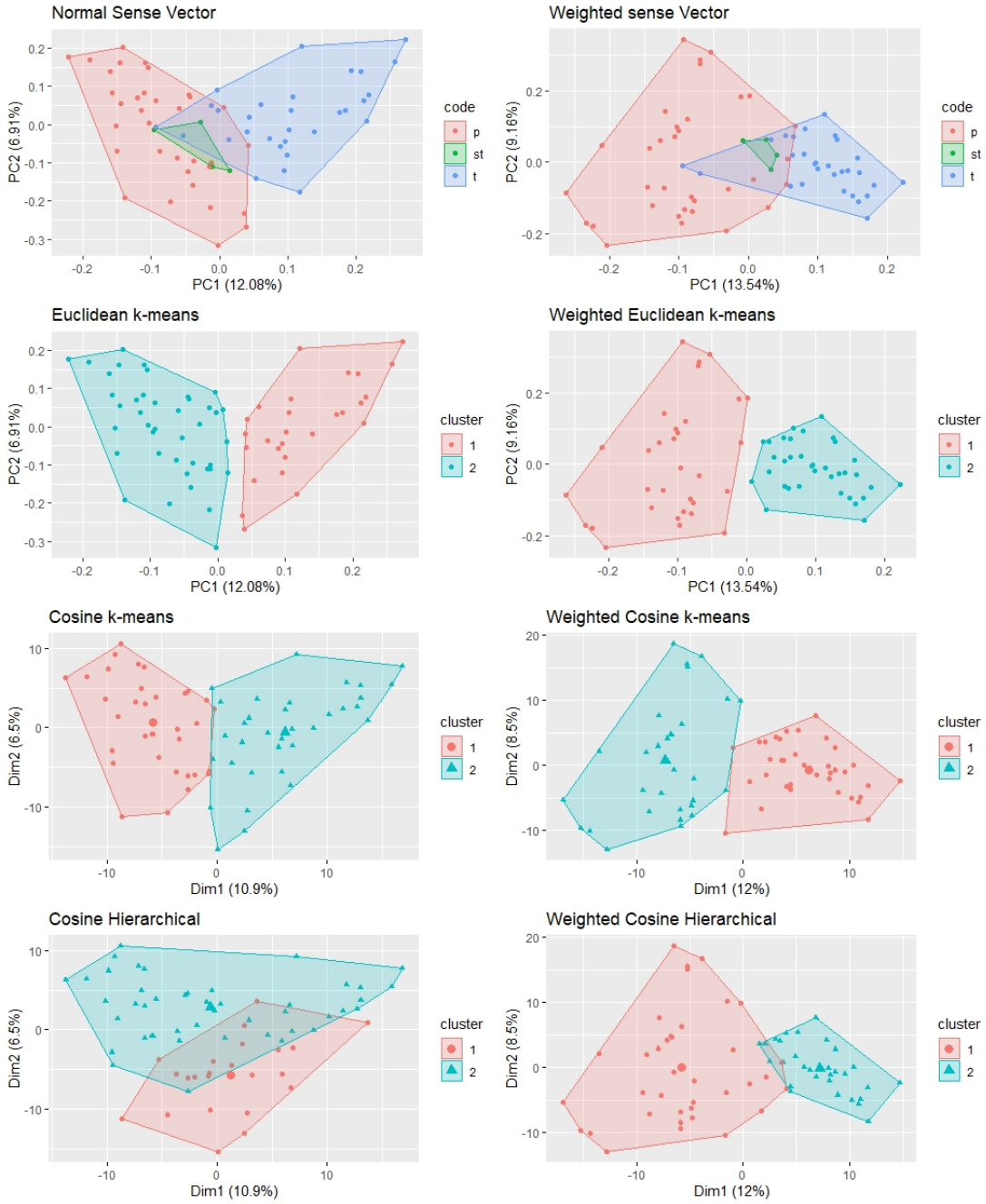

Figure 4.4: Clustering Visualization with Minword 1, Dim 500

4.4.2 Minword 2

Minword 2 Distance	k-means Clustering Euclidean		Cosine	
Dim	Ordinary	Proposal	Ordinary	Proposal
100	80.00%	71.43%	64.29%	80.00%
150	77.14%	82.86%	65.71%	75.71%
200	81.43%	84.29%	65.71%	81.43%
250	83.33%	74.29%	81.43%	82.86%
300	82.86%	75.71%	70.00%	82.86%
350	74.29%	74.29%	78.57%	74.29%
400	74.29%	81.43%	78.57%	74.29%
450	84.29%	84.29%	78.57%	82.86%
500	84.29%	84.29%	78.57%	82.86%

