Automatic Keyword Tagging With Machine Learning Approach

Xingyu Shen

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Applied Science (Electrical and Computer Engineering) at

Concordia University

Montréal, Québec, Canada

December 2021

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: Xingyu Shen

Entitled: Automatic Keyword Tagging with Machine Learning Approach

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Electrical and Computer Engineering)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

______________________ Chair
Dr. H. Rivaz

______________________ Examiner
Dr. Y. Zeng (CIISE)

______________________ Examiner
Dr. H. Rivaz

______________________ Thesis Supervisor(s)
Dr. W.-P. Zhu

______________________ Thesis Supervisor(s)
Dr. I. Moazzen (University of Victoria)

Approved by ______________________
Dr. Yousef Shayan, Chair
Department of Electrical and Computer Engineering

____
Dr. Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Date ______________________

Abstract

Automatic keyword tagging with machine learning approach

Xingyu Shen

With the explosive growth of information in the Internet age, the use of keywords has become the main tool for users to search for content of interest in a large amount of information. Keyword tagging can be divided into in-text keyword extraction and out-of-text keyword assignment. Keyword extraction is an important area in natural language processing (NLP), but the technology still has a lot of immaturity. Traditional keyword extraction methods are difficult to meet the commonly desired three characteristics simultaneously, i.e., understandability, relevance and good coverage, and thus even now in Web 2.0 many tags of web pages are still tagged manually.

In this thesis, we propose a novel unsupervised keyword extraction method that integrates word embedding (GloVe and fastText) with clustering (Affinity Propagation, Mean Shift and K-means). We use semantic relevance to cluster the terms in a document, and extract the noun phrase nearest to the center of the cluster as the keyword. This method ensures that the extracted keywords satisfy the above three characteristics at the same time. Our computer simulation results based on Hulth-2003, Krapivin-2009 and Nguyen-2007 datasets show that the proposed method outperforms all other existing methods in terms of common evaluation metrics such as Precision, Recall and F1-Score.

This thesis also proposes a CNN-BiLSTM model for keyword assignment, which uses word embedding method and attention mechanism. This model overcomes the limitation of single CNN model in ignoring the semantic and syntactic information of the input context, and effectively avoids the problem of gradient disappearance or gradient diffusion in traditional RNNs. Moreover, the use of attention mechanism can highlight important information and avoid the influence of invalid information on text sentiment and classification. Experimental results on three datasets, i.e., 20 Newsgroups, IMDB, SemEval 2018 task-1, show that the proposed keyword assignment method outperforms previous methods in terms of common evaluation metrics such as F1-Score, Accuracy and AUC, indicating the wide applicability of our method to various datasets.

Acknowledgments

I would like to express my heartfelt gratitude to my supervisor, Prof. Wei-Ping Zhu, for the academic research guidance he provided throughout my pursuit of the master's degree. Whenever I encountered difficulties in my research, his endless patience and invaluable advice helped me get through them. I am grateful to him not only for the theoretical knowledge I learned from him, but also for the impact he had on my entire life.

I would also like to thank my co-supervisor, Dr. Iman Moazzen, for his technical guidance. Without his constructive advice, I would not be able to complete this research.

In addition, I would like to thank my roommate Jianyu Tang and my good friend Runze Wang, who have helped me a lot not only in my academic research but also in my daily life.

Finally, I am deeply grateful to my parents. They always believe in me and support me no matter what happens. I hope they will always be healthy and happy.

Contents

List of Figures

List of Tables

List of Abbreviations

AP Affinity Propagation

BiLSTM Bidirectional Long Short-Term Memory

CBOW The Continuous Bag of Words

CNN Convolutional Neural Network

CRF Conditional Random Fields

DBN Deep Belief Network

DNN Deep Neural Network

GloVe Global Vectors for Word Representation

GRU Gate Recurrent Unit

LSTM Long Short-Term Memory

MS Mean Shift

NLP Natural Language Processing

NLTK Natural Language Toolkit

OOV Out-of-Vocabulary

POS Part of Speech

RAKE Rapid Automatic Keyword Extraction

SVM Support Vector Machines

TF-IDF Term Frequency–Inverse Document Frequency

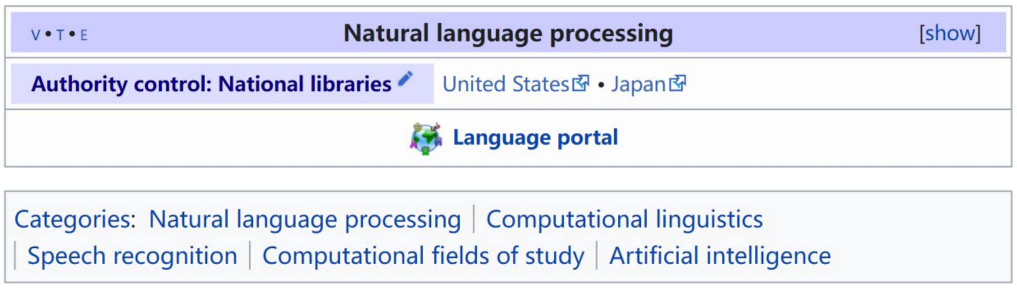
Chapter 1

Introduction

1.1 General information

The keywords of a document are usually a set of words or phrases, which briefly describe the main topic of the document. Keywords are widely used in news stories, academic papers, digital media and other documents to facilitate people to efficiently manage and retrieve the desired information. With the explosive growth of information in the Internet age, the use of keywords has become a main tool for users to search for content of interest in a huge amount of information, and for this reason keyword-based search engines have been developed by many companies, such as Google and Baidu.

Since the advent of the World Wide Web, tagging or links across the HTML texts has played a crucial role in the success of the Internet. In the end of 1990s, Google first adopted in-text content hyperlinks, namely, clicking on a word or a group of words in the text directly links to a web page that further explains the content behind these words. In this sense, these words in the hyperlink serve as the keywords of the new webpage. However, it should be noted that the new page in the form of HTML text may contain other keywords all together describing the content of this page. This paragon of cross-linking different webpages belongs to the so-called Web 2.0 technology, which provides the users with not only the searched result based on the input keyword but also more information and material relevant to the original search in a more adaptive and responsive way, largely facilitating users to search and navigate on internet. Although Web 2.0 is better than Web 1.0 which only consists of static pages without crosslinking/tagging, its linking and tagging activities are still manual due to the immature of technology in the field of natural language processing (NLP). For example, Wikipedia is considered to be among the most successful Web 2.0 services, but in the categories at the bottom of each Wikipedia article, the tags are added manually. Figure 1 shows a Wikipedia website explaining "natural language processing". In the categories, however, natural language processing is listed together with other tags including Computational linguistics, Speech recognition, Computational fields of study, and Artificial intelligence, which are the keywords manually added by the author of the page.



V • T • E	Natural language processing	[show]
Authority control: National libraries	United States • Japan	
Language portal		

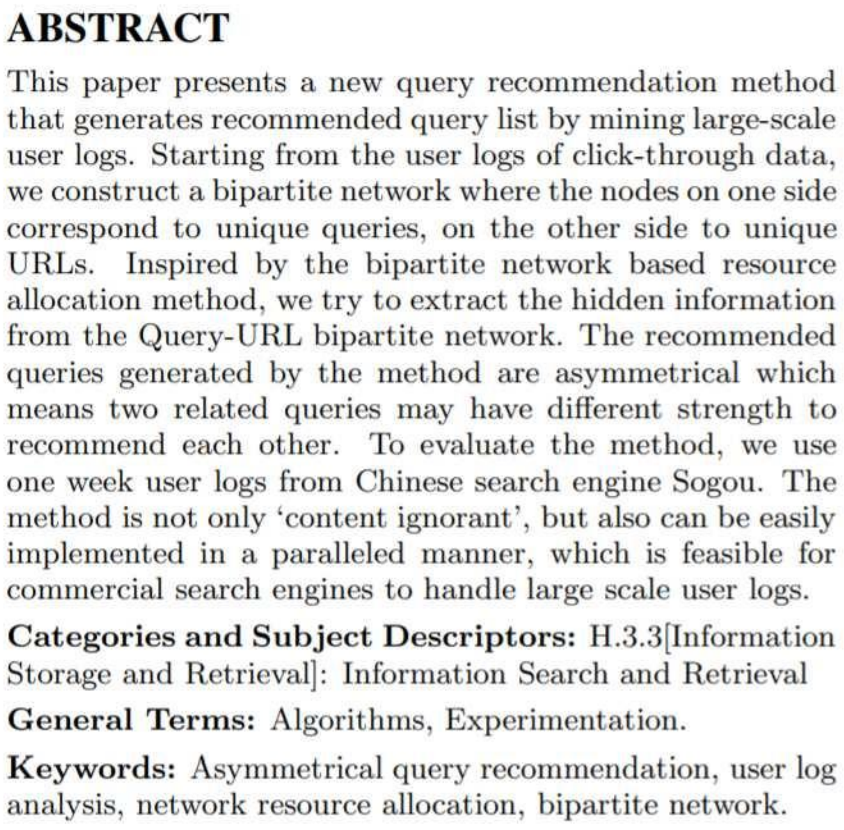
Categories: Natural language processing | Computational linguistics | Speech recognition | Computational fields of study | Artificial intelligence

Figure 1: The tags of Nature language processing in Wikipedia.

Fig. 2 shows another application. The abstract is for a conference paper published in an academic conference sponsored by the American Computer Society (ACM). At the end of the abstract, there are three kinds of keywords: "Categories and Subject Descriptions", "General Terms" and "Keywords". The first kind defines a four-level classification tree that is provided by the ACM to indicate the subject category of the paper. In H.3.3 [Information Storage and Retrieval] Information Storage and Retrieval, as it appears above, first the author needs to choose one topic from A. General Literature, B. Hardware, C. Computer Systems Organization, D. Software, E. Data, F. Theory of Computation, G. Mathematics of Computing, H. Information Systems, I. Computing Methodologies Systems Organization, J. Computer Applications and K. Computing Milieux that best fits the content of the paper. In the second step, after this author has chosen H, there are many sub-topics in Information Systems, including H.0 General, H.1 Models and Principles, H.2 Database Management, H.3 Information Storage and Retrieval, H.4 Information System Applications, H.5, Information Interfaces and

Presentation and H.m Miscellaneous. The author needs to select one of these subtopics that is most relevant to the content of the paper. The last step is author needs to customize a topic descriptor, usually the same as the description of the second step. This is how H.3.3 [Information Storage and Retrieval] Information Storage and Retrieval appears in the Categories and Subject Descriptors.

The second description "General Terms", is also provided by ACM comprising a list of 16 words: Algorithms, Design Documentation, Economics, Experimentation, Human, Factors, Languages, Legal, Aspects, Management, Measurement, Performance, Reliability, Security, Standardization, Theory and Verification. For the "Subject Description" and "General Terms", the authors have to select from the provided list. For the third description, which is the true "Keywords" of the paper, the authors can freely assign it or define according to the material of the paper. As example, Figure 3 shows the tags on the web page of the famous Globe NEWS, which has different types of tags like World, Canada, Local, Politics, Money, Health, Entertainment, lifestyle, etc. For the first piece of news, it is assigned a tag “LIFESTYLE” because the meaning of its content is closer to lifestyle than others. The second piece of news belongs to "ECONOMY" while the third one is given in "FEATURES".


ABSTRACT

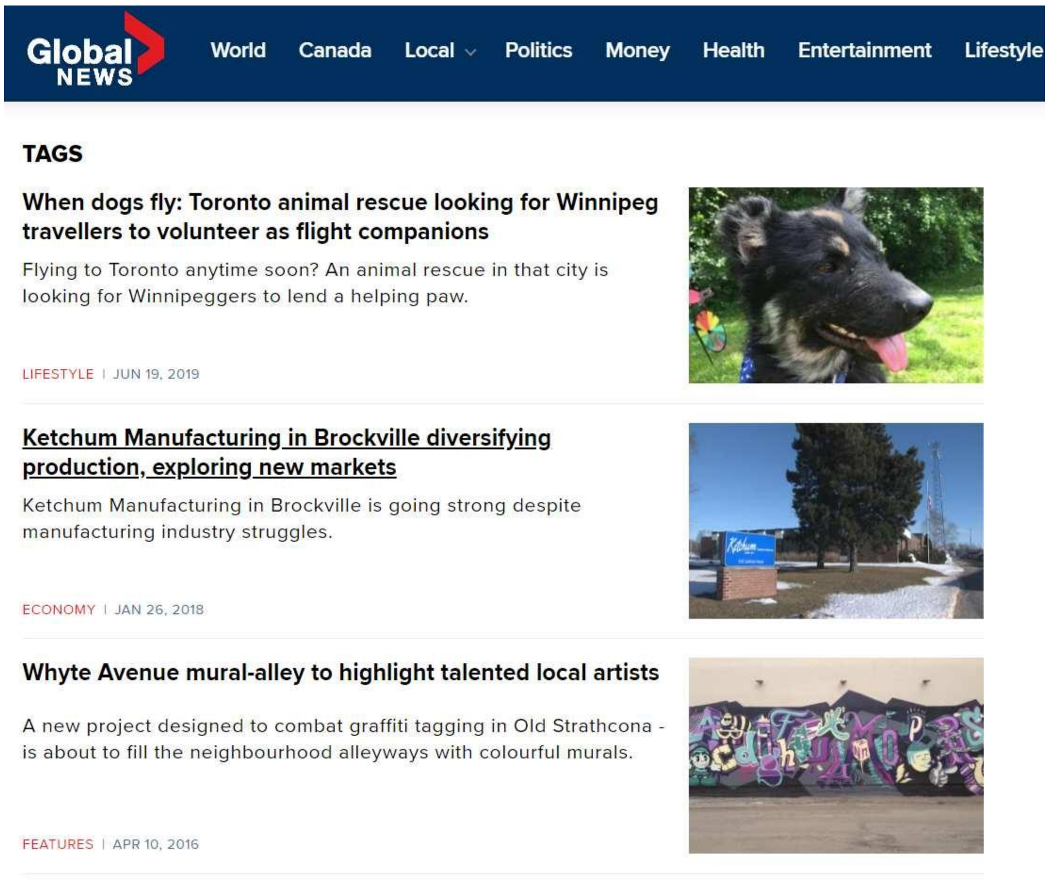
This paper presents a new query recommendation method that generates recommended query list by mining large-scale user logs. Starting from the user logs of click-through data, we construct a bipartite network where the nodes on one side correspond to unique queries, on the other side to unique URLs. Inspired by the bipartite network based resource allocation method, we try to extract the hidden information from the Query-URL bipartite network. The recommended queries generated by the method are asymmetrical which means two related queries may have different strength to recommend each other. To evaluate the method, we use one week user logs from Chinese search engine Sogou. The method is not only 'content ignorant', but also can be easily implemented in a paralleled manner, which is feasible for commercial search engines to handle large scale user logs.

Categories and Subject Descriptors: H.3.3[Information Storage and Retrieval]: Information Search and Retrieval

General Terms: Algorithms, Experimentation.

Keywords: Asymmetrical query recommendation, user log analysis, network resource allocation, bipartite network.

Figure 2: Example of keywords for an ACM conference paper [1].



TAGS

When dogs fly: Toronto animal rescue looking for Winnipeg travellers to volunteer as flight companions

Flying to Toronto anytime soon? An animal rescue in that city is looking for Winnipeggers to lend a helping paw.

LIFESTYLE | JUN 19, 2019

Ketchum Manufacturing in Brockville diversifying production, exploring new markets

Ketchum Manufacturing in Brockville is going strong despite manufacturing industry struggles.

ECONOMY | JAN 26, 2018

Whyte Avenue mural-alley to highlight talented local artists

A new project designed to combat graffiti tagging in Old Strathcona - is about to fill the neighbourhood alleyways with colourful murals.

FEATURES | APR 10, 2016

Figure 3: Tagging of the different pieces of news in Global NEWS website.

With the explosive growth of information and technology resources, a large number of documents and related information are generated at all times. Manual tagging of such a huge amount of information has become unrealistic. It is necessary to leverage computers to automatically extract and assign keywords for input documents. Nowadays, automatic keyword tagging has become a hot research topic in natural language processing and information retrieval. At present, keyword tagging has been widely used in search engines, news services, electronic libraries and other fields. It plays an important role in tasks such as full-text retrieval, text classification, information filtering, and document summarization.

Generally speaking, there are two kinds of keyword tagging approaches: in-text tagging and out-of-text keyword tagging. In-text keyword tagging directly marks keywords in the text. Just like in Figure 2, under the third description, the "keywords" are four phrases selected by the author from the text that can best represent the content of the paper. Out-of-text tags refer to the metadata about a document that is not necessarily provided in the body of the text. For example, in Figure 3, any news article that goes through the review will be given a tag from a set of tags defined by the website, based on the content of the news. This is one of the applications of keyword assignment.

This thesis investigates methods for automatic keyword tagging, both in-text and out-of-text, and explores some of the factors that play a crucial role in the algorithm, including machine learning models, deep learning models, dataset strategies, and evaluation metrics. The next section briefly introduces the state-of-the-art methods for in-text and out-of-text keyword tagging.

1.2 The state-of-the-art methods for automatic keyword tagging

The review of in-text keyword tagging comprises three main approaches: statistics-based approach, clustering-based approach and graph -based approach. As for the out-of-text keyword tagging, we focus on machine learning and deep learning methods which are most commonly used nowadays.

1.2.1 In-text keyword tagging

As mentioned earlier, in-text keywords are directly extracted from the text provided. The tagging idea is mainly based on the observations and understandings of some keyphrases. Statistics-based approaches use deterministic mathematical functions to identify the phrases that have unusual frequencies of appearing. Different algorithms interpret the notion of unusual frequency in different ways. For example, TF-IDF (Term Frequency–Inverse Document Frequency) [2] is a statistical method to assess the importance of a word for a document or a set of the documents in a corpus. The importance of a word increases positively with its number of occurrences in a document, but decreases inversely with its frequency in a corpus. Similarly, Paukkeri [3] assumes that an article is first selected in a corpus and key phrases are selected according to the ratio of the rank value of each phrase in a document to the corresponding value in the reference corpus, where the rank value is calculated as the relative N-gram frequency of the phrase (N-gram frequency is defined that in a sequence of N consecutive words, the frequency of Nth word is predicted from the previous N-1 words). When the ratio of the N-gram frequency of a phrase in a document to its N-gram frequency in the whole corpus is larger, it means that the phrase is more likely to be considered as a keyword in the document. RAKE (Rapid Automatic Keyword Extraction) [4] first removes stopwords and punctuations in the text and assigns a score to each word. In a general sense, stopwords are roughly divided into two categories. One category includes function words contained in human language, which are extremely common and have no real meaning compared to other words, such as 'he', 'is', 'ours', 'what', 'in', etc. or compound nouns like 'The Who', 'or 'Take The'. Another category of words includes lexical words, such as 'like', which are too widely used to help narrow the search and also reduce the efficiency of the search. If the word is co-occurring (words appear in a matrix of a given length) with another word, the score is added by one. Finally, the score to frequency ratio of each word is obtained.

