12-14-2017

Machine Learning Methods for Finding Textual Features of Depression from Publications

Zhibo Wang
Georgia State University

METHODS FOR FINDING TEXTUAL FEATURES OF MACHINE LEARNING DEPRESSION FROM PUBLICATIONS

by

ZHIBO WANG

Under the Direction of Dr. Yanqing Zhang

ABSTRACT

Depression is a common but serious mood disorder. In 2015, WHO reports about 322 million people were living with some form of depression, which is the leading cause of ill health and disability worldwide. In USA, there are approximately 14.8 million American adults (about 6.7% percent of the US population) affected by major depressive disorder. Most individuals with depression are not receiving adequate care because the symptoms are easily neglected and most people are not even aware of their mental health problems. Therefore, a depression prescreen system is greatly beneficial for people to understand their current mental health status at an early stage. Diagnosis of depressions, however, is always extremely challenging due to its complicated, many and various symptoms. Fortunately, publications have rich information about various

depression symptoms. Text mining methods can discover the different depression symptoms from literature. In order to extract these depression symptoms from publications, machine learning approaches are proposed to overcome four main obstacles: (1) represent publications in a mathematical form; (2) get abstracts from publications; (3) remove the noisy publications to improve the data quality; (4) extract the textual symptoms from publications. For the first obstacle, we integrate Word2Vec with LDA by either representing publications with document-topic distance distributions or augmenting the word-to-topic and word-to-word vectors. For the second obstacle, we calculate a document vector and its paragraph vectors by aggregating word vectors from Word2Vec. Feature vectors are calculated by clustering word vectors. Selected paragraphs are decided by the similarity of their distances to feature vectors and the document vector to feature vectors. For the third obstacle, one class SVM model is trained by vectored publications, and outlier publications are excluded by distance measurements. For the fourth obstacle, we fully evaluate the possibility of a word as a symptom according to its frequency in entire publications, and local relationship with its surrounding words in a publication.

INDEX WORDS: Information retrieval, Text representation, Text mining, Document summarization, Outlier document detection, Textual feature extraction, Social network, TextRank, Word2Vec, Latent Dirichlet allocation, One-class SVM, Keyword Extraction, Depression, Textual symptoms

MACHINE LEARNING METHODS FOR FINDING TEXTUAL FEATURES OF DEPRESSION FROM PUBLICATIONS

by

ZHIBO WANG

A Dissertation Defense Submitted in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

in the College of Arts and Sciences

Georgia State University

2017

MACHINE LEARNING METHODS FOR FINDING TEXTUAL FEATURES OF DEPRESSION FROM PUBLICATIONS

by

ZHIBO WANG

Committee Chair: Yanqing Zhang

Committee: Rajshekhar Sunderraman

Saied Belkasim

Ruiyan Luo

Electronic Version Approved:

Office of Graduate Studies

College of Arts and Sciences

Georgia State University

November 2017

DEDICATION

I dedicate this dissertation to my wife, who supported me each step of the way. Also, my unreserved gratitude goes to my parents who always encouraged and support all my scholarly endeavors.

ACKNOWLEDGEMENTS

The writing of this dissertation has been one of the most significant academic challenges I have ever had to face. I would not be able to finish this dissertation without the help of so many people in so many ways.

I would like to express my deepest appreciation to my committee chair, Dr. Yanqing Zhang for his guidance, understanding, patience, and most importantly, his friendship during my graduate studies at Georgia State. He was always there, listening and encouraging me for both my study and life.

I would like to gratefully and sincerely thank Dr. Raj Sunderraman, who continually and convincingly conveyed a spirit of adventure and an excitement in regard to research.

I would like to thank Dr. Saied Belkasim, who patiently corrected my writing and gave me many valuable suggestions with my research and further work.

I would also like to thank Dr. Ruiyan Luo, for guiding my research for the past several years and helping me to develop my background in statistics, and public health. I would also thank Department of Computer Science at Georgia State, especially those members of my doctoral committee for their input, valuable discussions and accessibility. My research would not have been possible without their helps.

Finally, and most importantly, I would like to thank my parents, and my family. They were always supporting me and encouraging me with their best wishes. And a special thank you to my wife, Ye Cui. She was always there cheering me up and stood by me through the good times and bad. I appreciate my baby, my little girl Ava Wang for abiding my ignorance and the patience she showed during my thesis writing. Words would never say how grateful I am to all

of you. I consider myself the luckiest in the world to have such a lovely and caring family, standing beside me with their love and unconditional support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

1.1 Background

The sheer volume of new articles being published every day is growing exponentially. A surprising number is that there are about 300,000 literatures about “depression” in PubMed database. This number also exposes that many researchers are concerned about the research of depression diagnosis and treatments since depression is an extremely challenging disease for accurate diagnosis. The depression diagnosis usually takes a couple of weeks to watch patients’ signs or symptoms, and meanwhile understand patients’ potential risk factors such as personal or family history of depression, major life changes, trauma, stress and so on. Also, the treatment to depression is much more difficult because depression is different from most of other diseases such as cancer. Depression not only requires the treatment of medications or drugs, but also needs psychotherapy. Therefore, it is highly demanded that a depression specialist must be able to comprehensively diagnose patient’s conditions in a short time and propose a reasonable and feasible treatment plan since the earlier that treatment can begin, the more effective it is.

Besides learning depression related knowledge in schools or from experienced doctors, medical literatures can be used to find out peers’ recent research progress, learn the cutting-edge cure methods, and understand latest contributions to the depression diagnosis. Therefore, reading medical publications for doctors is very helpful to improve their professional skills and expand their knowledge to take better treatment plans. However, due to either their tight work schedules or the large amount of newly publications, doctors don’t have enough time to read and study all the most recent publications. Also, it is more likely that they have forgotten many publications that have been read.

Text mining is a growing area of computer science because unstructured data increases exponentially in both relevance and quantity. Unstructured data differs from the traditional structured data that does not reside in a traditional row-column database. The unstructured data accounts about 70-80 percent of the total data, and the amount of unstructured data is growing faster than that of the structured data. Unstructured data contains text and multimedia contents such as e-mails, tweets, webpages, and other text files without fixed formats. The unstructured data cannot be used by the models of structured data directly. Therefore, a preprocess step is needed to translate unstructured data to understandable data. In other words, a mathematical representation is required to describe the collection of words and symbols. Academically, text mining methods can categorize documents based on their meanings, extract important targeted entities, analyze the sentiment of documents, retrieve the hidden topics from a collection of documents, and so on. Many techniques have been applied to solve real problems in industry in terms of revealing insights, patterns and trends from unstructured data. Customer care service is a classical application to improve customer experiences using texting mining. Natural language processing methods are used to analyze different text resources such as customer surveys, trouble tickets, and phone-call conversation notes to improve the quality, effectiveness and speed in resolving problems. Many companies are employing robots as customer representatives to communicate with clients and solve their problems, which dramatically reduces the reliance on the call center. Customer behavior prediction is another successful application of text mining. With the increase of unstructured social media data, there is a big potential market to predict the opinions, emotions, and future behaviors by analyzing prolific extracted information from the social media data such as user profiles, relationship among multiple users, behaviors and interests of users, and so on. Except for the two applications mentioned above, text mining is

also applied to risk management, knowledge management, cybercrime preventions, fraud detection through claims, contextual advertising, business intelligence, content enrichment and so on.

The vast numbers of biomedical texts are the main source to study the relevant medical knowledge and the references of research. Text mining can help us to extract information and knowledge from the huge amount of texts. It has been widely applied in biomedical research. The number of publications by querying the keyword “text mining” or “literature mining” grew substantially from 13 papers in 2000 to about 256 papers in 2011 [1]. It exposes that more and more researchers are interested in text mining methods. Generally, biomedical text mining is to derive implicit knowledge hidden in unstructured text data, and present the knowledge in a quantitative form. Most researchers are focusing on four aspects: information retrieval, information extraction, knowledge discovery, and hypothesis generation. Information retrieval is to get targeted contents on a certain topic [2]; information extraction is usually to extract predefined patterns such as entities and relations contained in the texts; knowledge discovery methods can help us to extract novel knowledge hidden in the text; hypothesis generation is used to explore and discover the unknown biomedical facts from text. Thus, biomedical text mining mainly focuses on information retrieval, named entity recognition and relation extraction, knowledge discovery and hypothesis generation.

1.2 Challenges for text representation

Structured data unlike unstructured data is in forms of tables, which most of us know of through spreadsheet packages like Excel. The numbers in the table can be easily used for statistical or machine learning models, such as the text classification and outlier text detection. Generally, we only need to clean the data which includes missing value imputation, outlier

removal, and so on. However, in many ways, texts are like data, but it is important to keep in mind that texts are not data since it cannot be used easily. Even in the ways that text has some formats or laws to follow, it still needs "converting" and "structuring" to mold the texts into a shape fit for analysis.

Text data has rules of syntax, grammar and expression, resulting in the same content being able to carry different meanings. For example, "How true!" is not the same as "How is it true?" because of the domain-sensitive interpretation. So, the same text could acquire different meanings when used in media and entertainment or in say, medical research. Likewise, there are dialect-specific or culture-specific nuances, sarcasm and emotions that alter meaning that must be inferred from context than mere words. Additionally, most text mining applications are context-specific and then it needs large scale processing, therefore, a powerful algorithm should also learn texts from a global perspective over all training text data.

Thus, how to develop a method to represent texts in a comprehensive way is a big challenge for the text mining. In another word, we need to find a way to represent texts not only containing the local neighboring information of the words, but also considering their meaning by evaluating the topics over texts.

1.3 Problem statement

In the framework, we aim to extract the important textual features from given publications, which are downloaded online by the keyword search. Therefore, the first problem to be solved is text representation.

Text representation is one of the most challenging problems in text mining and Information Retrieval (IR). It mathematically describes the unstructured text data to be numerically computable. For a collection of text documents $D = \{d_i, i = 1, 2, \dots, n\}$, where d_i stands for a

document, the problem of text representation is to represent each d_i of D as a point p_i in a numerical space S, where the distance similarity between each pair of points in space S is well defined.

Nowadays, the most commonly used text representation model is called Vector Space Model [3] [4]. Bag of Words model (BOW) is one the most used VSM is the Bag of Words model (BOW). It uses all words appeared in the given document set D as the index of the document vectors. If a term appears in a document, there has a "1" in the position, which corresponds to this term, in the document vector. Otherwise, the term weight is "0".

Later, Term Frequency Inversed Document Frequency (TFIDF) model was proposed. It uses real values which capture the term distribution among documents to weight terms in each document vector. However, there are many limitations in the traditional BOW text representation model. For example, (1) BOW ignores the within document term correlation such as the order of terms in each document; (2) the polysemy and synonymy problems can greatly decrease the performance of text representation; and (3) the TFIDF model cannot capture the semantics of documents. Thus, in my proposal, we propose a text representation method which successfully integrates the local information within each document with global information among documents.

Keywords are widely used to download resources from search engine or database. The publications we collect are from National Center for Biotechnology Information (NCBI) using the keyword "depression". Extracting the contents of large entries of text into a small set of words is difficult and time consuming for human beings, it is more likely an impossible task to accomplish with limited manpower as the size of the information grows. Therefore, we want to find a way to let machines taking the place of humans to automatically discover the important textual features of given topics. Moreover, due to the complexities of natural language, whether a

word or a set of words accurately represent topics of a document and how to determine these words become more challenging.

2 RELATED WORK

2.1 Bag of Words Model

In the bag of words (BOW) model, any document is treated as a collection of unordered terms [5]. Give a document $D = \{d_i, i = 1, 2, \dots, n\}$, suppose there are m unique terms existed in the document. Mathematically, the corpus of documents can be represented by a $m \times n$ matrix $S \in R^{m \times n}$. Each document is denoted by a column vector $s_i, i = 1,2, \dots, n$ and each term is denoted by a row vector. The j^{th} entry of s_i is denoted by $s_{ji}, j = 1,2, \dots, m$.

For example, there are two documents. S_1 = "He investigates the text representation approaches" and S_2 = "What is the meaning of text representation approach for text documents?". From S_1 and S_2, there are 13 unique terms, which are "He, investigates, the, text, representation, approaches, What, is meaning, of, approach, for, documents". There, the list of terms roughly represents the two document by a 13×12 matrix. The problem is how to weight each entry of this matrix. Considering the one-hot encoding model first. If a term exists in a document, we set the weight with 1 at its corresponding place. Otherwise, we give a weight with 0. So we transform the two documents to a matrix

$$S^T = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

where the order of term index is the same as the term list given above. There are three obvious problems in the text representation strategy: (1) not all terms have the physical meaning in representing documents such as "the", "is" etc.; (2) some terms such as "approaches" and "approach" have the same meanings; (3) the importance of all terms is equal in a document with the strategy we used. But, in fact, in a document, some of the terms are used to emphasize the topics, and others are used to construct the sentences. To improve the quality of the text representations, there are three main steps.

2.1.1 Stop words removal:

The stop words are the name of the terms that should be filtered out before text documents indexing or natural language processing. There has no fixed stop words list for all text processing applications. Generally, the stop words list will include the terms like “a”, “the”, “an”, etc.

2.1.2 Stemming

The stemming aims at reducing inflected words to their stem. For example, “approaches” is stemmed to “approach,” “investigates” is stemmed to “investigate” and “representation” is stemmed to “represent.”

After 2.1.1 and 2.1.2, the list of terms has been reduced to “He, investigate, text, represent, approach, what, mean, document”. Thus, the two documents d_1 and d_2 can represented by an 8×2 matrix, which is as below. So far, the problem (1) and (2) have been solved by 2.1.1 and 2.1.2.

$$S^T = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

2.1.3 Term Frequency – Inverse Document Frequency (IF-IDF)

The TFIDF aims to assign a weight to each term according to a document in a collection or corpus [6]. In other words, TFIDF aims to assign different weights for all entries s_{ji} in matrix S. An intuition is that the more times a term appears in a document, the more important is this term to this document. Thus, weight should increase proportionally to the number of times a term appears in the document. On the other hand, if a word appears in many documents in the corpus, the discriminative power of the term will be weak. Thus, the weight is offset by the frequency of the word in the corpus. The former step is called the Term Frequency (IF). It can be counted directly from the documents. For example, the TF of term “text” in d_2 is 2. A normalizing factor

is always used for calculating the TF of a term in a document. Since d_2 there are 7 terms after stop words removal and stemming. Among them, 2 of them are "text". Thus, the TF for term "text" in d_2 is 2/7. Next, we are talking about the Inverse Document Frequency (IDF). It is a log function. The IDF for term t is:

$$\log \frac{n}{\#\ document\ involve\ term\ t}$$

Thus, the TFIDF weighting schema can be as simple as $TF \times IDF$. There are various variations of TFIDF text indexing. The major differences are how to normalize and smooth the weight equation.

2.2 Word2Vec Model

Word2Vec model is a particularly computationally-efficient predictive model to learn word embeddings from a collect of texts [7]. It includes two alternative models to update parameters. 1) Continuous Bag of Words (CBOW) is a way to predict words by using contexts of its surroundings; 2) in contrast, Skip-gram uses a word's information to predict its neighboring words. As shown in Fig. 2, both models contain three layers: an input layer, a projection layer and an output layer. We take CBOW as example to briefly explain how Word2Vec works.

2.2.1 Introduction of Word2Vec CBOW Algorithm

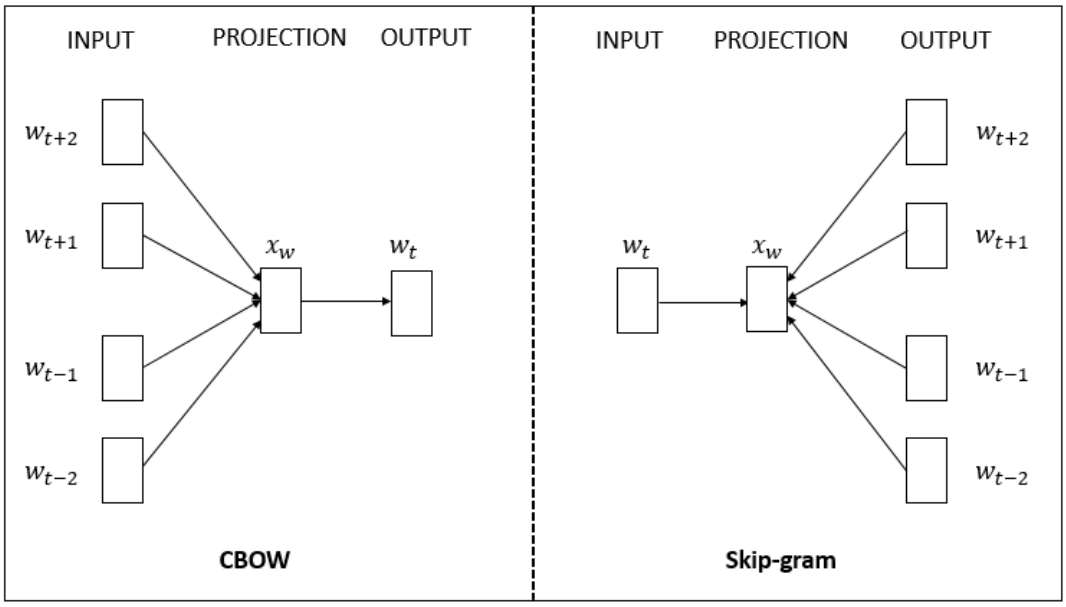


Figure 2-1 Two structure of Word2Vec model

Given a sentence $W = \{w_{t-2}, w_{t-1}, w_t, w_{t+1}, w_{t+2}\} \in \mathbb{R}^m$, where w_t is the target word.