Table 4.6: k-means Clustering Accuracy with Minword 2

Minword 2 Linkage	Euclidean Hierarchical Clustering Single		Average		Complete	
Dim	Ordinary	Proposal	Ordinary	Proposal	Ordinary	Proposal
100	52.86%	52.86%	54.29%	54.29%	68.57%	75.71%
150	52.86%	52.86%	54.29%	52.86%	54.29%	55.71%
200	52.86%	52.86%	54.29%	54.29%	54.29%	57.14%
250	54.29%	52.86%	54.29%	54.29%	54.29%	81.43%
300	52.86%	52.86%	54.29%	54.29%	74.29%	78.57%
350	52.86%	52.86%	54.29%	54.29%	67.14%	51.43%
400	52.86%	52.86%	54.29%	54.29%	62.86%	85.71%
450	52.86%	52.86%	54.29%	54.29%	54.29%	55.71%
500	52.86%	52.86%	54.29%	54.29%	61.43%	58.57%

Table 4.7: Euclidean Hierarchical Clustering Accuracy with Minword 2

Minword 2	Cosine Hierarchical Clustering					
Linkage	Single		Average		Complete	
Dim	Ordinary	Proposal	Ordinary	Proposal	Ordinary	Proposal
100	51.43%	52.86%	52.86%	57.14%	65.71%	81.43%
150	52.86%	52.86%	57.14%	55.71%	57.14%	57.14%
200	52.86%	52.86%	55.71%	57.14%	75.71%	61.43%
250	52.86%	52.86%	57.14%	54.29%	77.14%	71.43%
300	52.86%	52.86%	57.14%	51.43%	77.14%	62.86%
350	52.86%	52.86%	52.86%	51.43%	75.71%	82.86%
400	51.43%	52.86%	52.86%	52.86%	75.71%	62.86%
450	52.86%	52.86%	54.29%	61.43%	77.14%	62.86%
500	52.86%	52.86%	54.29%	52.86%	80.00%	62.86%

Table 4.8: Cosine Hierarchical Clustering Accuracy with Minword 2

For minword 2, also hierarchical clustering method does not perform better than the k-means clustering. But at the setting of minword 2, the effect of the weighting strategy is not clear. In some cases, it brings the performance higher but opposite for some other cases.