A clustering-based approach groups candidate phrases by similarity, and then selects the noun phrase closest to the cluster center from each cluster as the keyword. Zhang [5] first used the Word2Vec word embedding method to cluster semantically related phrases by applying the results of word embedding (word embedding means mapping individual words to dense and continuous vectors in which the semantic relations between words are represented) to two clustering algorithms, namely hierarchical and K-Means, after calculating co-occurrence frequencies. The extracted key phrases are the phrases that are close to the centroid of each cluster.

The graph-based approach rates phrases by a graph-based ranking algorithm. First it represents the document as a graph with each phrase in the document corresponding to each vertex. If a co-occurrence relationship is found between two phrases in a window of a set size, the two vertices corresponding to these two phrases are connected. The importance of these vertices is analyzed recursively by counting the number of adjacent vertices connected to each vertex and by the weights of the connected line segments. Linking algorithms were originally used to determine the quality of a web page. If a web page has many in-links, which are links on many other pages sending traffic to this page, it means that this page is considered as a high-quality page. The linking algorithm applied to graph-based keyword extraction methods is to consider each word as a web page, and the co-occurrence relationship between words is considered as a connection between them. If one word has a high frequency of co-occurrence with other words, then the it is likely to be considered a keyword.

The most famous graph-based algorithm is TextRank introduced by Mihalcea and Tarau [6], which represents documents as undirected and unweighted graphs and only considers word occurrence and co-occurrence frequency features. Wan and Xiao [7] first obtained ten highest scoring feature words by using the TF-IDF method for each piece document in the document set, and then generated a new set of all the feature words. After obtaining the eigenvalue vector of each document from the new set, the cosine similarity between the documents is calculated and the similar documents are grouped to form a new small set. A global graph is created in every small set of documents and the candidate words in the documents are scored using the TextRank method, and finally, for each document, the word with the highest score is considered as the keyword. Wang et al [8] used Synset in WordNet [9] to obtain semantic relationships between words in the text. WordNet is an English dictionary containing semantic information built at Princeton University. WordNet groups words according to their meaning, and each group of words with the same meaning is called a Synset (a collection of synonyms) [9]. Wang et al [8] used Synset in a large document set to obtain the number of synonym pairs between each document. He grouped the documents with more synonym pairs into a small document set. Thus, a large document set is divided into many small document sets. Then the keywords are extracted from these smaller document sets using a graph-based approach.

1.2.2 Out-of-text keyword tagging

Based on the learning methods employed, keyword assignment can be treated as a multiclass classification problem. Some common machine learning methods can be used to train the classifier, such as Naïve Bayes [10], Support Vector Machine (SVM) [11], Conditional Random Fields (CRF) [12] and neural networks.

Jo and Lee [13] proposed a deep learning approach for keyword assignment in which a deep belief network (DBN), which is a neural network in which the elements within any layer are not connected to each other, connected to a logistic regression layer to learn the classifier. This method does not require any manually selected features. It uses a greedy hierarchical unsupervised learning method, such as adding an output layer after each hidden layer to automatically learn the features of a layer, so that the locally optimal parameters of each layer can be obtained. After training, all pre-trained layers are simply concatenated for fine-tuning, where the locally optimized parameters are used as initial values for all layers. The logistic regression layers output potential latent phrases. Meng et al. [14] propose a generative model for keyword prediction with an encoder-decoder model using Gated Recurrent Unit (GRU) neural network [15] that adopts a copying mechanism [16]. In the field of NLP, due to the limitation of word list size, some low frequency words cannot be included in the word list, which is known as the out-of-vocabulary (OOV) issue. These words will be displayed uniformly by “UNK” instead while their semantic information will be discarded. The copy mechanism includes Generate-Mode and Copy-Mode. Generate-Mode calculates the output probability of the words in the traditional word list which contains a large number of common words, and UNK while Copy-Mode calculates the output frequency of the words in the source sequence word list. Finally, the word probabilities given by the two modes are summed to obtain the final word probability distribution.

1.3 Challenges faced by keyword tagging

A document often involves multiple topics. For example, an academic paper on keyword extraction may involve both the topic of "keyword extraction" and the topic of "graph method". It may also contain some other methods for keyword extraction and then those methods may carry the meaning of other topics. As a summary of the subject of the document, keywords should have proper coverage. Taking various aspects into consideration, document keywords should meet the following three characteristics at the same time:

1. Understandability. The keywords should be understandable to every individual. This means that the extracted keywords should be grammatically correct and meaningful. For example, "machine learning" is a grammatical phrase, but "machine learned" is meaningless. So, if the extracted keywords comprise incorrect or meaningless phrases, it means that this keyword extraction method does not meet the requirement of understandable.

2. Relevance. The keywords are semantically related to the document topic. For example, for a document about "machine learning", we want to extract keywords that are all about this topic machine learning or closely related topics, such as artificial intelligence, speech recognition, etc.

3. Good coverage. The keywords should cover the whole document very well. Suppose we have a document that describes "Beijing" from different aspects such as "location", "atmosphere" and "culture". Then the extracted keywords should cover all three aspects and not just a part of them. For example, if a certain method extracts keywords that are all related to "location" and have nothing to do with "atmosphere" and "culture", then this method does not have good coverage.

In view of the above three characteristics, the current keyword tagging algorithms face the following two important challenges: whether keywords can basically satisfy all the above three characteristics, and whether they can interpret true meaning of the text.

1.3.1 Difficulties in meeting keyword characteristics

The statistics-based method considers only the statistical properties of the words and thus it cannot guarantee all extracted keywords/keyphrases are grammatical. Also, this method ignores the semantic relevance among the words, making the extracted keywords less likely to be relevant to the main theme of the document. Moreover, the statistics-based method ranks words only in terms of their frequency of occurrence in the document, but the top ranked keywords may not necessarily cover the entire document.

The graph-based approach yields keywords that can meet the characteristic of being understandable. However, it is not efficient in terms of relevance, TextRank, for example, uses co-occurrence windows to emphasize the semantic relationship between local words, but the extracted words do not show good relevance to the whole text. In addition, the graph-based approach tends to select words that appear more frequently in the document, leading to a poor coverage in general.

The clustering-based approach groups all candidate terms in a document into clusters based on semantic relevance. The noun phrases closest to the cluster center of each cluster represent the keywords to be extracted. This approach ensures good coverage and has the characteristic of relevance. In addition, only noun phrases are selected for keyword, which also ensures the keyword has the understandability. But to the best of my knowledge, no one has combined the fastText and GloVe (Global Vectors for Word Representation) word embedding methods, which are two newer word embedding methods after the Word2Vec, with clustering idea except for Wan and Xiao's work [8], which is one of the inspirations for my research.

1.3.2 Difficulties in interpreting true meaning of the text

The relevance of the keyword to the document is an important indicator for assigning keywords. Out-of-text keyword tags often involve the problem of opinion mining. Determining phrases that identify the polarity in the text is generally difficult. Sentiment polarity analysis is the process of analyzing, processing, generalizing, and reasoning about subjective texts with emotional overtones. Many adjectives are domain-dependent (e.g., two comments about a machine: long-time battery life vs. taking a long time to turn on). The former is satisfaction with the machine, implying a positive evaluation, while the latter is dissatisfaction with the machine, i.e., a negative evaluation; similarly, emotion and subjectivity are context-sensitive. For example, in the evaluation of a movie adapted from a book, the sentence "It was very enjoyable to read this book" is negative in the context of the movie review, but positive in the context of the review of this book. Therefore, it is not possible to make judgments about the polarity of textual sentiment based only on individual phrases or words such as long time or enjoyable, which is a difficult issue for keyword assignment.

It is known that the attention mechanism can assign different weights to each part of the input, which can help the model pay more attention to important information and ignore irrelevant information, enabling the model to make more accurate judgments without increasing the computational load of the model. This may motivate us to deal with the problem of ambiguity in understanding the actual meaning behind the text/document by applying the attention mechanism to the deep learning model.

1.4 Organization of this thesis

The rest of the thesis is organized as follows:

Chapter 2: This chapter first describes commonly used statistics-based keyword extraction methods TF-IDF, RAKE, graph-based method TextRank, and one existing clustering based-method that integrates Word2Vec and K-means clustering. We then study two word embedding methods, GloVe and fastText, in comparison with the previously reviewed Word2Vec on the different datasets in order to identify the most suitable word embedding scheme for keyword extraction. The selected word embedding scheme, i.e., fastText is combined with three clustering methods (Affinity Propagation, Mean Shift and K-means) to extract the key words. Intensive computer simulation of the proposed keyword extraction schemes with comparison to other existing methods is also presented.

Chapter 3: This chapter first introduces the background and basic principles of deep learning including CNN and RNN. Then it gives a detailed description of our proposed CNN-BiLSTM model using word embedding and attention mechanism for keyword assignment. Computer simulation of the proposed model is also conducted in comparison with other existing methods.

Chapter 4: This chapter concludes the thesis and gives some directions for future research.

Chapter 2

Unsupervised Keyword Extraction Integrating Word Embedding with Clustering

In this chapter, we propose an unsupervised method that to extract keywords from texts by integrating word embedding and clustering methods. In Section 2.1, we will introduce the background and recent works in keyword extraction. Section 2.2 describes the proposed unsupervised keyword extraction model. Performances of the proposed method are evaluated in Section 2.3 through computer simulation based on popular datasets.

2.1 Previous work

In the previous chapter, we briefly mentioned three unsupervised methods for keyword extraction, i.e., statistics-based methods TF-IDF and RAKE and graph-based method TextRank. Here in Subsection 2.1.1, we will introduce these methods in detail. We will also introduce the Word2Vec word embedding and the K-Means clustering in Subsection 2.1.2.

2.1.1 Statistics-based and graph-based keyword extraction methods

TF-IDF:

In a given document, term frequency (tf) is the frequency of a particular word in the document. This number is the normalized to the term count. For word t_i in a particular document d_j, its importance can be expressed as:

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

where $n_{i,j}$ is the number of times of the word t_i in the file d_j, and the denominator is the total number of occurrences of all words in the file d_j.

Inverse document frequency (idf) is a measure of the general importance of words. The idf of a particular word can be obtained by dividing the total number of documents containing the word, and the resulting quotient is calculated by taking the logarithm of the base 10:

$$idf_i = lg \frac{|D|}{|\{all\ j: t_i \epsilon d_j\}|}$$

where $|D|$ represents the total number of files in the corpus, and $|\{j: t_i \in d_j\}|$ represents the number of files that contain word t_i (here, the number of files is not zero).

Define the product of $tf_{i,j}$ and idf_i as:

$$tfidf_{i,j} = tf_{i,j} \cdot idf_i$$

The score of $tfidf_{i,j}$ represents the relative importance of word t_i to document d_j.

After the score of $tfidf_{i,j}$ is calculated for all words, all scores are sorted in descending order and the words at the top are selected as the keywords for document d_j.

TextRank:

TextRank is a graph-based ranking algorithm that builds a graph model by dividing text into words or phrases, and uses a voting mechanism to rank the words or groups of words. The algorithm does not require prior training of multiple documents, but only uses the information from a single document itself to conduct keyword extraction.

The general model of TextRank can be expressed as a directed weighted graph $G = (V, E)$, consisting of a point set V and an edge set E, where E is a subset of $V \times V$. Each candidate phrase is a point and the line between every two phrases is called edge. The score of point V_i is defined as follows:

$$\mathrm{WS}(\mathrm{V_i}) = (1 - \mathrm{d}) + \mathrm{d} \times \sum_{\mathrm{V_j} \epsilon \mathrm{In}(\mathrm{V_i})} \frac{w_{\mathrm{ji}}}{\sum_{\mathrm{V_k} \epsilon \mathrm{Out}(\mathrm{V_j})} w_{\mathrm{jk}}} \mathrm{WS}(\mathrm{V_j})$$

The WS score indicates the weight of a word, and the summation on the right side of the above equation indicates the degree of contribution of each adjacent sentence to this sentence. The weight of the edge between any two points V_i and V_j is denoted as w_{ji}. For a given point V_i, In (V_i) is the set of points pointing to V_i, and $Out(V_i)$ is the set of points that V_i points to.

The parameter d is a damping factor ranging from 0 to 1. It represents the probability that one point points to any other point in the graph, and in general d is set to 0.85.

RAKE:

RAKE first divides a text into multiple clauses using punctuations, and then each clause is divided into phrases using stopwords as separators. These phrases are used as candidates for the finally extracted keywords.

Next, each phrase is divided into several words by spaces. Each word is assigned a score and the score for each phrase is obtained by adding up the scores of each word in the phrase. A key point is to take into account the co-occurrence relationship of each word in this phrase. The score of each word is defined as:

$$wordScore(w) = wordDegree(w) / wordFrequency(w)$$

That is, the score of word w is the degree of the word (its degree increases by 1 whenever the word appears in a window of set length with another word at the same time) divided by the frequency of the word (the total number of times the word appears in the document). Then for each candidate key phrase, the score of each word is accumulated and ranked. RAKE uses the first third of the total number of candidate phrases as the extracted keywords.

2.1.2 Keyword extraction using word embedding and clustering

Here we introduce another keyword extraction method proposed in [17] that makes use of word embedding and K-means clustering. This method will be used as benchmark for performance comparison with our proposed method.

Word2Vec

Word co-occurrence is at the heart of several machine learning algorithms for natural language processing, including the Word2Vec recently introduced by Mikolov et al. [18]. The neural models in [18] were shallow networks with only a single linear hidden layer, allowing it to be trained on datasets consisting of more than a billion of words. In particular, the authors proposed two new architectures for Word2Vec: the continuous bag-of-words (CBOW) and skip-gram models in Word2Vec. They also publicly released a toolbox with implementations of their algorithms and models that were pre-trained on large datasets, making it easy to implement in other NLP tasks. The CBOW of Word2Vec was trained to predict a word given its context, as shown in the left part of Figure 4. The input consists of C one-hot [18] encoded word vectors. As the hidden layer is shared for all input words, the input vectors are averaged in the hidden layer, hence leading to so-called continuous bag-of-words. As shown in the right part of Figure 4, the skip-gram method is essentially a reversed CBOW architecture. Instead of predicting a word given its context, this model is trained to predict the words surrounding a given word.

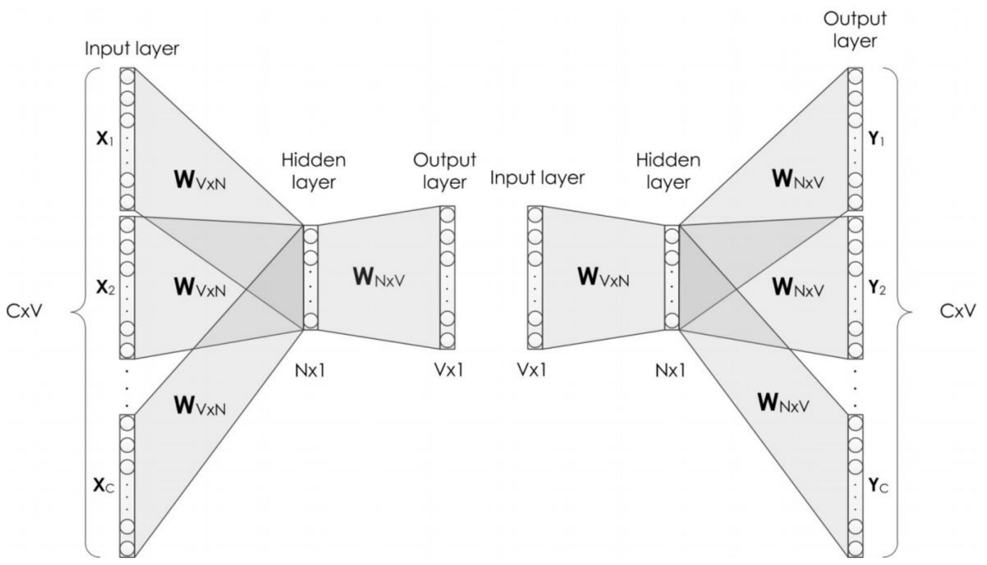


Figure 4: TwoWord2Vec model architectures proposed by Mikolov et al. (image taken from [19]).

The CBOW model in Figure 4 takes the input X_1, which is a one-hot input neuron of $V \times 1$. Assume that the hidden layer has N neurons, then the hidden layer H is a vector of $N \times 1$. The output vector and the input vector are of equal dimension $V \times 1$. $W_{v\times n}$ and $W_{n\times v}$ are initialized randomly and their elements are initialized to [-1,1], so that $H = X^T W_{v\times n}, O = H^T W_{n\times v}$. After the output vector is normalized, the error vector of the output layer is calculated by subtracting the probability vector, obtaining from the whole dataset, from the target vector. Once the errors are known, the weights of $W_{v\times n}$ and $W_{n\times v}$ can be updated iteratively using backpropagation. If the input is C vectors of $V \times 1$, the hidden layer vectors obtained by multiplying all the input layer vectors and $W_{v\times n}$ are averaged and the result is denoted as H. The final $W_{v\times n}$ is the word embedding vector.

In an extension of their original work, Mikolov et al. later proposed modifications to make theWord2Vec algorithm more computationally efficient [20]. First of all, they found that the frequent computation of the softmax function on high dimensional vectors was expensive. To improve on this, they suggested that a hierarchical softmax function be used instead [21]. The hierarchical softmax uses a binary Huffman tree that can estimate the standard softmax output while being much easier to compute. As an alternative to this, they also proposed to use negative sampling during the training. Usually, the loss is determined based on the output for the complete vocabulary, including all positive and negative samples. With a large vocabulary there are many negative samples, which is the main reason why computing the softmax function is expensive. By only training all positive samples and a fixed amount of random negative samples, we only need to compute significantly reduced outputs and update the weights each step, largely increasing the efficiency. As a final improvement, to balance common and rare words, they performed subsampling on frequent words. Some frequent words (such as stopwords like "a", "in", "the" or "it") provide much less information than rare words, and the word vectors of frequent words do not change significantly after hundreds of training sessions. A subsampling method is proposed: each word in the training set is removed with a certain probability. The more frequently the words appear, then the greater the probability of being removed. The method of subsampling frequent words accelerates parameter learning and significantly improves the accuracy of the word vector for rare words.