Input layer: $\text{Context}(v(w_t)) = \{v(w_{t-2}), v(w_{t-1}), v(w_{t+1}), v(w_{t+2})\} \in \mathbb{R}^m$.

Projecting layer: a contextual vector $v(x_w)$ is calculated by

$$v(x_w) = \sum_{i=t-2}^{t+2} context(v(w_i))$$

where $i \neq t$.

Output layer: A word in vocabulary is treated as a leaf node in a Huffman tree according to its occurrence in the corpus. Therefore, each word has a unique path from root node to leaf node. At each node except for the leaf node, the probability of selecting left child or right child can be estimated by the logistic model by

$$\text{left child}: \sigma(v(x_w)^T\theta) = \frac{1}{1 + e^{-v(x_w)^T\theta}}$$

$$\text{right child}: 1 - \sigma(v(x_w)^T\theta)$$

$p\left(v(x_w)\middle|\text{Context}\left(v(x_W)\right)\right)$ can be learned in the tree by a production of probabilities at each node, where $d_j^w \in \{0,1\}$ is the j^{th} digit in word w's Huffman code and j is any node on the path except as the leaf node.

The objective function below can be learned by maximizing the log-likelihood, and then use gradient descent method to update θ, $v(x_w)$ and its contextual words.

$$\mathcal{L} = \sum_{w \in C} \log \prod_{j=2}^{n} \left\{ \left[\sigma\left(v(x_w)^T \theta_{j-1}^w\right)\right]^{1-d_j^w} \left[1 - \sigma\left(v(x_w)^T \theta_{j-1}^w\right)\right]^{d_j^w} \right\}$$

2.2.2 Sample Outputs of Word2Vec

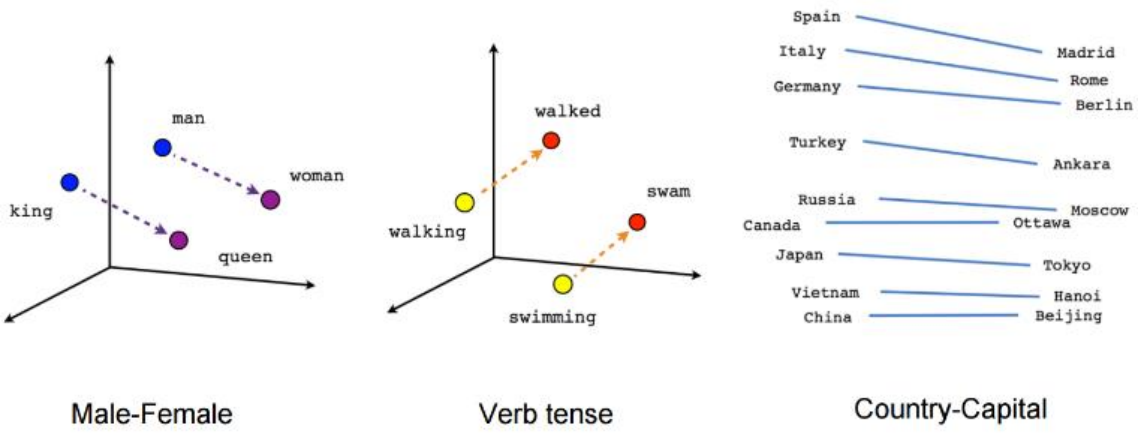


Figure 2-2 Sample outputs of Word2Vec model

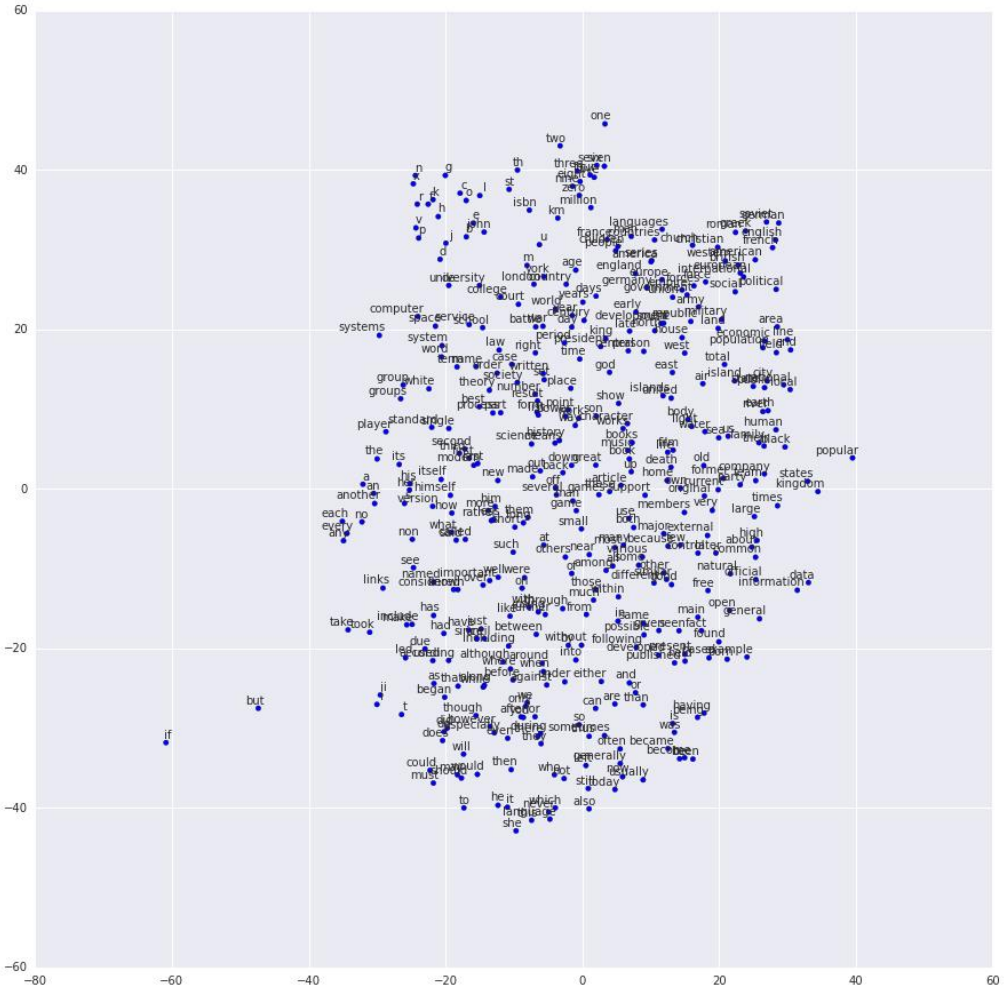


Figure 2-3 Visualized word representations use t-SNE

As shown in figure 2.2, the outputs show that the word representation can well maintain the relationships before vectorization [8]. The left figure tells Word2Vec can learn associations behind the words. The figure in the middle shows even without stemming processing Word2Vec can well study different tenses of verbs. The right figure perfectly describes the relationship between the capitals and countries. t-Distributed Stochastic Neighbor Embedding (t-SNE) is a dimensionality reduction technique which can well suited for the visualization of high-dimensional data [9]. In figure 2.3, it shows the 2-D results of word representations whose

dimensions are reduced by t-SNE. It is as expected that the similar words are closed to each other, which can be easily applied to word clustering tasks.

2.3 Latent Dirichlet Allocation

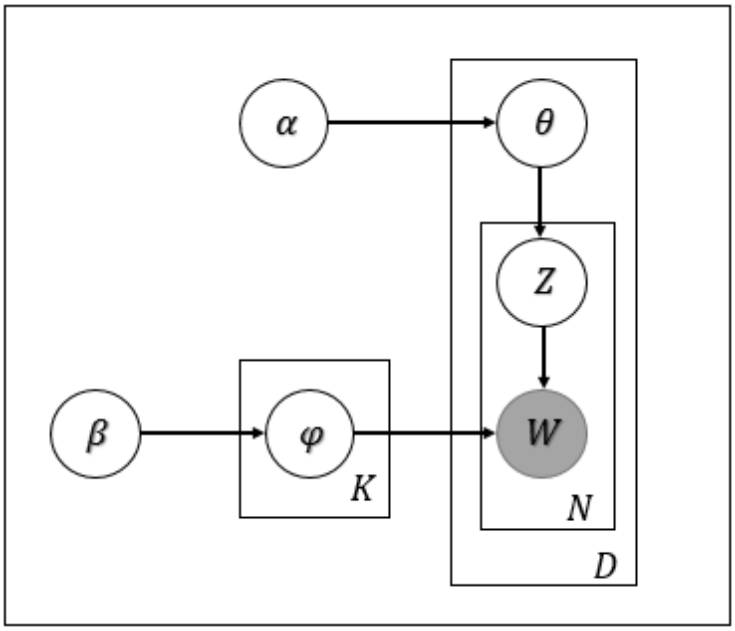


Figure 2-4 A graph of Latent Dirichlet allocation

2.3.1 Introduction to the Latent Dirichlet Allocation Algorithm

LDA is an unsupervised method to discover the latent topics Z from a collection of documents D [10]. In LDA, each document d is represented as a probability distribution θ_d over topics, where each topic z is a probability distribution φ_z over all words in vocabulary. Figure 2.4 shows the generative process. Both θ and φ have prior distributions with hyperparameters α and β. For every word w_{d_i} in document d, a topic z_{d_i}can be extracted by two equations below, a word w_{d_i}can be returned. Repeat equation (1) and (2) N times, a document d is generated, where N is the size of document d.

$$\theta_d \sim Dirichlet(\alpha) \quad z_{d_i} \sim Multinomial(\theta_d) \square \tag{1}$$

$$\varphi_z \sim Dirichlet(\beta) \quad w_{d_i} \sim Multinonial\left(\varphi_{z_{d_i}}\right) \tag{2}$$

By using Gibbs Sampling, θ and φ can be inferred to discover the latent topics in documents, and predict any new document with a topic proportion distribution.

2.3.2 Real Examples

An example from Edwin Chen's blog which is about the analysis of several thousand emails from Sarah Palin's time as governor of Alaska [11]. By learning the emails, we can easily see the main topics contained in the emails, and the associated words are very reasonable to describe their corresponding ponding topics.

Table 2-1The topics and their keywords discovered from Sarah Palin's emails by LDA

Topics	Keywords
Wildlife/BP Corrosion	game, fish, moose, wildlife, hunting, bears, polar, bear, subsistence, management, area, board, hunt, wolves, control, department, year, use, wolf, habitat, hunters, caribou, program, denby, fishing, …
Energy/Fuel/Oil/Mining	energy, fuel, costs, oil, alaskans, prices, cost, nome, now, high, being, home, public, power, mine, crisis, price, resource, need, community, fairbanks, rebate, use, mining, villages, …
Trig/Family/Inspiration	family, web, mail, god, son, from, congratulations, children, life, child, down, trig, baby, birth, love, you, syndrome, very, special, bless, old, husband, years, thank, best, …
Gas	gas, oil, pipeline, agia, project, natural, north, producers, companies, tax, company, energy, development, slope, production, resources, line, gasoline, transCanada, said, billion, plan, administration, million, industry, …
Education/Waste	school, waste, education, students, schools, million, read, email, market, policy, student, year, high, news, states, program, first, report, business, management, bulletin, information, reports, 2008, quarter, …
Presidential Campaign/Elections	mail, web, from, thank, you, box, mccain, sarah, very, good, great, john, hope, president, sincerely, wasilla, work, keep, make, add, family, republican, support, doing, p.o, …

2.4 Support Vector Machine (SVM)

2.4.1 Introduction of SVM

Support vector machines represent a set of supervised learning techniques that create a function from training data [12] [13]. The training data usually consist of pairs of input vectors

and desired output. The learned function can be used for predicting the new input. SVM are typically used for classification where the function outputs one of finite classes. Two special properties of SVMs (1) high generalization by maximizing the margin, where margin is the distance between the hyperplane and the closest data vectors in the feature space. (2) Support efficient nonlinear classification by kernel trick. The kernel trick is used to convert a linear classifier into a non-linear one by using a non-linear function to map the original observations into a high-dimensional space.

2.4.2 Binary SVM

A binary classification is to classify data objects into either positive or negative class. Each data point is represented by a n-dimensional vector. Each of these data points belongs to only one of two classes. A linear classifier separates them with an $n-1$ dimensional hyperplane.

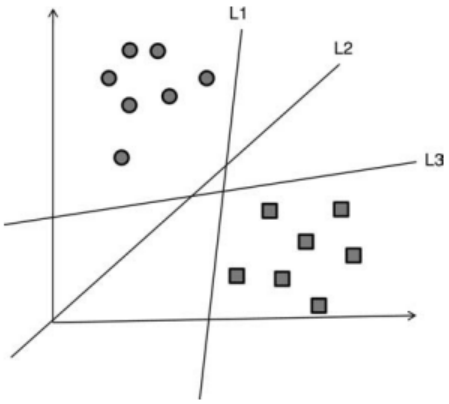


Figure 2-5 Linear classifiers in two-dimensional spaces

For example, in figure 2.5, two groups of data points are separated by three hyperplanes L_1, L_2, and L_3 correctly. In order to achieve maximum separation between the two classes, SVM picks the hyperplane with the maximum margin is maximized. Such a plane is likely to

generalize better, meaning that the hyperplane not only correctly classify the given data points, but also is likely to correctly classify the unknown data points.

2.4.3 One-Class SVM

One-Class SVM is a special form of support vector machine [14], which has been applied to various applications [15] [16] [17] [18]. It learns a minimum volume hypersphere that enclosed most of the data. In another word, One-Class SVM only recognizes one class from the normal training data. If a newly data is too different to this class, model labels this data as out-of-class. One-Class SVM is an optimization problem because we expect the ball as small as possible but the ball contains most of the data.

Most time we use One-Class SVM when we only have the data of one class. The objective function can be learned by (3) and (4):

$$F(R, a, \xi_i) = R^2 + C\sum_i \xi_i \tag{3}$$

$$s.t.\,(x_i - a)^T(x_i - a) \leq R^2 + \xi_i, \forall_i \geq 0 \tag{4}$$

where a is the center, R is the radius, and x_i is the training data.

We can solve the optimization with Lagrangian multipliers in (11):

$$L(R, a, \alpha_i, \xi_i) = R^2 + C\sum_i \xi_i - \sum_i \alpha_i\{R^2 + \xi_i - (x_i^2 - 2ax_i + a^2)\} - \sum_i \gamma_i \xi_i \tag{5}$$

where $\alpha_i \geq 0$ and $\gamma_i \geq 0$.

After deriving the parameters and let derivatives set to zero, we can get (12), (13) and (14):

$$\sum_i \alpha_i = 1 \tag{6}$$

$$a = \frac{\sum_i \alpha_i x_i}{\sum_i \alpha_i} = \sum_i \alpha_i x_i \tag{7}$$

$$C - \alpha_i - \gamma_i = 0 \tag{8}$$

By substituting (6), (7), and (8) to the Lagrangian multiplier, we can get:

$$L = \sum_i \alpha_i K(x_i, x_i) - \sum_{i,j} \alpha_i \alpha_j K(x_i, x_j) \tag{9}$$

To distinguish whether a new data is normal or abnormal, the Kernel equation is applied.

$$K(z, z) - 2\sum_i \alpha_i K(z, x_j) + \sum_{i,j} \alpha_i \alpha_j K(x_i, x_j) \leq R^2 \tag{10}$$

3 A NOVEL METHOD FOR DOCUMENT SUMMARIZATION USING WORD2VEC

3.1 Introduction

Tons of textual resources are created and uploaded to Internet every day, such as Tweets, Blogs, webpages and so on. These text data not only require lots of storage space to save and backup, but also bring in a lot of data organizing, processing and analyzing tasks. Hence, to extract meaningful and non-trivial knowledge from the humongous textual database becomes the overarching goal of text mining. However, unstructured text data, unlike those quantitative data, cannot be simply analyzed at the word level. A data pre-processing step has to be implemented to clean the texts, and convert them to a more understandable format by computers.