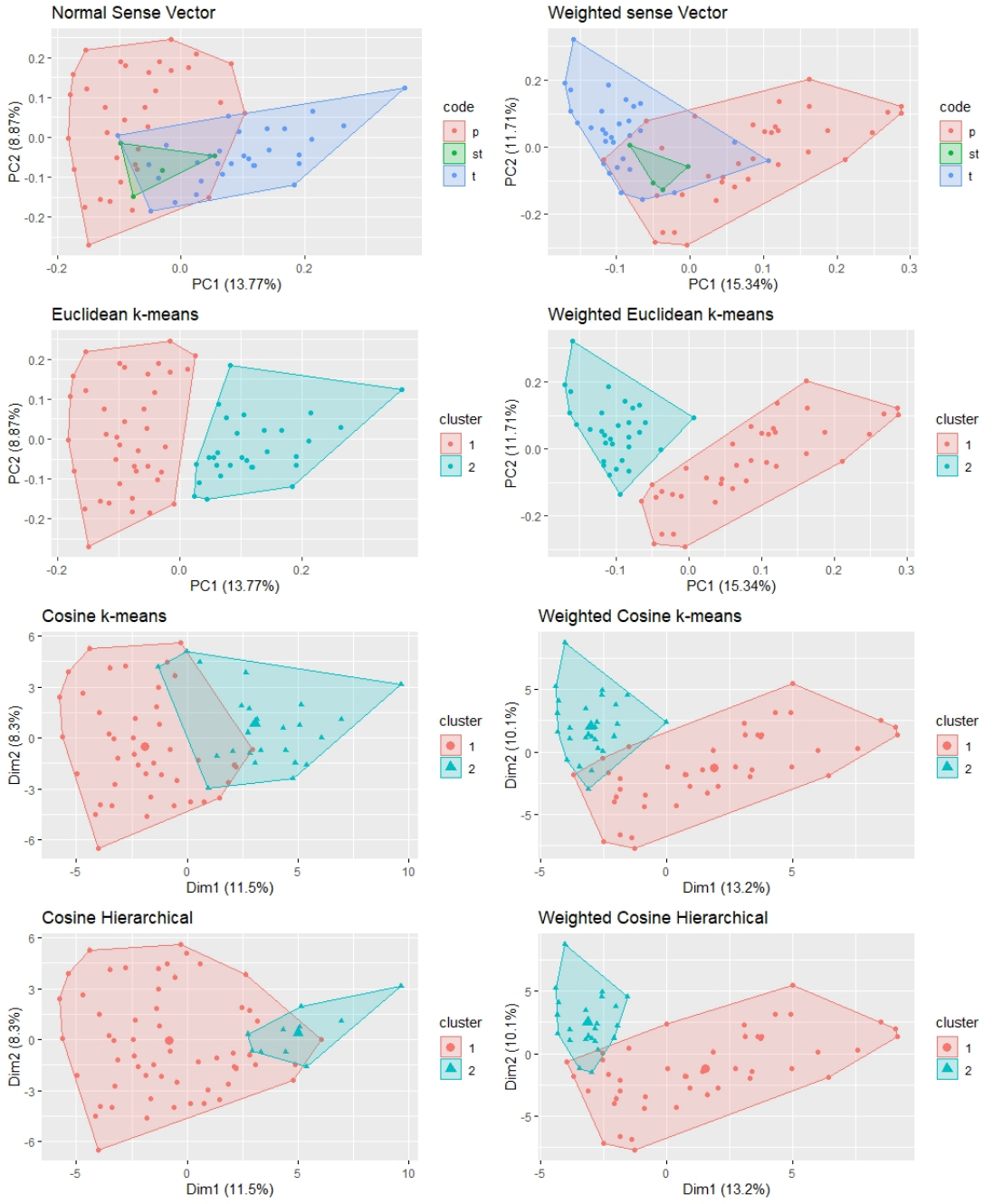

Figure 4.5: Clustering Visualization with Minword 2, Dim 100

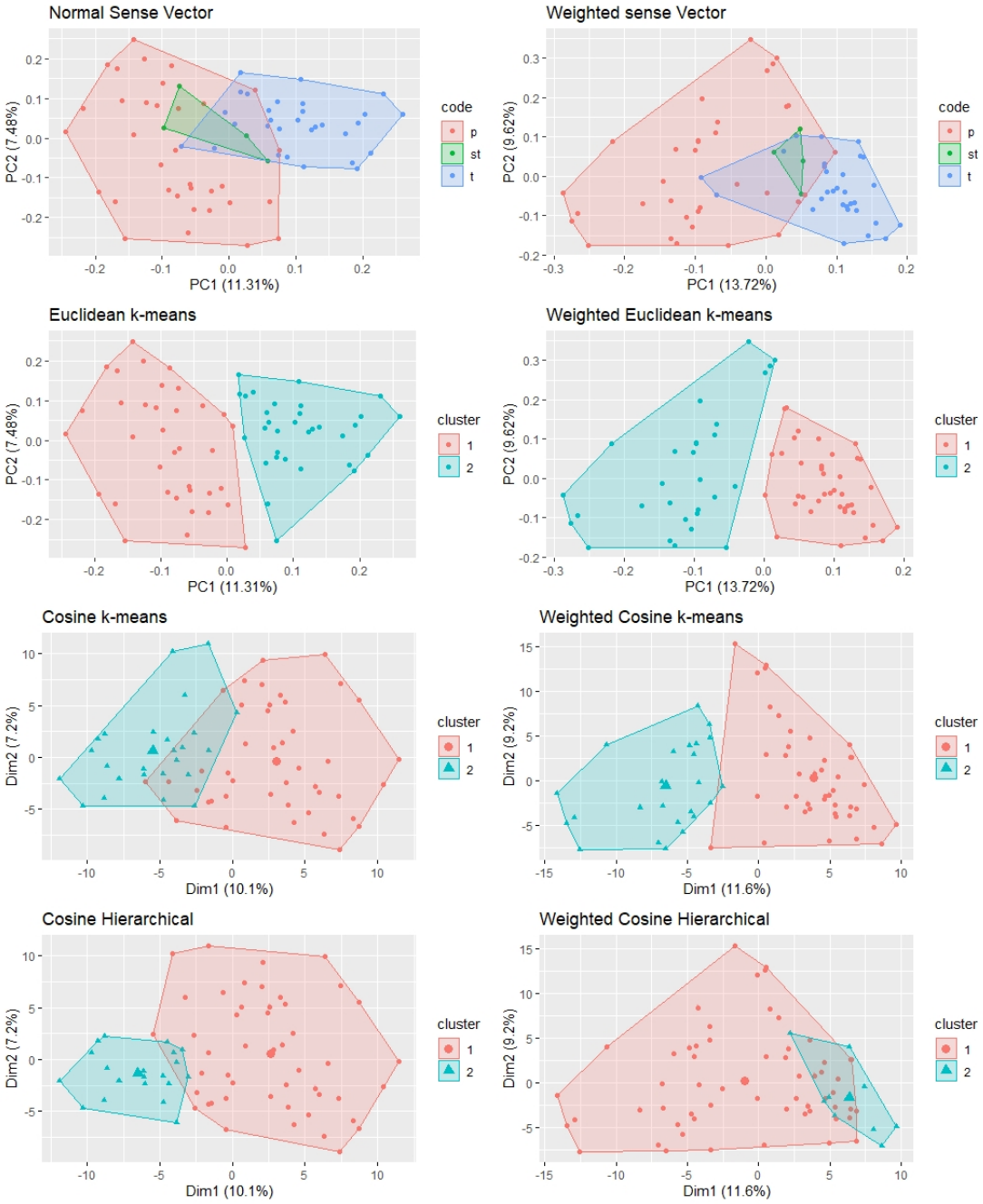

Figure 4.6: Clustering Visualization with Minword 2, Dim 300

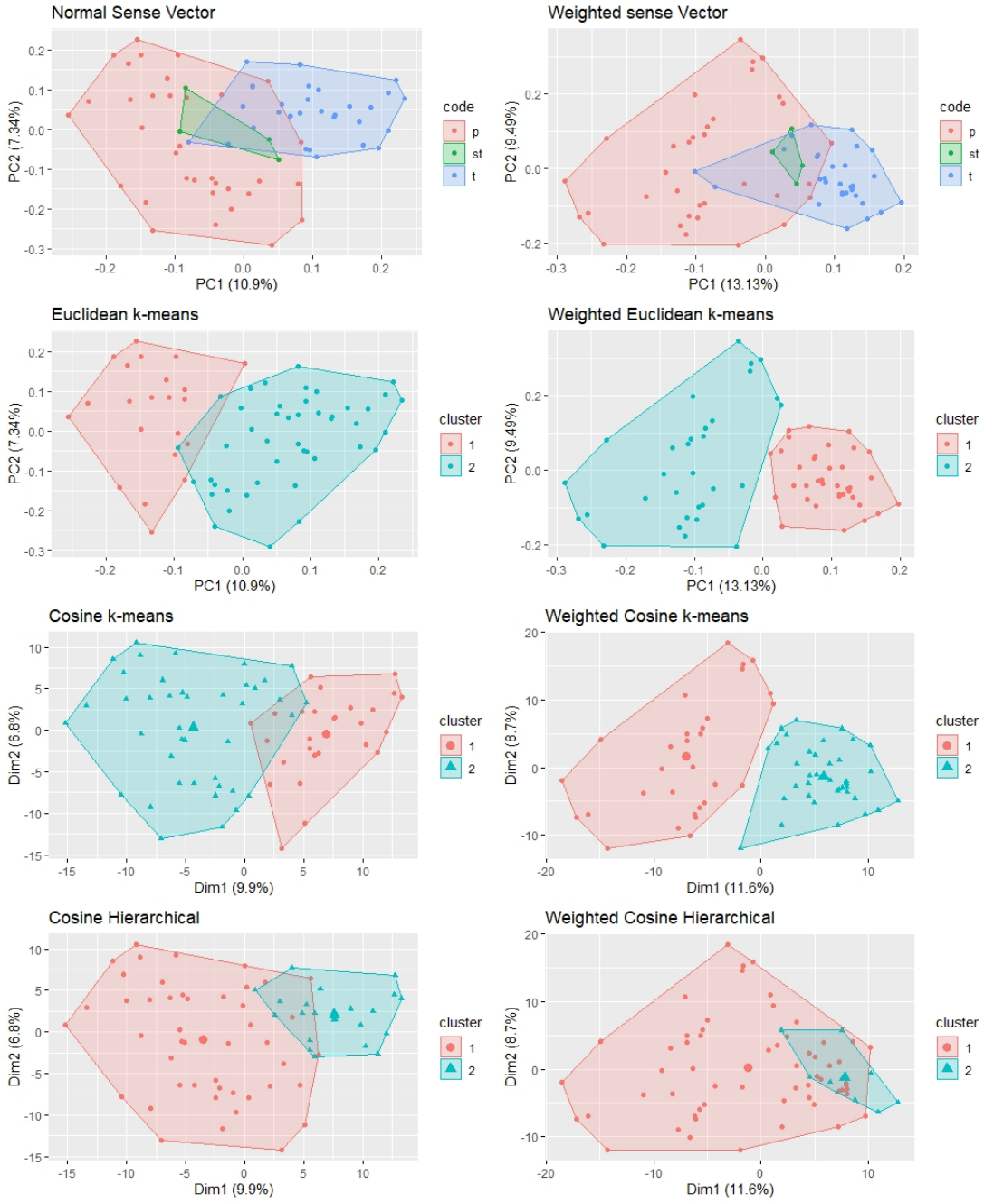

Figure 4.7: Clustering Visualization with Minword 2, Dim 500

4.4.3 Minword 3

Minword 3	k-means Clustering			
Distance	Euclidean		Cosine	
Dim	Ordinary	Proposal	Ordinary	Proposal
100	77.14%	72.86%	52.86%	68.57%
150	68.57%	84.29%	57.14%	71.43%
200	67.14%	75.71%	57.14%	72.86%
250	68.57%	82.86%	61.43%	80.00%
300	71.43%	81.43%	64.29%	81.43%
350	77.14%	75.71%	71.43%	74.29%
400	67.14%	75.71%	64.29%	82.86%
450	72.86%	81.43%	70.00%	82.86%
500	51.43%	75.71%	67.14%	80.00%