K-means clustering

K-means clustering, as an unsupervised learning clustering method with particularly low complexity, is widely used in the field of data mining. Its goal is to form groupings of data points based on the number of clusters represented by the variable k, which needs to be defined in advance before execution. K-means clustering uses an iterative refinement method to generate clusters based on the number of user-defined cluster centers. First, it randomly selects k centroids and finds the closest data points to each centroid to form k clusters. Then, the algorithm recalculates the new centroids of each cluster by continuously iterating until the algorithm converges to an optimal value. The algorithm consists of the following main steps:

1. For a set value of k, k points are randomly initialized into k clusters.

2. To form k clusters, each data point of the data set is assigned to its nearest centroid using the Euclidean distance.

3. The centroids are recalculated by averaging all of the data points assigned in each cluster so that the total variance within clusters can be reduced.

4. Steps 2 and 3 iterate when the centroid of each cluster no longer changes, or when the data points assigned to each cluster are the same as the previous assignment, the algorithm stops. The stopping criterion can also be maximum iteration number [22].

2.2 Proposed unsupervised keyword extraction method

As mentioned before, good keywords should satisfy three characteristics at the same time: 1. understandability, 2. relevance, and 3. good coverage. Statistics-based methods do not satisfy these three characteristics. Graph-based methods cannot guarantee that the top-ranked keywords can cover the whole document, namely they don't satisfy characteristic 3. Moreover, since the graph-based methods only consider local relevance, they cannot fully satisfy characteristic 2. Clustering-based methods group terms according to semantic relevance, which ensures good coverage of the document and satisfies features 2 and 3. Also, clustering-based methods extract only the keywords in accordance with noun group, so this ensures that the extracted keywords satisfy characteristic 1.

To develop an unsupervised keyword extraction method, we first study three word embedding methods, Word2Vec, GloVe and fastText that give the word vectors of candidate keywords. We then evaluate the three different word embedding methods according to the semantic relevance of the words. That is, we calculate the cosine similarity of the pairs of words to determine which word embedding method is most suitable for the dataset. Next, we apply three clustering algorithms, namely, Affinity Propagation, Mean Shift and K-means to the candidate keywords given by the word embedding scheme in order to determine the final keywords.

Our proposed keyword extraction method mainly includes the following steps:

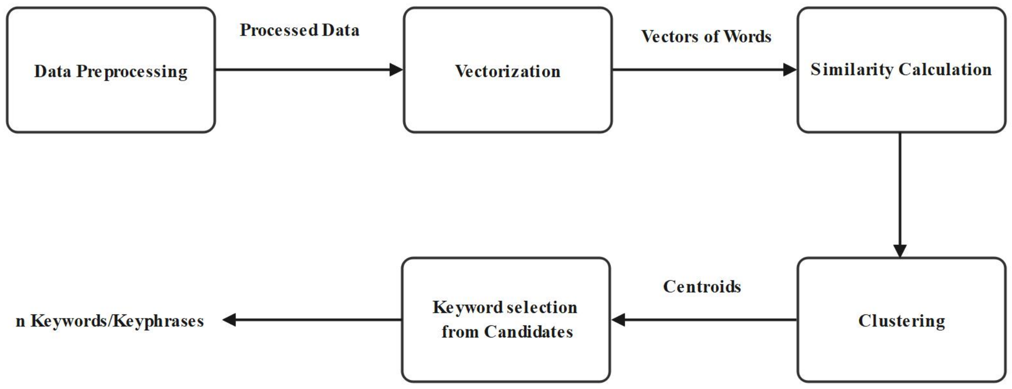


Figure 5: Workflow of our proposed method.

Step 1. Preprocessing and candidate keywords selection: the text is preprocessed and candidate keywords are selected in the form of nouns or noun phrases.

Step 2. Vectorization: all candidate keywords are converted into word vectors using three word embedding methods respectively.

Step 3. Similarity calculation: we calculate the cosine similarity between pairs of words to determine which word embedding is most suitable for the dataset.

Step 4. Clustering: we group the candidate keywords resulting from word embedding into clusters using three clustering algorithms.

Step 5. Keyword identification: finally, we select the candidate words closest to the centroids as the keywords to be extracted.

In the following, the whole process will be described in detail.

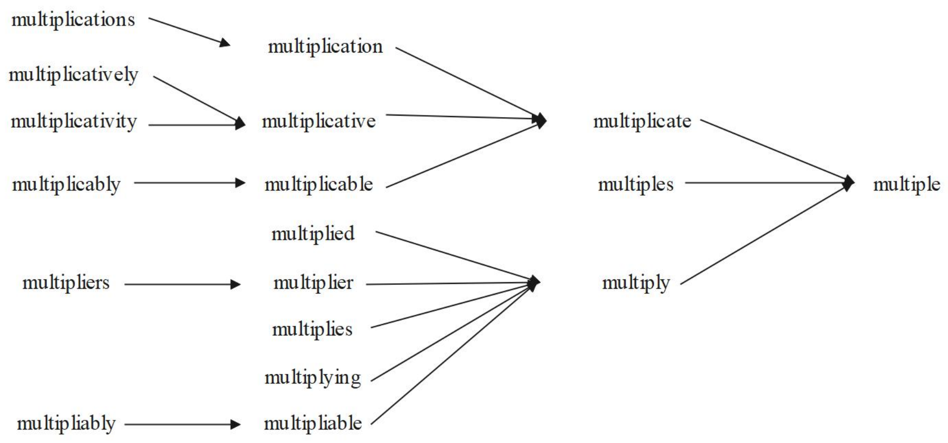
2.2.1 Preprocessing

Preprocessing the text simply means bringing the text into a form that is predictable and analyzable for the task. Its purpose is to identify the set of terms, single words or compound words, which can be tagged as keywords or keyphrases in the next steps of the method. The preprocessing step is composed of the following operations:

1. Tokenization: to split the entire text into individual words.

2. Token normalization: which, depending on the task, can be as simple as case folding (discarding information about letter casing) or lemmatization (the process of grouping together the inflected forms of a word so they can be analyzed as a single item,

identified by the word's lemma). Hence, we take the word "multiple" as an example to explain lemmatization. As seen from the following graph, the word "multiple" may appear in many different forms such as "multiplications", "multiplicatively", "multipliers" and "multipliable". These different forms are converted to their most basic form "multiple".



3. Removal of stopwords: stopwords are certain words that are automatically filtered out during preprocessing in order to save storage space and improve search efficiency in information retrieval. All forms of stopwords have already been described in the previous chapter.

4. Spelling correction: deal with ambiguous spelling variations such as “Nokia N8” vs. “Nokia N-8” vs. “Nokia N 8”, or “Windows 2000” vs. “Windows 2k”, “don’t” vs. “dont” and “stateof-the-art” vs. “state of the art” etc. Depending on the application, the desired course of action may be to use the exact form as it appears in the text, or normalize the tokens to a single canonical form.

In my thesis, the preprocessing process uses two standard tools: Stanford University's Core NLP Suite and Python's Natural Language Toolkit (NLTK) [23].

The core NLP suite from Stanford University provides a set of tools are to be used in accomplishing natural language tasks. It can effectively give the basic form and part of speech (POS) of each word and most importantly it is able to recognize noun phrases, since they are the basic forms of the keywords that we need to extract [25]. Natural

Language Toolkit NLTK provides a set of language processing libraries. This tool can be used for stemming, tagging, parsing, etc. during preprocessing.

2.2.2 Word vectorization

Word vectorization or word embedding is to transform a one-dimensional text into a high-dimensional word vector, and in most cases, the dimension can be as high as 300. Each word corresponding to a distinct word vector. The word vector of a word group is an average vector obtained simply by adding together the vector of each word and then taking the average value. A good word embedding approach should be able to represent the words with similar meanings as similar vectors and can effectively capture the semantic similarity between a word and its context in a document. In what follows, we introduce two word embedding methods: GloVe and fastText.

GloVe

Pennington, Socher, and Manning introduced Global Vectors (GloVe) in 2014 [25] by constructing a global word co-occurrence frequency matrix. The core idea behind their approach lies in the use of word co-occurrence ratios rather than pure frequencies. A simple example below can illustrate the word co-occurrence frequency matrix. Suppose a corpus has only three sentences, which are “I like studying deep learning”, “I like NLP” and “I like flying”. After tokenization, a text group “I like studying deep learning I like NLP I like fly” is obtained. Suppose we create a word frequency co-occurrence matrix using a window of length 2. This matrix has the structure of Table 1 below. The top row and the left-most column are from different words of the tokenized form by sequence. A window of length 2 is moving through the entire tokenization. The number of co-occurrences of the two words within this window will give the values in the table which constitute the co-occurrence matrix. Obviously, this matrix is symmetric.

Table 1: A sample of word frequency co-occurrence matrix.

	I	like	studying	deep	learning	NLP	fly
I	0	3	0	0	1	0	0
like	3	0	1	0	0	1	1

studying	0	1	0	1	0	0	0
deep	0	0	1	0	1	0	0
learning	1	0	0	1	0	0	0
NLP	0	1	0	0	0	0	0
fly	0	1	0	0	0	0	0

First, let X_{ij} be the number of times a word i occurs within the fixed window (usually the window size is set to 2 - 20) with a word j. Let X_i be the total number of co-occurrences of word i with all other words. Then, the co-occurrence probability X_{ij} of X_i and X_j can be defined as P_{ij} and which is calculated as follows:

$$P_{ij} = P(j|i) = \frac{X_{ij}}{X_i} = \frac{X_{ij}}{\sum_k X_{ik}} \tag{1}$$

Table 2 shows an example of co-occurrence probabilities of several typical words. Two words $i = ice$ and $j = steam$ are chosen as target words to illustrate their relationships with other four words, denoted as k, selected from the text. The co-occurrence probabilities of these words with are calculated from the corpus of around six billion words (tokens). The ratio of co-occurrence probabilities is denoted as $\frac{P_{ik}}{P_{jk}}$. When word k has similar semantics to ice and $steam$, the ratio $\frac{P_{ik}}{P_{jk}}$ will be large, which is the case of $k = water$. On the other hand, when word k is not relevant to both target words, this ratio is also close to 1, which is the case of $k = fashion$.

Table 2: Co-occurrence probabilities for target words ice and $steam$ with selected context words from a 6 billion token corpus [25].

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

With the above intuition, Pennington, Socher, and Manning [25] have proposed a method to model the ratio of the co-occurrence probabilities involving two target words and one context word, namely,

$$F\left(\omega_i, \omega_j, \widetilde{\omega_k}\right) = \frac{P_{ik}}{P_{jk}} \tag{2}$$

with $\omega \epsilon R^d$ and $\widetilde{\omega} \epsilon R^d$ which are the target word vectors and the text vector, respectively.

Further, by defining

$$F(\omega_i, \omega_j, \widetilde{\omega_k}) = \exp[(\omega_i - \omega_j)^T \cdot \widetilde{\omega_k})] \tag{3}$$

We obtain

$$\frac{P_{ik}}{P_{jk}} = \exp\,[(\omega_i - \omega_j)^T \cdot \widetilde{\omega_k}] = \exp\,(\omega_i^T \cdot \widetilde{\omega_k} - \omega_j^T \cdot \widetilde{\omega_k}) = \frac{exp(\omega_i^T \cdot \widetilde{\omega_k})}{exp(\omega_j^T \cdot \widetilde{\omega_k})} \tag{4}$$

In this sense, we can rewrite P_{ik} and P_{jk} as

$$P_{ik} = exp(\omega_i^T \cdot \widetilde{\omega_\mathrm{k}}), P_{jk} = exp(\omega_j^T \cdot \widetilde{\omega_\mathrm{k}}) \tag{5}$$

Or in general we have

$$P_{ij} = exp(\omega_i^T \cdot \widetilde{\omega_\mathrm{j}}) \tag{6}$$

By taking logarithm of (6), we have

$$\log\,(P_{ij}) = \log\left(\frac{X_{ij}}{X_i}\right) = \log(X_{ij}) - \log(X_i) = \omega_i^T \cdot \widetilde{\omega_J} \tag{7}$$

Based on which we can write $\log(X_{ij})$ as

$$\log(X_{ij}) = \omega_i^T \cdot \widetilde{\omega_J} + b_i + \widetilde{b_J} \tag{8}$$

where b_i and $\widetilde{b_J}$ are the bias terms introduced and $\log(X_i)$ is contained in b_i.

Using the above expressions, we define the loss function as

$$\sum_{i,j=1} [\omega_i^T \cdot \widetilde{\omega_J} + b_i + \widetilde{b_J} - \log(X_{ij})]^2 \tag{9}$$

In order to reduce the influence of too frequently occurring words on the model and to distinguish the role of words with different frequencies of occurrence, a weighting function $f(X_{ij})$ is introduced to the loss function by the authors. Specially, the following non-decreasing function and upper-bounded function is used.

$$f(X_{ij}) = \begin{cases} \left(\frac{x}{x_{max}}\right)^a & if\ x < x_{max} \\ 1 & otherwise \end{cases} \tag{10}$$

where a is an adjustable parameter which is set to 0.75 in the paper.

Using (10), the loss function is modified as

$$J = \sum_{i,j=1} f(X_{ij})\left[\omega_i^T \cdot \widetilde{\omega_j} + b_i + \widetilde{b_j} - \log(X_{ij})\right]^2 \tag{11}$$

In the paper, AdaGrad algorithm for gradient descent was used to randomly sample all non-zero elements in matrix X. The learning rate was set to 0.05 and iterated 50 times on vector size smaller than 300 and 100 times on other sizes of vectors until the loss function converged. For word i, two vectors ω_i and $\widetilde{\omega_i}$ are obtained after completing the training. X is symmetric, so in principle ω_i and $\widetilde{\omega_i}$ are also symmetric and their difference is that the initialized values are different, leading to different values after training. To improve the robustness, we finally choose the sum of ω_i and $\widetilde{\omega_i}$ as the word vector for word i.

fastText

Mikolov was a former employee of Google Brain and now is with the Facebook AI Research team (FAIR) which contributes to the developments in NLP field, including machine translation, natural language understanding and generation, question answering, dialogue and so on. The current state-of-the-art in word embeddings is the fastText library which has been made available open-source by FAIR. The algorithms used in fastText build upon the Continuous Skip-gram model and can be trained on corpora with billions of words in minutes [26][27].

A disadvantage of the Word2Vec algorithms is that they ignore the structure of words by assigning different vectors to each word independently. Grammatical inflections are treated as completely separate words, i.e., words like $'sleep'$, $'sleeps'$, $'sleeping'$, $'slept'$, which are examples of verb inflections, and are modeled independently of each other. Given a large enough corpus, the Word2Vec model is likely to be able to assign the inflections of $'sleep'$ similar word vectors because they appear in similar contexts in the corpus. However, certain languages have much more complicated and rare inflections, making it possible for certain inflections to not occur frequently enough even in very large corpora. English is a very easy language to model with word embeddings since the number of inflections is relatively low and compound words usually occur in open form ($post\ office$, rather than $postoffice$). On the other hand, the complex grammatical rules of other languages also amplify the disadvantages of word2Vec when applied on languages other than English, e.g., Spanish has more than 40 different verb inflections, while Finnish has 15 noun inflections [26].

The fastText algorithm known as Subword Information Skip-gram (SISG) solves the problem of modeling languages with rare word inflections by using character n-grams. A character n-gram is a sequence of n letters contained within a word. The SISG algorithm first adds an empty character before and after each word, then each word is decomposed into all of its character n-grams where $n = 3, 4, 5$. For example, <sleep> would be represented by the 3-grams $< sl, sle, lee, eep, ep >$, 4-grams $< sle, slee, leep, eep >$ and 5-grams $< slee, sleep, leep >$. Then, a word vector is trained for each of the n-grams of $< sleep >$. Finally, the original word $< sleep >$ is assigned the word vector equal to the sum of the word vectors of its n-grams. Following this rule, the words $'sleep'$, $'sleeps'$, $'sleeping'$,$'slept'$ share many n-grams and so their word embeddings will be correlated.

Furthermore, SISG remarkably allows for creating word embeddings for words which were not at all present in the training corpus. These are referred to as out-of-vocabulary words (OOV). Given an OOV word, as long as sufficiently many of its n-grams are present in the corpus, it can be modeled as the sum of the word vectors of those n-grams. This is a truly astonishing result as it gives the model a much deeper knowledge of the language and allows for almost any sequence of characters to be systematically assigned a word embedding.

In order to evaluate the applicability of different word embedding methods to a dataset, there are two different similarity calculation methods to evaluate the semantic relatedness for the generated word embedding vectors: the Euclidean distance and the cosine similarity.

The Euclidean distance is defined as the shortest straight-line distance between two points, namely,

$$ed(a,b) = \sqrt{\sum_{1}^{n}(a_i - b_i)^2} \tag{12}$$

where a and b are two different vectors obtained by the word embedding method and n represents the length of the vector.

Suppose there are two words represented by a high-dimensional word embedding vector and the cosine similarity represents the cosine of the angle between them in the inner product space, provided that they are both non-zero vectors. If the cosine similarity is closer to 0, the two words vectors tend to be orthogonal, indicating that the two words are semantically unrelated. If the cosine similarity of two words is close to ± 1, the two words are in the same or opposite direction, indicating that they are semantically the same or opposite terms.

$$\cos\theta = \frac{\vec{a}\,\vec{b}}{\left\|\vec{a}\right\|\left\|\vec{b}\right\|} = \frac{\sum_{i=1}^{i=N} a_i b_i}{\sqrt{\sum_{i=1}^{i=N} {a_i}^2}\sqrt{\sum_{i=1}^{i=N} {b_i}^2}} \tag{13}$$

2.2.3 Clustering

The purpose of clustering is to assign target word vectors to groups so that word vectors in the same group are semantically more similar than word vectors in other groups. After we select the most suitable word embedding method, the word vectors of all candidate words will be clustered using Affinity Propagation, Mean Shift and K-means clustering, and the centroids are extracted as final keywords. The K-means clustering algorithm has already been explained in section 2.1.2. Here we introduce affinity propagation and mean shift algorithms below.

Affinity Propagation Clustering

Traditional clustering methods usually first select K initial data as clustering centers and assign other data points according to the Euclidean distance from the initial clustering centers, and then iterate to determine the final clustering centers. The basic idea of AP algorithm is to treat all samples as points of a network, and then calculate the clustering center of each sample by the message transmission of each edge in the network. During the clustering process, two types of messages are passed among the points, namely attractiveness and availability which will be explained below in detail, and the AP algorithm continuously updates the attractiveness and availability of each point through an iterative process until high quality cluster centers are generated, and the remaining data points are assigned to the corresponding clusters [28].