Generally, a document always contains some words which are not quite related to the documents' topics. Removing these words reduces the corpus size such that decreases the dimension of features at the same time; additionally, it reduces noises of the documents in order to improve the usability of documents. Stop-words like prepositions, articles, and pro-nouns are high-frequency words in documents. These words do not carry any information regarding contents, but used only for grammar purpose. Therefore, removing these words do not have any effect on the document itself. Moreover, documents always contain various data formats and non-informative features such as date formats, number formats or currency format. There are alternative ways to address these formats. Besides elimination, other ways to handle them include putting all the dates in a container named DATE, and converting numbers into letter-format such as 5 is May or 10 is ten, etc. But there is not a standard way to pre-process the punctuations according to the purpose of analysis. For example, punctuations are much useful for the sentimental analysis that ":)" is positive and "T_T" is negative. For other purpose, however,

punctuations should be removed at the very beginning. Therefore, the way to pre-process the textual databases is highly dependent on its applications.

Normally, there are two ways to convert the textual data into quantitative data used for advanced analysis: one is based on one-hot word representation and the other one is based on the word embedding.

One-hot word representation is a vastly used word representation method, which builds a vocabulary according to the word occurrences in all given documents. It simply represents a word with a vocabulary size vector with 1 at its corresponding positon and 0 at other positions. Bag-of-words model extends one-hot presentation to represent a document by aggregating all one-hot word vectors in the document. It is obvious that the resulting document vector length is the same as the vocabulary size and each element in the document vector is the frequency of corresponding word [5]. Bag-of-words model has two drawbacks, on one hand, as the number of dimensions of document vector replies on the occurrence of words all documents, the one-hot vectors usually have very high dimensions. On the other hand, bag-of-words model ignores spatial relationship among words since element in vector only represent the counts of the word.

Distributed representation predicts each word with a very low-dimension vector ranging from 50 to 300 [19]. Generally, a word vector is randomly initialized, and is trained and updated according to each word's contexts. Since contextual information is taken into account, models like Neural Network are able to update each word vector with semantic meanings. Global Vectors for Word Representation (Glove) and Word2Vec are the most popularly used among the distributed representation methods. Glove predicts words and contexts by constructing co-occurrence matrix according to the given documents, and updates the vectors by reducing the dimensionality of the matrix. Word2Vec uses a three-layer neural network to train each word

vector with respect to its surroundings, and uses Gradient Descent algorithm to update the weights of the network [20] [7]. The resulting word vectors by Word2Vec are highly correlated with the practical semantics, for example “king – man + woman = queen”, and “Paris – France + Italy = Roma” [21]. Comparing with one-hot representation methods, distributed representation models are able to identify the relationship between two words even that one occurs much more frequently than the other. For example, a sentence “There is a dog running in the park.” occurs many times, while the other sentence “It is a cat sitting in the park.” appears only once. Word2Vec analyzes the contexts of the word ‘cat’ which is very similar to the contexts’ of “dog”, and then predicts “cat” at a position closer to “dog” in the semantic space. Recently, researchers have proposed many variations of Word2Vec to determine relationships between labels and words. Doc2Vec is one of these algorithms. Additionally, it integrates paragraph vectors at the input layer that cooperates with the word vectors to investigate correlation between labels and words [22]. Another hybrid method clusters the word vectors and considers each cluster as a bag-of-words. It thus uses frequency of words in each bag as features of the document [23].

However, these methods also analyze many vague and trivial words that are not highly correlated with document topics. In this section, we propose a method to further optimize the document size by eliminating the non-topic related words. In a document, some of the words are only used for connecting purpose or descriptions that are not related with the document topics. Although these words are necessary for communications, they are not meaningful for text data analysis. Our proposed method which targets on the issue of these irrelevant words is able to identify and remove them to optimize the process by extending the idea of Word2Vec. Word2Vec is a widely-used algorithm to transform each word in the corpus into a word vector with semantic meanings. We consider each cluster formed by word vectors using K-means as a

feature that supervises the subsequent optimization by measuring its distance from other target semantic groups. Generally, a document can be divided into multiple partitions such as paragraphs or part of paragraphs. In addition to word vectors, we also transform the document and its partitions into vectors and process them as continuous bags-of-words. Specifically, document partitions, document, and features are represented by single vectors, which dimensions are the same as any word vector in the document. Afterwards, we measure the Cosine distances between them that shorter distance indicates higher correlation, and vice versa. Hence, only partitions meet certain criterion are retained. Experimental results show that using our method size of document is significantly reduced, while accuracy of classification remains at a high level.

3.2 Overall Framework

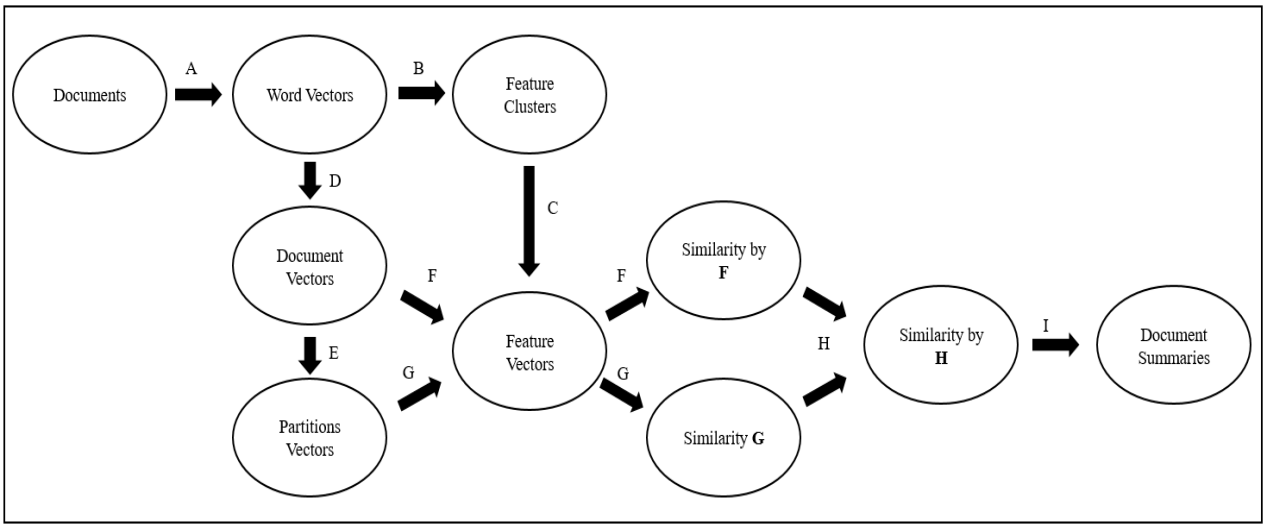


Figure 3-1 Process of the proposed framework

Generally, Word2Vec model builds a semantic space projected from the word vectors that trained and outputted by the model, where each word is considered as a point in the space. It measures and interprets similarity in grammar or semantics through distance between any two distinct points. Both Euclidean and Cosine distance are commonly adopted as ways to calculate

the distance between two points in researches. Extending the idea of Word2Vec, we generate different levels of vectors other than only word vectors to compare resemblance. The resulting output contains only "qualified partitions" which are highly correlated with the document topics. Following, we explain specifically the way to select the qualified partitions to represent the document, which is decomposed into 9 consequent steps.

3.2.1 Projecting words to word vectors

Given a set of D documents $\{d_1, d_2, \dots, d_D\}$, a dictionary containing n words is built accordingly. By applying Word2Vec model, we predicts the word w_x by its contextual words $\{w_{x-2}, w_{x-1}, w_{x+1}, w_{x+2}\}$, here window size 2 is set as an example. Thus, each word in the collection of documents is projected into a m dimension vector $v(w_x)$, where m is a predefined value usually ranging from 50 to 300.

3.2.2 Clustering words as features

Semantically related words are clustered utilizing K-means. Moreover, we are able to extract conceptual features or categories from the word clusters. For example, in processing words in the pool of documents, we group "cat" and "dog" in a cluster using K-means because of their high correlation. In addition to the cluster, an abstract feature is actually derived like "animal" or "pet'. Conclusively, we adopt K-means to assemble word vectors into N clusters, which centroids represent cumulative conceptual features generated from a series of similar meaning words.

3.2.3 Transforming features into feature vectors

It is intuitive to consolidate the word vectors that are grouped in the same cluster into a feature vector. However, sizes of clusters may vary dramatically in occasions. In order to diminish the impact of sizes of clusters and unify a criterion for assessment, we adjust the cluster vectors

using a size scaler. Equation (11) provides a mathematically demonstration of the aggregating and rescaling process.

$$v(cluster_c) = \frac{\sum_{i=1}^{X} v(w_x^c)}{X} \tag{11}$$

Where X is the number of words in $\mathrm{cluster_c}$.

3.2.4 Converting documents to document vectors

Furthermore, we develop the idea of converting documents to document vectors. Though documents are consisted of both topic related words and uninteresting trivial words, it is expected that they are closer to location of the document topics in the semantic space. Therefore, we define a document vector $v(document_d)$ as a representation of $document_d$ which can be used to evaluate the closeness between topics of document partition document and topics of the document. Similar to the calculation of the cluster vector in step 3, equation (12) shows the way to determine the document vector.

$$v(document_d) = \frac{\sum_{i=1}^{Y} v(w_x^d)}{Y} \tag{12}$$

Where Y is the number of words in $\mathrm{document_d}$.

3.2.5 Projecting partitions of document to partition vectors

Accordingly, we assume partitions of a document share the same property of document that each partition concentrates around the neighborhood of its topics in the semantic space. Thus, we propose a partition vector by accumulate information of the word vectors in it. In the calculation of a partition vector, the size scaler is denoted as Z in equation (13).

$$v\left(partition_k^d\right) = \frac{\sum_{i=1}^{z} v(w_x^k)}{Z} \tag{13}$$

3.2.6 Measuring similarity between documents and features

Construction of document vectors and feature vectors allows us to measure the similarity between a document and any semantic category that K-means generates. In this step, we introduce the intermediate factor — features rather than compare the similarity between a document and its partitions directly. Since a feature is derived to represent a more conclusive concept, adopting it enables us to generate the semantic meanings of both documents and document partitions by a global view. Additionally, it allows us to perceive the topics hidden in the document partitions which is measured in the following step. Cosine distance is calculated by equation (13). Thus, each document can be further represented by a distance distribution over all features in the form of a vector (14).

$$dis_d^c = 1 - \frac{v(document_d) \cdot v(cluster_c)}{\| v(document_d) \|_2 \| v(cluster_c) \|_2} \tag{13}$$

$$v(document_d)_{NEW} = (dis_d^1, dis_d^2, \dots, dis_d^{N-1}, dis_d^N) \tag{14}$$

where N is the total number of clusters generated from the pool of documents $\{d_1, d_2, \dots, d_D\}$,

3.2.7 Evaluating similarity between a document partition and features

Likewise, we use equation (15) to calculate the cosine distance from a document partition to a feature which is a representation of similarity between the document partition and a topic. Furthermore, each partition is restructured into a new vector, where each element is the cosine distance (16).

$$dis_k^{d,c} = 1 - \frac{v(partition_k^d) \cdot v(cluster_c)}{\| v(partition_k^d) \|_2 \| v(cluster_c) \|_2} \quad (15)$$

$$v(partition_k^d)_{NEW} = (dis_k^{d,1}, dis_k^{d,2}, \dots, dis_k^{d,N-1}, dis_k^{d,N}) \quad (16)$$

3.2.8 Assessing similarity between documents and document partitions using distance vectors

In this step, semantic similarities between a document and its partitions are measured using the newly constructed distance vectors (17).

$$dis_k^d = 1 - \frac{v(partition_k^d)_{NEW} \cdot v(document_d)_{NEW}}{\| v(partition_k^d)_{NEW} \|_2 \| v(document_d)_{NEW} \|_2} \quad (17)$$

3.2.9 Identifying non-topic related partitions

Distance measured in the preceding step represents similarity between a document and a partition of it. Shorter distance indicates higher correlation in their topics, and vice versa. Thus, we set a criterion to evaluate the correlations. Finally, partitions with longer distance comparing to the cut-off value are eliminated. If no partition of a given document meet the predefined criteria, the one with smallest distance is retained. Therefore, the remaining partitions construct a new representation of the document topics.

From: lerxst@wam.umd.edu (where's my thing)
Subject: WHAT car is this!?
Nntp-Posting-Host: rac3.wam.umd.edu
Organization: University of Maryland, College Park
Lines: 15
I was wondering if anyone out there could enlighten me on this car I saw the other day. It was a 2-door sports **car**, looked to be from the late 60s nearly 70s. It was called a Bricklin. The doors were really small. In addition, the front bumper was separate from the rest of the body. This is all I know. If anyone can tell me a model name, engine specs, years of production, where this car is made, history, or whatever info you have on this funky looking car, please e-mail.

Thanks,
- IL
---- brought to you by your neighborhood Lerxst ---- "

Figure 3-2 An example of a document from 20 Newsgroup datasets

We implement our proposed method on the 20 Newsgroups dataset, which consists of 18,846 labeled newsgroup documents proposed by Ken Lang [24]. The entire datasets are utilized to generate word vectors using the Word2Vec model. To perform the experiment we apply Hierarchical Softmax framework and CBOW scheme as algorithm of the Word2Vec model, and execute it by Python Gensim package with default settings [25].

In addition, Python Scikit-learn package is selected to implement the K-means cluster algorithm and the Support Vector Machine (SVM) classification algorithm [10].

Figure 3.2 illustrates an example of a newsgroup document in the dataset. Table 1 shows the analysis results according to it. We divide the document in Figure 3.2 into 11 partitions after removing the symbols, numbers, stop-words and some low-frequency words. Each of these partitions includes 6 words, which are listed in column 2 of the table. Following the steps described in the previous section, the distances between each partition and the document are

measured and recorded in column 3. It is easy to acquire the topic from the original document which is related with the keyword — "car". Setting criterion to 0.01, partitions 5, 7, 8, 9, and 10 are selected as representations of the document which are "car" related. This result agrees with our assumption that shorter distances indicate higher correlations. Other non-topic related partitions, therefore, are removed to reduce the size of the document.

Table 3-1Measurement of similarity between a document and its partition using cosine distance

No.	***Contents of Partition***	***Distance***
1	lerxst, wam, umd, ed, thing, subject	0.0143
2	car, nntp, posting, host, rac, wam	0.0182
3	umd, ed, organization, university, maryland, college	0.0110
4	park, lines, wondering, anyone, could, enlighten	0.0101
5	car, saw, day, door, sports, car	**0.0071**
6	looked, late, early, called, bricklin, doors	0.0126
7	really, small, addition, front, bumper, separate	**0.0081**
8	rest, body, know, anyone, tellme, model	**0.0065**
9	name, engine, specs, years, production, car	**0.0071**
10	made, history, whatever, info, funky, looking	**0.0070**
11	car, please, mail, thanks, brought, neighborhood	0.0196
12	lerxst	0.0341

To quantify the effects of our proposed method, additional experiments are implemented. We conduct classification analysis on both our processed documents that consists only of topic-related partitions and the "original" documents. Both types of documents are pre-processed by removing symbols, numbers, stop-words and very low-frequency words. The only difference

between them is our processed documents eliminate non-topic related partitions. This analysis attempts to evaluate the impact of reduced document size to subsequent analyses by comparing classification accuracies between the two sets of documents.

