Word embedding-based techniques for text clustering and topic modelling

With application in the healthcare domain

Sattar Seifollahi

Principal Supervisor: *Prof. Massimo Piccardi*

A thesis presented for the degree of

Doctor of Philosophy

School of Electrical and Data Engineering

University of Technology Sydney

September, 2019

Word embedding-based techniques for text clustering and topic modelling

Sattar Seifollahi

Abstract

In the field of text analytics, document clustering and topic modelling are two widely-used tools for many applications. Document clustering aims to automatically organize similar documents into groups, which is crucial for document organization, browsing, summarization, classification and retrieval. Topic modelling refers to unsupervised models that automatically discover the main topics of a collection of documents. In topic modelling, the topics are simply represented as probability distributions over the words in the collection (the different probabilities reveal what topic is at stake). In turn, each document is represented as a distribution over the topics. Such distributions can also be seen as low-dimensional representations of the documents that can be used for information retrieval, document summarization and classification. Document clustering and topic modelling are highly correlated and can mutually benefit from each other.

Many document clustering algorithms exist, including the classic *k*-means. In this thesis, we have developed three new algorithms: 1) a maximum-margin clustering approach which was originally proposed for general data, but can also suit text clustering, 2) a modified global *k*-means algorithm for text clustering which is able to improve the local minima and find a deeper local solution for clustering document collections in a limited amount of time, and 3) a taxonomy-augmented algorithm which addresses two main drawbacks of the so-called "bag-of-words" (BoW) models, namely, the curse of dimensionality and the dismissal of word ordering. Our main emphasis is on high accuracy and effectiveness within the bounds of limited memory consumption.

Although great effort has been devoted to topic modelling to date, a limitation of many topic models such as latent Dirichlet allocation is that they do not take the words' relations explicitly into account. Our contribution has been two-fold. We have developed a topic

model which captures how words are topically related. The model is presented as a semi-supervised Markov chain topic model in which topics are assigned to individual words based on how each word is topically connected to the previous one in the collection. We have combined topic modelling and clustering to propose a new algorithm that benefits from both.

This research was industry-driven, focusing on projects from the Transport Accident Commission (TAC), a major accident compensation agency of the Victorian Government in Australia. It has received full ethics approval from the UTS Human Research Ethics Committee. The results presented in this thesis do not allow reidentifying any person involved in the services.

Dedication

To my family, Sona and Selin, and my parents.

Declaration

CERTIFICATE OF ORIGINAL AUTHORSHIP

I, Sattar Seifollahi declare that this thesis, is submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in the School of Electrical and Data Engineering at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise reference or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

Production Note:
Signature: Signature removed prior to publication.

Date: 20/1/2020

Acknowledgements

I would like to acknowledge my great appreciation for my principal supervisor Professor Massimo Piccardi for providing me with the necessary guidance throughout my PhD research. The valuable comments and suggestions provided by him were immensely helpful to me in successfully writing the journal and conference papers on which this thesis is based. It has been an honor and a pleasure working with him. Thanks to my associate supervisors, Associate Professor Adil Bagirov, Associate Professor Federico Girosi and Dr Ehsan Zare Borzeshi for supporting me for this thesis and for helpful discussions. I would like also to thank the Transport Accident Commission (TAC) managers, an accident compensation agency of the Victorian Government in Australia, for their support and for providing the data, and the Capital Markets Cooperative Research Centre (CMCRC) and the School of Electrical and Data Engineering of University of Technology Sydney for providing great support and financial assistance throughout my candidature. This research has received ethics approval from UTS (UTS HREC REF NO. ETH16-0968). Finally, I would like to give special thanks to my family members, Sona and Selin, for their great support and giving me huge energy to work on my thesis, and thanks to all the friends and other staff members of UTS and Federation University Australia who have helped me in many different ways during my PhD study.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the rapid growth of the Internet and the World Wide Web in the recent years, availability of text data has extensively increased. As more information becomes available, it becomes increasingly difficult to retrieve what we are looking for. Therefore, the need for tools and techniques to organize, search and understand vast quantities of information is becoming more urgent. Document clustering and topic modelling are two of the most important text mining techniques, which provide us with ways to organize, understand and summarize large collections of textual information. Text-based document clustering groups a collection of documents based on their similarity. On the other hand, topic modelling can be described as an approach for determining sets of words (i.e., topics) from a collection of documents that best describe the information embedded in the collection. It can also be thought of as a form of document clustering – a way to obtain recurring patterns of words in textual material.

1.1 Motivation

The recent years have witnessed an incessant growth in the creation of digital text, from the increasing number of organizational documents and workflows to the large amounts of messages continuously generated on social media. As an example, the number of tweets generated on the popular Twitter platform is estimated to have reached over 200 billion per year. The immediate challenge stemming from such a huge growth in textual data is how to understand their contents in effective and efficient ways.

Given the increasing size of document collections, it is vital to manage them into structured forms. Cluster analysis and classification are likely the most widespread automated tasks on textual data and play important roles in organizing such a huge amount of documents into meaningful clusters/classes. In clustering, a cluster is a collection of data objects (can be both words and documents here) that are similar to one another within the same cluster and dissimilar from those in other clusters.

In recent years, topic models, which are hierarchical Bayesian models of discrete data, have been widely used for exploratory and predictive analysis of texts. They provide us with methods to organize, understand and summarize large collections of textual information. Topic models such as the latent Dirichlet allocation (LDA) [Blei *et al.*2003], posit that topics can be used to explain the observed collection. More precisely, they take as input a collection of documents and infer the set of their "topics". Each document is represented as a discrete distribution over the topics, where each topic is, at its turn, a discrete distribution over the vocabulary. The model assumes that each word in the document is generated (in the sense of generative probabilistic models) by one of the topics. Topic models help in discovering hidden topical patterns that are present across the collection, annotating documents according to these topics, and using these annotations to organize, search and summarize texts.

A major issue for effective document clustering and classification is the extraction of appropriate features and document representations. Techniques using the bag-of-words (BoW) model are the most widespread, [Arthur and Vassilvitskii2007], with an early reference in a linguistic context dating back to 1954 [Harris1954]. This model is nothing more than a normalized count, or empirical frequencies, of the unique words in a given text.

Most of the current topic models are extensions of the LDA such as the Pachinko allocation, [Wei *et al.*2007] which introduces topics' correlations further to the word correlations which form topics. Examples of other extensions include, but are not limited to, hierarchical formulations to produce an unknown number of topics [Teh *et al.*2006], topics evolution over time [Blei and Lafferty2006, Xuerui and McCallum2006, Wang *et al.*2008], topic correlation using the logistic normal distribution [Blei and Lafferty2007], topics that follow a Markov chain and change over sentences [Gruber *et al.*2007] and

topics employing prior knowledge [Chen and Liu2014].

Although the LDA model and many such models produce highly accurate topics, there are fewer research works investigating words' relations embedded in topics. For example, LDA and many such related models do not take subsequent words relations into account. In [Gruber *et al.*2007], the topics of the sentences follow a Markov chain. They assume that topic changes occur only between sentences, and that the topic of each sentence is decided at the beginning of each sentence. However considering such a restrictive assumption may lead to poor, and possibly incoherent, topic prediction in collections where the vocabulary is complex. Therefore, one of the contributions of this thesis is a semi-supervised hidden Markov chain using prior knowledge to find coherent topics which is in better agreement with human-judged topic coherence.

1.2 Thesis objectives

A founding aspect of this PhD research is that it has been performed in close collaboration with an industry partner, the Transport Accident Commission (TAC) of the Victorian Government. The TAC is a large accident compensation agency that possesses a huge amount of unstructured text data. Using unstructured data in conjunction with structured data can provide a deeper understanding of the clients' needs and help plan for better health outcomes, such as *return to work* (RTW). Transforming unstructured data to structured form and using the structured data models to improve the quality of existing models is another valuable contribution of this thesis.

This research has received ethics approval (UTS HREC REF NO. ETH16-0968) for the research work with the industry partner, "Unstructured text analytics for the Transport Accident Commission". This application has been granted by the UTS Human Research Ethics Committee on 28 April 2017. The application allows us to work on the TAC's deidentified data and publish/present the outcomes of the research in national and international journals and conferences.

I have developed and completed several projects with significant and promising results for the industry partner. The unstructured data consisted mainly of phone calls between the clients and the TAC claim managers, regarding clients' issues such as diverse as RTW, health recovery and engagement of external solicitors (to sue the TAC). Accordingly, the

ultimate goal of this PhD thesis has been to develop original research outcomes while fulfilling the industry partner's expectations for accuracy, using high-level unstructured data analysis tools with main focus on topic modelling and concept taxonomies. We have developed two new topic models: 1) a semi-supervised hidden Markov topic model (SHMTM) and 2) a clustering-based topic model. These models have been applied to the TAC data to improve over the quality of existing models.

Reducing the number of features in document clustering and classification has been another aspect of this thesis. The spontaneous number of features in text analytics is typically excessive, and most clustering techniques struggle to effectively work on such high-dimensional data. For example, given $100,000$ documents and a $20,000$-word vocabulary, it is very difficult to use conventional clustering or classification techniques as such huge data cannot be stored in the memory of a computer and existing algorithms cannot handle such a data. One of the main objectives of this research has been representing a document by a much smaller number of features leveraging word embeddings and word taxonomies.

Based on our objectives, our research questions are defined as:

1. Can we accurately represent a document with a small number of features to the purposes of accurate and efficient clustering and classification?
2. Can we accurately predict the recovery outcomes of clients of accident compensation agencies based on their conversations with their claim managers via text clustering or topic modelling? This question can be addressed by developing new algorithms that better fit the industry partner datasets and regress the topics towards outcomes.
3. Can we anticipate/forecast clients' outcomes in real time using topic models? How long ahead can we predict clients' outcomes with a reasonable accuracy?
4. Can we, in general, leverage topic models and text clustering to build more accurate predictive models?

1.3 Publications

Journal papers

1. Seifollahi S., Bagirov A., Zare Borzeshi E., Piccardi M. (2019), A simulated annealing-based maximum margin clustering algorithm, Computational Intelligence, 35(1), pp. 23–41.

Conference Papers

1. Seifollahi S., Piccardi M., Taxonomy-Based Feature Extraction for Document Classification, Clustering and Semantic Analysis, The 20th International Conference on Computational Linguistics and Intelligent Text Processing (CICLing), April 2019, France.

2. Seifollahi S., Piccardi M., Zare Borzeshi E., Kruger B., Taxonomy-augmented features for document clustering, The Australian Data Mining Conference (AusDM), November 2018, Bathurst, Australia.

3. Bagirov A., Seifollahi S., Piccardi M., Zare Borzeshi E., Kruger B., SMGKM: An efficient incremental algorithm for clustering document collections, The 19th International Conference on Computational Linguistics and Intelligent Text Processing (CICLing 2018), March 2018, Hanoi, Vietnam.

4. Seifollahi S., Piccardi M., Zare Borzeshi E., A semi-supervised hidden Markov topic model, The Australian Data Mining Conference (AusDM 2017), August 2017, Melbourne, Australia.

1.4 Thesis outline

This thesis consists of six chapters. Chapter 2 provides a brief overview of the literature on both document clustering and topic modelling. In Chapter 3, we introduce two methods, an incremental optimization and a maximum-margin based, for document clustering. Chapter 4 presents word embedding-based algorithms for document analytics, particularly for document clustering, using a very small number of features. Two algorithms for

topic modelling are developed and discussed in Chapter 5, with main focus on involving semantics into consideration. Finally, Chapter 6 concludes the thesis.

1.5 Common notations and symbols

Some notations and symbols are frequently used throughout this thesis. For convenience, they are recapped in Table 1.1.

Table 1.1: Table of the main notations used in this thesis.