The similarity between data point i and its respective cluster center at point j is denoted as $s(i,j)$. Generally, the Euclidean distance is used to calculate the similarity between these points, and all the similarity values are taken as negative values. Therefore, the larger the similarity value is, the closer the points are to each other, which is convenient for the later comparison calculation. The reason for taking negative values is to facilitate the later calculation. Suppose S is a matrix composed of all $s(i,j)$. Since i and j traverse every data point, the S matrix is a symmetric square matrix.

The diagonal element $s(i,i)$ or $p(i)$ refers to the preference degree of point i as the exemplar, which is used to indicate whether point i can become the exemplar. The larger the value of $p(i)$, the greater the possibility of this point becomes the exemplar. Also, $s(i,i)$ is usually taken as the median value of S. The number of clusters is influenced by the reference degree p. If each data point is considered as a possible exemplar, then p should be taken as the same value, the median value of S.

The value $r(i,k)$ is called "responsibility", which is used to describe the extent that point k is suitable as an exemplar for data point i and $a(i,k)$ is called "availability", which is used to describe the suitability of point i to select point k as its exemplar. The "responsibility" $r(i,k)$ is updated as follows:

$$r_{t+1}(i,k) \leftarrow s(i,k) - \max_{j \neq k}\{a_t(i,j) + r_t(i,j)\}, i \neq k$$

$$r_{t+1}(i,k) \leftarrow s(i,k) - \max_{j \neq k}\{S(i,j)\}, i = k \tag{14}$$

And the "availability" $a(i,k)$ is updated as follows:

$$a_{t+1}(i,k) \leftarrow min\left\{0, r_{t+1}(k,k) + \sum_{j \neq i,k} max\{0, r_{t+1}(j,k)\}\right\}, i \neq k$$

$$a_{t+1}(i,k) \leftarrow \sum_{j \neq k} max\{0, r_{t+1}(j,k)\}, i = k \tag{15}$$

After several iterations, exemplars can be determined by calculating maximum of $a(i,k) + r(i,k)$ for point i.

Furthermore, in order to avoid numerical oscillations in some circumstances when updating the messages, the damping factor λ is introduced to the iteration process:

$$r_{t+1}(i,k) = \lambda \cdot r_t(i,k) + (1-\lambda) \cdot r_{t+1}(i,k) \tag{16}$$

$$a_{t+1}(i,k) = \lambda \cdot a_t(i,k) + (1-\lambda) \cdot a_{t+1}(i,k) \tag{17}$$

where t denotes the iteration index.

The advantage of the AP algorithm is that it does not require a prior setting of the number of clusters, which is different from K-means clustering that requires a prior setting of K. And the results obtained from multiple executions of the AP clustering algorithm are identical, so there is no need to perform random selection of initial values. The widespread use of AP [28] demonstrates its ability in dealing with large datasets fast and efficiently.

Mean Shift Clustering

The Mean Shift algorithm estimates the density of a sample using a kernel function. It sets a kernel function at each sample point on the dataset and then sums all the kernel functions to get the kernel density estimation of the dataset. Suppose we have a d-dimensional dataset of size n and the bandwidth of the kernel function K is the parameter h. The kernel density estimation function for the dataset is defined as

$$f(x) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) \tag{18}$$

where $K(x)$ is a radially symmetric kernel, and $K(x)$ satisfying the kernel function condition denoted by

$$K(x) = c_{k,d} k\left(||x||^2\right) \tag{19}$$

where $c_{k,d}$ is a normalization constant such that the integral of $K(x)$ is equal to 1. The basic goal of the Mean Shift algorithm is to shift the sample points in the direction of increasing local density and the direction of the gradient in the density of the dataset is the direction of the fastest increase in density.

The gradient of the density estimator (18) is given by

$$\begin{aligned} \nabla f(x) &= \frac{2c_{k,d}}{nh^{d+2}} \sum_{i=1}^{n} (x_i - x) g\left(\left\|\frac{x - x_i}{h}\right\|^2\right) \\ &= \frac{2c_{k,d}}{nh^{d+2}} \left[\sum_{i=1}^{n} g\left(\left\|\frac{x - x_i}{h}\right\|^2\right)\right] \left[\frac{\sum_{i=1}^{n} x_i g(\left\|\frac{x - x_i}{h}\right\|^2)}{\sum_{i=1}^{n} g\left(\left\|\frac{x - x_i}{h}\right\|^2\right)} - x\right] \end{aligned} \tag{20}$$

where $g(s) = -k'(s)$. The first sum term is proportional to the density estimate at x computed with kernel $G(x) = c_{g,d} g(||x||^2)$ and the second term

$$m_h(x) = \frac{\sum_{i=1}^{n} x_i g\left(\left\|\frac{x - x_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{x - x_i}{h}\right\|^2\right)} - x \tag{21}$$

is the mean shift [29]. The mean shift procedure is obtained by successive computation of the mean shift vector $m_h(x^t)$ and translation of the window $x^{t+1} = x^t + m_h(x^t)$. Finally, the mean shift is able to find the point that makes the gradient of the density function converge to zero. The mean shift mode finding process is illustrated in Figure 6.

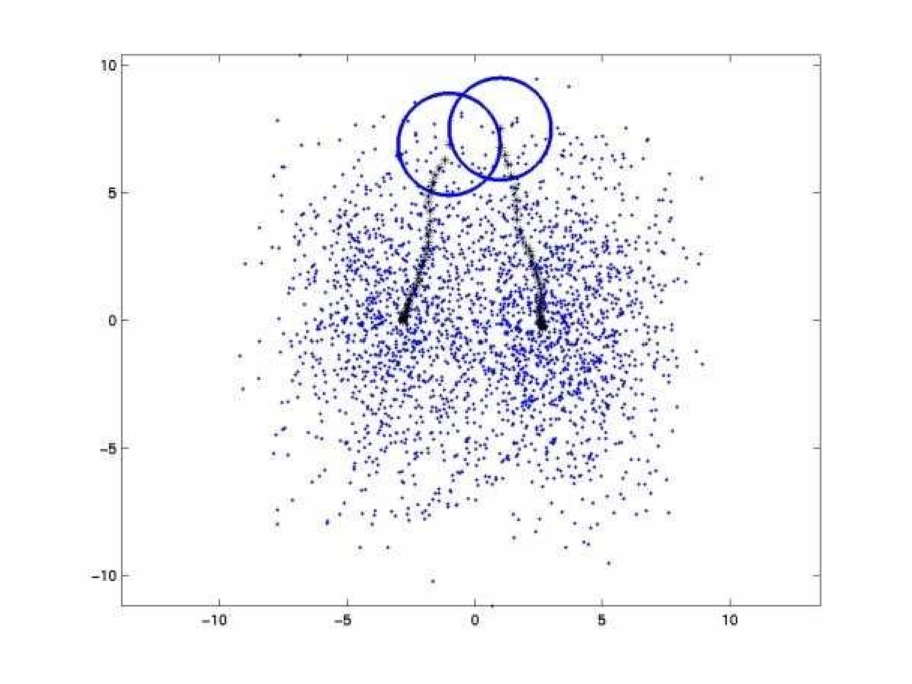


Figure 6: Mean Shift Mode Finding.

2.3 Experimental Results

This section presents our experimental results to demonstrate the performances of word embedding and clustering algorithms, based on which we compare the proposed keyword tagging methods with several existing techniques. To this end, we first briefly introduce the datasets and evaluation metrics.

2.3.1 Datasets and evaluation metrics for in-text keyword tagging

Datasets:

Hulth-2003: The first dataset was developed by A. Hulth in 2003 [30]. It consists of 2000 scientific abstracts, which are divided into training set (1000 abstracts), test (500 abstracts), and validation (500 abstracts). Each abstract is accompanied by a list of keywords that Hulth, together with other authors, manually labeled based on semantics. We combine the training and test collections from Hulth-2003 (a total of 1500 abstracts) to form the training set for our experiments, and use the manually annotated keywords as the gold standard.

Krapivin-2009: The second dataset is another well-recognized dataset designed by by Krapivinet al. in 2009 [31]. It consists of 2304 scientific papers from computer science domain published by ACM during 2003-2005. Different parts of the papers, such as title and abstract, are separated, and contained in the dataset, enabling extraction of keywords based on a part of the article text. Formulae, tables, figures and LaTeX markups were removed automatically. Each paper has its key phrases assigned by the authors and verified by the reviewers.

Nguyen-2007. The third dataset, proposed by Nguyen in 2007 [32], is composed of 211 scientific conference papers. The gold keywords were manually assigned by volunteers' students, each given three papers to read.

Table 3 shows the details and statistics of the aforementioned three datasets. In this table, $|D|$ is the number of documents, L_{avg} is the average document length in words, N_{avg} is the average gold-standard keywords (i.e., unigrams, or single words) per document, K_{avg} is the average gold-standard keyphrases (*n*-grams, or phrases of length greater than or equal to 2) assigned per document, and $KP_{avg}$is the average percentage of keyphrases present in the text.

Table 3: Detailed description of three common datasets for keyword extraction.

Collection	$\|D\|$	L_{avg}	N_{avg}	K_{avg}	KP_{avg}
Hulth-2003	1500	129	23	10	90.07
Krapivin-2009	2304	7961	11	5	96.91
Nguyen-2007	211	164	23	11	95.89

Evaluation Metrics:

To evaluate the performance of keyword extraction, we used three most commonly used evaluation metrics in the field of information retrieval: $Precision\ (P)$, $Recall\ (R)$ and $F1-Score$.

Precision refers to the ratio of the number of keywords correctly identified by an extraction algorithm to the total number of keywords extracted by the algorithm. This ratio indicates the ability of the algorithm to accurately extract keywords:

$$Precision = \frac{Correctly\ extracted\ keywords}{All\ extracted\ keywords} \tag{22}$$

Recall refers to the ratio of the number of keywords correctly selected by the extraction algorithm to the total number of labelled keywords (grand truth). The recall rate reflects the overall ability of the algorithm to capture true keywords:

$$Recall = \frac{Correctly\ extracted\ keywords}{All\ correct\ keywords} \tag{23}$$

$F1 - Scrore$ is a comprehensive assessment of P and R, which is defined as the harmonic mean of precision and recall,

$$F1 - Scrore = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{24}$$

2.3.2 Performances of word embedding methods

First of all, we evaluate the performance of three pre-trained word embedding methods, i.e., Word2Vec, fastText and GloVe.

Word2Vec was trained on the April 2018 version of Wikipedia. Its parameters are set as follows. The dimensionality of the vector is 300, the maximum distance between the target item (word or entity) and the contextual word to be predicted is 5, the number of iterations of the Wikipedia page is 10, and the number of negative samples is 15 [9].

The GloVe vectors were trained on Wikipedia 2014+Gigaword 5 with a total of 6 billion tokens and 400K words [25].

The fastText vectors were trained on Wikipedia 2017, the UMBC web corpus and the statmt.org news dataset (16B tokens) with a 1-million-word vectors [33].

For all required word embedding methods, the vector length is 300 features. The preprocessed data is put into each of the three trained word embedding models respectively and each model obtains a 300-dimensional vector for each candidate word. Since it isn't possible to graphically show a high-dimensional vector, we apply Principal Component Analysis (PCA) to reduce the dimension of word vectors to 2-D data by selecting feature vectors with larger feature values to base transform the original data. The dimension-reduced word vectors obtained by the three word embedding methods are intuitively shown in Figure 7-9.

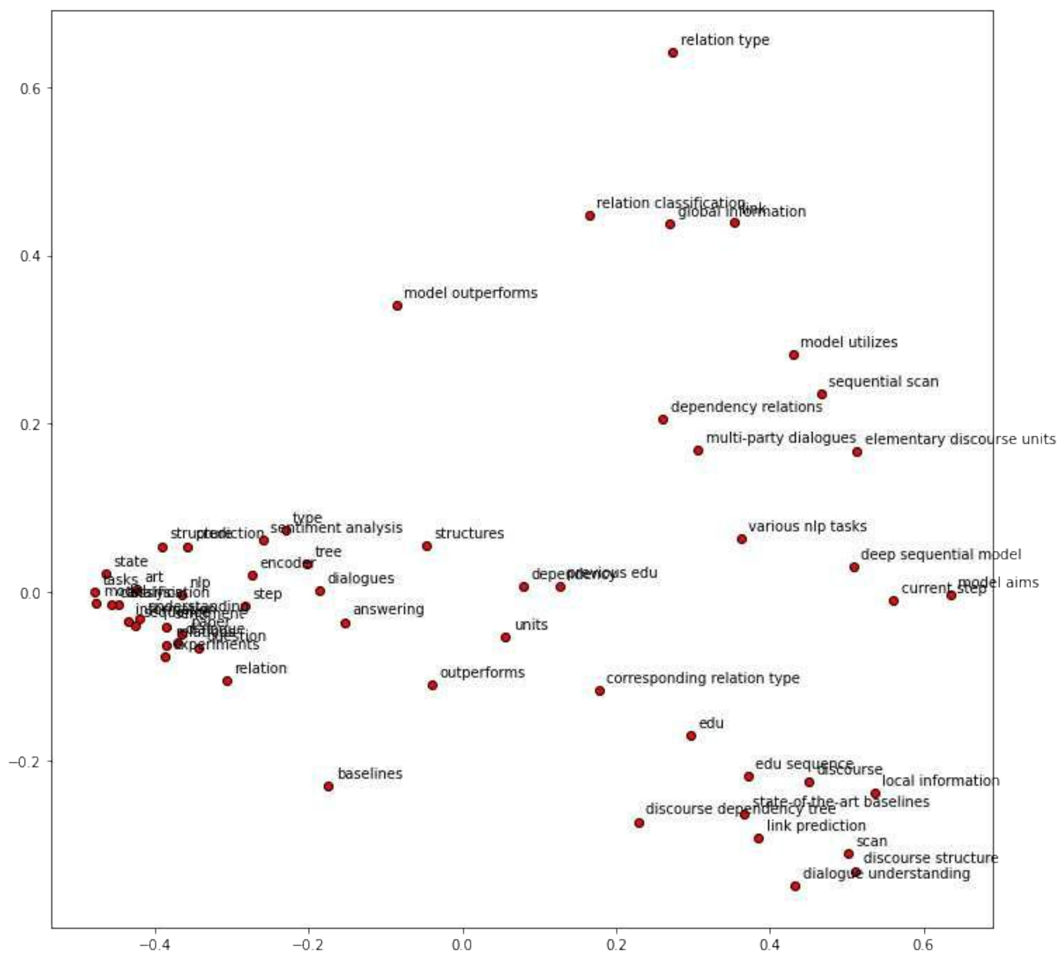


Figure 7: Two-dimensional word vectors obtained by Word2Vec word embedding method.

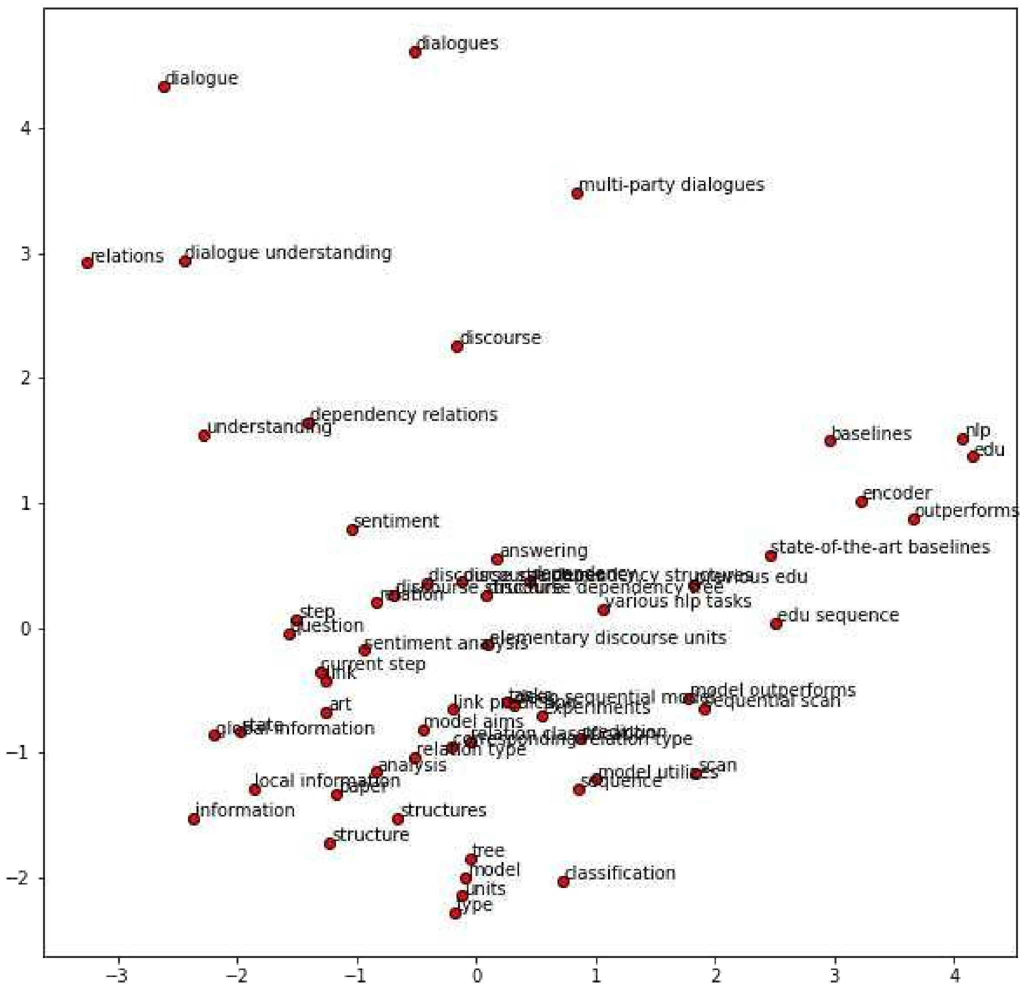


Figure 8: Two-dimensional word vectors obtained by GloVe word embedding method.