Table 4.9: k-means Clustering Accuracy with Minword 3

Minword 3	Euclidean Hierarchical Clustering					
Linkage	Single		Average		Complete	
Dim	Ordinary	Proposal	Ordinary	Proposal	Ordinary	Proposal
100	52.86%	52.86%	54.29%	52.86%	54.29%	54.29%
150	54.29%	52.86%	54.29%	54.29%	58.57%	62.86%
200	52.86%	52.86%	54.29%	54.29%	54.29%	61.43%
250	52.86%	52.86%	54.29%	54.29%	62.86%	60.00%
300	52.86%	52.86%	54.29%	54.29%	60.00%	61.43%
350	52.86%	52.86%	54.29%	54.29%	55.71%	57.14%
400	52.86%	52.86%	54.29%	54.29%	78.57%	51.43%
450	52.86%	52.86%	54.29%	54.29%	60.00%	55.71%
500	52.86%	52.86%	54.29%	54.29%	62.86%	54.29%

Table 4.10: Euclidean Hierarchical Clustering Accuracy with Minword 3

Minword 3	Cosine Hierarchical Clustering					
Linkage	Single		Average		Complete	
Dim	Ordinary	Proposal	Ordinary	Proposal	Ordinary	Proposal
100	54.29%	52.86%	54.29%	52.86%	58.57%	74.29%
150	52.86%	52.86%	55.71%	55.71%	68.57%	55.71%
200	54.29%	52.86%	55.71%	55.71%	64.29%	74.29%
250	51.43%	52.86%	57.14%	60.00%	64.29%	78.57%
300	52.86%	52.86%	57.14%	51.43%	64.29%	78.57%
350	52.86%	52.86%	57.14%	52.86%	67.14%	62.86%
400	52.86%	52.86%	57.14%	54.29%	68.57%	62.86%
450	52.86%	52.86%	57.14%	52.86%	64.29%	62.86%
500	52.86%	52.86%	54.29%	55.71%	67.14%	80.00%

Table 4.11: Cosine Hierarchical Clustering Accuracy with Minword 3

For minword 3, k-means clustering shows better performance than the hierarchical clustering method. Also, the weighting method generally advances the performance of the clustering method. To generalize, the hierarchical method does not give evidence to replace the k-means clustering method. In either method, using the weighting method is generally make performance better than the previous model.