Table 3-2 Classification results under different scenarios using bag-of-words model

Avg. Document Size	No. of Clusters	Distance Threshold	Std. Dev.	Avg. 10-Fold F1-Score
Proc. Doc				
24	50	0.005	0.009	0.638
53	50	0.010	0.009	0.732
92	50	0.015	0.010	0.792
115	50	0.020	0.008	0.825
50	100	0.005	0.010	0.707
90	100	0.010	0.011	0.803
110	100	0.015	0.008	0.835
126	100	0.020	0.007	**0.850**
25	150	0.005	0.009	0.646
68	150	0.010	0.007	0.751
101	150	0.015	0.008	0.807
124	150	0.020	0.007	0.833
Orig. Doc				
160	-	-	0.009	**0.841**

Table 3-3 Classification results under different scenarios using distributed representation model

Avg. Document Size	No. of Clusters	Distance Threshold	Std. Dev.	Avg. 10-Fold F1-Score
Proc. Doc				
24	50	0.005	0.010	0.702
53	50	0.010	0.011	0.765
92	50	0.015	0.012	0.812
115	50	0.020	0.013	0.801
50	100	0.005	0.012	0.694
90	100	0.010	0.014	0.762
110	100	0.015	0.009	**0.834**
126	100	0.020	0.013	0.810
25	150	0.005	0.011	0.699
68	150	0.010	0.013	0.766
101	150	0.015	0.012	0.823
124	150	0.020	0.012	0.805
Orig. Doc				
160	-	-	0.012	**0.806**

We explore the performance of the proposed method under several scenarios. Different clusters are tested to assess the impact of various topic specification scales. Word clusters group into different semantic categories with predefined number 50, 100, and 150. Additionally, we contrast the outcomes by various distance criteria as well, which are set to 0.005, 0.010, 0.015, and 0.020 respectively. Utilizing SVM we conduct the classification analysis under bag-of-words and distributed representation model, and measure the results with average 10-fold F1-score as well as standard deviation. Table 3.2 compares classification accuracy under different scenarios and also demonstrates the average document size after removing non-topic related partitions.

Results of Table 3.2 and Table 3.3 are derived from bag-of-words method and distributed representation method respectively. After removing symbols, numbers, stop-words and other low-frequency words, the average size of the original documents is 160 in the collection of the 18,846 documents. The average size of the processed documents varies according to different scenarios. The smaller distance criterion is selected, the less words are retained the documents. Since smaller and stricter criterion is associated with higher correlation. Thus, setting the criterion to 0.005 enables the method to remove more moderately related partitions than setting it to 0.020. However, the average document size is not linearly associated with the predefined number of clusters. Smaller number of clusters implies the generated features are broader and more general. Topic of a feature could be very ambiguous because the feature is an aggregation of a very wide range of words. In contrary, larger number of clusters indicates the derived features are very specific. Topics of this type of features could be very vague. Since these features probably are constructed with extremely less words than its natural semantic categories should contain.

Apparently the most accurate classifications identified from Table 3.2 and Table 3.3 belong to the processed documents group. 85% of the documents are detected to the correct labels using

bag-of-words method, 83.4% are properly classified implementing while distributed representation method, and in contrast results from the original documents are 84.1% and 80.6%. Both improve the accuracy of converting the original documents under different methods. Moreover, with the improvement in accuracy, the document sizes reduced significantly. Comparing with the average size of original documents 160, the number of words in the best scenarios are 126 and 110. With a decrease of document sizes about 21% - 31%, considerable storage space is released. Hence, our proposed method significantly abbreviates the original documents while maintains important topic-related information as a suggested step prior to further analyses.

4 A TEXT INFORMATION RETRIEVAL METHOD USING AUGMENTED MATRIX

4.1 Introduction

Text information retrieval (IR) techniques are widely used to generate abbreviate and essential representations of documents to improve performances of Natural Language Processing (NLP) tasks like classification, clustering, and summarization, and etc. [2]. In recent years, many methods have been proposed which are converged to two general types of techniques — one-hot models and distributed representation models.

One-hot models such as Latent Dirichlet Allocation (LDA) investigates latent topics from a collection of documents and represent each document with a probability distribution over the discovered topics [10]. A topic is extracted and summarized from distinct words with semantic similarities. Probability distributes differently according to occurrence of a word in a specific topic. It is very plausible that a given document is associated with a topic if the document contains high probability words from the topic, and vice versa. Specifically, LDA concentrates on handling those high probability words and disregard any low probability ones. Distributed representation models are also extensively utilized, including Word2Vec which predicts words using word vectors that are generated by the surrounding contexts of the target words. In contrast to LDA, Word2Vec is able to predict the low-frequency words accurately [7].

However, both models have some drawbacks according to their attributes. LDA underestimates the correlation between a document and topic when low probability words occurs frequently in the document. On the other hand, Word2Vec model merely uses local contexts within sentence to generate word vectors. It is impossible to globally improve the quality of word representations. In this section, therefore, we propose a hybrid method to represent documents by integrating the global statistical model and local semantic information. The experimental results

indicate our method improve accuracy of classification analysis significantly comparing to either LDA or Word2Vec.

4.2 Overall Framework

In this section, we propose a method attempts to summarize a document in a numeric form which incorporate both global and local information prompted by LDA and Word2Vec. Thus, we describe a document as a vector which is a mixture of word-to-topic association and word-to-word correlation; where the first part is the probability distributed over latent topics generated from the corpus and latter part is topographic association of each word in a document in semantic space. Following is the generative process of document vectors in 3 steps.

4.2.1 Generating word-to-topic document vectors

Adopting the idea of LDA, we convert a document into a vector, where each element of the vector corresponds to the probability that a topic is included in the document in equation (12).

$$p(w|\alpha,\beta) = \int p(\theta|\alpha)(\prod_{i=1}^{N}\sum_{k=1}^{M} p(z_k|\theta)p(w_i|z_k,\beta))\, d\theta \tag{12}$$

where w is a document or a collection of words, and α is the parameter vector of the Dirichlet distribution that is used to generate θ. Probability of selecting a topic for a document following Multinomial distribution $p(z_k|w) \sim Mult(\theta)$. β is a matrix generated by the entire corpus, that each element represents the conditioned probability of a word is included in a given topic. The document vector is $v_1(w) = p(w|\alpha,\beta)$.

4.2.2 Deriving word-to-word document vectors

Employing Word2Vec, we are able to transform each word in a document into a word vector which is an aggregation of its surrounding word vectors. For a semantic space, a k

dimensional vector is generated to initialize a word, where k is a predefined value for dimensionality. The vector is updated using logistic regression according to word-to-word relationship demonstrate by Huffman tree code. Subsequently, given a word w_x, the word vector is determined as the aggregation of its surrounding word vectors. Use 2 as the size of window, thus,

$$v(w_x) = \sum_i v(w_i), i = x-1, x-2, x+1, x+2 \tag{13}$$

Intuitively, we define a document vector by consolidate all word vectors in the document, and adjust it using a size scaler N.

$$v_2(w) = \frac{1}{N}\sum_x v(w_x) \tag{14}$$

N is the size of the document.

4.2.3 Construct augmented document vectors

We augment the word-to-topic and word-to-word vectors to construct vectors for each document accordingly in the corpus. Hence, each vector is considered to be a representation of both global and local information shown in equation (15).

$$v(w) = [v_1(w)|v_2(w)] \tag{15}$$

4.3 Experiments

20 Newsgroups dataset is used to implement our idea, which collects 18,846 labeled newsgroup documents organized and proposed by Ken Lang [24]. We extract latent topics from the entire corpus, and generate document vectors under the hybrid model of LDA and

Word2Vec. Classification analysis is conducted using Python Gensim package with default settings [25] and Scikit-learn package [26].

Table 4-1 Classification results using the new document representation

# Topics	Word2Vec Average F1-Score (Std. Dev)	LDA Average F1-Score (Std. Dev)	Our method Average F1-Score (Std. Dev)
50	0.807 (0.013)	0.620 (0.018)	0.815 (0.011)
75	0.807 (0.013)	0.662 (0.013)	0.832 (0.009)
100	0.807 (0.013)	0.665 (0.017)	0.843 (0.013)
125	0.807 (0.013)	0.668 (0.014)	0.854 (0.010)
150	0.807 (0.013)	0.672 (0.016)	0.855 (0.009)
200	**0.807 (0.013)**	**0.680 (0.015)**	**0.856 (0.010)**
250	0.807 (0.013)	0.654 (0.017)	0.827 (0.011)
300	0.807 (0.013)	0.652 (0.013)	0.808 (0.013)

We use Table 1 to demonstrate accuracy of classification analysis among LDA, Word2Vec, and our method. Column 3 shows the average F1-scores according to different number of topics defined, which is enhanced considerably with smaller variations comparing to either LDA or Word2Vec. The best scenario in the experiment is achieved when the number latent topics is set to 200 generated from the 18,846 documents. On average, 85.6% of documents are corrected labeled with a favorable standard deviation 1%, which indicates a very steady classification task is performed. Moreover, we observe an incremental change in the accuracy in the test as the number of topics increases from 50 to 200. But the precision declines when number of topics is set over 200. Since the predefined number of topics determines the coverage of each topic. The experimental results imply that the coverages of latent topics significantly the generation of satisfactory document representations.

5 A DOCUMENT REPRESENTATION METHOD USNG WORD2VEC AND LATENT DIRICHLET ALLOCATION

5.1 Introduction

With the explosive growth of online resources such as web pages, blogs, and social networks, text mining plays a more and more important role to analyze and organize these documents. An excellent representation of textual data should contain as much information as possible from the original document. Generally, there are two ways to represent a document — one-hot encoding and word embedding.

One-hot encoding describes a word in a high-dimensional vector, which is a dictionary composed of all words occurred in a set of documents. Each word is represented by a vector with 1 at its corresponding position and 0 in other positions. A model to represent a document called "bag-of-words" was proposed [5]. It sums up all the one-hot vectors in a document; and each element in the resulting vector becomes the occurrence of a word. Furthermore, Term Frequency-Inverse Document Frequency (TF-IDF) model was given to replace the counts with TF-IDF score [6]. The new model calculates TF scores to the selected high frequency words in a document, and also measures how unique these words occur across all documents using IDF scores. Using the product of the TF-IDF scores, the high frequency but less meaningful words can be eliminated, such as "that", "this", "the", etc.

LDA is a probabilistic topic model to discover latent topics from a large volume of documents and describe each document with a probability distribution over the discovered topics [10]. It is commonly considered as a feature reduction method by grouping words in different topics, thus a document can be mapped to a lower dimensional space. Additionally, words are assumed to occur independently in LDA, and documents are treated as bag-of-words. Therefore,

LDA does not study the contextual relationship among words. Moreover, LDA is also a doubly sparse model which prefers fewer topics in each document and fewer words to describe a topic. Thus, the document vector is very sparse.

Recently, Word embeddings has been a strong trend in Natural Language Processing. It distributes a word in a low-dimension vector that is highly correlated with the real semantics. Generally speaking, there are two approaches: one builds a co-occurrence matrix for the entire document and reduces the size of the matrix to generate words and context, such as Glove, Spectral Word Embeddings, and Word Embeddings through Hellinger PCA (HPCA) [20] [27] [28]. The other one, such as Latent Semantic Analysis (LSA), density based word embeddings, and Word2Vec, predicts a word by inspecting its surroundings [29] [30] [7]. For example, if two words “soccer” and “basketball” occur in a same “position” in two sentences “I like soccer” and “I like basketball”, “soccer” and “basketball” are more likely related either in semantics or syntactic. A method clusters the embedded word vectors as features and uses a count distribution as document representations [23]. A Doc2Vec model trains a document vector by a linear combination of the embedded word vectors [31].

LDA with strong capability to extract the main contents of the article is quite interpretable by humans. Therefore, many researchers use LDA on the text classification tasks. A feature-enhanced smoothing method was developed [32]. Those words existing in testing documents but not in the training corpus are very useful to improve accuracy of classification and the quality of features. An improved algorithm gLDA was designed by containing categories for each document [33]. The probability distribution of each document is generated by the most relevant categories of documents. The word-topic mapping performance was improved by using a large-scale trained corpus applied to the data with smaller corpus [34]. Websites were divided

into different subjects with slash tags and a relationship between each subject and topics used to classify the data was found [35]. A novel classifier named Multi-LDA Boost applied a boosting strategy by choosing the best scenario from multiple models with different parameters, and performed a weighted method to improve the accuracy of categorization [36].

Similar ideas but different methods and purposes by integrating LDA and Word2Vec are implemented. LDA models a global relationship from each document to all topics, and Word2Vec in the other hand captures the relationships by learning the target word from its contexts. Topic2Vec integrates the word contextual information from Word2Vec to learn topic representations in LDA, whose resulting topics are much more distinguishable than those generated by LDA [37]. LDA2Vec successfully uses the contextual word information to learn much more interpretable topics by adding the document vector in the step of generating word vectors [38].

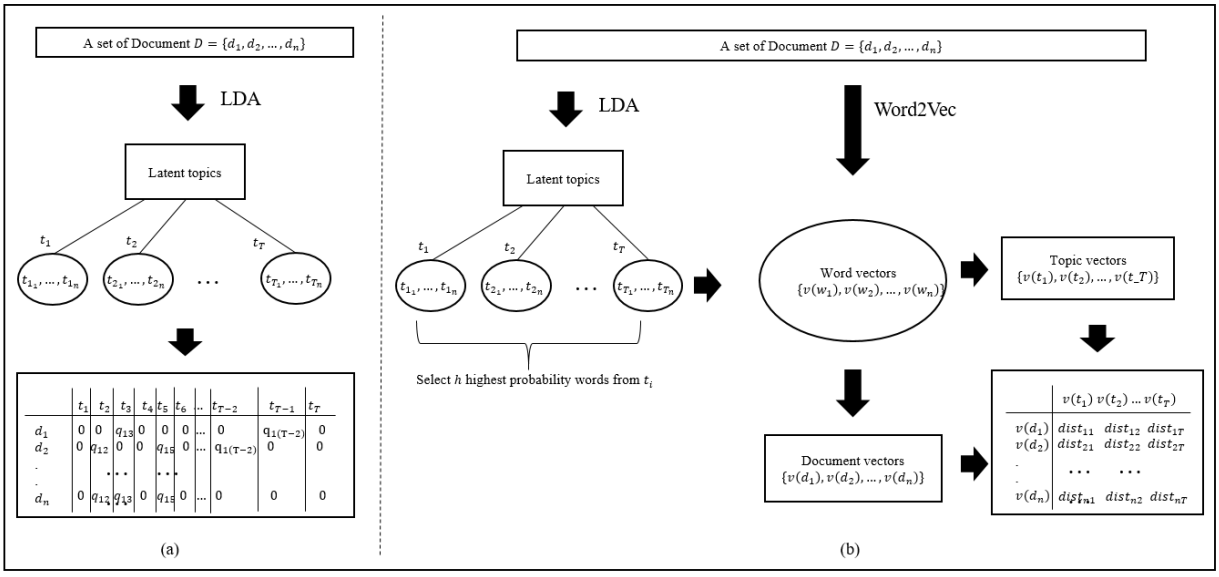


Figure 5-1 Processes and results of LDA (a) and out hybrid method (b)

5.2 Overall Framework

As we have discussed, LDA is a way to describe a global relationship among documents, while Word2Vec predicts words in a very local manner. So, we combine these two techniques to use a more comprehensive vector to represent documents, meanwhile, the new representation with a density vector enhances the capability of discrimination and predication applied to Natural Language Process tasks.

Our new method as shown in Fig. 3 (b) projects words, documents, and topics in a high-dimension semantic space. A document vector is considered as a single vector, which is the centroid of all words in the document as what Word2Vec does in the projection layer. In addition, each document has its individual length, thus its vector is divided by the number of words in the document to guarantee the measurements with same scale. We construct topic vectors in a similar way, but it is a little more complicated. A subset of h high-probability words in each topic is employed to represent the topic, and then their probabilities are rescaled as the weights of words. Hence different words have different contributions to the topic. We measure Euclidean distances from each document to topics so that a document can be represented with a distance distribution.

In details, given a set of documents $D = \{d_1, d_2, \dots, d_n\}$, whose vocabulary is built with N words $\{w_1, w_2, \dots, w_N\}$. By training D, LDA outputs latent topics $\{t_1, t_2, \dots, t_T\}$ and probabilities of words in each topic t_i, where the j^{th} word in t_i is denoted as θ_{i_j}. Word2Vec trains D and vectorizes each word in vocabulary into a fixed length vector $\{v(w_1), v(w_2), \dots, v(w_N)\}$. To generate topic vectors, h highest-probability words in $t_i$are selected. Meanwhile, the probabilities of words in t_i are rescaled as weights in (16). In (17), the topic vector $v(t_i)$is calculated by summing the productions of each word vector and its weight.

$$\omega_i = \frac{\theta_i}{\sum_{n=1}^{h} \theta_n} \tag{16}$$

$$v(t_i) = \sum_{n=1}^{h} \omega_{i_n} v(w_{i_n}) \tag{17}$$

Next, we calculate document vectors $v(d_i)$ by (18), where c is the number of words in the document.

$$v(d_i) = \frac{\sum_{n=1}^{c} v(w_{i_n})}{c} \tag{18}$$

Therefore, each document can be represented by a distance distribution from the document to all topics in a semantic space, and a distance is calculated as (19).

$$distance(v(d_i), v(t_i)) = |v(d_i) - v(t_i)| \tag{19}$$

Therefore, the new defined vector is no longer sparse by comparing the results in (a) and (b) of Fig. 3, which distributes risks to all elements $dist_{ij}$ in the vector, also possesses more information in $dist_{ij}$ to be discriminated from other documents. Moreover, the probability distribution over topics in LDA is still held in the space generated by the new method; and these topics are closer to the specific document. Meanwhile, other related important topics are found as well because of more word-level information is involved.