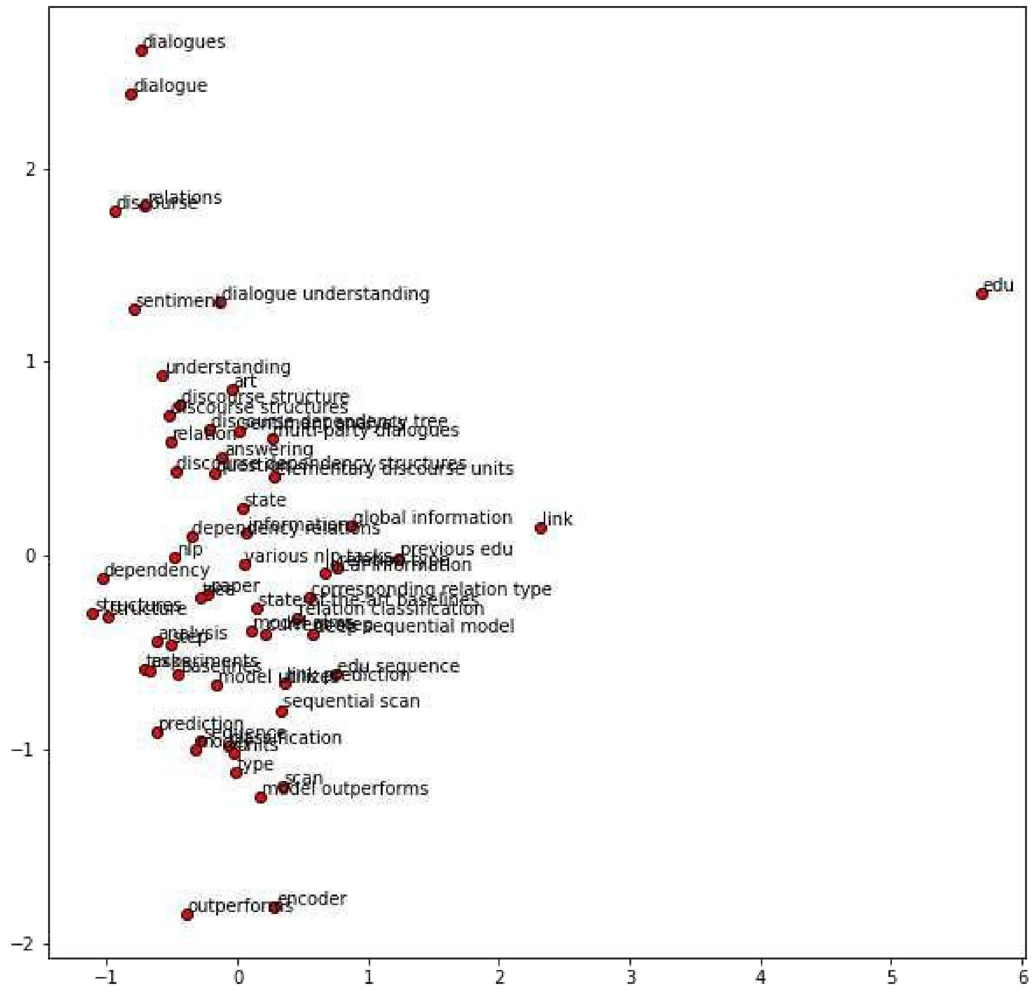


Figure 9: Two-dimensional word vectors obtained by fastText word embedding method.

It should be mentioned that in the original 300-dimensional space, the words having similar semantics should have close word vectors. However, this semantic similarity could not be illustrated in the 2-D space due to the dimension reduction. Usually, we calculate the cosine similarity between two word vectors. To assess effectiveness of word embedding methods, we calculate the cosine similarity between multiple pairs of words and analyze thesis scores versus semantic similarity between the corresponding word pairs. To explain this idea, we take six pairs of words as example. These six pairs of words are shown in table 4 along with their similarity scores computed by the three embedding methods.

Table 4: Scores of the cosine similarity of six pair of words after using three different word embedding methods.

Method	GloVe	fastText	Word2Vec
dialogue, dialogues	0.450	0.856	0.717
structure, structures	0.653	0.833	0.544
type, classification	0.323	0.372	0.714
model, sequence	0.177	0.218	0.863
scan, relations	0.027	0.054	0.087
understanding, encoder	0.004	0.056	0.770

The four pairs of words in the first four rows of Table 4 are all very close semantically. Dialogue and dialogues, as well as structure and structures, are only different in singular and plural, so the similarity between the two pairs of words should be very high and close to 1, from our semantic understanding. Type and classification are both related to categories and in the area of data science, and sequence is a branch of the model, so they both have some relevance and the third score should be lower than the first two and higher than the fourth one. From the results in the table, fastText has the highest score in the first two pairs. Word2Vec achieves the highest score for the third and fourth pairs, but the value is too large, while fastText and GloVe are within our expectation. GloVe achieved the lowest score in the last two pairs and the score of fastText also shows that there is almost no semantic relationship in last two pairs of words.

Thus, fastText appears to be a more reliable word vector model. Considering our test results, even if "self-attention" is not used in the pre-training vocabulary, fastText can still create a vector representation for it while GloVe and Word2Vec cannot. This shows that fastText has better coverage than two others. As a consequence, we choose fastText word embedding method.

2.3.3 Performances of clustering methods

We apply Affinity Propagation, Mean Shift and K-means to the fastText word embedding results for each document separately. K-means requires the number of clusters ($n\_clusters$) as its input, while Affinity Propagation and Mean Shift do not. For each dataset, we take the number of cluster centers automatically obtained by applying the AP algorithm and tuning the bandwidth parameter of Mean Shift so that the number of cluster centers obtained by the two clustering methods is close. We therefore applied K-Means twice to ensure fairness in the evaluation process, once to set $n\_clusters$ based on the results returned by Affinity Propagation and another time to set $n\_clusters$ based on the results returned by Mean Shift, and the obtained results were all averaged.

For Hulth-2003 dataset, I discovered that a bandwidth of 2.20 of Mean Shift would result in similar average n_clusters for the two algorithms. For Nguyen-2007 dataset, a bandwidth of 8.07 of Mean Shift leads to the similar average n_clusters for the two algorithms. For Krapivin-2009 dataset, we set the bandwidth of Mean Shift as 2.15. Table 5 show the average number of clusters obtained by AP clustering and mean shift clustering for three datasets.

Table 5: The average number of clusters obtained by AP and MS clustering based on three datasets.

Method	Average number of clusters		
	Hulth-2003 dataset	Nguyen-2007 dataset	Krapivin-2009 dataset
Affinity Propagation	14.623	4.145	14.844
Mean Shift	14.311	3.936	14.675

As discussed above, Affinity Propagation and Mean Shift generated their own n_clusters, which would lead to keyword lists of different lengths. Thus, we sliced the lists returned by RAKE and TextRank differently on n for each respective algorithm. We will compare our methods with RAKE and TextRank. The reason why we do not use TF-IDF is because it requires a larger corpus of documents to be more effective in our experiments.

Tables 6-8 list experimental results obtained using our methods and some other existing methods on the three datasets. For each method, in addition to accuracy, recall, and F1-Score, the tables give the total number of extracted keywords, the average number of keywords extracted per document, as well as the total number of accurately extracted keywords and the average number of correct keywords for each article. Affinity Propagation, Mean Shift and K-means are represented by AP, MS and KM, respectively.

Table 6: Results and comparison of keyword extraction on Hulth-2003 dataset.

	Extracted keywords		Correct keywords		Evaluation Metrics		
Method	Total number	Average	Total number	Average	Precision	Recall	F1-Score
Hulth's[30]	7815	15.6	1973	3.9	0.252	0.517	0.339
TextRank	6784	13.7	2116	4.3	0.312	0.431	0.362
RAKE	6552	13.1	2207	4.3	0.337	0.415	0.372
AP based on fastText	7303	14.6	2417	4.8	0.342	**0.697**	**0.459**
MS based on fastText	7158	14.3	2397	4.8	0.313	0.586	0.408
KM based on fastText	8013	13.0	2484	4.0	**0.350**	0.660	0.457

In table 6 shows that the AP combined with fastText gives the highest Recall 0.697 and F1-Score 0.459. The fastText-based KM gets the highest accuracy 0.350.

It indicates that for dataset Hulth-2003, our method is better than the existing three methods. For the fastText-based AP, F1-Score even exceeds the result of TextRank, RAKE and Hulth's by about 23%.

Table 7: Results and comparison of keyword extraction on Nguyen-2007 dataset.

	Extracted keywords		Correct keywords		Evaluation Metrics		
Method	Total number	Average	Total number	Average	Precision	Recall	F1-Score
TextRank	955	4.8	132	0.67	0.139	0.134	0.193
RAKE	922	3.1	273	0.80	0.259	0.266	0.262
AP based on fastText	1028	4.1	305	1.21	**0.297**	0.331	**0.313**
MS based on fastText	1008	3.9	234	0.91	0.233	0.304	0.263
KM based on fastText	1128	4.2	311	1.16	0.276	**0.333**	0.301

In table 7, the fastText-based AP achieves the highest Precision 0.297 and F1-Score 0.313, and KM clustering based on fastText word embedding gets the highest Recall 0.333. This comparison shows that for dataset Nguyen-2007, our method is better than TextRank and RAKE. For the AP clustering based on fastText, F1-Score exceeds the results of TextRank and RAKE by about 4%.

Table 8: Results and comparison of keyword extraction on Krapivin-2009 dataset.

	Extracted keywords		Correct keywords		Evaluation Metrics		
Method	Total number	Average	Total number	Average	Precision	Recall	F1-Score
TextRank	6784	13.7	950	1.9	0.140	0.187	0.160
RAKE	6552	13.1	1107	2.2	0.169	0.410	0.239
AP based on fastText	7158	14.8	1868	3.8	**0.261**	**0.441**	**0.328**
MS based on fastText	6799	14.6	1339	2.9	0.197	0.375	0.289

KM based on fastText	7743	13.4	1850	3.2	0.239	0.423	0.299

In table 8, the fastText-based AP again achieves the highest Precision 0.261, Recall 0.441 and F1-Score 0.328. Obviously, our method is better than TextRank and RAKE. For the AP based on fastText, F1-Score exceeds the results of TextRank and RAKE by about 37.24%.

The above simulation results show that for each dataset, the method integrating the fastText word embedding and the AP clustering algorithm achieve high accuracy, recall and F1-score. This can also demonstrate the stability of the algorithms in different datasets. However, Mean Shift performed much worse than K-means and Affinity Propagation. Upon further investigation, we discovered that this is probably because the embedding dimension of 300 is too high for Mean Shift to work effectively. According to a review of mean-shift algorithms for clustering [34][35], the kernel density estimator applied by the Mean Shift algorithm would break down in high dimensions.

2.4 Summary

This chapter first described three commonly used statistics-based keyword extraction methods TextRank and RAKE, and one existing clustering based-method that integrates Word2Vec and K-means clustering. We then studied two word embedding methods, GloVe and fastText, in comparison with the previously reviewed Word2Vec on different datasets in order to identify the most suitable word embedding scheme for keyword extraction. The selected word embedding scheme, i.e., fastText is combined with three clustering algorithms (Affinity Propagation, Mean Shift and K-means) to extract the keywords.

Our experimental results showed that the keywords extracted by fastText word embedding integrated with Affinity Propagation clustering can greatly improve the accuracy, recall and F1-Score compared with existing methods, on all the three commonly-used datasets.

Chapter 3

Keyword Assignment Based on CNN-BiLSTM Model Using Word Embedding and Attention Mechanism

In this chapter, we propose a new CNN-BiLSTM model using word embedding and self-attention that can improve the accuracy of keyword assignment. In Section 3.1, we will introduce the background of deep learning and the basic principles of CNN and RNN. Section 3.2 describes the proposed supervised keyword assignment model. Computer simulation of the proposed model is conducted in comparison with other existing methods in Section 3.3.

3.1 Deep learning background

Deep learning is essentially implemented with the help of on multilayer perceptron (MLPs) or deep neural networks (DNNs). By far, the most used as well as the most effective multilayer neural networks are convolutional neural networks (CNN) and recurrent neural networks (RNN). Currently, CNN and RNN have very good results in processing one-dimensional to multidimensional signals such as text, image, audio and video. Statistically speaking, deep learning is able to make predictions on a given sample and can give the most appropriate class label based on the prediction result. Given a training set, a DNN can be well trained so that the new and similar data and samples in the test set will get the same labels as the training set.

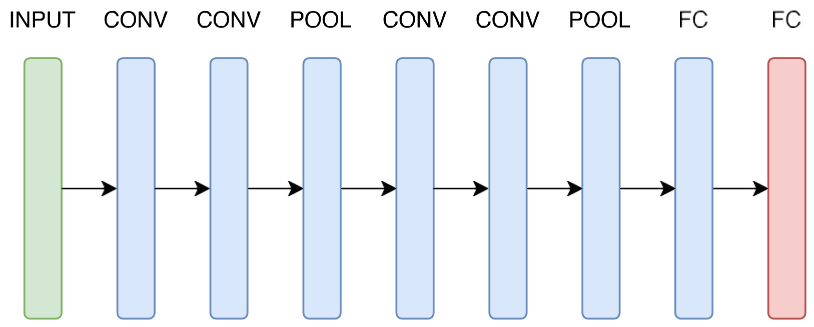
Compared to traditional machine learning methods, deep learning methods have three great advantages. First, deep learning algorithms do not need to manually extract features when training data with different labels through datasets. Second, deep learning methods have strong generalization, and the same deep learning methods can be used for different applications and different data types. Thus, they can be effectively applied to real-life classification problems. In addition, deep learning can make use of multiple GPUs to perform huge parallel computations. When the amount of data is large, it produces better output compared to traditional machine learning methods. In the following, we will introduce the basic principles of CNN and RNN in deep learning.

3.1.1 CNN

CNNs consist of three main layers, namely, convolutional layer, pooling layer and fully connected layer, denoted by CONV, POOL and FC, respectively. These three layers form a convolutional neural network that generally obeys the following pattern:

$$INPUT \rightarrow [CONV * B \rightarrow POOL?] * C \rightarrow FC * D \rightarrow FC,$$

where $INPUT$ denotes the input layer, the FC represents the output layer, and the remaining are all hidden layers. In the above pattern, symbol "$*$" denotes a repeat while the symbol "?" stands for whether an additional optional layer will be added before this symbol, and "$\rightarrow$" indicates edge between layers. B, C, D denote the number of repetitions, and $B, C, D \geq 0$ are hyperparameters which can be adjusted. In the CNN model in Figure 10, the hyperparameters are set to $B = 2$, $C = 2$, and $D = 1$ for the above pattern. The green part of the figure is the input layer, the red part is the output layer, and the blue parts are the hidden layers. Hidden layers consist of four



] Figure 1: Scheme of a convolutional neural network with convolutional layers, two pooling layers and one fully connected layer. In the following we will explain the three different types of hidden layers in detail.

Convolutional layers

The convolutional layer is composed of filters, where the role of the filter is to extract the basic features from the input vector. As a neuron that can be connected to part of the input, the number of filters is also a tunable hyperparameter. The filters in a convolutional layer (marked in blue in the figure and labeled with f) are depicted in Figure 11. The four green nodes are input vectors of size 4. The left figure shows three input nodes x_1, x_2, and x_3 to be processed with o_1 as output while the right figure shows input nodes x_2, x_3, and x_4 to be processed with o_2 as output. In both figures, ω_1, ω_2, and ω_3 are the weights of the filter, and both the weights and bias b are kept constant.

The convolution layer contains three hyperparameters, which are the sizes of the filter, span and zero padding. The first hyperparameter is the size of the filter, which defines the size of the region composed of the left and right parts as in Figure 11. The second hyperparameter is the span size, which determines the number of different nodes in the two regions. The third hyperparameter is the size of zero padding, which represents the number of zero nodes to be added to the edges of the input vector. The effect of zero nodes is to make the input vector's edges be processed for the same number of times as the rest of the input. For both figures in Fig.11, the three hyperparameters are set to 3, 1 and 0 respectively.

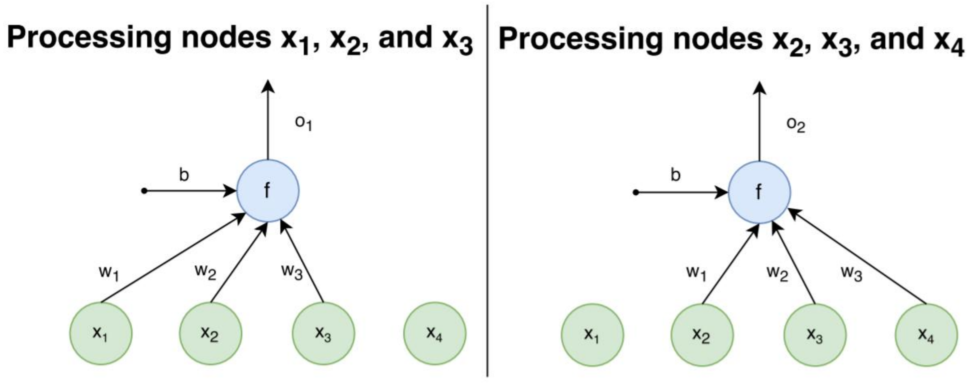

Processing nodes x_1, x_2, and x_3
Processing nodes x_2, x_3, and x_4
o_1
o_2
b
f
w_1
w_2
w_3
x_1
x_2
x_3
x_4

Figure 11: Two regions processed by a filter of size 3, zero-padding and stride of one.

Pooling layer

The pooling layer is usually behind the convolution layer, and its role is to reduce the dimensionality of the feature map formed by the output of the filter, which in turn prevents the network from overfitting and also significantly decreases the complexity of the computation in the network. Two hyperparameters control the pooling layer, namely, the size and the stride or the down sampling factor. There are usually two types of pooling layers, the average pooling layer and the maximum pooling layer. The max pooling layer outputs the maximum value of the input, while the average pooling layer applies a nonlinear function to get the average value of the input. In general, the max pooling layer gives significantly better results than the average pooling layer [36].

Fully-connected layer

The extracted and down-sampled features are forwarded to one or more fully-connected layers. In a CNN structure, one or more fully connected layers are connected after multiple convolutional and pooling layers. Each neuron in a fully connected layer is fully connected to all neurons in the layer before it. The fully connected layers can integrate the local information of the convolutional or pooling layers with category differentiation. In order to improve the performance of CNN networks, the excitation function of each neuron in the fully connected layer is usually a ReLU function. The output value of the last fully connected layer is passed to an output that can be classified using softmax function.

3.1.2 RNN

Structure of RNN

Figure 12 depicts the entire process of RNN using a simple neural network containing a hidden neuron and a one-dimensional output. In Figure 12, x is an input neuron, o is the output, and s is the hidden state. s loops through the information once and still returns to the hidden state. The weights u, v, w and bias b are parameters that need to be constantly adjusted during the learning process.