Notation	Description
m	number of points/instances
M	number of documents
n	dimension of space
N	number of words in each document
d	a document in the corpus
D	set of documents
X	set of points/instances or document-term matrix
x	a point in X
c	center of a cluster
w	a word in the vocabulary
V	the dimension of the word vocabulary
K	number of topics/clusters
ϕ	topic-word matrix
θ	document-topic matrix

Chapter 2

Literature Review

Machine learning techniques are based on an automated learning, where the aim is to learn a model using historical data, so that the model can generalise to new unseen data. We have, in general, two types of machine learning approaches, depending on the problem to be solved; namely, supervised and unsupervised techniques. Figure 2.1 shows these learning techniques. Classification is the most-well known technique in the supervised case, while clustering and topic modelling are two of the most popular unsupervised techniques.

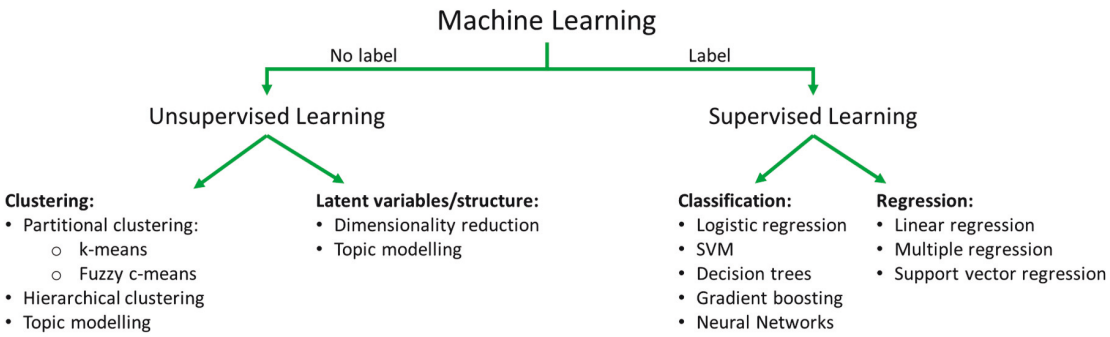


Figure 2.1: Main machine learning approaches.

In this section, we briefly review the literature for document clustering and topic modelling, with more focus on those techniques that are related to our research aims. Document clustering and topic modelling are highly related to each other. Topic modelling deals with constructing clusters of words rather than clusters of documents. A document is, therefore, a mixture of all the topics, each having a certain contribution. In other words, if document clustering is assigning a single category to a text document, topic modelling

is assigning multiple tags to that document.

2.1 Cluster analysis

Cluster analysis or clustering is one of the most frequently used exploratory data analysis techniques, often applied to get an intuition about the structure of the data. It is an unsupervised learning technique, and deals with the problem of organizing a collection of data into clusters based on a notion of similarity. More precisely, the task is grouping together a set of objects in a way that objects in the same cluster are more similar to each other than to objects in other clusters. Similarity is an amount that reflects the strength of relationship between two data objects.

The similarity measure is fundamental to formulate clustering problems. This measure, in particular, can be defined using distance(-like) functions. Clustering problems with the similarity measure defined using the Euclidean norm are usually called the minimum sum-of-squares clustering (MSSC) problems. To date, many algorithms based on different approaches have been developed to solve this problem. Amongst them, the *k*-means algorithm and its variations have been widely used to solve the MSSC problem (see, for example, [Kogan2007, Jain *et al.*1999] and references therein). The global *k*-means algorithm and its various modifications are amongst the most effective algorithms for solving the MSSC problem [Ordin and Bagirov2015, Bagirov *et al.*2011, Bai *et al.*2013]. They deal with a global solution or a near global solution, while *k*-means can only find a local solution to the clustering problem.

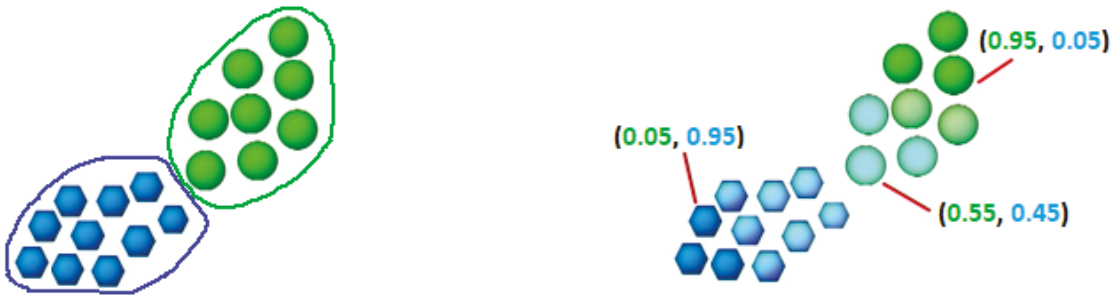


Figure 2.2: Difference between hard clustering (left) and soft clustering (right).

Clustering can be broadly divided into two subgroups, namely hard clustering and soft clustering. Figure 2.2 illustrates the differences between them. In hard clustering, each data object or point either belongs to a cluster completely or not, while in soft clustering,

a data point can belong to more than one cluster to a certain degree or likelihood value. Topic modelling can be compared to soft clustering where each document is a mixture of topics (or clusters). The *k*-means and Fuzzy *c*-means algorithms are two widely-used examples of hard and soft clustering techniques, respectively.

Clustering algorithms can also be categorized based on their cluster model or how they form the clusters or groups. Most of them are based on two main lines of approaches, the hierarchical and the partitional ([Jain *et al.*1999, Moseley and Wang2017]). They have been extensively studied in the literature (see for example [Moseley and Wang2017] and references therein). The *k*-means algorithm and its variants are the most broadly used partitional algorithms [Dhillon *et al.*2001, Seifollahi *et al.*2017a]. Partitional clustering algorithms find the partition that optimizes a clustering criterion or a objective function, e.g. minimise the MSSC problem.

Algorithms based on the hierarchical approach generate a dendrogram representing the nested grouping of patterns and similarity levels at which groupings change. Hierarchical algorithms can be divided into two main categories, namely agglomerative and divisive. Agglomerative algorithms are a bottom-up approach, i.e. the clustering produced at each layer of the hierarchy merges similar clusters from the previous layer. Conversely, divisive algorithms sub-divide clusters incrementally, starting from the initial dataset. Agglomerative algorithms generally perform better than divisive algorithms, and often "better" than single-layer algorithms such as *k*-means [Moseley and Wang2017].

In some cases, a combination of both techniques has been used. For example, in the document domain, the work of [Cutting *et al.*1992] uses a hybrid approach involving both *k*-means and agglomerative hierarchical clustering to create a document browsing system based on clustering. *k*-means is used because of its efficiency and agglomerative hierarchical clustering is used because of its quality. It is also shown that the"bisecting" *k*-means, which can be considered as *k*-means in a hierarchical form [Steinbach *et al.*2000], can produce clusters of documents that are better than those produced by the standard *k*-means and as good or better than those produced by agglomerative hierarchical clustering techniques. In this report, we focus on partitional clustering algorithms, although we also use them in a hierarchical form for word clusters, in a way similar to the "bisecting" algorithm.

Clustering is used in many fields such as pattern recognition, image analysis, information retrieval, bio-informatics, data compression, computer graphics and document clustering [Alshari *et al.*2017, Wang *et al.*2016, Yang *et al.*2016]. Document clustering is one of the most challenging tasks in the literature as the number of features can reach a power of 10, i.e. $100,000$. It has been investigated for use in a number of different areas of text mining and information retrieval.

Document clustering is highly related to data clustering. It was initially used for improving the precision or recall in information retrieval systems [Van-Rijsbergen1989, Kowalski1997] and as an efficient way of finding the nearest neighbors of a document [Buckley and Lewit1985]. It has also been used to automatically generate hierarchical clusters of documents [Koller and Sahami1997]. Other earlier references for document clustering include the work of [Cutting *et al.*1992] for browsing a collection of documents and the work of [Zamir *et al.*1997] for organizing the results returned by a search engine in response to a user's query.

Document clustering can be performed in a two-stage scenario. The first stage is to preprocess the documents in order to transform them into a usable data representation for analysis. The preprocessing consists of tasks such as the exclusion of words without informative value known such as stop words; the reduction of the words to their radicals, known as stemming; the uppercase/lowercase conversion known as case-folding; and transforming the words to a numeric space. The second stage is to analyze the numeric data and partition them into clusters.

2.2 Document representations for clustering

Text document representations play an important role in many natural language processing (NLP) based tasks such as document clustering and classification ([Zhang *et al.*2018, Gui *et al.*2014, Gui *et al.*2016]), sense disambiguation ([Gong *et al.*2017, Gong *et al.*2018]), machine translation ([Mikolov *et al.*2013b]), document matching ([Pham *et al.*2015]), and sequential alignment ([Peng *et al.*2018b, Peng *et al.*2015]). One of the main requisites in document clustering is to convert each document into a numeric vector or set of features. Since there are no explicit numerical features in text documents, much work has been aimed at developing effective numerical text representations. In the

following we discuss three well-known methods to convert text document into structured data, or numerical vectors.

2.2.1 Methods based on term frequency

Most clustering methods applied to unstructured document collections start with creating a vector-space model, known as a bag-of-words (BoW) model, [Salton and Buckley1988, Steinbach *et al.*2000, Dhillon and Sra2005], consisting of the frequencies in the document of each word in a vocabulary. This model is popular due to its simplicity, efficiency and often surprisingly high accuracy [Wang and Manning2012]. In this model, documents are described by a very sparse matrix due to the large size of typical vocabularies. The words are treated as attributes and each document is described by a vector that stores the frequency of each word in the document. The set of all documents is usually called document-to-term matrix, or sometimes document-term matrix.

It is very common to reweigh the term frequencies to somehow reflect the "discriminative power" of each word. The term frequency-inverse document frequency (tf-idf) approach, [Robertson and Walker1994], is the most popular reweighting scheme for the BoW model. The term frequency (tf) is the raw count of the word's appearances in a document, and the inverse document frequency (idf) increases the importance of words that appear rarely in the collection, assuming that they would be more discriminative in any ensuing clustering and classification tasks [Erra *et al.*2015, Qimin *et al.*2015, Bagirov *et al.*2018]. The simple formula of the tf-idf weighting approach is:

$$tf_{i,j} \times \log(\frac{M}{df_i}) \tag{2.1}$$

where $tf_{i,j}$ is the number of occurrences of the i-th word in the j-th document, df_i the number of documents containing the i-th word, and M the total number of documents. Intuitively, a term has a large weight when it occurs frequently across the document but infrequently across the corpus. For example, in a health domain words such as "dr" and "pain" might appear often in a document, but because they are likely fairly common also in the rest of the corpus, they will not have a high tf-idf score. Conversely, words referring to a specific pathology may have higher tf-idf score.

2.2.2 Ontology based techniques

The BoW model has inherent flaws such as ignoring the word ordering [Cheng2008] and suffering from the "curse of dimensionality" (the difficulty of learning in high dimensional spaces) [Friedman1997]. Using an ontology, derived from existing databases, is a promising solution to diminish such flaws [Fodeh *et al.*2011, Elsayed *et al.*2015].

In [Hotho *et al.*2003], the authors integrated the popular WordNet ontology with document clustering. WordNet organizes words into hierarchical sets of synonyms called 'synsets'. They tried to solve the problem of the BoW representation by leveraging synonymy, in order to represent relationships between terms which do not co-occur literally. The authors considered the synsets as concepts and extended the BoW model by including the parent concepts (hypernyms) of synsets up to five levels. For example, they utilised the WordNet ontology to find the similarity between related terms such as "beef" and "pork", since these two terms have the same parent concept "meat". Therefore, a document having the term "beef" will be related to a document with the term "pork" appearing in it; the proposed approach is shown to enhance the clustering performance.