5.3 Experiments

The 20Newsgroups dataset contains 18846 newsgroup documents collected by Ken Lang, which is organized into 20 different newsgroups [24]. We use all the documents in the dataset to train LDA and Word2Vec to extract latent topics and word vectors. Both LDA and Word2Vec are implemented with Gensim which is a free Python package widely used for topic models [25].

In LDA, the hyperparameter α is set to 0.1 and $passes$ is set to 20 to guarantee convergence. Word2Vec uses the CBOW model with default settings in Gensim. We use the Python scikit-learn package to perform SVM for the classification with $gamma = 0.001$ [39].

From: lerxst@wam.umd.edu (where's my thing)
Subject: WHAT car is this!?
Nntp-Posting-Host: rac3.wam.umd.edu
Organization: University of Maryland, College Park
Lines: 15
I was wondering if anyone out there could enlighten me on this car I saw the other day. It was a 2-door sports **car**, looked to be from the late 60s nearly 70s. It was called a Bricklin. The doors were really small. In addition, the front bumper was separate from the rest of the body. This is all I know. If anyone can tell me a model name, engine specs, years of production, where this car is made, history, or whatever info you have on this funky looking car, please e-mail.

Thanks,
- IL
---- brought to you by your neighborhood Lerxst ---- "

Figure 5-2 An original document sample

In the first part of our experiment, we tested whether our new representation carried the relationships from LDA, and enriched results with extra benefits Word2Vec brings. Next, we compared our method with three other methods on classification tasks.

Table 5-1 Topic distribution and distance distribution of the example document

Topic Dist. (Index, Prob.)	**Distance Dist. (Index, Distance)**						
	(1, 2.005)	(2, 1.597)	(3, 2.376)	(4, 1.285)	(5, 2.119)	(6, 1.646)	(7, 1.889)
	(8, 2.704)	**(9, 1.698)**	(10, 2.254)	**(11, 1.692)**	(12, 1.691)	(13, 2.646)	(14, 1.743)
	(15, 1.154)	(16, 1.928)	(17, 1.269)	(18, 1.706)	(19, 2.147)	(20, 1.775)	(21, 1.792)
	(22, 1.603)	(23, 1.927)	(24, 2.030)	(25, 1.421)	(26, 2.169)	(27, 1.418)	(28, 1.803)
	(29, 1.774)	(30, 2.323)	(31, 1.793)	(32, 1.568)	(33, 2.010)	(34, 1.604)	(35, 2.067)
	(36, 1.107)	(37, 1.591)	(38, 1.594)	(39, 1.743)	(40, 2.096)	(41, 1.879)	(42, 1.703)
	(43, 1.655)	(44, 1.286)	***(45, 3.728)***	(46, 2.107)	**(47, 1.269)**	(48, 1.638)	**(49, 0.985)**
	(50, 2.002)						
	*Mean = 1.820 Minimum=0.985						

Table 5-2 Words contained in topic 15 and topic 45

Topic 15	Topic 45
writes, **bike**, article, dod, lines, organization, org, posting, nntp, host, apr, rochester, **bmw**, mitre, ride, upenn, clarkson, dog, att, riding, sas, **motorcycle**, john, **bikes**, reply, shaft, inc, rec, rider, noise, helmet, chain, well, mail, list, ahl, **motorcycles**, like, tek, ysu, **sport**, dave, corporation, road, bill, wave, wax, yfn, lock, pink	max, bhj, giz, scx, rlk, chz, qax, bxn, biz, air, fij, okz, gcx, nrhj, rck, ync, frustrated, uww, fil, cho, mvs, hernia, nei, mbs, tct, rmc, lhz, umu , wwiz, nuy, ahf, qtm, ghj, kjz, vmk, ecs, mcx, fpl, syx, pmf , dct, barman, srcs, gizw, mkg, qvf, bhjn, mgb, mas, khf'

Fig. 4 shows an original news document from the 20 Newsgroups dataset. After learned by LDA, the topic distribution is shown in the left column of Table 1, which has 5 very relevant topics 9, 11, 22, 47, and 49. Except for these, other topics with values of zero are considered as irrelevant. The right column is the distance distribution using our method. The bolded topics underlined are corresponding to the topics listed in the left column, where the first value is the index of topics and the second value is the normalized distance. We can see that all the highlighted values are below the mean of 1.82 and topic 49 is the minimum value 0.985 among all topics. Therefore, the conclusions of LDA are well held in our new representations. Moreover, we are also interested in the italicized and bolded topics in the right column, such as the topic 15 with a very short distance and topic 45, who has the longest distance. To interpret these, we investigate Table 2 at first, which has two word lists of topic 15 and topic 45. Topic 15 is found by our method but missed by LDA. The words in topic 15 such as “motorcycle”, ‘bmw’, ‘bike’, ‘sport’, etc. are quite related to the word ‘car’ mentioned in the example document, where ‘car’ and ‘motorcycle’, ‘bike’ belong to vehicles, also ‘bmw’ is a brand of ‘car’. Therefore, topic 15 and the example document are highly relevant at the word-level. Topic 45 obviously contains a lot of ‘trash’ words without any semantic meanings, therefore, it makes sense that it has the longest distance to the document.

To investigate the performance, we compare TF-IDF, Word2Vec, LDA and our method using the SVM model. From Table 3, TF-IDF performs the best prediction with about 2% more accuracy rate over our method, but it takes more than 4 times of the running time comparing with our method with about 20000 features. In contrast, Word2Vec only trains a predefined number of features no more than 500, while LDA and our method only train the same number of features as topics. With the growing words occurred in documents, the performance of TF-IDF begins to decrease more drastically. Moreover, similar to LDA, TF-IDF describes the occurrences of words without semantic meanings. Considering the prediction accuracy, running time, and the volume of semantic information involved, our method performs effectively among any single methods.

Table 5-3 Average 10-fold micro-F1 score of different methods

Methods	Average 10-fold micro-F1 score
TF-IDF+SVM	0.822
Word2Vec+SVM	0.717
LDA +SVM	0.639 (# topic = 100)
Our method	0.803 (# topic = 250)

Table 5-4 Average 10-fold micro-F1 score of LDA and our method under different number topics

	Average 10-fold micro-F1 score (Standard deviation)					
	50 topics	**100 topics**	**150 topics**	**200 topics**	**250 topics**	**300 topics**
LDA+SVM	0.615 (0.016)	0.639 (0.016)	0.639 (0.016)	0.619 (0.015)	0.589 (0.010)	0.536 (0.019)
Our method	0.748 (0.016)	0.777 (0.014)	0.794 (0.013)	0.796 (0.015)	0.803 (0.012)	0.789 (0.015)

Another experiment is conducted to evaluate the performances of LDA and our method under different number of topics. As shown in Table 4, 100 - 150 topics are the optimal range of LDA to categorize the newsgroups data, while the range of 150 -250 topics is the best for our

model. Therefore, our method requires more topics to predict documents because our document representation takes more contextual information in consideration.

6 A HYBRID MACHINE LEARNING METHOD FOR FINDING DEPRESSION RELATED PUBLICATION BY ELIMINATION OUTLIER PUBLICATIONS

6.1 Introduction

The sheer volume of new articles being published every day is growing exponentially. A surprising number is that there are about 300,000 literatures about "depression" in PubMed database. This increasing number of publications exposes that researchers are continuously contributing to either the depression diagnosis or treatment. The depression diagnosis always takes couple weeks to watch patients' signs or symptoms, and meanwhile understand potential risk factors such as personal or family history of depression, major life changes, trauma, or stress and so on. Depression treatment is also different from other regular diseases since is not only treated with medications, but also with psychotherapy, or a combination of the two. Therefore, depression specialists should comprehensively diagnose patient's conditions in a short period and propose a treatment plan since the earlier that treatment can begin, the more effective it is.

To improve the doctors' specialty, text mining models have been used for years to extract the significant contents from textual data, which can help doctors learn much more cases than before. Keyword-search based method to download the literatures is the easiest but most used way to collect the resources from either online or existing medical database. But the drawback of this method is that some of the downloaded literatures are not truly discussing the keyword, but only contain the targeted keyword. These literatures can result in poor accuracy or deviated results since some models update their parameters or weights by learning all training data.

Text representation is to translate text data to a model understandable language, normally as a vector of numbers. Bag-of-words and its extended form Term Frequency-Inverse Document Frequency (TF-IDF) are most used because of its simplicity and intuitive [5] [36]. Latent

Dirichlet Allocation (LDA) is a probabilistic topic model that discovers hidden topics from literatures [10]. Each literature is represented with a set of probabilities to the latent topics. A feature-enhanced smoothing method finds that the words in testing literatures but not in training corpus are significant to the accuracy in classification models [32]. An improved algorithm gLDA uses large and grouped data to train the model and apply it to smaller corpus to improve the accuracy [33]. Multi-LDA uses the boosting strategy to train multiple modes with different parameters, and performs a weighted method to improve the accuracy [36]. Word Embedding is another method to represent word in a vector format. Global Vectors for Word Representation (Glove), Spectral Word Embedding and Word Embedding through Hellinger PCA (HPCA) use a co-occurrence matrix for the entire document and reduce the size of the matrix to generate word and context [20] [27] [28]. Latent Semantic Analysis (LSA), density based word embedding, Word2Vec predict a word by its surroundings with contextual information [40] [30] [7]. Averaging the word vectors in a document is a way to calculate the document representation. There are also other ways such as Doc2Vec, that combines words with a linear relationship [31].

Outlier analysis has been studied and applied in many applications. Numerical methods have been proposed. Distance-based methods declare outliers which are far away from the dense regions [41] [42]. Density-based methods declare outliers with low local density with respect to the remaining points [43]. Subspace methods declare outliers based on subspace behavior of the underlying data [44] [45] [46] [47] [48]. Recent years, Non-negative Matrix Factorization (NMF) is used to detect outliers. It is derived from the low rank approximation technique but constrain the fact matrices with non-negative values to find out abnormal individuals [49] [50] [51] [52].

Next, we propose a novel text representation method paired with One-class Support Vector Machine to detect and eliminate outlier literatures to improve the data purity. Word2Vec

and LDA are two important models to our method which are explained in section 6.2. In section 6.3, One-class SVM is introduced. In section 6.4, experiment results are discussed. Section 6.5 are conclusions and future insights.

6.2 Text Representation Method

Feature extraction is a dimensionality reduction approach to represent data in a compact feature vector. Literatures composed of words are not recognized by models. Therefore, literature representation is to convert texts to a model understandable format, and contain as much information of the original texts as possible. Next, we will explain how to incorporate two advanced models with better performance.

6.2.1 Word2Vec

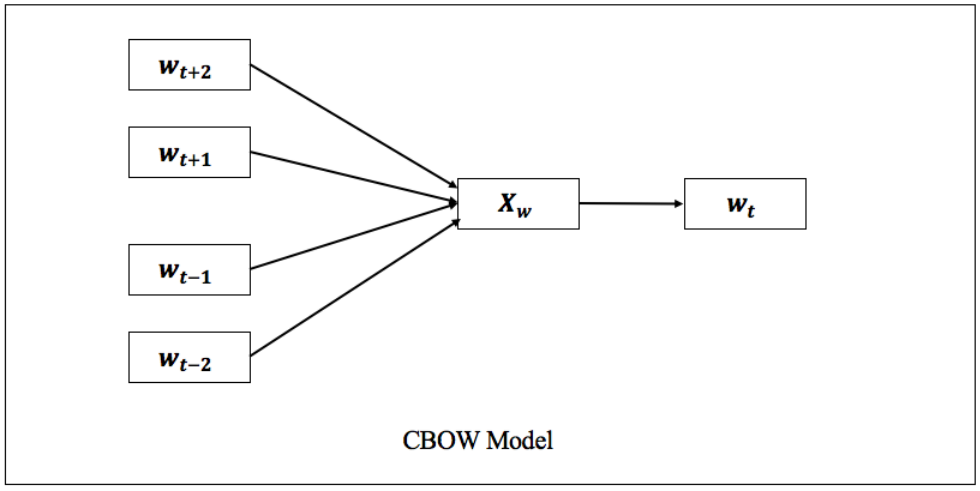


Figure 6-1 Word2Vec CBOW model.

Word2Vec based on a deep neural network predicts word vectors by their surroundings. Thus, words are mapped to a lower dimension space, where similar meaning words are assigned at closer positions. Any occurrence of a word will update its word vector as well as its's neighbors. A Word2Vec model can guess word associations (such as "man" is to "boy" what

"women" is to "girl") as well as understand meanings across different languages (such as "ONE" in English and "UNO" in Spanish are at the very closed positions in the vector space).

Next, one of the Word2Vec architecture CBOW will be introduced. Each model contains three layers: input layer, a projection layer, and an output layer.

Input layer: w_t is the targeted word vector and it is predicted by four neighbors w_{t-2}, w_{t-1}, w_{t+1}, and w_{t+2}, where window size is set to 2.

Projection layer: a contextual vector

$$X_w^{context} = \sum_{i=t-2}^{t+2} w_i \text{ where } i \neq t.$$

Output layer: A Huffman tree is constructed and all occurred words are assigned to leaf nodes, where higher frequency words with shorter depth. Thus, each word in the tree is represented with a unique Huffman code to be accessed.

A child to its parent is estimated by a logistic model

$$\sigma(X_w^T\theta) = \frac{1}{1 + e^{-x_w^T\theta}}$$

and the other child is calculated by $1 = \sigma(X_w^T\theta)$

Then, the probability of w_x to its neighbors at each node is calculated by (1).

$$p(X_w|X_w^{context}) = \left[\sigma\left(X_w^T\theta_{j-1}^w\right)\right]^{1-d_j^w}\left[1 - \sigma\left(X_w^T\theta_{j-1}^w\right)\right]^{d_j^w} \quad (20)$$

where $d_j^w \in \{0,1\}$ is the j^{th} digit in word w's Huffman code and j is any node on the path except as the leaf node.

Finally, the objective function is optimized by the gradient descent method to maximize the log-likelihood (2).

$$\mathcal{L} = \sum_{w \in C} \log \prod_{j=2}^{n} \left\{ \left[\sigma\left(v(x_w)^T \theta_{j-1}^w\right)\right]^{1-d_j^w} \left[1 - \sigma\left(v(x_w)^T \theta_{j-1}^w\right)\right]^{d_j^w} \right\} \tag{21}$$

6.2.2 Latent Dirichlet Allocation

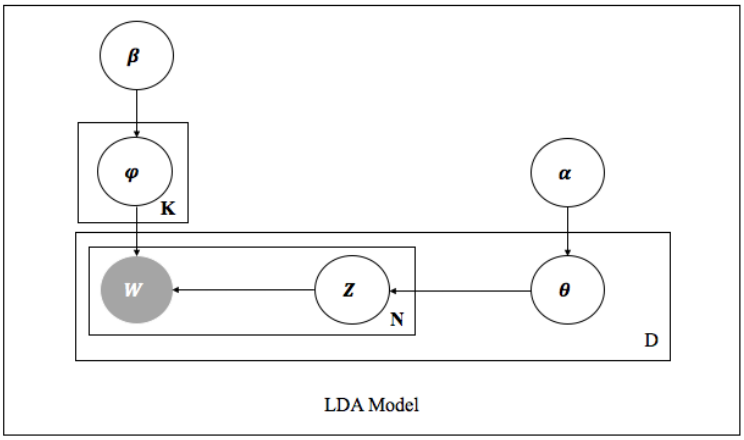


Figure 6-2 A graphic of LDA model

LDA is an unsupervised method to discover the latent topics Z from a collection of documents D. In LDA, each document d is represented as a probability distribution θ_d over topics, where each topic z is a probability distribution φ_z over all words in vocabulary. Figure 2 shows the generative process. Both θ and φ have prior distributions with hyperparameters α and β. For every word w_{d_i} in document d, a topic z_{d_i} can be extracted by equations (22) and (23), a word w_{d_i} can be returned. Repeat (3) and (4) N times, a document d is generated, where N is the size of document d.

$$\theta_d \sim Dirichlet(\alpha) \quad z_{d_i} \sim Multinomial(\theta_d) \tag{22}$$

$$\varphi_z \sim Dirichlet(\beta) \quad w_{d_i} \sim Multinonial\left(\varphi_{z_{d_i}}\right) \tag{23}$$

By using Gibbs Sampling, θ and φ can be inferred to discover the latent topics in documents, and predict any new document with a topic proportion distribution.