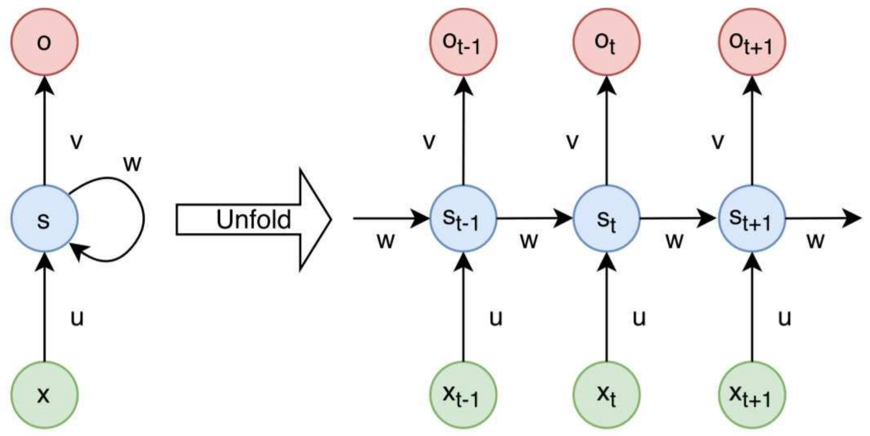


Figure 12: A simple RNN with the unfolded sequential output.

The right part of Figure 12 shows the input, output and hidden states of three different times for evaluating the network. For the step of time t, the hidden state s_t is calculated using the input x_t and the previous one hidden state s_{t-1} as

$$s_t = f(u \cdot x_t + w \cdot s_{t-1} + b),$$

where f is the activation function. The output of s_t is forwarded and used as the input of s_{t+1}. This process ensures that the RNN can always remember and process the previous information.

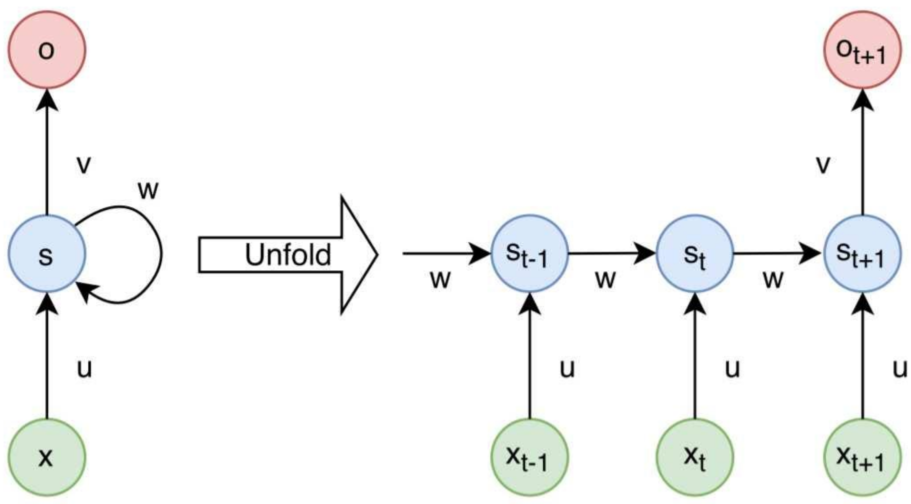


Figure 13: A simple RNN with the unfolded non-sequential output.

The RNN in Figure 12 uses sequential input and output, which is suitable for tasks like machine translation, but not good for tasks like time series classification that requires non-sequential output. Figure 13 shows a variation of the RNN with non-sequential outputs. The hidden state s_{t+1} produces an output only with the last input, while s_{t-1} and s_t don't produce any output.

Long Short-Term Memory Neural Networks (LSTM)

The difference between LSTM and the general form of RNN is in the hidden unit. Figure 14 depicts a hidden unit of the LSTM. It is composed of four parts: cell state C_t, forget gate f_t, input gate i_t and output gate h_t. The unit state contains crucial information in the sequential input which is controlled by these three gates and each gate has different weights and bias. The weights of the forget gate are u_f and w_f, and the bias of the forget gate is b_f. The weight of the input gate is u_i, w_i , u_b and w_b and the bias of the output gate b_i and b_b. The weight of the output gate is u_h and w_h, and the bias of the output gate is b_h. Compared with a common RNN, LSTM has more weights and biases to transmit and iterate during the learning process, which increases the computational complexity of the LSTM. But the advantage of LSTM is its ability to learn input data's long-term dependencies.

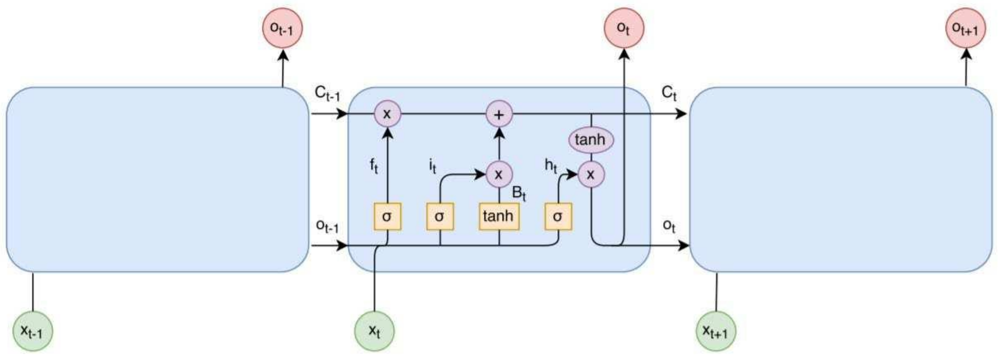


Figure 14: LSTM's hidden unit.

The function of the forget gate is to remove the previously saved information from the state of the unit, and the forget gate is defined as

$$f_t = \sigma(u_f \cdot x_t + w_f \cdot o_{t-1} + b_f)$$

where x_t is the input and o_{t-1} is the output of the previous hidden cell. The value of output of σ function is between 0 and 1. If σ is equal to 0, then all information in that cell state will be removed. If σ equals 1, all the information about the unit state will be retained.

The function of the input gate is to store the new information into the cell state.

$$i_t = \sigma(u_i \cdot x_t + w_i \cdot o_{t-1} + b_i)$$

$$B_t = \tanh(u_b \cdot x_t + w_b \cdot o_{t-1} + b_b)$$

Hence the σ function in the input gate i_t is able to select the part of the cell state that needs to be updated, while tanh in B_t determines what value will be stored and then the cell state will be updated as follows:

$$C_t = i_t \cdot B_t + f_t \cdot C_{t-1}$$

The role of the output gate h_t is to select the partial cell state that needs to be transferred to the next hidden cell, which is expressed as

$$h_t = \sigma(u_h \cdot x_t + w_h \cdot o_{t-1} + b_h)$$

The actual information will be transmitted to the next hidden cell by multiplying the output gate h_t with the cell state, i.e.,

$$o_t = \sigma(h_t \cdot \tanh(C_t))$$

In a sequential output, the same information is used as the output of the unit o_t. In this output approach, all weights and biases are shared for the entire input sequence and can be learned all the way through the learning process. When the weights and biases are tuned, the LSTM can learn the information that is contained in the input vectors and the information is used to generate an appropriate output.

Peephole Connections

Figure 15 shows a way to modify the LSTM structure by adding peephole connections [37], where the red part is the connection line. The effect of this method is to enable these three gates to make use of the information in the input state directly, without passing through layers. The gates in this method are defined as follows:

$$f_t = \sigma(u_f \cdot x_t + w_f \cdot o_{t-1} + p_f \cdot C_t + b_f)$$

$$i_t = \sigma(u_i \cdot x_t + w_i \cdot o_{t-1} + p_i \cdot C_t + b_i)$$

$$h_t = \sigma(u_h \cdot x_t + w_h \cdot o_{t-1} + p_h \cdot C_t + b_h)$$

where p_f, p_i and p_h are the weights of all connected line segments of the peephole.

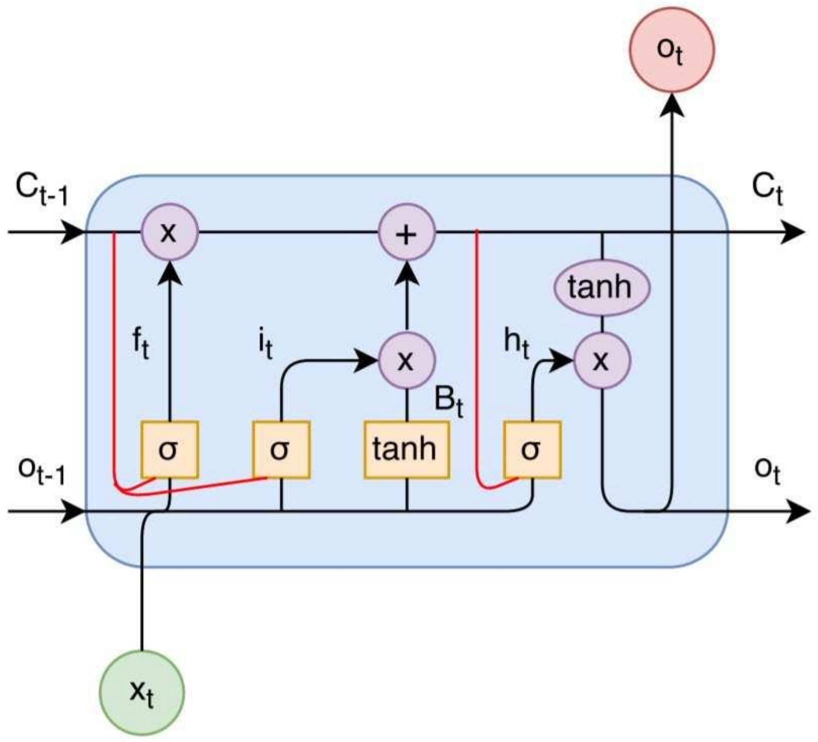


Figure 15: Adding peephole connections to the hidden units of LSTM.

3.2 Proposed keyword assignment method

3.2.1 A CNN-BiLSTM framework for keyword assignment

The general framework of the proposed keyword assignment model is shown in Figure 16. The model is mainly composed of the following parts: embedding layer, BiLSTM layer, self-attention layer, CNN layer, maximum pooling layer and output layer.

In the embedding layer, the text is output to the BiLSTM layer after being converted into word vectors by a word embedding method. In the BiLSTM layer, the features are extracted by the forward and backward sequences of each sentence separately and then concatenates the first and the last to obtain a new vector for each sentence. In the self-attention layer, different weights are assigned to each hidden vector by computing the dot product attention. In the CNN layer, the features of the matrix are extracted using convolution with proper size of the convolution kernel. Important features in the document are activated in the max pooling layer, and finally the sentence is classified using softmax function.

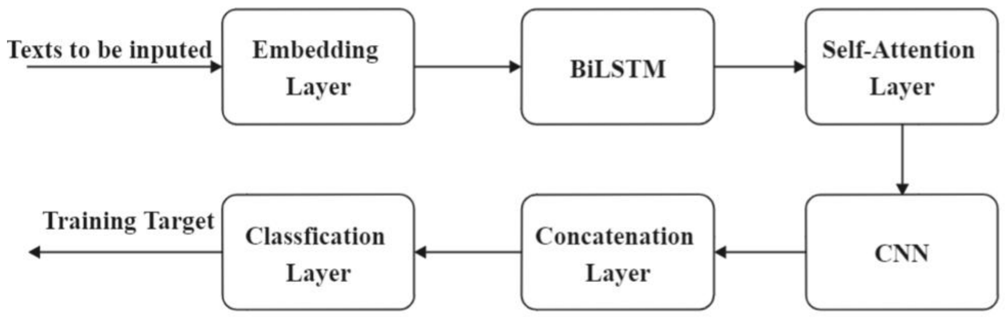


Figure 16: The proposed CNN-BiLSTM framework for keyword assignment.

The advantages of our proposed CNN-BiLSTM model are as follows: CNN is used to extract local features of text vectors, while BiLSTM is used to extract global features related to the text context. In particular, the use of CNN together with BiLSTM can overcome the problem of a single CNN model that ignores the semantic and syntactic information of the context. It can also effectively solve the problem of gradient disappearance or gradient diffusion in traditional RNNs. Furthermore, attention mechanism is introduced to highlight the important information and avoid the effect of invalid information on sentiment and text classification.

3.2.2 Detailed structure of the CNN-BiLSTM model

Embedding layer

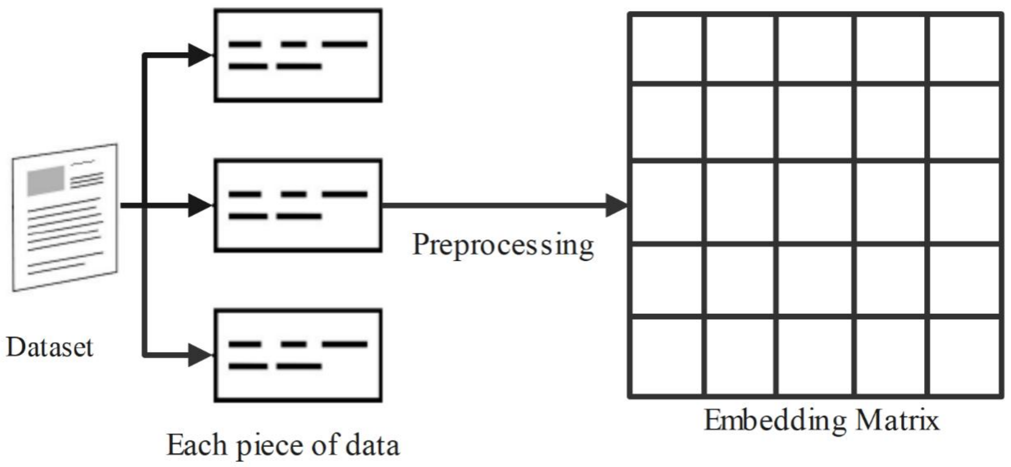


Figure 17: General architecture of Embedding layer.

The embedding layer of the model aims to transform the input text into a high-dimensional vector representation using word embedding, which is easier to compute in a neural network. If the embedding layer is not applied, each word is input as one hot form, which will increase the computational burden to a great extent. Since the length of each input sentence is different, we need to define a max_terms value. If the length of the sentence exceeds the max_terms value, we only consider the part within the max_terms value. If the length of the sentence is less than the max_terms value, then we add 0 to the end of the sentence. We set the value of max_terms to 80 words and use GloVe and fastText respectively for word embedding.

Bi-LSTM layer

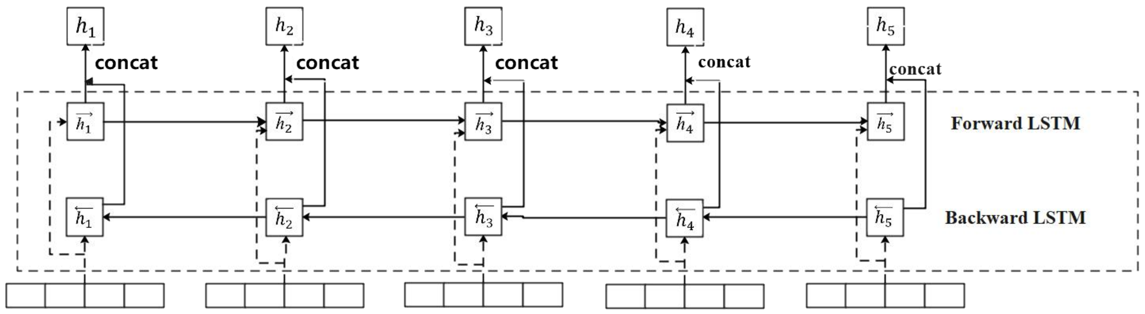


Figure 18: The structure of BiLSTM layer.

The RNN is able to take into account the sequential relationships between sentences and phrases, which means that the context of the previous phrase determines the next one. The hidden states in the LSTM update the weights of the internal loops, through the weights learned by the network from the previously input words. This step enables the layer to remember and exploit the relationship with the previous input. The BiLSTM is able to concatenate the relationships between the input elements from both directions at the same time. The BiLSTM is formulated as follows.

$$h_i = \overleftarrow{h_i} || \overrightarrow{h_i}$$

$$h_i \in R^{2d}$$

where $||$ represents the concatenation and d is the dimension of the hidden layers h of LSTM.

In our experiments, we set the hyperparameters of the LSTM layer as follows: the hidden unit value is set to 200 and the dropout value is set to 0.3. The reason for the value of hidden unit setting is to reduce the dimension of the output. The dropout value is used to reduce the overfitting during training.

In the LSTM layer, the activation function we chose is $tanh$. Figure 20 shows $tanh$ allows the output of the layer to be closer to 0 and its gradient determines a faster convergence compared to the $sigmoid$ activation function.

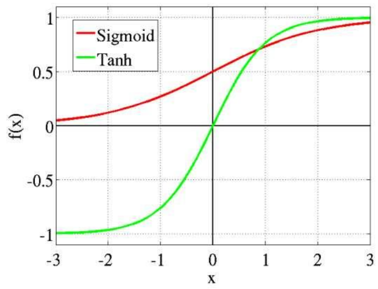


Figure 19: Sigmoid activation function and tanh activation function.

Self-attention layer

Since each word has a different level of importance to its context, we use the self-attention mechanism to assign different weights to each word so that we can better estimate the different levels of importance of each word. We add a weight c to the attention layer to represent the weight of the "context vector". The context vector c is treated as the average of h_i.

$$c = \frac{1}{N}\sum_{1}^{N} h_i$$

The annotations u_i is represented by concatenation of c and h_i :

$$u_i = h_i || c$$

e_i is obtained by using the tanh activation function on u_i, yielding

$$e_i = \tanh(W u_i + b)$$

where W and b denote the weights and biases. And attention weight a_i is calculated by using a softmax function as follows:

$$a_i = \frac{\exp{(e_i)}}{\sum_{t=1}^{N} \exp{(e_t)}}$$

The final representation r can be calculated as the weighted sum of attention weight and annotation,

$$r = \sum_{i=1}^{N} a_i h_i \, , r \in R^{2L}$$

CNN layer

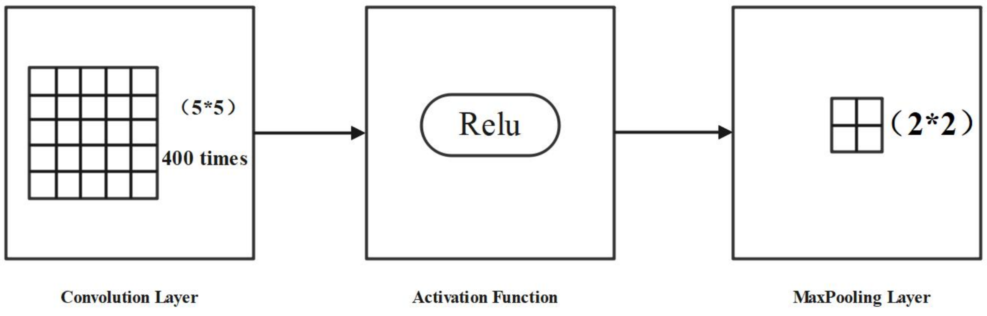


Figure 20: The structure of CNN layer.