The work of [Recupero2007] presents a clustering technique based on an information extraction system, ANNIE, and WordNet that finds the lexical category of each term and uses it to replace the term. The document vectors were reduced to 41 dimensions given that this is the number of lexical categories for nouns and verbs in Wordnet. At its turn, ANNIE is an information extraction system which helps understand whether two words or compound words refer to the same entity [Recupero2007]. In [Fodeh *et al.*2011], the authors claim that the drawback of augmenting WordNet synsets with original terms is increasing the dimensionality of terms. They used an ontology to reduce the number of features. In their work they showed that by identifying and using noun features alone, document clustering is improved. Furthermore, they used word sense disambiguation techniques in order to resolve polysemy, which happens when a single word has multiple meanings. In addition, they used an ontology to resolve synonymy problems, which occur when two different words refer to the same concept, by using the corresponding concepts and expelling the synonymous. In turn, the work of [Elsayed *et al.*2015] leveraged WordNet to decrease the document features to just 26 features representing the WordNet lexical noun categories. They integrated the WordNet ontology with bisecting k-means using the

MapReduce parallel programming model. MapReduce is a programming model and an associated implementation for processing and generating big data sets with a parallel, distributed algorithm on a computer cluster [Dean and Ghemawat2008].

2.2.3 Methods based on word embeddings

More recently, various document clustering techniques based on word embeddings [Mikolov *et al.*2013a] have emerged as capable of overcoming the flaws of the BoW model. More precisely, these techniques embed each distinct word in a vector space of dimensionality ($\approx 10^2 - 10^3$) typically much smaller than that of the BoW model ($\approx 10^5 - 10^6$).

A recent empirically successful body of research makes use of distributional or contextual information together with simple neural network models to obtain vector-space representations of words and phrases such as Wor2Vec [Mikolov *et al.*2013a] and GloVe [Pennington *et al.*2014]. There are two models in Wor2Vec, [Mikolov *et al.*2013a], namely, the Continuous Bag-of-Words (CBOW) model and the Skip-gram (SG) model. A graphical description of these models is shown in Fig. 2.3. Both models have three layers: input layer, projection layer and output layer. For the CBOW model, the context $\{w_{t2}, w_{t1}, w_{t+1}, w_{t+2}\}$ is used to predict the word w_t, while it is the reverse in the other model; i.e. the word w_t is used to predict the context $\{w_{t2}, w_{t1}, w_{t+1}, w_{t+2}\}$. There is no specific, best value for the size of the context, and it needs to be chosen before the learning. One of the standing advantages of a word embedding approach such as Wor2Vec over more recent *contextualised* approaches such as Embeddings from Language Models (ELMo) [Peters *et al.*2018] and Bidirectional Encoder Representations from Transformers (BERT) [Devlin *et al.*2018] is that it encodes each single word in a vocabulary with a vector that does not change in different instances of the word. This allows using the word embeddings for tasks such as counting, clustering and building ontologies which are the key targets of this thesis.

A number of researchers have proposed extensions of Wor2Vec towards learning semantic vector-space representations of sentences or documents and used them for, among other: sentiment analysis [Tang *et al.*2014, Zhang *et al.*2015], document distance measurement [Kusner *et al.*2015], topic modelling [Xun *et al.*2017, Das *et al.*2015], and

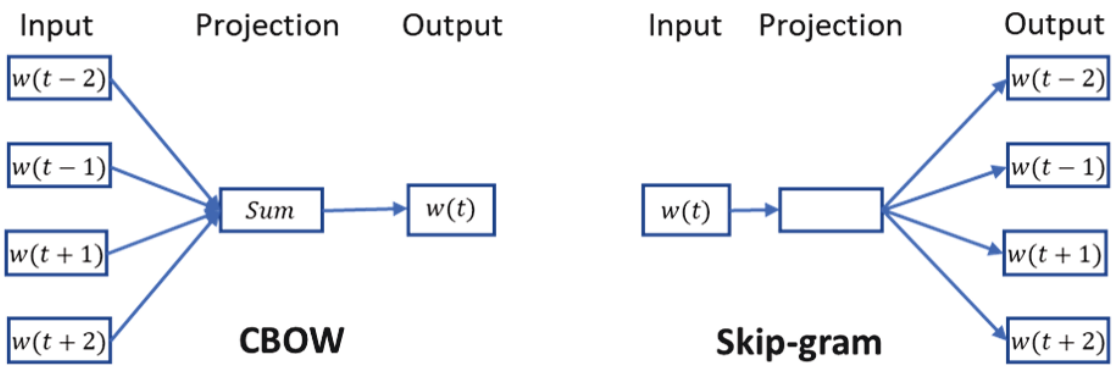


Figure 2.3: The two models of Wor2Vec.

document clustering and classification [Kim *et al.*2015, Lilleberg *et al.*2015, Wang *et al.*2016, Lenc and Král2017]. However, extending word embeddings to entire paragraphs and documents for tasks such as document and short-text classification is a challenge.

A simple but often effective approach is to use a weighted average of some or all of the word embeddings in the document. While this is simple, important information could easily be lost in such a document representation, particularly since it does not consider word ordering. A more sophisticated technique ([Le and Mikolov2014]) has focused on jointly learning embeddings for both words and paragraphs using models similar to Wor2Vec, namely, Distributed Memory Model Paragraph Vectors (PV-DM) and Distributed BoWs paragraph vectors (PV-DBoW). In PV-DM, the model is learned to predict the next context word using word and paragraph vectors, while in PV-DBoW, the paragraph vector is directly learned to determine randomly-sampled context words. These models use the word order within a small context window and potentially capture only local semantics [Singh and Mukerjee2015]. The quality of the word embeddings learned in such models may be limited by the size of the training corpus, which cannot scale to the large sizes used in the simpler word embedding models, and which may consequently weaken the quality of the document embeddings. The works of [Wang *et al.*2016] map word embeddings to a latent topic space to capture different senses in which words occur. They represent documents in the same space as words. However, these methods are also computationally intensive.

The work of [Kusner *et al.*2015] presents a document distance metric, the Word Mover's Distance (WMD), that measures the dissimilarity between two documents in the Wor2Vec embedding space. Despite its state-of-the-art KNN-based classification accuracy over other methods, combining KNN and WMD might incur very high computational cost. More importantly, WMD is simply a distance that can be only combined with KNN or

k-means, whereas many machine learning algorithms require a fixed-length feature representation as input.

In order to better capture the order relationships between words, and improve the topic discovering in an effective way, the work of [Li *et al.*2017] classified the order relationships between terms into forward dependence and backward dependence, and presented a feedback recurrent neural network-based topic model. To deal with capturing backward dependences, they considered each word with a one-hot vector and applied long short-term memory (LSTM) recurrent neural network to compute its corresponding embedded vector, and designed a feed-back mechanism for the recurrent neural network for capturing forward dependences.

The work of [Kim *et al.*2017] proposed the bag-of-concepts method for document representation, which overcomes the weaknesses of BoW and Wor2Vec models as shown by the authors. The method creates concepts through clustering word vectors generated from Wor2Vec, and uses the frequencies of these concept clusters to represent document vectors. In the other words, document vectors are represented by the frequencies of the concepts. The proposed method aims to incorporate the impact of semantically similar words on preserving document proximity effectively.

The work of [Peng *et al.*2018a] introduces a Deep Graph convolutional neural networks (CNN) approach to text classification. They used recursive regularization to deep learning for large scale hierarchical text classification. This is a general framework for deep learning applied to classifications problems when classifying data into a hierarchy of labels.

2.3 The *k*-means algorithm and its variants

In this section, we describe the *k*-means clustering algorithm and some of its variants. Let X be a finite set of points in the n-dimensional space $\mathbb{R}^n$:

$$X = \{x^1, \ldots, x^m\}, \;\; x^i \in \mathbb{R}^n, i = 1, \ldots, m.$$

The data points x^i, $i = 1, \ldots, m$ are called *instances* and each instance has *n dimensions*.

The *hard unconstrained clustering problem* is the distribution of the points of the set

X into a given number K of disjoint subsets X^j, $j = 1,\ldots,K$ with respect to predefined criteria such that

1. $X^j \neq \emptyset$, $j = 1,\ldots,K$;
2. $X^j \bigcap X^l = \emptyset$, for all $j,l = 1,\ldots,K$, $j \neq l$;
3. $X = \bigcup_{j=1}^{K} X^j$.

The sets $X^j, j = 1,\ldots,K$, are called *clusters*. Each cluster X^j can be identified by its *center* $c^j \in \mathbb{R}^n, j = 1,\ldots,K$. The problem of finding these centers is called the *K-clustering* (or K-partition) *problem*.

2.3.1 The k-means algorithm

The k-means clustering algorithm is a form of unsupervised learning, which is used for unlabeled data (i.e., data without defined categories, groups or classes). The goal of this algorithm is to partition the data into groups, with the number of groups represented by the variable K. This is a versatile algorithm that can be used for any type of grouping. Some examples of use cases are behavioral segmentation, inventory categorization, sorting sensor measurements, detecting bots or anomalies and document categorization.

The algorithm works iteratively to assign each data point to one of K groups based on the features that are provided, i,e. data points are clustered based on notion of feature similarity. The results of the k-means clustering algorithm are: 1) the centroids of the K clusters, which can be used to label new data and 2) labels for the training data, e.g. each data point is assigned to a single cluster. Each centroid of a cluster is a collection of feature values which define the resulting groups. Examining the centroid feature weights can be used to qualitatively interpret what kind of group each cluster represents.

The k-means starts with K initial cluster centroids and assigns each object to its closest one based on the similarity measure defined as the squared Euclidean distance. In each round, the centroids are recalculated and objects are re-assigned. The algorithm keeps running until the clusters converge (covnergence to a local minimum is guaranteed). In other words, it uses iterative refinements to produce a final result. The algorithm's inputs are the number of clusters, i.e. K, and the dataset. The initial estimates for the K centroids can either be randomly generated or randomly selected from the dataset. The algorithm assigns each object to its closest one. The steps of the algorithm are as Algorithm 1

[Arthur and Vassilvitskii2007]:

Algorithm 1: Standard k-means algorithm.

1. Arbitrarily choose K initial centers $\mathscr{C} = \{c^1, \cdots, c^K\}$.
2. For each $i \in \{1, \cdots, K\}$, set the cluster X^i to be the set of points in X that are closer to c^i than they are to c^j for all $j \neq i$.
3. For each $i \in \{1, \cdots, K\}$, set c^i to be the center of mass of all points in X^i:

$$c^i = \frac{1}{|X^i|} \sum_{x \in X^i} x.$$

4. Repeat Steps 2. and 3. until no more changes in $\mathscr{C}$.

In Step 2., ties may be broken arbitrarily, as long as the method is consistent. Steps 2. and 3. are both guaranteed to minimise the objective function f:

$$f = \sum_{x \in X} \min_{c \in \mathscr{C}} \|x - c\|^2. \tag{2.2}$$

where $\|.\|$ is generally L_2 norm. The algorithm iterates between two steps, 2. and 3., until a stopping criteria is met. Examples of the stopping criteria can be:

- no data points change clusters;
- the sum of the distances is minimised (oracle solution);
- some maximum number of iterations is reached.

The algorithm is guaranteed to converge to a solution, but it might be a local optimum. Therefore, it is not necessarily the best possible solution or the global solution. To overcome this, repeating more than one run of the algorithm with randomized starting centroids may give a better outcome.

2.3.2 The k-means++ algorithm

The k-means++ variant was proposed in 2007 by Arthur and Vassilvitskii [Arthur and Vassilvitskii2007]. It is identical to the k-means algorithm, except for the selection of initial conditions. This algorithm comes with a theoretical guarantee to find a solution that is $O(\log K)$ competitive to the optimal k-means solution. At any given time, let $d(x)$ denote the shortest distance from a data point x to the closest center; the steps of the

algorithm are, then, defined as follows [Arthur and Vassilvitskii2007]:

Algorithm 2: The *k*-means++ algorithm.

1. Choose an initial center c^1 uniformly at random from X.
2. Choose the next center c^i, selecting $c^i = \bar{x} \in X$ with probability $d(\bar{x})^2 / \sum_{x \in X} d(x)^2$.
3. Repeat Step 2. until K centers are selected.
4. Proceed as with the standard *k*-means algorithm.