6.2.3 Our Hybrid Method

LDA is a topic model that discovers the hidden topics from literatures. It overlooks the entire literatures to extract a global relationship based on the word occurrences and distributions. In contrast, Word2Vec predicts a target word by focusing on its neighboring words as shown in figure 1. Therefore, a word vector well maintains the associations to its neighbors. LDA and Word2Vec are complements to each other, this is one reason that we combine the two models. The other reason is the rule of LDA to assign topics to each literature. LDA is a doubly sparse model that uses fewer topics to describe a literature. Thus, the literature vector is too sparse to be applied to the models. Next, our proposed literature representation method will be explained on how to overcome the two points mentioned above.

n topics $\{T_1, T_2, \dots, T_n\}$ are extracted from the literatures by LDA. And then a Word2Vec model is trained by the entire literatures. d word vectors are $\{w_1, w_2, \dots, w_d\}$. A literature is a collection of words, which can be calculated by summing the containing word vectors in Equation (24),

$$v = \frac{\sum_{n=1}^{c} w_L}{c} \tag{24}$$

where $w_L \in \{w_1, w_2, \dots, w_d\}$. and c is the number of words in L

Like generating the literature vectors, topics also are collections of words. Each topic contains its most related keywords that are ranked by their probabilities. The probability θ of a word can be interpreted as its importance to the topic. If a literature contains a keyword with high probability to a topic, there is more likely the literature has strong association to the topic.

In our method, top rated h words from each topic are used to generate the topic vector. Probabilities of words are used as the weights calculated by (23). The topic vectors are calculated by (24).

$$wt = \frac{\theta}{\sum_{n=1}^{h} \theta_n} \tag{23}$$

$$v(T) = \sum_{n=1}^{h} (wt_n * w_{T_n}) \tag{24}$$

where wt is the weight calculated from the probability and $w_T \in \{w_1, w_2, \dots, w_d\}$.

After vectoring literatures and topics, Euclidean distance measures the similarity between literatures and topics by (25). A set of similarity values between literature and topics are new representations to describe literatures.

$$distance(v(L), v(T)) = |v(d_i) - v(t_i)| \tag{25}$$

6.3 One-Class Support Vector Machine (OCSVM)

OCSVM is a special form of support vector machine [14], which has been applied to various applications [15] [16] [17] [18]. It learns a minimum volume hypersphere that enclosed most of the data. In another word, OCSVM only recognizes one class from the normal training data. If a newly data is too different to this class, model labels this data as out-of-class. OCSVM is an optimization problem because we expect the ball as small as possible but the ball contains most of the data.

Most time we use OCSVM when we only have the data of one class. The objective function can be learned by (26) and (27):

$$\mathrm{F}(\mathrm{R}, \alpha, \xi_\mathrm{i}) = \mathrm{R}^2 + \mathrm{C} \sum_{\mathrm{i}} \xi_\mathrm{i} \tag{26}$$

$$s.t.\,(x_i - a)^T(x_i - a) \le R^2 + \xi_i, \forall_i, \xi_i \ge 0 \quad (27)$$

where a is the center, R is the radius, and x_i is the training data.

We can solve the optimization with Lagrangian multipliers in (28):

$$L(R, \alpha, a_i, \xi_i) = R^2 + C\sum_i \xi_i - \sum_i \gamma_i\xi_i - \sum_i a_i\{R^2 + \xi_i - (x_i^2 - 2ax + a^2)\} \quad (28)$$

Where $a_i \ge 0$ and $\gamma_i \ge 0$

After deriving the parameters and let derivatives set to zero, we can get (29), (30) and (31):

$$\sum_i a_i = 1 \quad (29)$$

$$a = \sum_i a_i x_i \quad (30)$$

$$C - a_i - \gamma_i = 0 \quad (31)$$

By substituting (29), (30), and (31) to the Lagrangian multiplier, we can get:

$$L = \sum_i a_i K(x_i, x_i) - \sum_{ij} a_i a_j K(x_i, x_j) \quad (32)$$

To distinguish whether a new data is normal or abnormal, the Kernel equation is applied.

$$K(z, z) - 2\sum_i a_i K(z, x_j) - \sum_{ij} a_i a_j K(x_i, x_j) \le R^2 \quad (33)$$

6.4 Experiment

The National Center for Biotechnology Information (NCBI) advances science and health by providing access to biomedical and genomic information. Our medical literatures used in the experiments are collected from NCBI database and downloaded through a Python package named Biopython. There are about 300,000 “depression” related abstracts are used to train the

model. Plus, we also randomly download 3000 “obesity” related abstracts and 3000 “myocarditis” related abstracts as the testing data. The reason we use obesity literatures as the testing data is that obesity is also a disease requires both the mental and physical aspects. Comparing with depression and obesity, myocarditis is a regular disease which only needs physical treatment. 20Newsgroups dataset is also as part of my testing data. Majority of the articles are not related to medical topics such as politics, religions etc. [24]. We randomly select 3000 articles from 20Newsgroups articles as the test data.

To simulate the real problem, we are facing, we replace the words “obesity” and “myocarditis” with “depression” in the testing data. For the articles from 20Newgroups, we add “depression” at a random position in each article.

Python packages Gensim and sciket-learn are used to implement the LDA and Word2Vec models [39]. The hyerparameter α in LDA model is set to 0.1 and passes is set to 20 to guarantee that the convergence can be finished. Word2Vec sets the window size to 5. OCSVM uses a non-linear kernel Radial Basis Function (RBF). 200 topics are used in LDA model. We use Word2Vec to map all the words in a 200 dimensions’ numerical space.

Firstly, to display the distributions, we randomly select 100 depression abstracts, 100 obesity abstracts, 100 myocarditis abstracts, and 100 articles from 20Newsgroup to visually demonstrate the performance of our proposed framework in Figure 3. 300,000 depression abstracts except the selected 100 abstracts (totally 299,900 depression abstracts) are used to train the OCSVM model. Principle Component Analysis (PCA) model is used to map each literature to a 2-D space. In Figure 3, the pink area is the learned model, where a publication in the pink area is considered as a normal point, and a publication out of this area is treated as an outlier literature. From the experiment, we find about 40% depression data not contained in the

hypersphere. To evaluate the model's distinguishability, 100 depression literatures are used as testing data. The results in the Figure 3A is that 17 of 100 abstracts are not recognized by the model. The accuracy is 83%. The figure 3B shows the results to detect the obesity abstracts. 35 of 100 literatures are found as outliers. Depression and obesity share many similar symptoms and medicines. Thus, it is very challenging to distinguish these two types of publications. In figure 3D, non-medical articles are scattered to the space with less associations to the depression abstracts, and the model performs very high accurate that 80 of 100 are considered as outliers. The reason is the entire training data only describe the shape of the regular terms, thus, it highly requires that the text representation method can keep and capture as much descriptive information as possible. Another experiment in table 1 shows the performances using our text representation method compared with TF-IDF in classification problem. We use support vector machine (SVM) as the classifier. The data in the experiment is to randomly select 20,000 abstracts from each disease and 20Newgroups dataset. The results indicate that the frequency-based TF-IDF as the text representation performs better to classify the medical and non-medical publications. However, for different types of medical publications, TF-IDF cannot discover the differences in semantics.

Back to the discussion of outlier detection, we further test the performance of our proposed model, table 2 continues to show the results with more testing data. The OCSVM model is trained with 200, 000 depression data by both TF-IDF and out proposed method. The remain 100,000 depression abstracts are used as the testing data. Experiment runs 20 times with same parameters. Each time we randomly select 3000 articles from each topic as the testing data. The numbers in table 2 are the average values of 20 experiments. From table 2, TF-IDF still

performs better than our method for non-medical outlier publications. But for medical outlier publications, our method is much better than TF-IDF based method.

Table 6-1average 10-fold micro-f1 score and std. to classify depression abstracts with other data

	Obesity	**Myocarditis**	**20Newsgroup**
TFIDF+SVM	0.421 (0.016)	0.502 (0.014)	0.862 (0.011)
Proposed Method	0.532 (0.015)	0.634 (0.012)	0.825 (0.013)

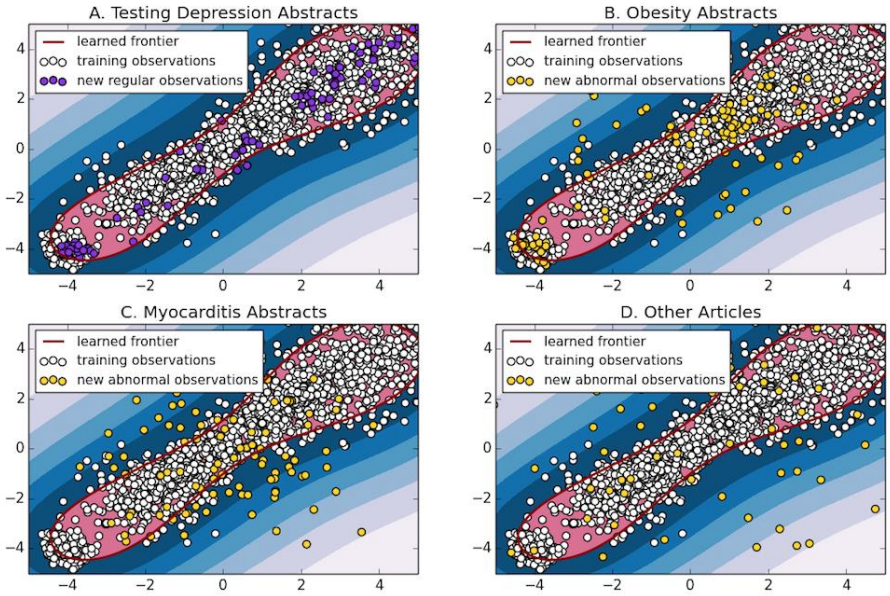


Figure 6-3. Use PCA to display depression, obesity, myocarditis and non-medical articles in a 2-D space.

Python Gensim and sciket-learn packages are used to implement the LDA and Word2Vec models. The hyerparameter αin LDA model is set to 0.1 and $passes$ is set to 20 to guarantee that

the convergence can be finished. Word2Vec will set the window size to 5. OCSVM uses a non-linear kernel Radial Basis Function (RBF).

Firstly, to display the distributions, we randomly select 100 depression abstracts, 100 obesity abstracts, 100 myocarditis abstracts, and 100 articles about other topics to visually demonstrate the performance of our proposed framework. 300,000 depression abstracts except the selected 100 abstracts are used as normal data to train the OCSVM model. Principle Component Analysis (PCA) model is used to map each literature to a 2-D space [53]. In figure 6.3, One-class SVM is trained by about 300,000 depression abstracts. White points represent training depression abstracts, and purple points are the testing depression abstracts. In figure A, B, C, yellow points are corresponding articles associated to different topics. The pink area is the learned model, where inside the area is considered as a normal literature, and outside the area outlier literatures. From the experiment, we find about 40% depression data not contained in the hypersphere. To evaluate the model's distinguishability, 100 depression literatures are used as testing data. The results in the figure 6.3A is that 17 of 100 abstracts are not recognized by the model. The accuracy is 83%. The figure 6.3B shows the results to detect the obesity abstracts. 35 of 100 literatures are found as outliers. Depression and obesity share many similar symptoms and medicines that means they have many same keywords contained both. It is a challenge to understand the real differences for the model. Figure 6.3C shows better performance that our method successful find 55 of 100 myocarditis abstracts. In figure 6.4D, non-medical articles are scattered to the space with less associations to the depression abstracts, and the model performs very high accurate that 80 of 100 are considered as outliers.

Table 6-2 results of our method and TF-IDF based method to evaluate the outlier detection capability on different data

	Depression	Obesity	Myocarditis	20News
TF-IDF	62.5%	28.7%	43.1%	83.2%
Proposed method	77.4%	36.1%	51.5%	81.7%

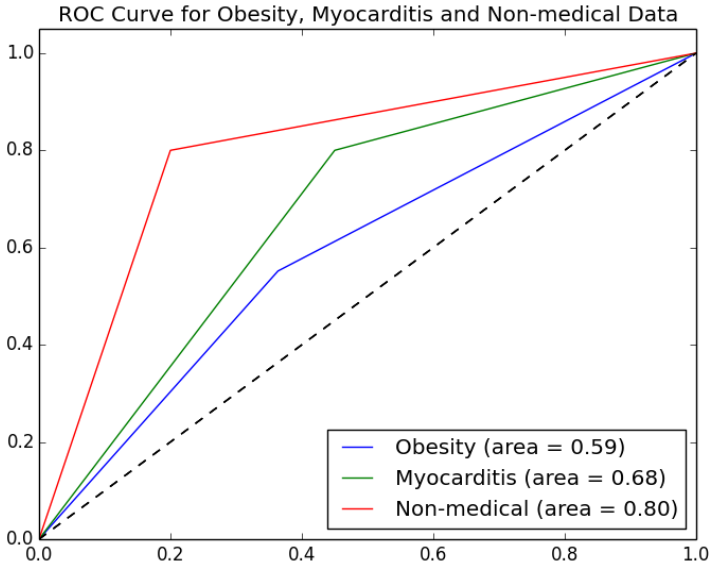


Figure 6-4 ROC curve of our method against different types of noise data

To further demonstrate the model performance, table 1 continues to show the results with more testing data. The OCSVM model is trained with 200, 000 depression data, and the remain data are used as testing data. We also have enough other types of testing data. Experiment runs 20 times with same parameters. Each time we randomly select 3000 articles from every topic as testing data. The numbers in table 1 are average values of 20 experiments, which is very closed

to the results discussed above. Therefore, it tells that our model has a very stable prediction capability.

In statistics, Area under Curve (AUC) is the area under the receiver operating characteristic (ROC) that is an important factor to illustrate the discrimination performance of system. The AUC is a probability between 0 to 1, and a larger probability indicates the model has better capability to detect the normal literatures and abnormal literatures. If the model is no better than random guessing, the true positive rate will increase linearly with the false positive rate and the area under the curve will be around 0.5. Moreover, outlier detection is a problem that is different from other traditional classification problems. The analyzing data is usually extremely imbalanced, in which a very small portion of data is the target to be distinguished. In figurer 4, the ROC curves show the detecting capabilities of our model to the 3 types of articles with different color lines. To against the medical articles blue line and green line, it performs as high as 0.6 of AUC value, and against the non-medical articles with red line, it reaches as high as 0.8.

7 A SEMI-SUPERVISED MACHINE LEARNING METHOD FOR IDENTIFYING TEXTUAL DEPRESSION SYMPTOMS FROM PUBLICATIONS

7.1 Introduction

The number of literatures are growing explosively, which has been far beyond a person's reading volume. Most of the time, keywords as the smallest meaningful units are widely used to prescreen literatures. However, a literature always contains multiple topics, while majority of time we are only interested in one or some of the topics. Therefore, we develop an approach to extract keywords from literatures regarding limited given topics. Many existing work on keyword extraction from a text has been conducted. Word frequency based methods are the earliest methods we used, and many later methods are extended from it [54]. TFIDF is the most used techniques that not only considers the word frequency, but also look at the distributions both in a single document and the full document set [6]. The advantages of the above approaches are easily to be implemented, and it conforms to the habit of human's writings that people prefer to repeat the contents they want to emphasize. However, the bad thing is that these methods always ignore the low-frequency words, such as the new-found biomarker, new medicines, and so on. Another type of keyword extraction method is to consider the lexical semantic information among words or sentences. Semantic relations between words can be obtained from a manually constructed thesaurus such as WordNet. Ye et al. used the frequency of all words belonging to the same WordNet concept set [55]. Probabilistic Latent Semantic Analysis (PLSA) as a topic model is used to build a dictionary, in which words are ranked according to the topical similarity to the topics [56]. Rose et al. fully consider dependencies among words, word co-occurrence in a document, and apply TextRank to identify the importance of words to the document [57]. Li et

al. use Word2Vec model to study the word vectors and represent document by a cluster of word vectors. Then the center of the cluster is selected as the keyword [58].