CNN is very adaptable for processing data with grid shapes [38], where a local convolution operation is performed for adjacent cells, resulting in a denser but smaller vector, which the CNN uses to capture the hidden relationships between adjacent cells. In our experiments, the input of the CNN layer is the output vector of the attention layer, whose size of the vector is 80x400 and we apply a 1D convolutional network with 400 filters and 5x5 kernels for convolution. ReLu is used as the activation function, as shown in Figure 21, instead of tanh to speed up the computation.

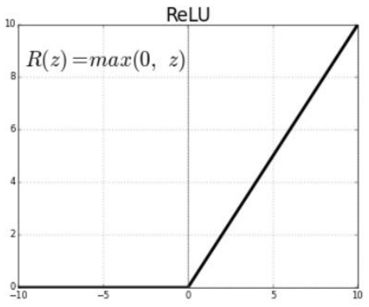


Figure 21: ReLU activation function.

In the last part of the CNN layer, a max pooling function with a 2x2 kernel is added in order to subsample the values and reduce the parameters of the model.

Classification

We use a dropout function for the output of the max pooling layer. Dropout function achieves regularization by ignoring some of the feature detectors in each training batch. After that, we use the global max pooling layer to reduce the model parameters, which is able to greatly reduce overfitting of the model. After the subsequent dense layer and the following dropout function to reduce the dimensionality of the data, the probability distribution of each category in the dataset is estimated by the softmax activation function. We trained the model 100 times using the categorical cross-entropy loss function [39] and the Adam optimizer. The best model was used for classification.

Figure 22 shows the full architecture of the CNN-BiLSTM model with self-attention. The detailed configurations of each layer in the proposed keyword assignment model based on the 20 Newsgroups dataset are listed in Table 9 as an example.

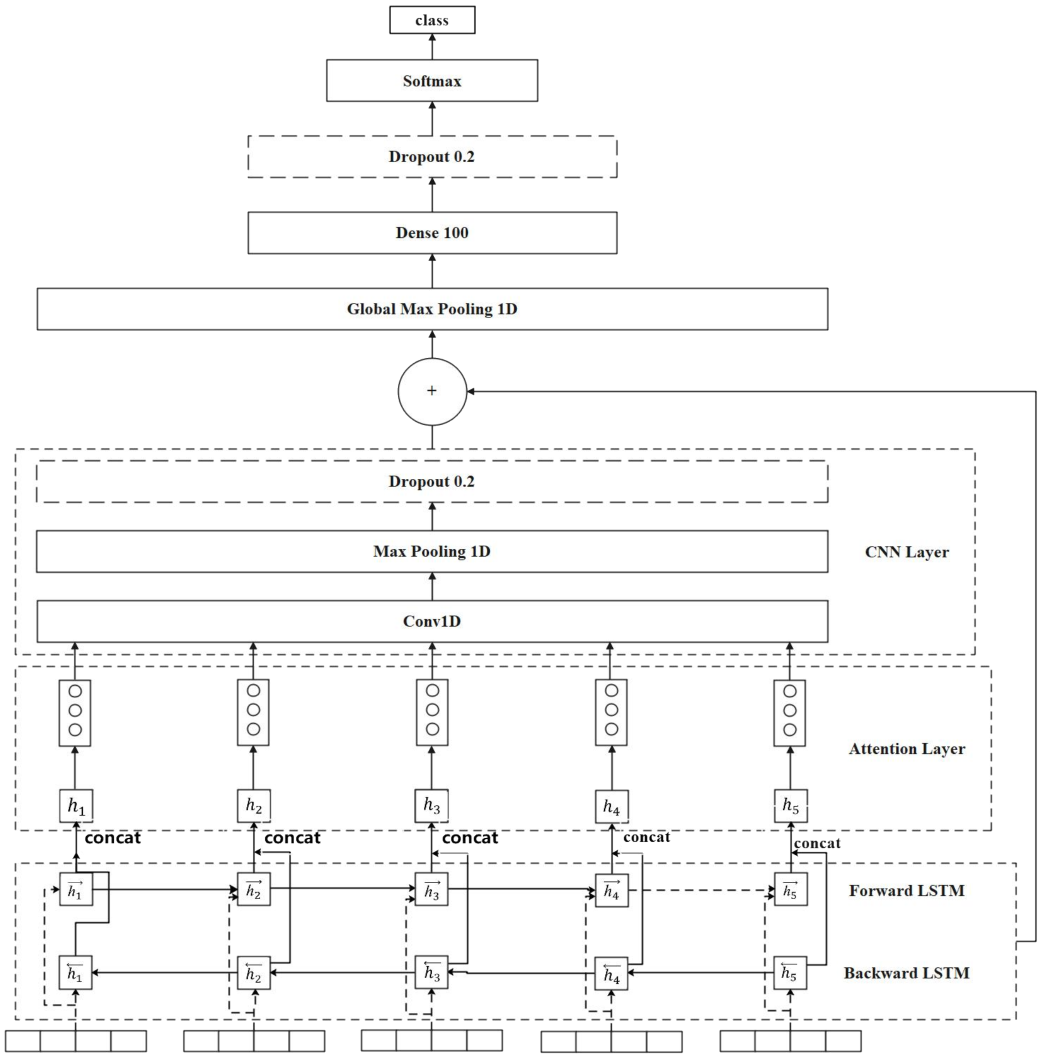


Figure 22: The architecture of the CNN-BiLSTM model with self-attention.

Table 9: Shapes of the proposed model after training on 20 Newsgroups dataset.

Layer (type)	Output Shape	Param #
Input_1 (InputLayer)	(None, 80, 300)	0

Bidirectional_1(Bidirectional)	(None, 80, 400)	801600
Seq_self_attention_1(SeqSelAttention)	(None, 80, 400)	25665
Conv1d(Conv1D)	(None, 76, 400)	800400
Max_pooling1d_1(MaxPooling1D)	(None, 38, 400)	0
Dropout_1(Dropout)	(None, 38, 400)	0
Concatenate_1(Concatenate)	(None, 118, 400)	0
Global_max_pooling1d_1 (GlobalMaxPooing1D)	(None, 400)	0
Dense_1(Dense)	(None, 100)	40100
Dropout_2(Dropout)	(None, 100)	0
Dense_2(Dense)	(None,9)	909
Total params: 1667976 Trainable params: 1667976 Non-trainable params: 0		

3.3 Experimental Results

3.3.1 Datasets and evaluation metrics for out-of-text keyword tagging

Datasets

We have used 3 following popular datasets for evaluating the performance of the proposed out-of-text keyword tagging method.

20 Newsgroups. The first dataset is the 20 Newsgroups dataset. It has 20 topics each of which has approximately 1,000 documents. We divided the 20 types of news into nine categories according to the semantics as shown in Table 10 where 70% of the documents in each class were randomly selected for training, and the remaining 30% were used as a testing set. For this dataset, the keyword is the category of each piece of news inputted [40].

IMDB. The IMDB dataset contains 50K movie reviews for natural language processing. It is divided into 3 categories, positive, negative and neutral. Among them

25K movie reviews are provided for training and the rest are for testing. For this dataset, the keyword is “positive”, “negative” or “neutral” only [41].

SemEval 2018 task-1. The training set for SemEval 2018 Task-1 consists of a dataset of 2761 different sentiment records, where these sentiments is composed of four emotions anger, fear, happiness, and sadness, with a number share of 34%, 16%, 26%, and 14%, respectively. And the test set consists of 1580 records representing these four emotions, with a number share of 25%, 15%, 36%, and 24%, respectively. Thus, the keyword is one of the four emotions, in this case anger, fear, joy or sadness [42].

Table 10: New categories and number of pieces of news in each category.

New categories	All original categories	Number of pieces of news
computer	comp.windows.x comp.sys.ibm.pc.hardware comp.os.ms-windows.misc comp.graphics comp.sys.mac.hardware	2936
science	sci.crypt sci.electronics sci.space	1779
politics	talk.politics.misc, talk.politics.guns, talk.politics.mideast	1575
sport	rec.sport.hockey rec.sport.baseball	1197
automobile	rec.autos rec.motorcycles	1192
religion	soc.religion.christian talk.religion.misc	976
medicine	sci.med	594

sales	misc.forsale	585
atheism	alt.atheism	480

Evaluation metrics

In many practical classification problems, error or accuracy are not the best evaluation metrics [43]. If the number of each classification sample is unequal, many machine learning techniques tend to ignore classes with small numbers. Therefore, we introduced confusion matrix that can effectively judge the classification results on unbalanced datasets. It summarizes the accuracy of the hypothesis with four performance metrics as follow:

• True positive (TP): The positive label correctly predicted as positive.

• True negative (TN): The negative label correctly predicted as negative.

• False positive (FP): The negative label incorrectly predicted as positive.

• False negative (FN): The positive label incorrectly predicted as negative.

These figures are demonstrated in table 11:

Table 11: Confusion Matrix of four performance metrics.

	Predicted-	Predicted+
Actual-	TN	FP
Actual+	FN	TP

In general, the performance of the model is directly proportional to the pair TN and TP and inversely proportional to FP and FN. higher values of FN indicate higher error rates for detection of positive categories, and higher values of FP indicate higher error rates for negative categories.

• Classification Accuracy:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

• Precision – Positive Predicted Value:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

• Recall - True Positive Rate:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

• F1-Score:

$$\mathrm{F1-Score} = \frac{2\mathrm{TP}}{2\mathrm{TP} + \mathrm{FP} + \mathrm{FN}}$$

Sensitivity is defined as the rate of true positives for all categorical thresholds and specificity is defined as the rate of false positives, and the horizontal and vertical coordinates of the graph of AUC are the sensitivity and specificity. And AUC is the area under the curve on the graph. Figure 23 shows the AUC at different threshold values. The effectiveness of the classifier is proportional to the value of AUC, and if the value of AUC is higher, it means that the classifier is more effective [44].

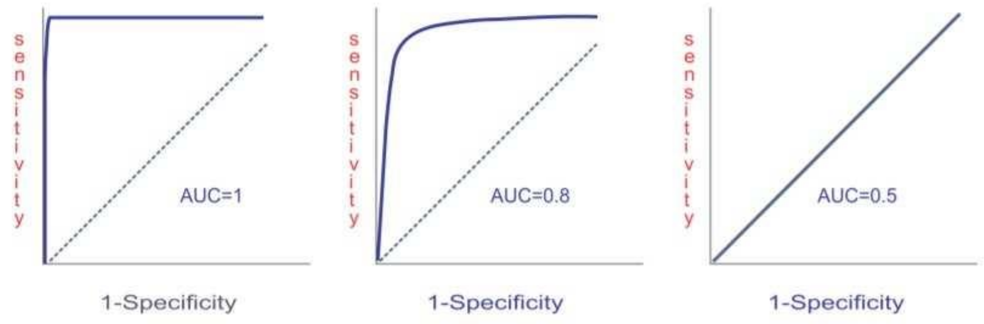


Figure 23: Higher value of AUC represents better performance of the classifier.

We used metrics Precision, Recall and F1-score along with Accuracy and AUC as evaluation metrics for each category based on the IMDB and SemEval 2018 task-1 dataset. We run the proposed model for each dataset using two word embedding methods, GloVe, and fastText, respectively. As a baseline for performance comparison, we used two methods of machine learning with SVM and random forest, where SVM uses the "rbf" kernel, the values of C and Gamma are adjusted according to the dataset, and the number of trees in the random forest is also tuned for different datasets. Also, we use CNN and LSTM based Word2Vec word embedding methods as comparison. The experimental results are provided in the next subsection.

3.3.2 Performance of the proposed model

In Table 12, we randomly select four paragraphs in the test set of 20 Newsgroups dataset, and each paragraph represents one piece of news. The list of keywords contains “computer”, “science”, “politics”, “sport”, “automobile”, “religion”, “medicine”, “sales” and "atheism" shown in Table 10 and for each paragraph, one of these keywords is assigned. The correct class in Table 12 represents the correct label for each paragraph provided in the dataset, while the predicted class represents the label predicted by our model. Three of them were correctly assigned keywords while one was assigned incorrectly.

Table 12: Four paragraphs with correct and predicted classes in the 20 Newsgroups dataset.

Four paragraphs in the test set	Correct class	Predicted class
DEC LK250-AA PC keyboard for sale: - automatically senses machine type and switches between AT/XT modes - same exact key layout as DEC's VT2xx, VT3xx, etc., with DEC names on keys as well as PC names - standard AT/XT cable included	sales	sales
First, a longer game in no way suggests "more baseball to watch," unless you include watching the grass grow as baseball. The lengthier games are so because of batters stepping out of the box, pitchers taking longer between pitches and excessive trips to the mound by managers and pitching coaches.	sport	sport
I'm considering switching to Geico insurance, but have heard that they do not assign a specific agent for each policy or claim. I was worried that this might be a real pain when you make a claim. I have also heard that they try to get rid of you if you have an accident. I've read in this group that Geico has funded the purchasing of radar guns by police depts (I'm not sure where).	automobile	politics

"... Do not be deceived; neither fornicators, nor idolators, nor adulterers, nor effeminate, nor homosexuals, nor thieves, nor the covetous, nor drunkards, nor revilers, nor swindlers, shall inherit the kingdom of God. And such were some of you..." I Cor. 6:9-11.	Religion	Religion

Table 13: Classification results obtained from different methods based on the 20 Newsgroups dataset.

	20 Newsgroups dataset		
	Precision	Recall	F1-Score
SVM	0.63	0.65	0.64
Random Forest	0.79	0.81	0.80
CNN with Word2Vec	0.76	0.80	0.78
LSTM with Word2Vec	0.71	0.68	0.69
CNN-BiLSTM using GloVe and self-attention	0.89	0.93	0.91
CNN-BiLSTM using fastText and self-attention	**0.92**	**0.95**	**0.94**

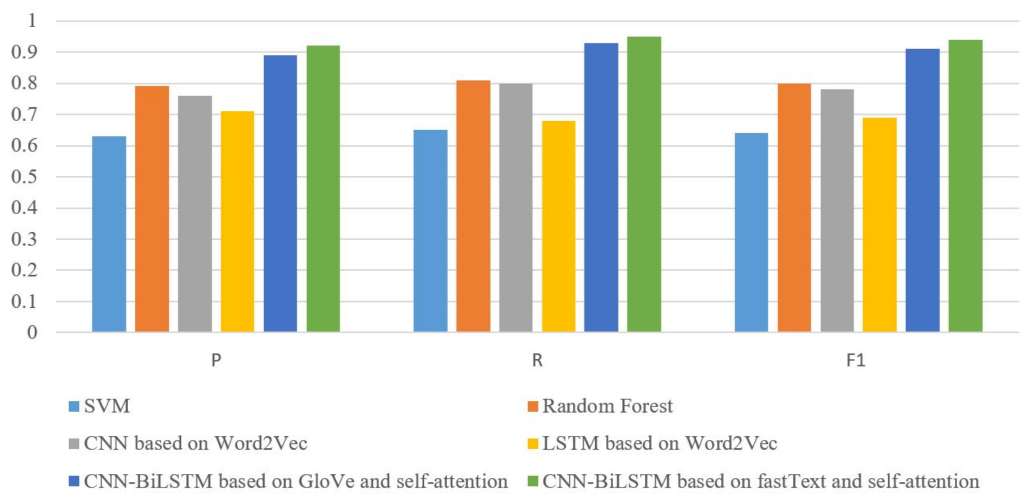


Figure 24: Classification score obtained from different methods based on the 20 Newsgroups dataset.

We will describe how precision, recall and F1-Score are calculated in this dataset and take the class "computer" as an example.

$$Pricision_{Computer} = \frac{The\ number\ of\ paragraphs\ predicted\ to\ be\ computer\ with\ their\ actual\ keyword\ is\ computer}{The\ number\ of\ paragraphs\ predicted\ to\ be\ computer}$$

$$Recall_{Computer} = \frac{The\ number\ of\ paragraphs\ predicted\ to\ be\ computer\ with\ their\ actual\ keyword\ is\ computer}{The\ number\ of\ paragraphs\ with\ their\ actual\ keyword\ is\ computer}$$

$$Weight_{Computer} = \frac{The\ number\ of\ paragraphs\ \ with\ their\ actual\ keyword\ is\ computer}{The\ number\ of\ all\ the\ paragraphs}$$

$$\begin{aligned} P = {} & Pricision_{Computer} \cdot Weight_{Computer} + Pricision_{science} \cdot Weight_{science} + Pricision_{politics} \\ & \cdot Weight_{politics} + Pricision_{sport} \cdot Weight_{sport} + Pricision_{automobile} \\ & \cdot Weight_{automobile} + Pricision_{religion} \cdot Weight_{religion} + Pricision_{medicine} \\ & \cdot Weight_{medicine} + Pricision_{sales} \cdot Weight_{sales} + Pricision_{atheism} \\ & \cdot Weight_{atheism} \end{aligned}$$

$$\begin{aligned} R = {} & Recall_{Computer} \cdot Weight_{Computer} + Recall_{science} \cdot Weight_{science} + Recall_{politics} \\ & \cdot Weight_{politics} + Recall_{sport} \cdot Weight_{sport} + Recall_{automobile} \cdot Weight_{automobile} \\ & + Recall_{religion} \cdot Weight_{religion} + Recall_{medicine} \cdot Weight_{medicine} + Recall_{sales} \\ & \cdot Weight_{sales} + Recall_{atheism} \cdot Weight_{atheism} \end{aligned}$$

$$F1 = \frac{2 \cdot P \cdot R}{P + R}$$

In Table 13 and Figure 24, we have given classification scores obtained from different methods including our proposed CNN-BiLSTM with self-attention model using the fastText word embedding. Clearly, our method achieves the maximum value in both the precision (0.92), recall (0.95) and F1-Score (0.94), and therefore is more effective for keyword assignment compared with traditional machine learning methods and neural networks proposed by others.

Table 14: Classification results obtained from different methods based on the IMDB dataset.