2.3.3 Spherical *k*-means algorithm

Among many clustering algorithms in the literature, the spherical *k*-means (spkmeans) algorithm [Dhillon *et al.*2001], that performs *k*-means using the squared Euclidean distance as the dissimilarity measure of the projections of the feature vectors onto the unit sphere, or equivalently, the cosine dissimilarity, has been found to work well for document clustering. The work of [Banerjee *et al.*2005] shows that the spkmeans is also obtained as an Expectation Maximization (EM) variant for Maximum Likelihood Estimation of the mean direction parameters of a uniform mixture of von Mises-Fisher (or Langevin) distributions.

Let consider symbols M and K to denote the number of documents and the number of clusters, respectively. We will use the symbol X to denote the set of M documents that we want to cluster, with $X^1, X^2, \ldots, X^K$, to denote the collection of the K clusters.

Given the vector space model, the document vectors may be represented by $x^1, x^2, \ldots, x^M$, with each $x^i \in \mathbb{R}^V$. Recall that V stands for the number of unique words in the vector space model and M is the total number of documents. A clustering of the document collection is its partitioning into the disjoint subsets $X^1, X^2, \ldots, X^K$, *i.e.*

$$\bigcup_{j=1}^{K} X^j = \{x^1, x^2, \ldots, x^M\} \quad \& X^j \cap X^l = \emptyset, \quad j \neq l.$$

In spkmeans, the data is projected to the unit sphere. There are many schemes for selecting the term, global, and normalization components. The spkmeans algorithm uses the popular tf-idf scheme known as normalized term frequency-inverse document fre-

quency [Salton and McGill1983]. Note that the tf-idf normalization implies that $\|x^i\| = 1$, i.e., each document vector lies on the surface of the unit sphere in $\mathbb{R}^V$. This normalization is the most common weighting method used to describe documents in the vector space model [Salton and McGill1983]. It reduces the weight (or importance) of common terms in the collection, ensuring that the matching of documents be more influenced by that of more discriminative words which have relatively low frequencies across the documents in the collection. This ensures that documents dealing with the same subject matter (that is, using similar words), but differing in length lead to similar document vectors [Dhillon *et al.*2001]. The partitioning of spkmeans is achieved by maximising the following objective function:

$$\sum_{j=1}^{K} \sum_{x \in X^j} x^T \times c^j \tag{2.3}$$

where $\|x\| = 1$, $x^T \times c^j$ is the inner product between two vectors and c^j is the normalized centroid of cluster X^j,

$$c^j = \sum_{x \in X^j} x \Big/ \| \sum_{x \in X^j} x \|. \tag{2.4}$$

2.3.4 Global *k*-means algorithm

The global *k*-means algorithm, introduced in [Likas *et al.*2003], is an improvement of the *k*-means algorithm. This algorithm computes clusters successively and in order to compute K-partition this algorithm uses centers of $K-1$ clusters from the previous iteration. To compute $q \leq m$ clusters this algorithm proceeds as follows:

Algorithm 3: Global *k*-means algorithm.

1. Consider the centers $c^1, c^2, \ldots, c^{K-1}$ from the previous iteration.
2. Add in turn, each point of A, thus obtaining m initial solutions with K points. Apply *k*-means to each of them and keep the best K-partition so-obtained and its centers $c^1, c^2, \ldots, c^K$.
3. Set $K = K+1$ and go to Step 1 as long as $K \leq q$.

The global *k*-means algorithm is not applicable for clustering on medium sized and large data sets. It is very time consuming as at each iterations the number of the *k*-means algorithm applications made is the number of data points.

2.3.5 Modified global k-means algorithm

In this subsection, we briefly describe the modified global k-means algorithm [Bagirov2008]. Consider the *nonsmooth optimization formulation of the MSSC problem* [Bagirov and Yearwood2006]:

$$\begin{cases} \min & f_K(c) \\ \text{subject to} & c = (c^1, \ldots, c^K) \in R^{nK}, \end{cases} \tag{2.5}$$

where

$$f_K(c^1, \ldots, c^K) = \sum_{x \in X} \min_{j=1,\ldots,K} \|c^j - x\|^2. \tag{2.6}$$

The function f_K is called the *K-th clustering objective function*. For $K = 1$ this function is convex and for $K > 1$ it is both non-convex and nonsmooth. To partition the data X to K clusters one needs to solve the K-clustering problem (2.5).

Assume that the solution $c^1, \ldots, c^{K-1}$ to the $(K-1)$-clustering problem and the corresponding value $f^*_{K-1} = f_{K-1}(c^1, \ldots, c^{K-1})$ of the function f_{K-1} are known, $K > 1$. The so-called auxiliary cluster function is formulated as [Bagirov2008]:

$$\bar{f}_K(y) = \frac{1}{m} \sum_{i=1}^{m} \min \left\{ d^i_{K-1}, \|y - x^i\|^2 \right\}. \tag{2.7}$$

Here d^i_{K-1} is the squared distance between x^i and the closest center among $K-1$ cluster centers $c^1, \ldots, c^{K-1}$:

$$d^i_{K-1} = \min \left\{ \|c^1 - x^i\|^2, \ldots, \|c^{K-1} - x^i\|^2 \right\}. \tag{2.8}$$

It is obvious that

$$\bar{f}_K(y) = f_K(c^1, \ldots, c^{K-1}, y), \;\; y \in \mathbb{R}^n. \tag{2.9}$$

Consider the following set

$$P = \left\{ y \in \mathbb{R}^n : \exists I \subset \{1, \ldots, m\},\, I \neq \emptyset : \|y - x^i\|^2 < d^i_{K-1} \;\; \forall i \in I \right\}. \tag{2.10}$$

The set P contains all points from $\mathbb{R}^n$ which are not cluster centers and can attract at least one data point from the set A. It is clear that $c^j \notin P$ for all $j = 1, \ldots, K-1$ and $x^i \in P$ for

all $x^i \in X: \ x^i \neq c^j, \ j = 1, \ldots, K-1$. It is also clear that $\bar{f}_K(y) < f^*_{K-1}$ for all $y \in P$. For any $y \in P$ consider the following set:

$$S(y) = \left\{ x^i \in X : \|y - x^i\|^2 < d^i_{K-1} \right\}. \tag{2.11}$$

The set $S(y)$ contains all points from X which are attracted by the point $y \in P$. Note that $S(y) \neq \emptyset$ for any $y \in P$.

A starting point for the K-th cluster center is found as a solution to the following minimization problem:

$$\text{minimize} \ \ \bar{f}_K(y) \ \ \text{subject to} \ \ y \in \mathbb{R}^n. \tag{2.12}$$

The steps for solving Problem (2.12) involves the k-means algorithm and proceeds as Algorithm 4.

Algorithm 4: An algorithm for finding a starting point.

1. For each $x^i \in P \bigcap X$ compute the set $S(x^i)$, its center c^i and the value $\bar{f}_{K,x^i} = \bar{f}_K(c^i)$ of the function $\bar{f}_K$ at the point c^i.
2. Compute
$$\bar{f}_{K,min} = \min_{x^i \in P \bigcap X} \bar{f}_{K,x^i}, \ \ x^j = \operatorname*{argmin}_{x^i \in P \bigcap X} \bar{f}_{K,x^i},$$
select the corresponding center c^j and the set $S(c^j)$.
3. Recompute the set $S(c^j)$ and its center until no more data points escape or return to this cluster. The final value of c^j is a starting point for the K-th cluster center.

The modified global k-means algorithm solves the problem (2.5) incrementally. It starts with the computation of the centroid of the whole data set. Then a new cluster center is added at each iteration. The K-clustering problem is solved using the $K-1$ centers for the $K-1$ clustering problems and the remaining K-th center is placed in an appropriate place. An auxiliary cluster function is defined using $K-1$ cluster centers from the $(K-1)$-th iteration and is minimized to compute the starting point for the K-th center. Then this new center together with previous $K-1$ cluster centers is taken as a starting point for the K-clustering problem. The k-means algorithm is applied starting from this point to find the K-partition of the data set. Such an approach allows one to find a global or a near global solution to the clustering problem. The steps of modified global k-means are explained in

Algorithm 5.

Algorithm 5: Modified global k-means algorithm.

1. (Initialization). Select a tolerance $\varepsilon > 0$. Compute the center $c^1 \in \mathbb{R}^n$ of the set X. Let f_1 be the corresponding value of the objective function (2.6). Set $l := 1$.
2. (Stopping criterion). Set $l = l + 1$. If $l > K$, then **stop** since the K-partition problem has been solved.
3. (Computation of the next cluster center). Set $l := l + 1$. Let $c^1, \ldots, c^{l-1}$ be the cluster centers for $(l-1)$-partition problem. Apply Algorithm 4 to find a starting point $\bar{y} \in \mathbb{R}^n$ for the l-th cluster center.
4. (Refinement of all cluster centers). Select $(c^1, \ldots, c^{l-1}, \bar{y})$ as a new starting point, apply the k-means algorithm ($K = l$) to solve the l-partition problem. Let $y^1, \ldots, y^l$ be a solution to this problem and f_l be the corresponding value of the objective function (2.6).
5. (Stopping criterion). If
$$\frac{f_{l-1} - f_l}{f_1} < \varepsilon,$$
then stop, otherwise set $c^i = y^i,\ i = 1, \ldots, l$ and go to Step 2.

It is clear that $f_K^* \geq 0$ for all $K \geq 1$ and the sequence $\{f_K^*\}$ is decreasing, that is,

$$f_{K+1}^* \leq f_K^* \ \ \forall \ \ K \geq 1. \tag{2.13}$$

This means that stopping criterion in Step 5. will be satisfied after a finite number of iterations and therefore, Algorithm 5 computes as many clusters as the data set A contains with respect to the tolerance $\varepsilon > 0$. The choice of this tolerance is crucial for Algorithm 5. Large values of ε can result in the appearance of large clusters whereas small values can produce small and artificial clusters.

2.4 Maximum margin clustering

There is a growing interest in applying support vector machines (SVM) techniques to clustering motivated by their success in supervised learning. However, unlike large-margin supervised learning, large-margin unsupervised learning is a non-convex problem. Most algorithms for large-margin unsupervised learning are based on some relax-

ation techniques (e.g., [Li *et al.*2009, Xu and Schuurmans2005]). These techniques allow applying convex optimization methods, however, they are typically accurate only on relatively small datasets. The use of relaxation techniques in large datasets may lead to solutions which are significantly different from the global solution of the original non-convex problem. Therefore, it is imperative to develop methods which are both efficient and accurate at solving large-margin unsupervised learning problems on large datasets. In particular, such methods can be designed using opportunistic combinations of local and global search algorithms. In these methods, local search algorithms are used to determine local solutions while global search algorithms are used to escape from such local solutions and find better re-starting points for the local search algorithms.

Consider a set of training samples, $\mathscr{X} = \{(x^i, y^i)\}_{i=1}^m$, where x^i is the instance and y^i the class. For simplicity assume that $y^i \in \{-1, 1\}$. The SVM finds a maximum-margin hyperplane $h(x) = \omega^T \Phi(x) + b = 0$, where $\Phi(x)$ is the mapping induced by a kernel and T is transpose of a vector, by solving:

$$\begin{cases} \min_{\omega,b,\xi_i} & \|\omega\|^2 + 2\mathscr{C}\xi^T e \\ \text{subject to} & y^i(\omega^T \Phi(x^i) + b) \geq 1 - \xi_i, \;\; \xi_i \geq 0. \end{cases} \tag{2.14}$$

Here, the ξ_i's are slack variables for the errors, $\mathscr{C} > 0$ is a regularization parameter and $e = (1, \ldots, 1)^T$. The optimization problem (2.14) is convex. If the class labels, y^i, are unknown, the problem becomes a non-convex maximum-margin clustering problem (MMC) [Zhang *et al.*2007]:

$$\begin{cases} \min_y \min_{\omega,b,\xi_i} & \|\omega\|^2 + 2\mathscr{C}\xi^T e \\ \text{subject to} & y^i(\omega^T \Phi(x^i) + b) \geq 1 - \xi_i, \;\; \xi_i \geq 0. \\ & y^i \in \{-1, +1\}, \\ & -l \leq e^T y \leq l. \end{cases} \tag{2.15}$$

The last constraint, $-l \leq e^T y \leq l$, is a "balance" constraint preventing the convergence of the algorithm to a trivially "optimal" solution where all instances are assigned to the same class.