In this chapter, we propose a hybrid method for keyword extraction that rewards both word similarity, to identify the major topics regarding the given subject, and word frequency, to identify prominent words to related to the subject if necessary. The chapter is organized as follows. In section 7.2 we introduce how our method integrates Word2Vec with LDA. Section 7.3 shows the details of experiments and results. Section 7.4 is the conclusion and future work.

7.2 Proposed Framework

In the framework, LDA as a topic model discovers latent topics by learning all the literatures. Each latent topic is composed of a set of words with probabilities. If a literature includes a higher probability word of a topic, it is more likely that the literature is about the topic. Therefore, each literature can be described by a probability distribution over all topics. Word2Vec is deep neural network which projects words into a high-dimensional space in form of word vectors. Hence, the relationship between any two words can be easily measured by the cosine similarity. We studied depression symptoms from WebMD, which is an authoritative online publisher of news and information pertaining to human health and well-being. Based on its descriptions of depression symptoms, we manually choose 10 most important symptoms “fatigue”, “worthlessness”, “helplessness”, “hopelessness”, “insomnia”, “irritability”, “restlessness”, “anxious”, “sad”, and “suicidal” to form a depression symptom vector. A similarity between any word and symptom vector can be used to measure the probability a word to be depression symptom. Therefore, by calculating similarities of words in a topic, we can measure the degree of a topic about depression. Finally, the word frequency is counted by two

conditions: (1) only words in important topic will be counted; (2) only the literature has probability to the topic. Higher frequency words are more likely to be a depression symptom.

In details, given a set of literatures L, a word dictionary $D = \{d_1, d_2, \dots, d_n\}$ is constructed where d is word existed in L. By training L, LDA finds n latent topics $T = \{t_1, t_2, \dots, t_n\}$. Top 50 highest probability words are used to represent the topic. The n^{th} word in m^{th} topic can be denoted by d_n^m, and its probability is denoted with wp_n^m. By training the Word2Vec model, words in D are vectored to $V = \{v_1, v_2, \dots, v_n\}$. By learning the symptoms online, 10 words $D^{sy} = \{d_1^{sy}, d_2^{sy}, \dots, d_{10}^{sy}\}$ are used to represent depression symptoms, and from V, symptom words D^{sy} are mapped to $V^{sy} = \{v_1^{sy}, v_2^{sy}, \dots, v_{10}^{sy}\}$. Symptom vector can be calculated by (34).

$$SV = \frac{\sum_{n=1}^{10} v_n^{sy}}{10} \tag{34}$$

Next, words d_n^m in topics are vectored to v_n^m. We represent each topic in a vector form by aggregating the word vectors in it. The vector value of m^{th} topic is calculated by (35).

$$VT_m = \frac{\sum_{n=1}^{50} v_n^m}{50} \tag{35}$$

We measure the similarity by calculating the cosine values between any two vectors. θ is the angle between current topic and depression vector. The similarity is calculated by dot product and magnitude (36). A threshold is set to determine if a topic is a depression symptom topic. If the similarity larger than the threshold, $flag_t = 1$, else $flag_t = 0$.

$$similarity_m = \frac{VT \cdot SV}{\| VT \|_2 \| SV \|_2} \tag{36}$$

For any single literature, LDA represents it with a series of probabilities to indicate the relationship to all topics, where 0 indicates no relationship between the literature and the current

topic, and any value between 0 and 1 is the correlation of the literature to the current topic. So the correlation of m^{th} literature's n^{th} topic is denoted by tp_n^m.

Next, we measure how likely a word within a topic is a symptom in a literature.

- For the m^{th} literature, check the probability distribution of topics. We replace the probabilities with binary values. If any probability is 0, we set $flag_p = 0$ with 0; if it is not 0, $flag_p = 1$.
- We use logical AND operator with $flag_t$ and $flag_p$. If returned value is true, the topics will be kept.
- We calculate how important a word in kept topics to a literature by equation (37).

$$score_n^m = wp_n^m \times tp_n \tag{37}$$

  where n is the n is n^{th} kept topic and m is the m^{th} word in n^{th} topic.
- Iterate all literatures and aggregate scores of each word. All words are ranked by the scores. The score will be used to show the possibilities a word to be a symptom.

7.3 Experiment

In the experiment, we downloaded 300,000 depression related literatures from National Center for Biotechnology Information (NCBI) database. Python NLTK package is used to clean and prepare text data [59]. We also use Python Gensim package to implement LDA model and Word2Vec model [25]. Besides these, we also manually choose 10 depression symptoms from the WEBMED website which is an authoritative online platform fulfills the promise of health information. The 10 symptoms are "fatigue", "worthlessness", "helplessness", "hopelessness", "insomnia", "irritability", "restlessness", "anxious", "sad", and "suicidal". Based on these 10 symptoms, a symptom dictionary including 100 symptom words is constructed by adding their synonyms words.

In order to evaluate the performance of our model, we compare it with two other models TFIDF and original LDA-Word2Vec. The difference between the original LDA-Word2Vec and our model is that we use symptom vector to select topics. Our experiment requires each model returns 20, 50, 100 most important words based on the rank either by frequency or by score. Therefore, precision (P), recall (R), and F-measure are most used for this type of tasks. The corresponding equations are (38), (39), and (40).

$$P = \frac{extracted\ words\ \cap \text{human annotated keywords}}{\text{human annotated keywords}} \quad 38)$$

$$R = \frac{extracted\ words\ \cap \text{human annotated keywords}}{extracted\ words} \quad 39)$$

$$F - mearsure = \frac{2 \cdot P \cdot R}{P + R} \quad 40)$$

Table 7-1 comparing the precision of the three models to extract 20, 50, 100 keywords

	20 words	**50 words**	**100 words**
TF-IDF	0.04	0.08	0.15
Word2Vec + LDA	0.06	0.09	0.19
our model	0.09	0.21	0.35

Table 7-2 comparing the recall of the three models to extract 20, 50, 100 keywords

	20 words	**50 words**	**100 words**
TF-IDF	0.02	0.16	0.15
Word2Vec + LDA	0.03	0.18	0.19
our model	0.45	0.42	0.35

Table 7-3 comparing the F-score of the three models to extract 20, 50, 100 keywords

	20 words	50 words	100 words
TF-IDF	0.02	0.11	0.15
Word2Vec + LDA	0.04	0.12	0.19
our model	0.15	0.28	0.35

In Table 7.1 – Table 7.3, we find out that our model performs much better than the other two models in precision, recall and F-measure. With the increasing number of words extracted from literatures, our model has a clear upward trend to discover more symptoms, while the other models do not have a big rise.

The second experiment is to measure the average distances in terms of similarity from extracted word vector to the symptom vector. Similarly, we extract 20, 50, 100 words from the three models. The cosine similarity is ranged from -1 to 1, where negative value means opposite meanings, 0 means no relationship, and positive value means similar meanings. In Table 4, we can see TF-IDF model hardly find out symptom words so that its aggregated word vector has completely different directions of the symptom vector. Also, without the guidance of the symptom vector, the original LDA+Word2Vec model lacks concentration on any single topic. Compared to the above methods, our mod has much stronger learning capability to find out symptoms.

Table 7-4 average distances from extracted word vector to the symptom vector

	Avg. Distance		
	20 words	**50 words**	**100 words**
TFIDF	-0.2767	-0.1792	-0.0561
LDA + Word2Vec	0.1529	0.1388	0.0782
Our model	0.3886	0.3209	0.2633

8 EXTRACTING DEPRESSION SYMPTOM WORDS BY TEXTRANK AND WORD2VEC

8.1 Introduction

Keyword extraction is a task that automatically finds a small set of the words that best describe the contents of document. Keywords as the smallest unit to express the meanings in document, it plays a crucial role to improve the performances of text indexing, text abstracting, information retrieval, document classification and clustering, and so on. To identify keywords, basically it needs word tokenization, word stemming, removing stop words and so on.

Linguistic based methods take the position of words into account. It labels words as part-of-speech tagging, and according to the grammatical structures to decide the importance of words in document. Statistical methods count the occurrences of words in all documents. It treats high frequency words with more importance to the document. Term frequency–inverse document frequency (TF-IDF), as an example, is used to evaluate how important a word is to a document in corpus [6]. The importance increases proportionally to the occurrences of a word exists in document, meanwhile, the importance decreases by the frequency of the word in corpus. Another type of statistical model is probability-based method. Latent Dirichlet Allocation (LDA) investigates latent topics from corpus and represent each document with a probability distribution over the discovered topics [10]. Each topic is composed of words to describe the topic's contents where we can extract keywords based on the importance of words to the topics. TextRank as the reprehensive of the graph-based model, it uses the structure of the document and the known parts of speech of words to assign a score to words that is used to evaluate the importance of words to the document [60].

8.2 Proposed Framework

In our frame, we treat the keyword extraction problem by ranking the importance of words in document. The words with higher score are considered as prominent words to the document. A graph model is constructed as the keyword network where nodes are words as the basic meaning unit, and edges are the weights between every two words. By iterating words in any given document, it aims to optimize each word's weight to score the importance to document. Each word is learnt in the Word2vec and represented in a high-dimensional space. We measure the relativity between the current word and the target topic vector. Then the relativity is used in the iteration step to improve the output. Before we build the keyword graph model, we apply 5 steps to preprocess the training articles:

1. Select N training articles Using the NLTK package to divide N articles to sets of words, $c_1, c_2, \dots, c_N$, and build the training words set S_1. The stops words are removed from each set. M articles as testing data and construct test word set S_2.
2. Tag S_1 and S_2. Retain the important words such as nouns, verbs, adjectives.
3. Remove the duplicate words in S_1 and S_2. And generate the candidate word dictionary $D = \{w_1, w_2, \dots, w_n\} \in (S_1 \cup S_2)$.
4. Use model to train $M + N$ articles. And generate k-dimension word vectors $V = \{v_1, v_2, \dots v_m\}$.
5. Learn top 10 targeted topic related words $T = \{t_1, t_2, \dots, t_{10}\}$. We sum the word vectors as the topic vector TV in T if it exists in V. And builder a relativity list R where words in D to the topic vector.

   Use CBOW model and Skip-gram model to train the articles, we project words on D in to k-dimension word vector in V. The topic vector can be calculated by summing the values

at corresponding dimensions, and each dimension is divided by the number of topic keywords by equation (41).

$$TV = sum(T)/10 \tag{41}$$

where $T = t_1, t_2, \dots, t_{10}$ are selected top 10 related words to the topic.

The relativity between word vectors in V and topic vector TV can be calculated by (42)

$$sim(v_x, TV) = cos\theta = \frac{v_x \cdot TV}{\| v_x \| \cdot \| TV \|} \tag{42}$$

where v_x is the x^{th} vectors in Vector dictionary, and TV is the topic vector in (41). Based on (42), we save the similarities of all the words in dictionary D and the targeted topic TV in the list L_{sim} (43).

$$L_{sim}(v_x, TV) = [v_1 TV, v_2 TV, \dots, v_m TV] \tag{43}$$

Similarly, we also construct a similarity matrix which is used to initialize graph model by (44).

$$M_{sim}(x, x_j) = \begin{bmatrix} x_1 y_1 & \cdots & x_1 y_n \\ \vdots & \ddots & \vdots \\ x_n y_1 & \cdots & x_n y_n \end{bmatrix} \tag{44}$$

where each value in matrix is the similarity between two words, such as $x_i y_j$ represents the similarity between word x_i and word y_j.

After preprocessing the articles, TextRank is used to build the graph model. The main idea of TextRank is that the importance of a word node replies on the number of other word nodes connecting to it as well as their neighboring weights. The weight of a word node can be calculated by (45)

$$R(w_i) = \gamma \sum_{j:w_j \to w_i} \frac{e(w_j, w_i)}{O(w_j)} R(w_j) + (1-\gamma)\frac{1}{|V|} \tag{45}$$

where $R(w_i)$, is the weight of word w_i, $O(w_j)$ is the number of the word node connected to other nodes, $e(w_j, w_i)$ is $w_j \to w_i$'s weight on the edge, V is a set of word nodes, $\gamma \in [0,1]$ is the damping factor with value 0.85.

Normally, TextRank algorithm sets the weights of all word nodes with 1 as the default value. Then sequentially iterating words in the article to update its weight according to its neighbors. Meanwhile, the neighboring nodes also get updated and contributions is equally divided by the number connected nodes.

For example, 6 nodes $\{v, v_1, v_2, v_3, v_4, v_5\}$ graph model as shown in figure 8.1. The weight of each node is initialized with 1. The weight of v is equally contribute to the connected nodes $\{v_1, v_2, v_3, v_4, v_5\}$ and edge weigh is set to 0.2. For the nodes $v_1 - v_5$, they only linked to node v, so the weight is 1. Then the weights are updated by the same steps.

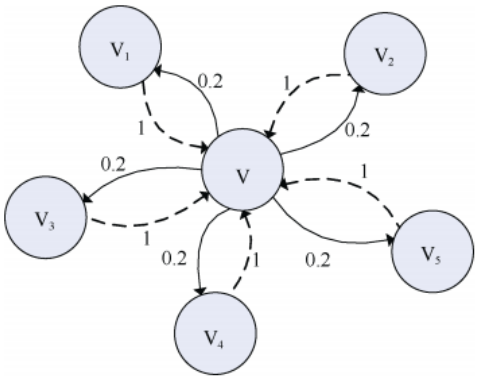


Figure 8-1 a graph model and its weight initialization

In this section, we will discuss how to optimize the initialization weights for nodes, how to decide the importance of word node, and how to improve the ranking methods of words to

extract keywords form documents. It is obvious that the initialization weight assigned with 1is not the best scenario. A better way is that we can employ the similarity matrix in (44) to set the starting edge weights.

Moreover, the similarity list in (43) is applied to the weight updating during the iteration. The updating process contains two components to affect the new weight. One part is to utilize original TextRank method to learn the importance of the word in the article. The other part is the similarity between the current word and the given topic by equation (46).

$$S(w_i) = (1-\gamma)\frac{1}{|V|} + \gamma(d_1 \sum_{j:w_j \to w_i} \frac{R(w_j)}{O(w_j)} + d_2 \sum_{j:w_i \to TV} \frac{Sim(w_i, TV)}{O(w_i)}) \quad (46)$$

where $\gamma \in [0,1]$ is the damping factor, in the experiment we set it as 0.85. $d_1 + d_2 = 1$, here we set is 0.5 and 0.5. $R(w_i)$, is the weight of word w_i, $O(w_j)$ is the number of the word node connected to other nodes, $Sim(w_i, TV)$ is the similarity of the current word and the topic vector, $O(w_i)$ is the number the current word's connections. V is the set of word nodes. When calculating the $Sim\ (w_i, TV)$, we can use the output of equation (3) $[v_1TV, v_2TV, \dots, v_mTV]$.

Therefore, we integrate Word2Vec with TextRank for keyword extraction. There are two times we apply Word2Vec outputs involved in TextRank to improve the performance of keyword extraction. We successfully find the global relationship among words by Word2Vec, and also we integrate it with the local relationship detected by TextRank to make the proposed algorithm much more powerful.

8.3 Experiments and Results

In the experiment, we downloaded 300,000 depression related literature summaries from the National Center for Biotechnology Information (NCBI) database. WEBMED is an organization that fulfills the promise of health information on the Internet. It provides credible

information, supportive communities, and in-depth reference material about health subjects that matter to you. Their sources for original and timely health information as well as material all are from well-known content providers. Based on its description of depression symptoms. So based on its description of depression symptoms as shown in figure 8.2, we extract 10 most important symptoms keywords manually based on human's understanding. They are "fatigue", "worthlessness", "helplessness", "hopelessness", "insomnia", "irritability", "restlessness", "anxious", "sad", and "suicidal".