	Positive	Negative	Neutral

	P	R	F1	P	R	F1	P	R	F1
SVM	0.87	0.93	0.90	0.81	0.88	0.84	0.73	0.75	0.74
Random Forest	0.85	0.88	0.87	0.77	0.82	0.80	0.64	0.77	0.71
CNN with Word2Vec	0.80	0.81	0.81	0.76	0.72	0.74	0.77	0.78	0.78
LSTM with Word2Vec	0.68	0.75	0.71	0.68	0.61	0.65	0.61	0.69	0.65
CNN-BiLSTM using GloVe and self-attention	0.91	0.96	0.93	0.87	0.87	**0.87**	0.82	0.82	0.82
CNN-BiLSTM using fastText and self-attention	**0.93**	**0.96**	**0.95**	**0.83**	**0.88**	0.86	**0.84**	**0.84**	**0.84**

	F1	Accuracy	AUC
SVM	0.83	0.803	0.940
Random Forest	0.80	0.753	0.844
CNN with Word2Vec	0.78	0.777	0.835
LSTM with Word2Vec	0.67	0.657	0.750
CNN-BiLSTM using GloVe and self-attention	0.87	0.867	0.949
CNN-BiLSTM using fastText and self-attention	0.88	0.867	0.964

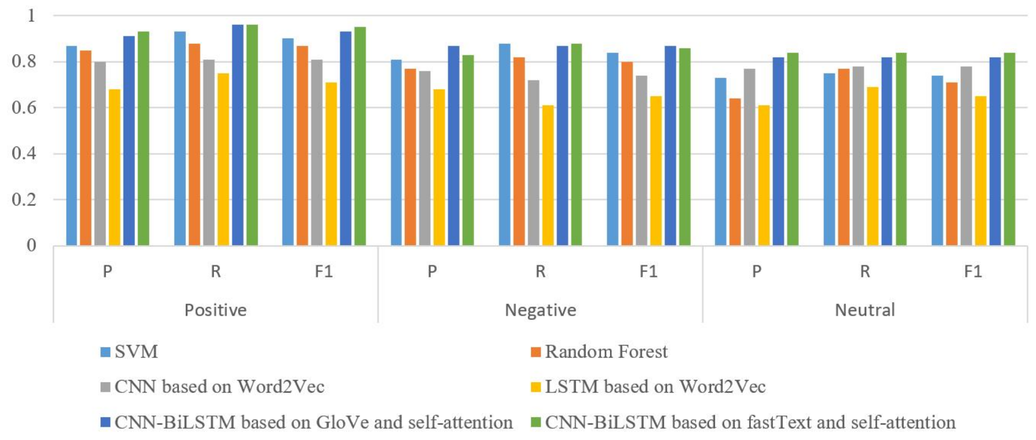


Figure 25: Classification score obtained from different methods based on the IMDB dataset.

Table 14 shows the classification results of our proposed CNN-BiLSTM model with self-attention along with those from previous machine learning methods. Those scores are also plotted in Fig. 25. It is seen that our method using the fastText word embedding method achieves maximum values for all the precision, recall and F1-Score in each class, except for recall of "Negative" using GloVe word embedding method. For the total dataset, the F1-Score of our proposed model is 0.88 while the baseline of SVM is 0.83. Our proposed method also exceeds the SVM baseline in Accuracy and AUC. It is also clear that our method is more effective for keyword assignment compared to traditional machine learning and neural networks methods.

Table 15: Classification results obtained from different methods based on the SemEval 2018 task-1 dataset.

	anger			fear			joy			sadness		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
SVM	**0.79**	0.68	0.73	0.59	**0.71**	0.64	0.91	0.97	0.94	0.68	0.66	0.67
Random Forest	0.54	0.67	0.59	0.62	0.41	0.50	0.74	0.88	0.80	0.58	0.51	0.54
CNN with Word2Vec	0.70	0.68	0.69	0.68	0.64	0.66	0.88	0.91	0.89	0.63	0.58	0.60
LSTM with Word2Vec	0.65	0.70	0.68	0.58	0.63	0.60	0.84	0.83	0.83	0.61	0.58	0.59

CNN-BiLSTM using GloVe and self-attention	0.77	0.80	0.78	0.75	0.69	0.72	0.97	0.94	0.96	0.68	0.72	0.70
CNN-BiLSTM using fastText and self-attention	0.77	**0.83**	**0.80**	**0.77**	0.70	**0.73**	**0.97**	**0.97**	**0.97**	**0.74**	**0.74**	**0.74**

	F1	Accuracy	AUC
SVM	0.78	0.778	0.835
Random Forest	0.71	0.710	0.773
CNN with Word2Vec	0.77	0.762	0.819
LSTM with Word2Vec	0.75	0.739	0.796
CNN-BiLSTM using GloVe and self-attention	0.82	0.818	0.940
CNN-BiLSTM using fastText and self-attention	**0.84**	**0.836**	**0.946**

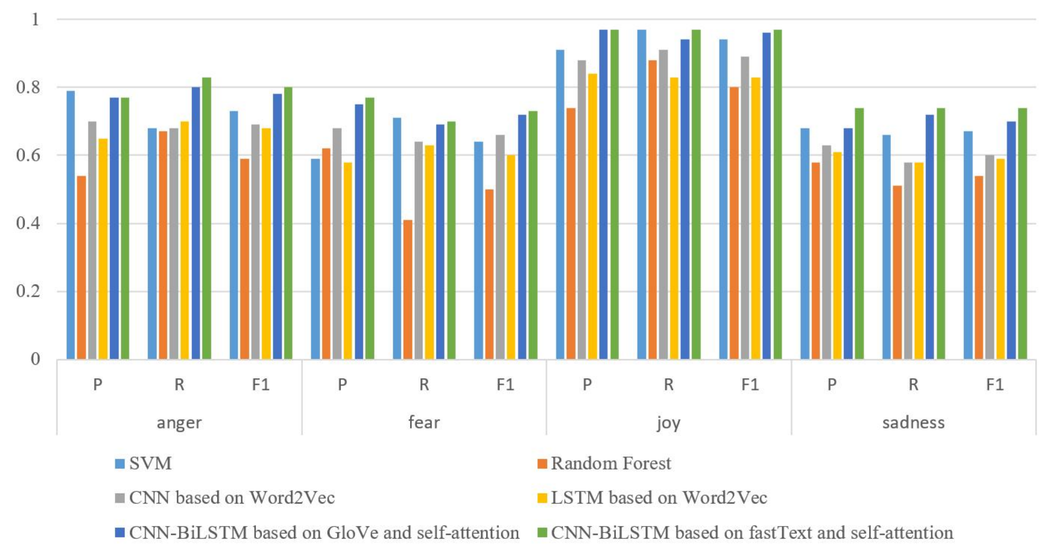


Figure 26: Classification score obtained from different methods based on the SemEval 2018-task 1 dataset.

Table 15 shows the classification results of our proposed CNN-BiLSTM model with self-attention along with those from previous machine learning methods. Those scores are also plotted in Fig. 26. It is seen that our method using the fastText word embedding method achieves maximum values for all the precision, recall and F1-score in each class, except for precision of "Anger" and recall of "Fear". For the total dataset, the F1-Score of our proposed model is 0.84 while the baseline of SVM is 0.78. Our proposed method also exceeds the SVM baseline in Accuracy and AUC.

Our proposed method achieves leading results in all three datasets, indicating the wide applicability of our method to various datasets.

3.4 Summary

In this chapter, we have first introduced the background of deep learning including the basic principles of CNN and RNN. Based on some existing methods for keyword assignment, we proposed a CNN-BiLSTM model using fastText word embedding and self-attention mechanism. This model solves the problem of a single CNN model that ignores the semantic and syntactic information of the context, and effectively overcomes the problem of gradient disappearance or gradient diffusion in traditional RNNs. A self-attention mechanism is also introduced to highlight the important information and avoid the influence of invalid information on the text sentiment and classification.

Our experimental results based on 20 Newsgroups, IMDB, SemEval 2018 task-1 datasets show that the results obtained by our proposed method are better than those obtained by previous machine learning and deep neural network methods in terms of F1-Score, Accuracy and AUC, indicating the wide applicability of our model to various datasets.

Chapter 4

Conclusion and Future Work

4.1 Summary of the work

In this thesis, automatic keyword tagging has been thoroughly studied along its two main branches: keyword extraction for in-text tagging and keyword assignment for out-of-text tagging. An unsupervised learning method integrating fastText word embedding method and Affinity Propagation clustering is shown to provide a superior performance for keyword extraction while a CNN-BiLSTM deep learning model with word embedding and attention mechanism is verified to have better accuracy, recall, F1-Score and AUC than some other state-of-the-art machine learning and deep learning methods.

Chapter 2 presents an unsupervised automatic keyword extraction method integrating word embedding method and clustering. We have investigated two word embedding methods, GloVe and fastText, in comparison with the previously reviewed Word2Vec on different datasets in order to identify the most suitable word embedding scheme for keyword extraction. The word embedding scheme, fastText is then selected to combine with three clustering methods (Affinity Propagation, Mean Shift and K-means) to extract the keywords, where the cluster centers are the keywords to be extracted. The obtained results are compared with the gold standard in terms of the accuracy, recall and F1-score, indicating that our method has a better performance compared to several widely used methods. We have shown that in the whole process, the word embedding method can obtain the semantics of the context while the clustering algorithm can identify the essence of the term, and then select the important ones that can better represent the text content.

In Chapter 3, in order to improve the accuracy of keyword assignment, we have proposed a CNN-BiLSTM model using word embedding and attention mechanism. In the word embedding layer, we investigated the two word embedding methods GloVe and fastText respectively. In the self-attention layer, different weights are assigned to each hidden vector to highlight the important information and ignore the invalid information. In the CNN layer, the features of the matrix are extracted using convolutions with appropriate the size of the convolution kernel. The most important features of the document are activated at the max pooling layer and finally the text is classified by the softmax output layer.

4.2 Suggestions for future work

For the keyword extraction using word embedding in conjunction with clustering methods, further work could be done in the following directions:

Dataset: Scale up by using a larger corpus of documents, for example, a genre different from scholarly writing. This may cause the clustering algorithms to behave differently from what was seen in this thesis. Therefore, further research should be conducted with large and different datasets for different applications. It could be interesting to train a model for GloVe that predicts embeddings for OOV words and compare the results to fastText embeddings. This would require experimental work with different sets of training data and hyperparameters to find the optimal combination.

Clustering algorithms: It is of interest to investigate if there is a way to extract centroids from results returned by hierarchical clustering. Applying clustering algorithms to similarity matrices with comparison to their direct application to word embeddings is also interesting since in scikit-learn clustering, only Affinity Propagation, Spectral Clustering, and DBSCAN may accept similarity matrices as input.

Evaluation metric: It is valuable to develop a more robust metric for evaluating the similarity between two lists of strings or for comparing more keyword extraction methods in literature.

Language expansion: The keyword extraction considered in this thesis is limited to English. As such, the performance and conclusion obtained may not be applicable to other languages. Therefore, studying the proposed keyword extraction methods for other languages is another option.

With respect to the keyword assignment using deep neural network model with word embedding and attention mechanism, the following work could be considered.

Model optimization: Since the transformer model has led to a huge improvement in the results of many natural language processing tasks, the self-attention can be replaced with multi-headed attention in the encoder-based attention layer to compare the performance results of our proposed model. As the latest bi-directional transformer-based BERT word embedding uses positional encoding in pre-training, we will also investigate the possibility of replacing BiLSTM with BERT in order to reduce the number of model parameters and the complexity.

Computing resources: Since deep learning is a relatively young field, most frameworks do not support mixed use of CPU and GPU for training. Due to this resource-limitation in nature, we ran into problems when we had deepened our model to a certain extent. One of the models we tried was actually fully trained on the CPU which took a long time. We hope to explore the full capacity of deep neural networks when powerful computing machines are available.

Vocabulary expansion: It is worth expanding the vocabulary to include all the words in the word embedding vector instead of the words that appear in the training set. This will allow us to detect the type of any text corpus by dividing it into logical parts and evaluating each part and use the normalized classification as the final genre.

Bibliography

[1] Liu, Zhiyuan and Maosong Sun. “Asymmetrical query recommendation method based on bipartite network resource allocation.” WWW, 2008.

[2] Manning C D, Schü tze H. Foundations of statistical natural language processing. Cambridge, MA, USA: MIT Press, 1999.

[3] Mari-Sanna Paukkeri, lari T Nieminen, Matti Pollä, and Timo Honkela. A language-independent approach to keyphrase extraction and evaluation. In COL ING (Posters), pages 83 86, 2008.

[4] Rose, Stuart & Engel, Dave & Cramer, Nick & Cowley, Wendy. (2010). Automatic Keyword Extraction from Individual Documents. 10.1002/9780470689646.ch1.

[5] L. Zhang, J. Li and C. Wang, "Automatic synonym extraction using Word2Vec and spectral clustering," *2017 36th Chinese Control Conference (CCC)*, 2017, pp. 5629-5632, doi: 10.23919/ChiCC.2017.8028251.

[6] Mihalcea R, Tarau P. TextRank: Bringing Order into Texts. Proceedings of EMNLP, 2004.404–411.

[7] Xiaojun Wan and Jianguo Xiao. 2008. Single document keyphrase extraction using neighborhood knowledge. In AAAI, volume 8, pages 855–860.

[8] Wang J., Zhou Y., Li L., Hu B., Hu X. (2009) Improving Short Text Clustering Performance with Keyword Expansion. In: Wang H., Shen Y., Huang T., Zeng Z. (eds) The Sixth International Symposium on Neural Networks (ISNN 2009). Advances in Intelligent and Soft Computing, vol 56. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-01216-7_31

[9] S. Ousia, “Pretrained embeddings,” Wikipedia2Vec. [Online]. Available: https://wikipedia2vec.github.io/wikipedia2vec/pretrained/. [Accessed: 27-Dec-

2021].

[10] I. Rish, “An Empirical Study of the Naïve Bayes Classifier,” Cc. Gatech.Edu, no. January 2001, pp. 41–46, 2014.

[11] Cortes, C. & Vapnik, V., Support-vector networks. Machine learning, 20(3), pp.273–297, 1995.

[12] John Lafferty, Andrew McCallum, and Fernando C.N. Pereira, "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data", June 2001.

[13] Jo, Taemin & Lee, Jee-Hyong. (2015). Latent Keyphrase Extraction Using Deep Belief Networks. The International Journal of Fuzzy Logic and Intelligent Systems. 15. 153-158. 10.5391/IJFIS.2015.15.3.153.

[14] Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. Deep keyphrase generation. arXiv preprint arXiv: 1704.06879, 2017.

[15] Kyunghyun Cho, Bart Van Merriènhoer, Caglar Gulcehre, Damitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshla Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv1406.1078, 2014.

[16] Jiatao G, Zhengdong Lu, Hang Li, and Victor OK Li. Incorporating copying mechanism in sequence-to-sequence learning. arXiv preprint arXiv:1603.06393, 2016.

[17] M. M. Haider, M. A. Hossin, H. R. Mahi and H. Arif, "Automatic Text Summarization Using Gensim Word2Vec and K-Means Clustering Algorithm," 2020 IEEE Region 10 Symposium (TENSYMP), 2020, pp. 283-286, doi: 10.1109/TENSYMP50017.2020.9230670.

[18] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," arXiv preprint arXiv:1301.3781, 2013.

[19] Elena Tutubalina and Sergey Nikolenko. “Demographic Prediction

Based on User Reviews about Medications”. In: Computacion y Sistemas 21 (June 2017), pp. 227–241. DOI: 10.13053/CyS-21-2-2736.

[20] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. “Distributed representations of words and phrases and their compositionality”. In: Advances in neural information processing systems. 2013, pp. 3111–3119.

[21] Frederic Morin and Yoshua Bengio. “Hierarchical probabilistic neural network language model.” In: Aistats. Vol. 5. Citeseer. 2005, pp. 246–252.

[22] D. L. Yse. A complete guide to K-means clustering algorithm, (2019, May), https://www.kdnuggets.com/2019/05/guide-k-means clusteringalgorithm.html

[23] Loper, E.; Bird, S. NLTK: The Natural Language Toolkit. arXiv 2002, arXiv:cs/0205028.

[24] Nguyen, T. D., & Kan, M.-Y. (2007). Keyphrase extraction in scientific publications. In Proceedings of the 10th international conference on Asian digital libraries, ICADL 2007, Hanoi, Vietnam, December 10-13, 2007 (Vol. 4822, p. 317-326). Springer.

[25] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global Vectors for Word Representation," in EMNLP, 2014, vol. 14, pp. 1532-43

[26] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. arXiv preprint arXiv:1607.04606, 2016.

[27] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759, 2016.

[28] B. J. Frey and D. Dueck, “Clustering by passing messages between data points,” Science, vol. 315, no. 5814, pp. 972–976, 2007.

[29] D. Comaniciu and P. Meer, "Mean shift: a robust approach toward

feature space analysis", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 5, pp. 603-619, 2002. Available: 10.1109/34.1000236.

[30] Hulth A. Improved Automatic Keyword Extraction Given More Linguistic Knowledge. Proceedings of EMNLP, 2003. 216–223.

[31] Krapivin, Mikalai, Aliaksandr Autaeu, and Maurizio Marchese. "Large dataset for keyphrases extraction.", 2009

[32] Nguyen, T. D., & Kan, M.-Y. (2007). Keyphrase extraction in scientific publications. In Proceedings of the 10th international conference on Asian digital libraries, ICADL 2007, Hanoi, Vietnam, December 10-13, 2007 (Vol. 4822, p. 317-326). Springer.

[33] "English word vectors · fasttext," fastText. [Online]. Available: https://fasttext.cc/docs/en/english-vectors.html. [Accessed: 27-Dec-2021].

[34] Carreira-Perpiñán, Miguel. (2015). A review of mean-shift algorithms for clustering.

[35] Hennig, C., Meila, M., Murtagh, F., & Rocci, R. (Eds.). (2015). Handbook of Cluster Analysis (1st ed.). Chapman and Hall/CRC. https://doi.org/10.1201/b19706

[36] Scherer, D.; Muller, A.; et al. Evaluation of pooling operations in convolutional architectures for object recognition. In International Conference on Artificial Neural Networks, Springer, 2010, pp. 92–101.

[37] Gers, F. A.; Schmidhuber, J.; et al. Learning to forget: Continual prediction with LSTM. Neural computation, volume 12, no. 10, 2000: pp. 2451–2471.

[38] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. arXiv preprint arXiv:1404.2188 (2014).

[39] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. 2016. Deep learning. Vol. 1. MIT press Cambridge.

[40] “20 newsgroups,” Home Page for 20 Newsgroups Data Set. [Online]. Available: http://qwone.com/~jason/20Newsgroups/. [Accessed: 27-Dec-2021].

[41] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. (2011). Learning Word Vectors for Sentiment Analysis. The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011).

[42] Semeval-2018 Task 1: Affect in Tweets. Saif M. Mohammad, Felipe Bravo-Marquez, Mohammad Salameh, and Svetlana Kiritchenko. In Proceedings of the International Workshop on Semantic Evaluation (SemEval-2018), New Orleans, LA, USA, June 2018.

[43] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. arXiv preprint arXiv:1404.2188 (2014).

[44] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. 2016. Deep learning. Vol. 1. MIT press Cambridge.