The dual problem of (2.15) is:

$$\begin{cases} \min_y \max_\alpha & 2\alpha^T e - \alpha^T (\mathscr{K} \circ yy^T)\alpha \\ \text{subject to} & \alpha^T y = 0, \;\; Ce \geq \alpha \geq 0, \\ & y^i \in \{-1, +1\}, \\ & -l \leq e^T y \leq l. \end{cases} \tag{2.16}$$

where α is the vector of the dual parameters, $\alpha = [\alpha_1, \ldots, \alpha_m]^T$, $\mathscr{K}$ is the kernel function and "$\circ$" is the element-wise product between matrices.

It is shown in [Xu *et al.*2005] that (2.16) is equivalent to an NP-hard convex integer programming. Since its introduction, it has been extended in many ways, for instance, by choosing different loss functions (e.g., [Zhang *et al.*2007], [Gieseke *et al.*2009]), incorporating additional constraints such as pairwise links ([Hu *et al.*2008]) and manifold smoothness ([Wang *et al.*2009]), or adding a feature weighting mechanism ([Zhao *et al.*2009]).

The works of [Xu *et al.*2005] and [Valizadegan and Jin2006] reformulated the original problem as a semi-definite programming (SDP) relaxation. The work of [Zhang *et al.*2007] employ alternating optimization - finding labels and optimizing a support vector regression (SVR). The paper of [Li *et al.*2009] iteratively infers the most-violating ground-truth labels and combines them via multiple kernel learning. Note that the above methods can only solve binary-cluster clustering problems. To handle the multi-cluster case, [Xu and Schuurmans2005] extends the SDP method in [Xu *et al.*2005]. The work of [Zhao *et al.*2008] proposes a cutting-plane method which uses the constrained convex-concave procedure to relax the non-convex constraint. The work of [Gopalan and Sankaranarayanan2011] examines data projections to identify the maximum margin. More recently, [Karnin *et al.*2012] has proposed a global solution of polynomial complexity; however, its soft-margin formulation differs from the conventional soft-margin SVM based on the hinge loss.

Overall, the existing methods for MMC can be categorized as either relaxations or alternating methods [Zhao *et al.*2008, Zhang *et al.*2007]. For the relaxation methods, one can review the papers of [Xu *et al.*2005], [Li *et al.*2009] as examples. These methods

aim to find best approximating functions to the non-convex formulation of (2.16) and solve the relaxed problem instead, by hoping that the solution will be close enough to the actual one. However, in general, the solution found is not qualified compared to the actual solution (but the relaxation in [Li *et al.*2009] is provenly tighter than that of [Xu *et al.*2005]).

In contrast to the relaxation methods, alternating methods solve the original problem in two steps [Zhang *et al.*2007]. They first initialize the labels using a clustering technique such as k-means and then they solve a supervised problem, often using SVM. The procedure can be iterated in various ways until no further improvement is achieved. Our proposed method falls in this group since we use an iterative method leveraging a combination of k-means++ and SVM at each iteration. However, our method differs from existing methods in that we solve a sub-problem at each step rather than considering the whole problem at once.

2.5 Topic modelling

Topic models have become a frequently used tool in text mining. They are mostly referred to as probabilistic topic models, which alludes to probabilistic algorithms for revealing the hidden structure of a document collection. A topic model takes as input a collection of documents and returns as output a set of *topics*. Figure 2.4 shows the input, output and the main steps in topic modelling. These topics are probability distributions over the words in the collection, Figure 2.5. Topic models can be used for deriving low-dimensional representations of documents that are, then, used for information retrieval, document summarization, and classification [Blei and McAuliffe2008, Lacoste-Julien *et al.*2009]. There are several scenarios when topic modelling can prove useful:

1. Document classification: The topic vectors of the documents can be used as features or input data for document classification. More precisely, topic modelling can improve classification by grouping similar words together in topics rather than using each word as a feature. The number of features will be the number of topics rather than the much larger number of unique words in the corpus.

2. Recommender systems: One can build recommender systems using a similarity mea-

sure based on topic modelling. If a system has to recommend articles for readers, it can recommend articles with a topic structure similar to the articles that the user has already read.

3. Uncovering themes in texts: Useful, for example, for detecting trends in online publications or customer's behaviours.

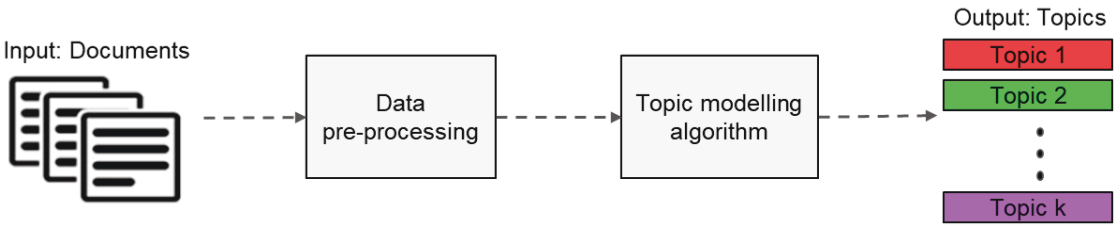


Figure 2.4: The main steps in topic modelling.

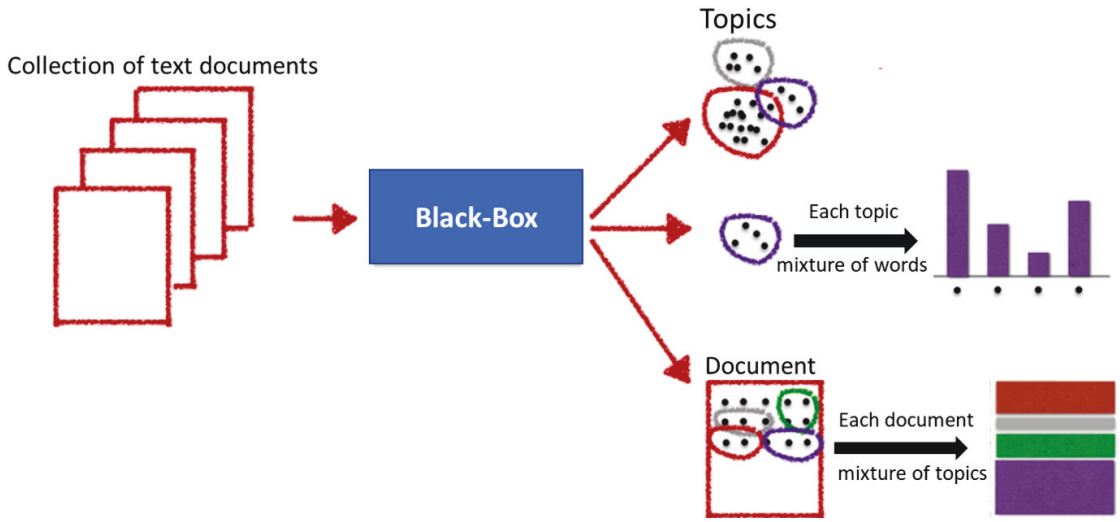


Figure 2.5: Another view on topic modelling.

There are several well-known algorithms for topic modelling. The most popular ones include: 1) Latent Dirichlet Allocation or LDA, where the foundations are Probabilistic Graphical Models; 2) LSA (or LSI) which stands for Latent Semantic Analysis (or Latent Semantic Indexing). It uses Singular Value Decomposition (SVD), [Wu and Stathopoulos2014] of the Document-Term Matrix; 3) Non-Negative Matrix Factorization or NMF, which uses linear Algebra and matrix factorization to find topic-word and document-topic matrices. In all the algorithms, the number of topics is a (hyper)parameter of model learning and needs to be given by the user in advance. All of the algorithms have as input the document-term matrix, and returns two matrices, a topic-word matrix and a document-topic matrix. A good model should produce those two matrices such that their multipli-

cation is as close as possible to the document-term matrix. None of the above algorithms can infer the number of topics in the document collection.

The beginning of topic modelling dates back to latent semantic indexing (LSI) [Deerwester *et al.*1990]) which was not originally introduced for topic modelling. The earliest topic model as such was introduced by Papadimitriou et al. in [Papadimitriou *et al.*2000], and a significant step forward in topic modelling was due to Hofmann [Hofmann1999], who introduced a probabilistic model for the LSI (pLSI). Latent Dirichlet allocation (LDA), [Blei *et al.*2003], which is nowadays the most widely used model, is an extension of pLSA. LDA assumes the standard BoW representation and explains each document as a mixture of topics with each topic describing a probability distribution over the words in the collection.

Most of the other contemporary topic models are extensions of LDA such as the Pachinko allocation [Wei *et al.*2007] which introduces topics' correlations further to the word correlations to form topics. Examples of other extensions include, but are not limited, hierarchical formulations to produce an unknown number of topics [Teh *et al.*2006], topics evolution over time [Blei and Lafferty2006], topic correlation using the logistic normal distribution [Blei and Lafferty2007], topics that follow a Markov chain and change over sentences [Gruber *et al.*2007] and topics employing prior knowledge [Chen and Liu2014, Seifollahi *et al.*2017b].

Although LDA has been the starting model for many later models, one common limitation of LDA and many such related models is that they do not take subsequent words relations into account. Conversely, in [Gruber *et al.*2007] the topics of sentences follow a Markov chain. They assume that topic changes occur between sentences, and that the topic for each sentence is decided at the beginning of each sentence by drawing a random number from a binomial distribution. However, considering such an assumption may lead to incoherent topics in collections where the vocabulary used by human is complex. For this reason, this author's work [Seifollahi *et al.*2017b] has introduced a model based on a Markov chain, where the topic of each words depends on the previous word. This is done by calculating a word transition probability matrix in advance. Such a transition matrix, which is estimated via an initial topic model, records how two words are topically correlated to each other (details will be provided in later chapters).

Below we describe three well-known topic models that can be regarded as strong baselines. The core idea is to take the document-term matrix and decompose it into a separate document-topic matrix and a topic-word matrix in various ways.

2.5.1 Latent semantic analysis

Latent Semantic Analysis (LSA) also known as Latent Semantic Index (LSI) uses the BoW model [Bellegarda2005]. LSA learns latent topics by performing a matrix decomposition on the document-term matrix using SVD, and is typically used as a dimensionality reduction or noise reducing technique.

Let's say we have M as the number of text documents with V the number of total unique words. We wish to extract K topics from all the text data in the documents. The number of topics, K, has to be specified by the user. Given the document-term matrix, one can find two matrices, document-topic and topic-word, using the SVD procedure. In fact, SVD decomposes a matrix into three matrices. Suppose we want to decompose a matrix X using SVD. It will be decomposed into θ, Σ, and ϕ,

$$X = \theta \times \Sigma \times \phi^T \tag{2.17}$$

where θ and ϕ^T are document-topic and topic-word matrices, respectively, and Σ is a $V \times V$ diagonal matrix with non-negative real numbers which can be compounded with either of the other two factors.

2.5.2 Non-negative matrix factorization

Non-negative Matrix Factorization (NMF) is a Linear-algebraic model, that factors high-dimensional vectors into a low-dimensionality representation. Similar to Principal component analysis (PCA), NMF takes advantage of the fact that the vectors are non-negative. By factoring them into the lower-dimensional form, NMF forces the coefficients to also be non-negative.

Given the document-term matrix X, we can obtain two matrices ϕ and θ, such that $X = \theta \times \phi$, where θ and ϕ are so-called document-topic and topic-word matrices. These matrices, θ and ϕ, are calculated by optimizing over an objective function (like the

Expectation-Maximization algorithm), and updating both θ and ϕ iteratively until convergence. To be more specific, the input data matrix X can be approximately represented by the product of two non-negative matrices θ and ϕ, such that

$$\min_{\phi,\theta>0} \|X - \theta\phi\|^2 + \lambda R(\theta,\phi) \tag{2.18}$$

where $R(.,.)$ is a regularization term, and λ is the parameter of the regularization term. The regularization term $R(.,.)$ can be defined as $\theta + \phi$ or can be defined based on L_1 and L_2 norms:

$$R(\theta,\phi) = \gamma(\|\theta\|_1 + \|\phi\|_1) + (1-\gamma)(\|\theta\|_F^2 + \|\phi\|_F^2) \tag{2.19}$$

where γ is the ratio for the L_1 penalty. The regularization parameter helps controlling the sparsity of the topics.