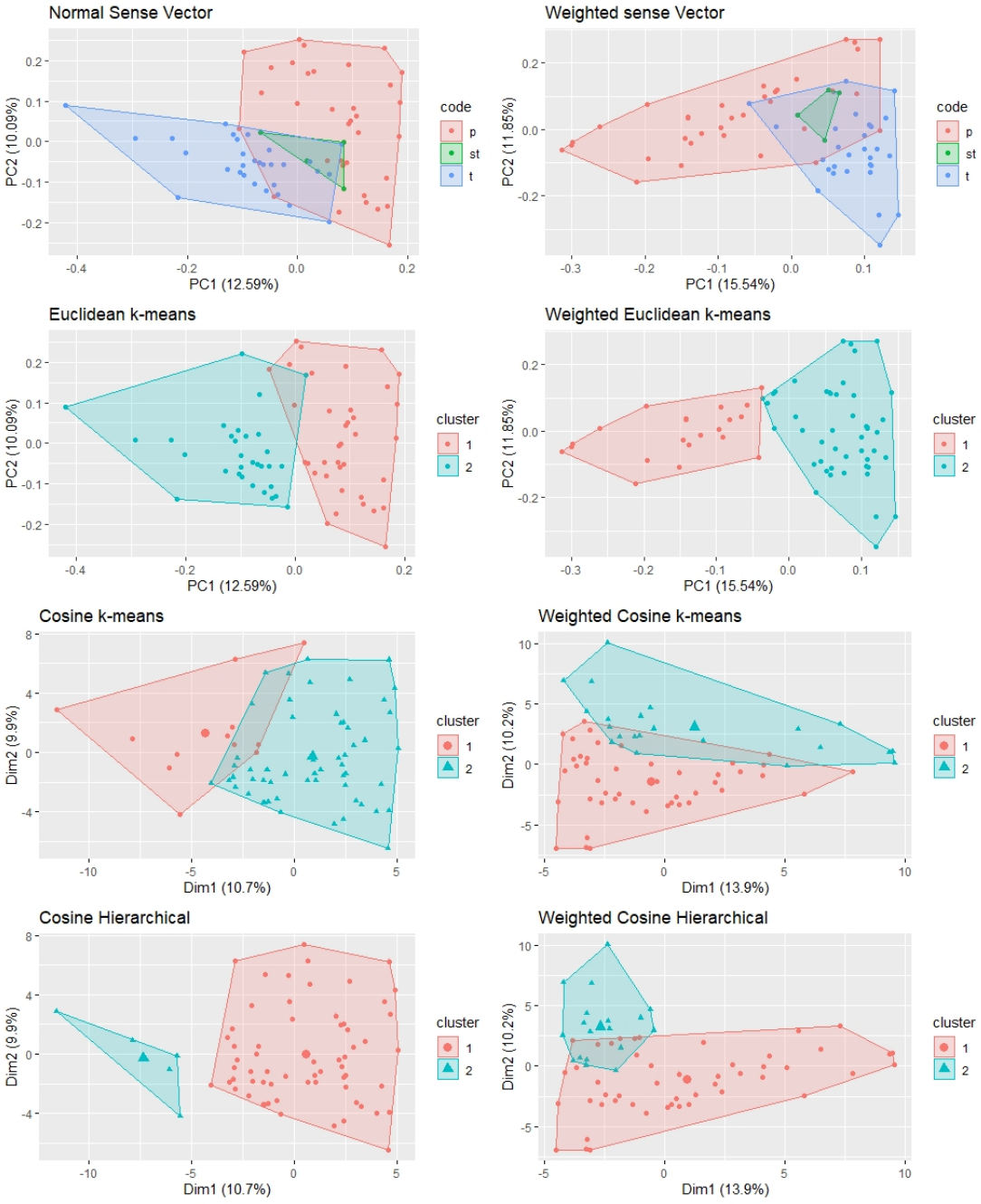

Figure 4.8: Clustering Visualization with Minword 3, Dim 100

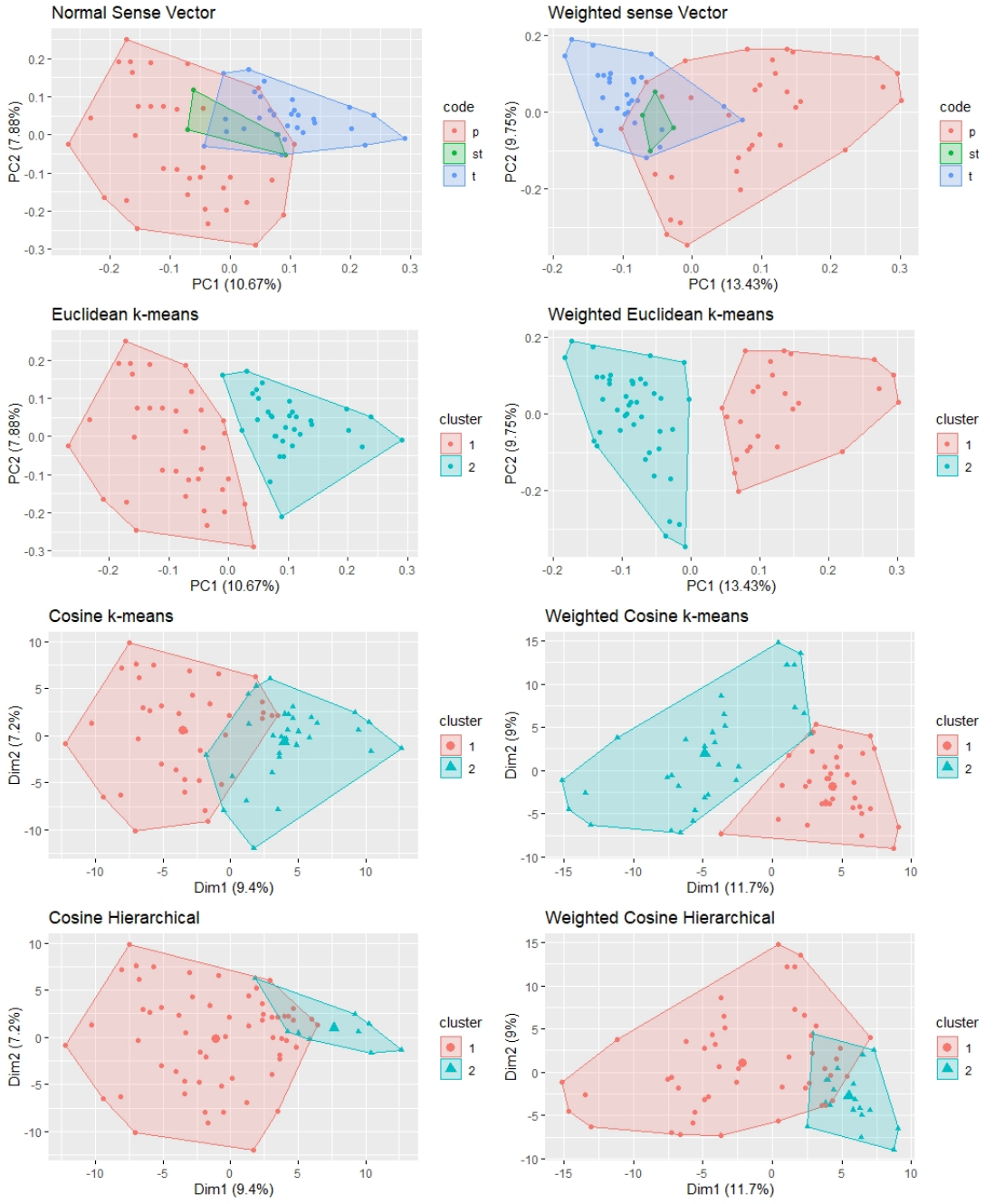

Figure 4.9: Clustering Visualization with Minword 3, Dim 300

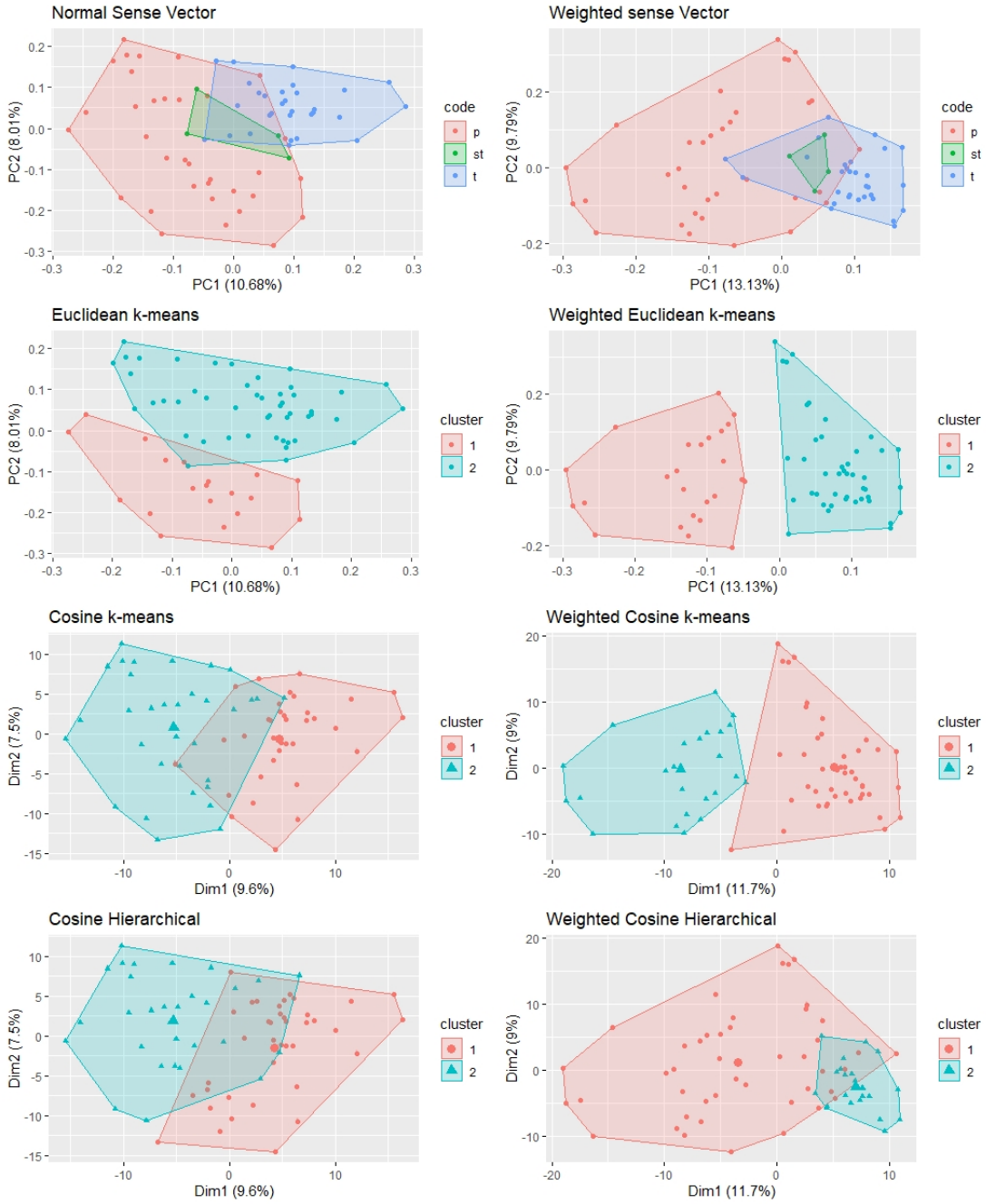

Figure 4.10: Clustering Visualization with Minword 3, Dim 500

Chapter 5

Conclusion

5.1 Conclusion

In this paper, the new weighting strategy and clustering method are proposed and compared with other previous models to solve WSD. There are many methods to bypass the problem of polysemy. But word2vec is still one of the best and easy methods for the analysis of text data rather than the other methods. Most of the recent transformation methods can have fine performances but heavy computation outweighs the benefits of them.