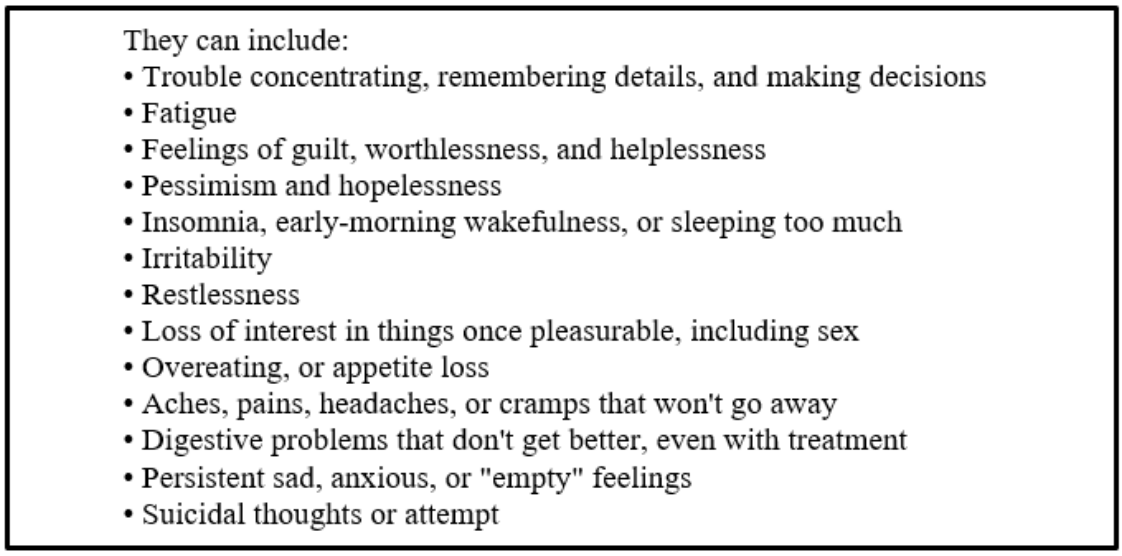


Figure 8-2 symptoms description from WEBMED

Python NLTK package is used to tokenize the words, process word stemming/lemmatization, and remove the stops words for each literature [59]. Python Gensim package is to train the Word2Vec model from the 300,000 articles [25]. TextRank algorithm is improved and developed used based on David's work on Github (https://github.com/davidadamojr/TextRank). A keyword dictionary is constructed by expanding the synonyms of the given 10 keywords, and human annotated word, totally 100 words.

To design the experiment, we use 100,000 literature as the training data. Then we randomly select 120 literatures from the pool as the training data. Each time we let each method to extract 3, 5, 7, 10 keywords from each literature, and check if the keyword is listed in the keyword dictionary we built above. The experiment will be repeated five times, and the precision, recall and F-measurement are averaged values. Basically, precision, recall and F-measurement are calculated by equation (47), (48), and (49) respectively. To further evaluate the performance, we compare with TF-IDF, original TextRank, and Word2Vec models on the dataset.

$$P = \frac{extracted\ words\ \cap \text{human annotated keywords}}{\text{human annotated keywords}} \quad 47)$$

$$R = \frac{extracted\ words\ \cap \text{human annotated keywords}}{extracted\ words} \quad 48)$$

$$F - mearsure = \frac{2 \cdot P \cdot R}{P + R} \quad 49)$$

Table 8-1 comparing the precision of the four models to extract 3, 5, 7, 10keywords

	TF-IDF	TextRank	Word2Vec	Word2Vec+TextRank
3	0.325	0.334	0.273	0.285
5	0.278	0.346	0.289	0.318
7	0.256	0.348	0.314	0.359
10	0.245	0.341	0.335	0.388

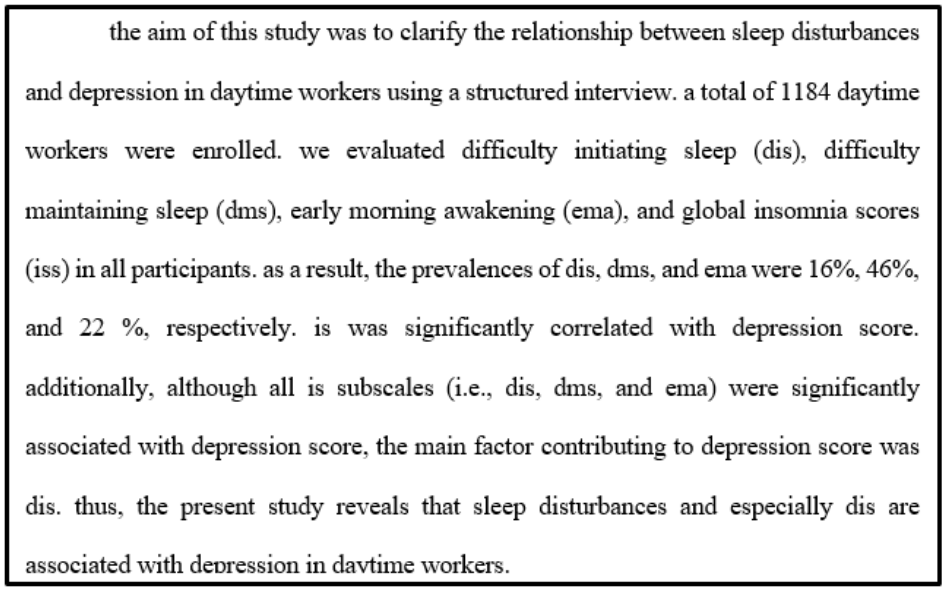
Table 8-2 comparing the recall of the four models to extract 3, 5, 7, 10keywords

	TF-IDF	TextRank	Word2Vec	Word2Vec+TextRank
3	0.336	0.329	0.281	0.289
5	0.277	0.341	0.287	0.312
7	0.249	0.351	0.321	0.358
10	0.228	0.338	0.333	0.392

Table 8-3 comparing the F-measurement of the four models to extract 3, 5, 7, 10keywords

	TF-IDF	TextRank	Word2Vec	Word2Vec+TextRank
3	0.330	0.331	0.277	0.287
5	0.277	0.343	0.288	0.315
7	0.252	0.349	0.317	0.358
10	0.236	0.339	0.334	0.390

From Table 8.1- Table 8.3, the comparing results show us the precision, recall, and F-measurement among the 4 methods. The performance of TF-IDF is getting worse when the number of keywords extracted from literatures increases. Also, its overall performance is the worst. TextRank method is not affected a lot by the number of keywords, while Word2Vec model raises its accuracy with increasing the extracted keyword amounts. By integrating TextRank and Word2Vec, our method has better performance when number of keywords are increased, and the overall performance is the best among the 4 methods.



the aim of this study was to clarify the relationship between sleep disturbances and depression in daytime workers using a structured interview. a total of 1184 daytime workers were enrolled. we evaluated difficulty initiating sleep (dis), difficulty maintaining sleep (dms), early morning awakening (ema), and global insomnia scores (iss) in all participants. as a result, the prevalences of dis, dms, and ema were 16%, 46%, and 22 %, respectively. is was significantly correlated with depression score. additionally, although all is subscales (i.e., dis, dms, and ema) were significantly associated with depression score, the main factor contributing to depression score was dis. thus, the present study reveals that sleep disturbances and especially dis are associated with depression in daytime workers.

Figure 8-3 An example of literature sample

Table 8-4 symptoms found by each model from top 5 keywords

	Top 5 Keywords	# found keywords
TF-IDF	relationship, depression, significantly, correlated, **sleep**	1
TextRank	score, **sleep**, difficulty, **dis**, study	2
Word2Vec	difficulty, correlated, depression, **disturbances**, **sleep**	2
Word2Vec + TextRank	**dis**, **insomnia**, **sleep**, **disturbances**, difficulty	4

Another way that we can evaluate the performance of the new method is replying on the judgement by human. We randomly select one literature from the training data as shown in figure 8.3. And in table 8.4, we list the top 5 words returned from different methods. Based on my knowledge, I highlighted the words that I consider it as a symptom or related to symptom. Our method successfully find out 4 words dis, insomnia, sleep, disturbances from the literature, while other methods just get 1 or 2 keywords.

REFERENCE

[1] F. Zhu and etc., "Biomedical text mining and its applications in cancer research," *Journal of Biomedical Informatics,* vol. 46, no. 2, pp. 200-211, 2013.

[2] G. Salton and M. J. McGill, Introduction to Modern Information Retrieval, New York: McGraw-Hill, Inc., 1986.

[3] T. o. Indexing, A theory of indexing, vol. 18, SIAM, 1975.

[4] G. Salton and M. J. McGill, Introduction to modern information retrieval, New York, 1983.

[5] Z. S. Harris, "Distributional Structure," vol. 10, pp. 146-162, 1954.

[6] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," Information Processing and Management: an International Journal , no. 24, p. 5, 1988.

[7] T. Mikolov, I. Sutskever, K. Chen, G. Corrado and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," in Advances in neural information processing systems, 2013.

[8] TensorFlow, "Vector Representations of Words," [Online]. Available: https://www.tensorflow.org/tutorials/word2vec.

[9] L. v. d. Maaten, "Accelerating t-SNE using Tree-Based Algorithms," Journal of Machine Learning Research, vol. 15, no. 1, pp. 3221-3245, 2014.

[10] D. M. Blei, A. Y. Ng and M. I. Jordan, "Latent dirichlet allocation," The Journal of Machine Learning Research, vol. 3, pp. 993-1022, 2003.

[11] E. Chen, "Introduction to Latent Dirichlet Allocation," 22 1 2011. [Online]. Available: http://blog.echen.me/2011/08/22/introduction-to-latent-dirichlet-allocation/.

[12] J. Suykens and J. Vandewalle, "Least Squares Support Vector Machine Classifiers," Neural Processing Letters, vol. 9, no. 3, pp. 293-300, 1999.

[13] T. Joachims, "Text categorization with Support Vector Machines: Learning with many relevant features," European Conference on Machine Learning, vol. 1398, pp. 137-142, 1998.

[14] B. Scholkopf and A. J. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, Cambridge: MIT Press, 2001.

[15] C. Campbell and K. P. Bennett, "A linear programming approach to novelty detection," in NIPS'00 Proceedings of the 13th International Conference on Neural Information Processing Systems, Dever, 2000.

[16] F. Desobry, M. Davy and C. Doncarli, "An online kernel change detection algorithm," IEEE Transactions on Signal Processing, vol. 53, no. 8, pp. 2961-2974, 2005.

[17] A. Ganapathiraju, "Support vector machines for speech recognition," Mississippi State University, Mississippi, 2002.

[18] L. M. Manevitz and M. Yousef, "One-class svms for document classification," The Journal of Machine Learning Research, vol. 2, pp. 139-154, 2002.

[19] Y. Bengio, R. Ducharme, P. Vincent and C. Janvin, "A neural probabilistic language model," The Journal of Machine Learning Research, vol. 3, pp. 1137-1155, 2003.

[20] J. Pennington, R. Socher and C. D. Manning, "Glove: Global Vectors for Word Representation," Conference on Empirical Methods in Natural Language Processing, vol. 14, pp. 1532-1543, 2014.

[21] Google, "word2vec," 2013. [Online]. Available: https://code.google.com/archive/p/word2vec/.

[22] Q. Le and T. Mikolov, "Distributed Representations of Sentences and Documents," Proceedings of The 31st International Conference on Machine Learning, vol. 32, pp. 1188-1196, 2014.

[23] Y. Z. Long Ma, "Using Word2Vec to process big text data," in BIG DATA '15 Proceedings of the 2015 IEEE International Conference on Big Data (Big Data), San Diego, 2015.

[24] K. Lang, "20 Newsgroups," 2008. [Online]. Available: http://qwone.com/~jason/20Newsgroups/.

[25] R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in Proceedings of the **LREC** 2010 Workshop on New Challenges for NLP Frameworks, Malta, 2010.

[26] R. Garreta and G. Moncecchi, Learning scikit-learn: Machine Learning in Python, Packt Publishing ©2013, 2013.

[27] P. S. Dhillon, D. P. Foster and L. H. Ungar, "Eigenwords: spectral word

embeddings," The Journal of Machine Learning Research, vol. 6, no. 1, pp. 3035-3078, 2015.

[28] R. Lebret and R. Collobert, "Word emdeddings through hellinger PCA," arXiv preprint arXiv:1312.5542, 2013.

[29] S. Dennis, T. Landauer, W. Kintsch and J. Quesada, "Introduction to latent semantic analysis," in 25th Annual Meeting of the Cognitive Science Society, Boston, 2003.

[30] L. Vilnis and A. McCallum, "Word representations via gaussian embedding," arXiv preprint arXiv:1412.6623, 2014.

[31] Q. V. Le and T. Mikolov, "Distributed Representations of Sentences and Documents," International Conference on Machine Learning, vol. 14, pp. 1188-1196, 2014.

[32] D. Liu, W. Xu and J. Hu, "A feature-enhanced smoothing method for LDA model applied to text classification," in Natural Language Processing and Knowledge Engineering, 2009.

[33] D. Zhao, J. He and J. Liu, "An improved LDA algorithm for text classification," in Information Science, Electronics and Electrical Engineering (ISEEE), 2014.

[34] D. Q. Nguyen, R. Billingsley, L. Du and M. Johnson, "Improving topic models with latent feature word representations," Transactions of the Association for Computational Linguistics, pp. 299-313, 2015.

[35] D. Ramage, D. Hall, R. Nallapati and C. D. Manning, "Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora," in EMNLP '09

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, 2009.

[36] Y. Wang and Q. Guo, "Multi-LDA hybrid topic model with boosting strategy and its application in text classification," in InControl Conference (CCC), 2014 33rd Chinese, IEEE, 2014.

[37] L. Niu, X. Dai and J. Zhang, "Topic2Vec: Learning distributed representations of topics," arXiv:1506.08422, 2015.

[38] C. MOODY, "A Word is Worth a Thousand Vectors," 30 Mar 2015. [Online]. Available: http://multithreaded.stitchfix.com/blog/2015/03/11/word-is-worth-a-thousand-vectors/.

[39] F. e. a. Pedregosa, "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, pp. 2825-2830, 2011.

[40] E. M. Knorr and R. T. Ng, "Algorithms for mining distancebased outliers in large datasets," in Proceedings of the International Conference on Very Large Data Bases, 1998.

[41] S. Ramaswamy, R. Rastogi and K. Shim, " Efficient algorithms for mining outliers from large data sets," in SIGMOD '00 Proceedings of the 2000 ACM SIGMOD international conference on Management of data, Dallas, 2000.

[42] M. M. Breunig, H.-P. Kriegel, R. T. Ng and J. Sander, " LOF: identifying density-based local outliers," in SIGMOD '00 Proceedings of the 2000 ACM SIGMOD international conference on Management of data, Dallas, 2000.

[43] C. Aggarwal and P. S. Yu, "Outlier detection for high dimensional data," in

SIGMOD '01 Proceedings of the 2001 ACM SIGMOD international conference on Management of data, Barbara, 2001.

[44] F. Keller, E. Muller and K. Bohm, "HiCS: High Contrast Subspaces for Density-Based Outlier Ranking," in Data Engineering (ICDE), 2012 IEEE 28th International Conference on, 2012.

[45] H.-P. Kriegel, P. Kröger and E. Schubert, "Outlier detection in arbitrarily oriented subspaces," in Data Mining (ICDM), 2012 IEEE 12th International Conference on., 2012.

[46] A. Lazarevi and V. Kumar, "Feature bagging for outlier detection," in Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining, 2005.

[47] M. E. Muller, I. Assent, P. Iglesias, Y. Mulle and K. Bohm, "Outlier ranking via subspace analysis in multiple views of the data," in Data Mining (ICDM), 2012 IEEE 12th International Conference on, 2012.

[48] C. Ding, T. Li and W. Peng, "NMF and PLSI: equivalence and a hybrid algorithm," in Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, 641-642, 2006.

[49] E. Gaussier and C. Goutte, " Relation between PLSA and NMF and implications," in Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval, 2005.

[50] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," Nature,401(6755), pp. 788-791, 1999.

[51] W. Xu, X. Liu and Y. Gong, "Document clustering based on non-negative matrix factorization," in Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval, 2003.

[52] E. J. Jackson, A user's guide to principal components, vol. 587, John Wiley & Sons, 2005.

[53] P. Lawrence, S. Brin, R. Motwani and T. Winograd, "The PageRank citation ranking: Bringing order to the web," Stanford InfoLab, 1999.

[54] R. Mihalcea and P. Tarau, "TextRank: Bringing order into texts," Association for Computational Linguistics, 2004.

[55] Wikipedia, "Named-entity recognition," [Online]. Available: https://en.wikipedia.org/wiki/Named-entity_recognition.

[56] C. Sutton and A. McCallum, "n introduction to conditional random fields," Foundations and Trends® in Machine Learning, pp. 267-373, 2012.

[57] K. S. Jones, "A statistical interpretation of term specificity and its application in retrieval," Journal of Documentation, vol. 28, no. 1, pp. 11-21, 1972.

[58] M. I. Yutaka Matsuo, "Keyword extraction from a single document using word co-occurrence statistical information," International Journal on Artificial Intelligence Tools, pp. 157-169, 2004.

[59] R. B. W. S. Christian Wartena, "Keyword extraction using word co-occurrence," in In Proceedings of the 2010 Workshops on Database and Expert Systems Applications, Washington, 2010.