2.5.3 Latent Dirichlet allocation

The Latent Dirichlet Allocation (LDA) model, [Blei *et al.*2003], is the most widely used probabilistic topic model to derive a low-dimensional topic space. It assumes the BoW representation which is the input for most topic modelling algorithms. Each document d is represented as a discrete distribution over topics, where each topic K is a discrete distribution over the words. Both distributions, the document-topic θ and topic-word ϕ, are drawn from Dirichlet distributions with parameters α and η respectively. Figure 5.1(a) illustrates the graphical model of LDA, where M is the number of documents, V the size of vocabulary, and K the number of topics in the collection. The documents are known, whereas the topic parameters, including the topic-word, document-topic and topic-word assignments, are hidden. The generative model for the LDA is as follows:

With the above notations and generative process for LDA, the joint probability of a document d with N words can be formulated as:

$$p(w) = \int_{\theta} \Big(\prod_{n=1}^{N} \sum_{z_n=1}^{K} p(w_n|z_n;\beta) p(z_n|\theta) \Big) p(\theta;\alpha) d\theta \tag{2.20}$$

The computation in (2.20) is highly intractable. Therefore, the need for an estimation method to determine the posterior (2.20) is crucial. There are generally two types of

Algorithm 6: Latent Dirichlet Allocation.

1. For each topic k in $\{1,\ldots,K\}$.
   1.1. Draw a distribution over words $\phi_k \sim Dir_V(\eta)$.
2. For each document d :
   2.1. Draw a vector of topic proportions $\theta_d \sim Dir_K(\alpha)$.
   2.2. For each word $w_{d,n}$
      2.2.1. Draw a topic assignment $z_{d,n} \sim Mult_K(\theta_d)$.
      2.2.2. Draw a word $w_{d,n} \sim Mult_V(\beta_{z_{d,n}})$.

estimation methods: i) the variational Bayesian estimation and ii) (collapsed) Gibbs sampling. The EM algorithm is generally used to learn the LDA parameters by maximising a variational bound; see [Teh *et al.*2006], while the aim in Gibbs sampling, [Griffiths and Steyvers2004], is to estimate the posterior distribution in a Markov chain whose limiting distribution is the posterior. The Markov chain runs for a large number of iterations, collects samples from the limiting distribution, and then approximates the distribution with the collected samples.

2.6 Deep learning and beyond

Deep learning is presently the undisputed technology for supervised machine learning, especially for classification of data of numerical nature. However, its application to unsupervised learning has been somehow more limited and more recent, with variational autoencoders the most notable approach [Kingma and Welling2014]. Deep learning has also recently made its way into unsupervised applications such as clustering and topic modelling [Aljalbout *et al.*2018, Yang *et al.*2016, Srivastava and Sutton2017]. However, in many cases the training objectives remain unchanged and deep learning seems to contribute mostly to the feature extraction. For instance, Jule is an approach that combines a variational autoencoder with a k-means output stage [Yang *et al.*2016]. Since in this thesis we focus on original formulations of clustering and topic modelling, we reserve to explore these avenues in future research.

Chapter 3

Algorithms for Documents Clustering

In this section, we present two novel methods for document clustering. First, we describe a method based on the maximum-margin concept, namely a semi-supervised maximum-margin clustering [Seifollahi *et al.*2018]. Then, we present a method based on incremental optimization and spherical k-means. Both methods use the bag-of-words (BoW) model to create the feature set.

3.1 A maximum margin clustering algorithm

Maximum margin clustering (MMC) is an extension of the support vector machines (SVM) technique for clustering. It partitions a set of unlabelled data into multiple groups by finding hyperplanes with the largest margins. Although existing algorithms have shown promising results, there is no guarantee of convergence of these algorithms to global solutions due to the non-convexity of the optimization problem. In this section, we propose a simulated annealing-based algorithm that is able to deal with the issue of local minima in the MMC problem. The novelty of our algorithm is twofold: (1) it comprises a comprehensive cluster modification scheme based on simulated annealing and (2) it introduces a new approach based on the combination of k-means++ and SVM at each step of the annealing process. More precisely, k-means++ is initially applied to extract subsets of the data points. Then, an unsupervised SVM is applied to improve clustering results.

We design a method to address two main issues of the MMC method: (1) its computational complexity and (2) the inherent risk of falling into local solutions. To address

the first problem, we perform the computation over only a subset of the points, hereafter called *representative points*, instead of the whole data. To address the second problem, we use a scheme employing simulated annealing and a combination of k-means++ and SVM to find "deeper" local solutions. The details for the initial process, post-processing, and the steps of the algorithm are presented in the following sub-sections.

3.1.1 Formulation of the clustering problem

In this section we present nonsmooth optimization formulations of the clustering problem. Consider a finite set X of points in the n-dimensional space R^n is given:

$$X = \{x^1, \ldots, x^m\}, \text{where } x^i \in R^n, i = 1, \ldots, m.$$

The data points x^i, $i = 1, \ldots, m$ are called *instances* and each instance has *n dimensions*.

The *hard unconstrained clustering problem* is the distribution of the points of the set X into a given number K of disjoint subsets X^j, $j = 1, \ldots, K$ with respect to predefined criteria such that

1. $X^j \neq \emptyset$, $j = 1, \ldots, K$
2. $X^j \bigcap X^l = \emptyset$, for all $j, l = 1, \ldots, K$, $j \neq l$.
3. $X = \bigcup_{j=1}^{K} X^j$.

The sets $X^j, j = 1, \ldots, K$ are called *clusters*. Each cluster X^j can be identified by its *center* $c^j \in R^n, j = 1, \ldots, K$. The problem of finding these centers is called the *K-clustering* (or K-partition) *problem*.

In this report, the similarity measure is defined using the Euclidean norm (the L_2-norm)

$$d_2(c, x) = \left(\sum_{i=1}^{n} (c^- x^i)^2 \right)^{1/2}. \tag{3.1}$$

The *nonsmooth optimization formulation of the MSSC problem* is [Bagirov and Yearwood2006]:

$$\begin{cases} \min & f_K(c) \\ \text{subject to} & c = (c^1, \ldots, c^K) \in R^{nK}, \end{cases} \tag{3.2}$$

where

$$f_K(c^1, \ldots, c^K) = \sum_{x \in X} \min_{j=1,\ldots,K} d_2(c^j, x). \tag{3.3}$$

The function f_K is called the *K-th clustering objective function*. For $K = 1$ this function is convex and for $K > 1$ it is both non-convex and nonsmooth.

3.1.2 Initial clusters

We generate an initial set of clusters using k-means++. These clusters are meant to act as "coarse-scale" points in the final clustering algorithm. Their number has to be very high so as to not over-simplify the problem, yet significantly smaller than the number of the original points to be approachable by MMC-like algorithms. An adequate value for the number of such initial clusters is likely to depend on the number of points, the number of attributes and the closeness of the points to each other. Herewith, we choose it as random number proportional to the number of points. The initialization procedure can be summarized as follows:

Algorithm 7: Computation the set $\mathscr{R}$ of representative points.

1. Select a number $\delta \in (0,1)$ and the final number of clusters K to be computed.
2. Compute K clusters $X^1, \ldots, X^K$ using the $k-$means++ algorithm.
3. For each cluster $X^j,\ j = 1, \ldots, K$ compute the weight $w_j = |X^j|/m$ and define the number of initial clusters as $K_j = \delta w_j m$.
4. For each cluster $X^j,\ j = 1, \ldots, K$ compute K_j clusters using $k-$means++.
5. Define the set $\mathscr{R}$ of representative points as the set of cluster centers computed in Step 4..

3.1.3 Post-processing of initial clusters

The set $\mathscr{R}$ of representative points obtained using Algorithm 7 are used as the input for a post-processing phase to generate the final clusters. The points are, then, grouped using

a combination of k-means++ and SVM, labelled as KMSVM. We use an iterative cluster modification scheme based on simulated annealing in which we update the representative points at each iteration using two steps: (1) perturbing a randomly-selected representative point to other points from the same initial cluster and (2) splitting an initial cluster into two new clusters if it exhibits high bi-modality when applying a Gaussian Mixture Model on the corresponding points.

We exploit simulated annealing to find a "deeper"' local solution to the clustering problem. Simulated annealing comprises two main iterations: the outer and inner iterations. In the outer iteration the temperature, $\mathscr{T}$, which is analogous to the temperature in the physical process of annealing, is updated. In order to do so, we take any initial value $\mathscr{T}_0$ for the temperature and a number $r \in (0,1)$ and use the following schedule for the temperature update: $\mathscr{T}_{i+1} = r \times \mathscr{T}_i, i = 0,1,2,....$ In the inner iteration the current state is modified to generate a new solution based on a proposal step. If the proposal reduces the value of the objective function, the transformation to the new state is accepted. If it increases the value of the objective function, the transformation is accepted with an acceptance probability:

$$p = \min\left(1, \exp\left(\frac{-\Delta f}{\mathscr{T}}\right)\right), \tag{3.4}$$

where $\Delta f = f^{new} - f^{old}$, f^{old} is the function value in the previous state and f^{new} is the function value based on the perturbed configuration. More precisely, a random number u from the uniform distribution $U[0,1]$ is generated. If $p \geq u$, the perturbed configuration is accepted as a new solution; otherwise the inner iterations are repeated. For more details on the simulated annealing method see [Kirkpatrick *et al.*1983] and [Seifollahi *et al.*2014].

3.1.4 The proposed algorithm for clustering large-scale data

Here we propose our algorithm which consist of two stages. First, Algorithm 7 is applied to generate the initial clusters. Then, the representative points of the initial clusters are used as input for the post-processing stage that outputs the final clusters.

We leverage bi-modality information (Step 3. in Algorithm 8) to decide whether the initial clusters are appropriate. The idea is to split a candidate initial cluster so that it substantially contains only one mode. The presence of two distinctive modes in a cluster can be identified using the Gaussian Mixture Model (GMM). Then the Kullback-Leibler

divergence (KLD) can be applied to measure the difference between distributions of these modes. KLD is a measure of the difference between two probability distributions. For Gaussian distributions P and Q it can be expressed in closed-form as [Duchi2007]:

$$D_{KLD}(P|Q) = \frac{1}{2}\Big(tr(\Sigma_q^{-1}\Sigma_p) + (\mu_q - \mu_p)^T\Sigma_q^{-1}(\mu_q - \mu_p) - n + \ln\big(\frac{\det\Sigma_q}{\det\Sigma_p}\big)\Big) \tag{3.5}$$

where μ_p and μ_q are the means of P and Q, Σ_p and Σ_q are their covariance matrices and n is the dimension of the vector space.

Algorithm 8: Simulated annealing-based maximum-margin clustering (SAMMC).

1. (Initialization). Compute K_0 initial clusters using Algorithm 7, where $K_0 = \sum_{i=1}^{K} K_i$ and compute the set $\mathscr{R}$ of representative points. Choose the final number of clusters, $K < K_0$.

2. (Computation of clusters). Apply KMSVM to the set $\mathscr{R}$ to find K clusters.

3. (Splitting clusters based on GMM). Apply the GMM on each initial cluster to find two new clusters for each one. Select an initial cluster with the highest KLD value as a candidate. Split the candidate cluster into two new clusters using $k-$means++. Set $K_0 = K_0 + 1$ and update the set $\mathscr{R}$. Apply KMSVM on the set $\mathscr{R}$. Accept or reject the proposal based on (3.4).

4. (Perturbation of representative points). Select a representative point at random and perturb it to another point (at random) within the corresponding initial cluster. Recompute clusters using KMSVM on new representative points, and accept or reject the proposal based on (3.4).