The hierarchical clustering method doesn't seem to work well because the structure of meaning is very different from the experimented word. Hierarchical clustering assumes continuous property that clusters are continuously contained. But this structure does not fit the senses of tree and power which do not share any meaning. The hierarchical clustering method does not provide enough performance compared to the k-means clustering method because of the structure of word sense are strictly separated.

For some special cases, there might be a way to implement a hierarchical clustering

method such as for word ‘moon’. The most known meaning would be *“astronomical body that orbits the Earth”*. On the other hand, the word ‘moon’ has more general meaning that is *“A natural satellite, or moon, is, in the most common usage, an astronomical body that orbits a planet or minor planet”*. The latter completely contains the meaning of the former meaning but used in a different sense. In this kind of situation, the structure of the hierarchical clustering might fit better than the k-means clustering.

Although the hierarchical clustering method could not provide enough performances for practical use, the weighting strategy showed that it can be adaptable. The performance of the weighted model is usually better than the previous model. Applying the weight method to other corpora that are bigger and general needs to be done for further investigation.

References

Chen, L.-P. (2019). Text mining in practice with r. *Journal of Statistical Computation and Simulation*, 0(0):1–2.

Goldberg, Y. and Levy, O. (2014). word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *ArXiv*, abs/1402.3722.

Guo, J., Che, W., Wang, H., and Liu, T. (2014). Revisiting embedding features for simple semi-supervised learning. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 110–120, Doha, Qatar. Association for Computational Linguistics.

Hornik, K. and Grün, B. (2011). topicmodels: An r package for fitting topic models. *Journal of Statistical Software*, 40(13):1–30.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Mnih, A. and Hinton, G. E. (2008). A scalable hierarchical distributed language model. In *NIPS*.

Morin, F. and Bengio, Y. (2005). Hierarchical probabilistic neural network language model. In *AISTATS*.

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

Reisinger, J. and Mooney, R. J. (2010). Multi-prototype vector-space models of word meaning. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 109–117, Los Angeles, California. Association for Computational Linguistics.

Rong, X. (2014). word2vec parameter learning explained. *CoRR*, abs/1411.2738.

Thanaki, J. (2017). *Python Natural Language Processing*. Packt Publishing.

Tomas Mikolov, Kai Chen, G. S. C. J. D. (2013). Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

초 록

Word2vec 다의어 포착을 위한 클러스터링

유 제 진
통계학과
성균관대학교

워드 임베딩(Word Embedding)은 기계가 단어를 이해할 수 있게 단어를 벡터에 매핑하는 기법입니다. 벡터에 단어를 매핑하는 방법은 여러 가지 있으나 최근에 구글에서 내놓은 워드2벡(word2vec)은 다른 방법보다 매우 강력하고 가벼운 방법으로 떠올랐습니다.

단어 중의성 해소(Word Sense Disambiguation;WSD)문제는 워드 임베딩에서 발생하는 문제로 기계의 단어 이해를 방해하는 문제를 지칭합니다. 단어 중의성 문제는 워드 임베딩의 기본적인 전제인 단어와 벡터의 일대일 매칭의 컨셉으로부터 발생합니다.

이를 해결하기 위해서 많은 다른 방법들이 제시되었으나 많은 경우 무거운 컴퓨팅 자원을 요구하거나 워드 임베딩 자체보다 다른 많은 정보를 요구하는 경우가 많습니다. 본 논문에서는 이를 해결하기 위해서 기본적으로 워드 임베딩만을 사용해서 중의성 해소를 하려고 합니다. 새로운 가중치 부여방법과 다른 군집화 방법 등을 통해 문제를 해결하여 더 간단하고 가벼운 단어 중의성 해소 방안을 제시하고자 합니다.

워드 임베딩의 이웃 단어들을 통해 얻은 의미 벡터들을 군집화하여 해당하는 단어의 의

미를 분류합니다. 의미벡터를 구성할 때 다른 가중치를 적용하여 결과를 비교합니다. 또한 군집화에서는 구형성 k 평균 군집화 방법과 위계적 군집화 방법의 효율을 비교합니다.

주제어 : 단어 중의성 해소(WSD), 워드2벡, 스킵그램(SG), 멀티센스스킵그램, 클러스터링