5. (Termination of algorithm). Repeat steps 3.- 4. until convergence.

As can be seen, Algorithm 2 uses the ratio (3.4) twice to accept or reject a new proposal: the first one (at Step 3.) is a supervised form of learning and the second one (Step 4.) is an unsupervised one, or random walk. More precisely, i) Step 3 is a proposal of a cluster split, in which the cluster having the highest KLD value is conditionally split into two new ones, while ii) Step 4 is a random perturbation of a representative point to update the cluster labels. In this step, the representative point of an initial cluster is switched to another point from the same initial cluster by a random walk. In this way, we give a chance to all points of that initial cluster to choose a different labelling. In both Steps 3

and 4, initial clusters with only one point are discarded as they will cause no change to the objective function.

3.1.5 Convergence of the proposed algorithm

The convergence of the proposed algorithm to the global solutions follows from the convergence of the simulated annealing algorithm. It is well-known that under some mild assumptions the simulated annealing method convergence to global solutions of continuous global optimization problems with probability 1 (see, [Locatelli2000], for details).

The proposed algorithm is the combination of a local search and the simulated annealing method. A local search algorithm is applied to find stationary points of the problem (2.15) and the simulated annealing method is applied to escape from these points and find points with better values of the objective function or point which are located in deeper basins of the objective function.

The objective function in the problem (2.15) has a finite number of local minimizers. As the simulated annealing method escapes such points with the probability 1 then we get that the proposed algorithm converges to the set of global minimizers of the problem (2.15) with probability 1.

3.1.6 Experiments

We compare the performance of the proposed algorithm with a pool of algorithms widely adopted for clustering: 1) the *k*-means++ algorithm (in the implementation of the scikit-learn Python machine learning library) [Arthur and Vassilvitskii2007]; 2) *mini-batch k*-means++ (initialized with *k*-means++) [Sculley2010]; 3) An algorithm based on a Dirichlet process Gaussian mixture model (DPGMM), from the same library [Görür and Rasmussen2010]; 4) the fuzzy c-means algorithm, from the fuzzy logic scikit Python machine learning library [Winkler *et al.*2011] and 5) the alternating maximum-margin clustering (MMC) [Zhang *et al.*2007]. The mini-batch *k*-means++ algorithm [Sculley2010] was proposed as an alternative to the *k*-means algorithm for clustering very large datasets. The advantage of this algorithm is a reduction of the computational load deriving from the use of sub-samples of fixed size. The DPGMM is an infinite mixture model with a Dirichlet Process as a prior distribution over the number of clusters. The fuzzy c-means

algorithm [Winkler *et al.*2011] is a clustering algorithm in which each data point is assigned to multiple clusters according to its membership grade. For hardening the final assignment, we assign the point to the cluster with the highest grade. The alternating MMC is a clustering algorithm that simply alternates steps of SVM classification and prediction, starting fron an initial, arbitrary (possibly random) assignment.

For ease of reference, hereafter we refer to the proposed algorithm as simulated-annealing MMC, or SAMMC for short; the $k-$means++ algorithm as KM; the *mini-batch* k-means++ algorithm as MBKM; the algorithm based on the Dirichlet process Gaussian mixture model as DPGMM; the fuzzy c-means algorithm as FCM; and the alternating MMC simply as MMC. All algorithms were implemented in Python 3.5 on a PC with an Intel(R) Core(TM) i5-5300, a CPU frequency of 2.3 GHz and 8 GB RAM.

To test the performance of the proposed algorithm and compare it with other algorithms, we have carried out experiments with sixteen datasets. A brief description of these datasets is given in Table 3.1, while their more detailed description can be found in [Lichman2001], with the exception of the dataset D15 which is described hereafter. In Table 3.1, m is the number of points and n is the number of features (dimensions).

The dataset D15 is from the Transport Accident Commission (TAC) which is a major accident compensation agency of the Victorian Government in Australia. It consists of a collection of 593,433 phone calls from 13,937 single TAC clients recorded by various operators over 5 years. The phone calls are made for different purposes including, but not limited to: compensation payments, recovery and return to work, different type of services, medications and treatments, pain, solicitor engagement and mental health issues. We refer to this data set as “Phone Calls” or briefly as “PCalls”.

The following preprocessing steps have been applied to the dataset D15 before its use in the experiments: 1) removal of numbers, punctuation, symbols and “stopwords”; 2) synonyms and misspelled words have been replaced with the base and actual words; 3) infrequently occurring words have been removed; 4) as common in text mining, we have also removed the most frequently occurring words such as names and addresses based on a predefined list. The data have then been projected to a vector space by using the tf-idf scheme [Beel *et al.*2016, Seifollahi *et al.*2017a]. The tf-idf is the most frequently applied weighting scheme among those approaches used to describe documents in the

vector space model [Beel *et al.*2016, Seifollahi *et al.*2017a], particularly in information retrieval. The idf part is applied to normalize the word frequencies by means of their relative frequency of presence in the document and over the entire collection. It reduces the weight (or significance) of common terms in the collection, ensuring that the matching of documents be more influenced by that of more discriminative words which have relatively low frequencies in a collection.

All datasets contain only numeric features and do not have missing values. In brief, the datasets were chosen so that i) the number of attributes would range from very few (2) to many (4,696); and ii) the number of data points would range from thousands (smallest: 2,310) to millions (largest: 4,178,504). Their diversity provides a thorough base for evaluation and comparison.

Table 3.1: Dataset summary.

Name	Datasets	m	n
D1	Image Segmentation	2,310	19
D2	Page Blocks	5,473	10
D3	Gas Sensor Array Drift	13,910	128
D4	EEG Eye State	14,980	14
D5	D15112	15,112	2
D6	Online News Popularity	39,797	58
D7	KEGG Metabolic Relation Network	53,413	20
D8	Shuttle Control	58,000	9
D9	Sensorless Drive Diagnosis	58,509	48
D10	MiniBooNE particle identification	130,065	49
D11	Skin Segmentation	245,057	3
D12	3D Road Network	434,874	3
D13	Cover Type	581,012	10
D14	Poker Hand	1,025,010	10
D15	Phone Calls (PCalls)	593,433	4,696
D16	Gas sensor array under dynamic gas mixtures	4,178,504	19

In order to implement Algorithm 7, one has to choose the parameter δ at Step 1. Values of δ close to one significantly increase computational time. Therefore, we decreased the value of this parameter with the increase of the number of data points. For the datasets (D1-D11), we set it between 0.05 and 0.20 and decreased it to 0.01 for the very large data sets (D12-D16).

For the implementation of the two alternating steps of the KMSVM in Algorithm 8, we used the MBKM and linear SVM algorithms from the scikit-learn module in Python. Algorithm 8 terminates if one of the following criteria is satisfied:

- (Temperature drop). If the temperature parameter in the simulated annealing method drops to a minimum user-defined value. We set the minimum temperature to 10^{-5}.
- (Number of iterations). If the number of iterations reaches a maximum number defined by the user. We set the maximum number of iterations to $20,000$.
- (Number of unsuccessful iterations). If the number of unsuccessful iterations exceeds a user-defined value (an unsuccessful iteration is an iteration that does not decrease the objective). It was set to $1,000$.
- (Time consumption). If the CPU time spent exceeds a pre-defined value. The maximum CPU time used by any algorithm is limited to: two hours for very large datasets including Cover Type, Poker Hand, Phone Calls, and Gas sensor array under dynamic gas mixtures; and half an hour for all the other datasets.

Results for comparing the SAMMC with the KM are given in Tables 3.2 and 3.3, while Tables 3.4 and 3.5 present results for all the clustering algorithms used in numerical experiments. These results are the best output out of 20 runs with different random initializations. In these tables, we have adopted the following notations:

- K_0 is the number of initial clusters before the use of the SAMMC;
- K is the number of final clusters;
- $f_{initial}$ is the objective function value of the MMC problem before the use of the SAMMC;
- f_{final} is the objective function value of the MMC problem after the use of the SAMMC;
- f_{best} (scaled by the number shown immediately after the name of the dataset) is the best value of the clustering objective function (3.8) among all algorithms used in this paper;
- E_A is the *error* (in %) of an algorithm A calculated as follows:

$$E_A = \frac{\bar{f} - f_{best}}{f_{best}} \times 100 \tag{3.6}$$

where $\bar{f}$ is the value of the objective function obtained by an algorithm A.

For all datasets, we computed up to 20 clusters. Since the proposed algorithm, SAMMC, requires an initial KM step in all cases, we report the objective function values before and after the use of the SAMMC (i.e., $f_{initial}$ and f_{final}) in Tables 3.2 and 3.3. Tables 3.4 and 3.5 instead report the errors of all the compared algorithms. While it is possible to use additional performance figures such as the Dunn and Davies-Bouldin indices, we found that they to tend to be very sensitive to outliers [Davies and Bouldin1979].

Results presented in Tables 3.2 and 3.3 show that the decrease of the objective function values over iterations generated by the proposed method, where $f_{initial}$ is the function value before the use of the SAMMC algorithm and f_{final} is the function value after the simulated annealing method's iterations. In fact, the initial function values $f_{initial}$ are the values obtained by applying the k-means++ algorithm. The parameter K_0 is the number of initial clusters used in the SAMMC algorithm; i.e. K_0 in Step 1. of Algorithm 8. Almost in all cases, the function values decrease and in most of them the amount of decrease is significant.

Tables 3.4 and 3.5 demonstrate the relative errors, formulated in (3.6), of algorithms. In these tables, E_1, E_2, E_3, E_4, E_5 and E_6 are the errors obtained using KM, MBKM, DPGMM, FCM, MMC and SAMMC, respectively. The best objective function value f_{best} among all algorithms are also presented in these figures. Results presented in both Tables 3.4 and 3.5 show that the proposed SAMMC algorithm is the most accurate among all algorithms, followed by MMC, KM, FCM, MBKM and DPGMM, respectively. It was able to find better values of the objective function for most datasets; with particularly significant improvements over datasets Image Segmentation, Page Block, KEGG Metabolic Relation Network, Shuttle Control, Sensorless Drive Diagnosis and 3D Road Network.

The proposed algorithm requires more, and in some cases significantly more, computational time than all other algorithms as it may call many times the simulated annealing step. However, the CPU time for the proposed method is reasonable even for very large datasets (two hours for datasets D15 and D16). In fact, this is a trade-off between the effectiveness and complexity. One can decrease the time by terminating the algorithm in early iterations, which may lead to less improvement over $k-$means++, or can have more effectiveness by running the algorithm for a long time.

Table 3.2: Objective function values for the SAMMC algorithm over datasets 1-8

K	$\#itr$	$f_{initial}$	f_{final}	$\#itr$	$f_{initial}$	f_{final}
	D1 ($\times 10^5$); $K_0 = 203$			D2 ($\times 10^6$); $K_0 = 849$		
3	2999	2.27535	1.92150	2000	6.92090	4.86011
5	2999	1.71600	1.66362	2000	4.21305	3.30935
7	2000	1.57640	1.49572	2000	3.01114	2.47169
10	2000	1.31979	1.27118	2000	2.76200	1.96377
12	2000	1.24097	1.16674	2000	2.17831	1.78075
15	2000	1.09656	1.08162	2999	2.06897	1.61311
17	2000	1.02597	1.02597	2000	1.73571	1.54352
20	2000	0.96981	0.96691	2000	1.63124	1.41833
	D3 ($\times 10^8$); $K_0 = 1974$			D4 ($\times 10^6$); $K_0 = 2250$		
3	2999	6.94459	6.77215	2000	1.69264	1.69264
5	2999	5.84274	5.28765	2000	1.13563	0.98754
7	2000	4.72804	4.62881	2000	0.87802	0.85799
10	2000	4.02730	4.02730	2000	0.75413	0.75413
12	2000	3.83332	3.70030	2000	0.71172	0.70948
15	2999	3.41227	3.36523	2000	0.67419	0.67418
17	2000	3.23031	3.22069	2000	0.65220	0.64999
20	2000	3.03252	3.03028	2000	0.62949	0.62949
	D5 ($\times 10^7$); $K_0 = 1975$			D6 ($\times 10^9$); $K_0 = 1763$		
3	2999	5.54539	5.54065	1907	3.57554	3.55978
5	2000	4.03324	4.03281	1845	2.56188	2.28574
7	2000	3.41973	3.41973	1826	2.05394	1.90084
10	2999	2.86699	2.86601	1789	1.68326	1.56733
12	2000	2.62155	2.62132	1764	1.47500	1.42740
15	2000	2.33309	2.33309	1717	1.34470	1.27022
17	2000	2.18249	2.18249	1691	1.26208	1.19329
20	2000	2.02283	2.02283	1642	1.13187	1.12003
	D7 ($\times 10^6$); $K_0 = 1930$			D8 ($\times 10^6$); $K_0 = 1646$		
3	1567	2.80643	1.86366	709	2.51994	1.90781
5	1244	1.63321	1.29238	647	2.46973	1.99779
7	1184	1.28721	1.04727	629	2.40504	1.60107
10	973	0.94686	0.80424	584	1.79075	1.51039
12	963	0.94071	0.74028	571	1.75150	1.39149
15	884	0.82894	0.66603	566	1.72041	1.20582
17	852	0.71239	0.63615	528	1.50015	1.23395
20	843	0.66856	0.59669	548	1.39315	1.11643

The datasets can be divided in two groups based on their dimensionality. The first group contains the datasets with small dimensionality (≤ 10): Page Blocks, D15112,

Table 3.3: Objective function values for the SAMMC algorithm over datasets 9-16

K	$\#itr$	$f_{initial}$	f_{final}	$\#itr$	$f_{initial}$	f_{final}
	D9($\times 10^5$); $K_0 = 1889$			D10($\times 10^7$); $K_0 = 3195$		
3	482	8.24640	6.44864	453	4.98413	4.90467
5	333	6.50427	5.89485	456	3.90094	3.88245
7	331	5.86409	5.36534	451	3.25558	3.22204
10	331	5.53054	4.86941	450	2.82944	2.77394
12	326	5.10210	4.54399	447	2.62172	2.62172
15	315	4.49256	4.24934	440	2.54134	2.43630
17	318	4.31557	4.06649	439	2.37069	2.34742
20	321	4.01503	3.93085	442	2.25009	2.23405
	D11($\times 10^7$); $K_0 = 2892$			D12($\times 10^6$); $K_0 = 3526$		
3	363	1.17959	1.17271	151	2.53221	2.45714
5	338	0.87830	0.87783	124	1.57130	1.55483
7	325	0.71552	0.71552	121	1.19048	1.15124
10	302	0.57940	0.57381	120	0.89404	0.85363
12	295	0.53372	0.53317	120	0.75937	0.73588
15	293	0.49592	0.49114	119	0.63639	0.62200
17	291	0.45631	0.44613	117	0.73587	0.56791
20	287	0.41331	0.41206	115	0.52566	0.51077
	D13($\times 10^8$); $K_0 = 3568$			D14($\times 10^6$); $K_0 = 4219$		
3	322	6.84525	6.82126	223	7.73012	7.73012
5	315	5.31869	5.31693	224	7.10038	7.09909
7	314	4.73661	4.73574	227	6.66651	6.66651
10	316	4.12908	4.12908	251	6.15894	6.15712
12	315	3.88909	3.88898	264	5.95588	5.94458
15	313	3.55111	3.52572	250	5.63555	5.63519
17	310	3.41428	3.41428	242	5.49580	5.49443
20	309	3.23589	3.22004	283	5.35294	5.35294
	D15($\times 10^4$); $K_0 = 1558$			D16($\times 10^{10}$); $K_0 = 3812$		
3	383	5.27089	5.27052	41	2.19410	2.19334
5	384	5.25709	5.25586	40	1.75273	1.75209
7	380	5.25277	5.24668	40	1.52607	1.52607
10	376	5.23029	5.22822	38	1.25024	1.25024
12	379	5.22126	5.21523	38	1.15669	1.14279
15	375	5.20914	5.20049	38	1.00832	0.99651
17	362	5.20057	5.20057	39	0.95343	0.92735
20	349	5.18291	5.17676	39	0.85912	0.85175

Shuttle Control, Skin Segmentation, 3D Road Network, Cover Type and Poker Hand. The number of points in these datasets ranges from $15,112$ to $1,025,010$. Results pre-

Table 3.4: Clustering errors obtained with the compared algorithms over datasets 1-8; for compactness of notation, E_1, E_2, E_3, E_4, E_5 and E_6 are the errors obtained using KM, MBKM, DPGMM, FCM, MMC and SAMMC, respectively

K	f_{best}	E_1	E_2	E_3	E_4	E_5	E_6	f_{best}	E_1	E_2	E_3	E_4	E_5	E_6
	D1 ($\times 10^5$)							D2 ($\times 10^6$)						
3	1.92065	18.46	0.34	35.42	0.00	12.56	0.04	4.86011	42.40	7.06	29.41	11.57	38.75	0.00
5	1.66362	2.83	3.07	22.57	1.29	1.55	0.00	3.30935	27.30	1.84	67.69	5.55	16.96	0.00
7	1.49572	5.39	7.81	31.29	1.01	2.85	0.00	2.47169	21.82	18.03	90.63	18.67	17.68	0.00
10	1.27072	3.86	0.00	49.03	1.76	2.53	0.03	1.96377	40.64	32.27	109.14	13.35	28.72	0.00
12	1.16674	6.36	1.00	55.16	3.31	0.41	0.00	1.78075	22.32	10.16	108.26	13.85	15.95	0.00
15	1.08162	1.38	2.95	58.82	2.02	1.36	0.00	1.61311	28.26	46.55	142.47	1.19	14.22	0.00
17	1.0236	0.23	0.00	62.20	5.55	0.23	0.23	1.54352	12.45	52.57	88.48	0.54	7.22	0.00
20	0.96691	0.30	0.35	89.45	4.74	0.30	0.00	1.41833	15.01	86.40	123.33	0.62	9.38	0.00
	D3 ($\times 10^8$)							D4 ($\times 10^6$)						
3	6.77215	2.55	0.07	56.14	1.52	1.70	0.00	1.69264	0.00	153.30	24.26	0.00	0.00	0.00
5	5.28765	10.50	2.32	59.50	2.79	6.51	0.00	0.98754	14.99	182.78	15.92	0.18	10.57	0.00
7	4.62881	2.14	1.75	77.33	1.78	1.47	0.00	0.85799	2.33	377.50	11.05	2.47	2.29	0.00
10	4.0273	3.35	4.37	64.53	3.74	2.15	0.00	0.74556	0.40	336.08	53.54	3.95	0.39	0.00
12	3.7003	3.59	6.99	66.01	3.62	1.84	0.00	0.70699	0.32	159.20	24.02	6.55	0.00	0.00
15	3.36523	1.40	0.01	66.60	3.41	0.07	0.00	0.67114	0.00	284.78	69.80	9.92	0.00	0.00
17	3.22069	0.30	6.80	43.15	6.42	0.27	0.00	0.64949	0.34	513.23	35.70	11.16	0.00	0.00
20	3.03252	0.07	10.25	66.34	2.64	0.07	0.00	0.62333	0.00	306.83	81.85	11.81	0.00	0.00
	D5 ($\times 10^7$)							D6 ($\times 10^9$)						
3	5.53855	0.13	0.07	24.29	0.00	0.10	0.04	3.55978	0.59	7.50	37.91	0.00	0.22	0.15
5	4.02884	0.11	1.64	142.09	0.00	0.09	0.10	2.28574	12.08	13.27	144.65	0.14	10.71	0.00
7	3.41973	0.00	1.77	14.89	1.37	0.00	0.00	1.90084	8.05	5.17	128.04	1.69	6.83	0.00
10	2.86414	0.03	1.73	240.54	2.06	0.00	0.00	1.56733	7.39	0.64	222.86	8.29	3.81	0.00
12	2.62132	0.01	1.66	11.66	0.25	0.00	0.00	1.42740	3.33	2.88	227.17	10.70	2.74	0.00
15	2.32964	0.15	1.33	318.44	0.00	0.08	0.15	1.27022	5.86	2.93	298.94	7.63	3.72	0.00
17	2.17353	0.41	2.54	9.45	0.00	0.08	0.41	1.19320	5.76	6.11	239.74	12.15	0.00	0.00
20	2.00347	0.97	3.55	383.42	0.00	0.54	0.97	1.12003	1.06	4.24	335.47	13.85	1.06	0.00
	D7 ($\times 10^6$)							D8 ($\times 10^6$)						
3	1.86366	50.58	8.30	10.66	15.14	49.57	0.00	1.90781	34.47	2.91	20.36	0.00	18.65	2.39
5	1.29238	26.37	5.36	25.04	14.04	19.70	0.00	1.77075	42.88	5.87	14.64	0.00	24.39	3.43
7	1.04727	22.91	11.42	88.22	16.04	12.89	0.00	1.60107	50.21	12.03	30.62	0.43	48.79	0.00
10	0.80424	17.73	12.62	59.07	21.79	12.31	0.00	1.51039	18.56	4.61	23.33	4.19	14.51	0.00
12	0.74028	27.07	18.37	136.27	17.39	13.00	0.00	1.39149	25.87	10.68	39.76	9.84	16.44	0.00
15	0.66603	24.46	26.33	74.73	16.09	12.86	0.00	1.20582	42.68	9.65	34.08	24.37	21.75	0.00
17	0.63615	11.98	22.19	128.04	13.34	6.11	0.00	1.23395	21.57	9.50	53.30	18.83	19.33	0.00
20	0.59669	12.04	25.43	101.41	14.94	10.09	0.00	1.11643	24.78	25.16	53.49	29.26	24.78	0.00

sented in Tables 3.4 and 3.5 show that SAMMC improves the results for most of them, with a particularly strong improvement on Page Blocks, Shuttle Control and 3D Road Network. The second group contains datasets with larger number of attributes: Image Segmentation, Gas Sensor Array Drift, EEG Eye State, Online News Popularity, KEGG Metabolic Relation Network, Sensorless Drive Diagnosis, MiniBooNE particle identification, Phone Calls and Gas sensor array under dynamic gas mixtures. The number of dimensions in these datasets ranges from 14 to 4,696. Results presented in Tables 3.4

Table 3.5: Clustering errors obtained with the compared algorithms over datasets 9-16; for compactness of notation, E_1, E_2, E_3, E_4, E_5 and E_6 are the errors obtained using KM, MBKM, DPGMM, FCM, MMC and SAMMC, respectively

K	f_{best}	E_1	E_2	E_3	E_4	E_5	E_6	f_{best}	E_1	E_2	E_3	E_4	E_5	E_6
	D9 ($\times 10^5$)							D10 ($\times 10^7$)						
3	6.44864	30.29	2.85	6.05	0.00	20.58	2.40	4.90467	1.62	49.06	93.91	41.52	1.58	0.00
5	5.65453	15.03	0.00	11.23	0.73	7.91	4.25	3.88245	2.29	67.57	107.18	0.00	0.32	1.80
7	5.11723	14.59	0.00	27.83	1.30	14.43	4.84	3.22204	1.04	84.64	43.40	97.54	0.55	0.00
10	4.86941	13.58	3.01	20.64	4.68	6.86	0.00	2.77394	2.00	105.57	179.43	123.64	0.82	0.00
12	4.49953	13.39	0.00	38.67	3.47	10.97	0.98	2.62172	0.00	131.61	61.21	133.11	0.00	0.00
15	4.24934	5.72	0.12	15.91	15.31	5.67	0.00	2.43630	4.31	124.53	215.33	148.27	2.16	0.00
17	4.06649	6.12	0.42	41.43	6.68	4.38	0.00	2.31230	2.52	0.00	76.76	160.27	2.07	1.51
20	3.93085	2.14	1.86	14.36	21.44	1.11	0.00	2.23405	0.72	138.92	204.42	167.56	0.08	0.00
	D11 ($\times 10^7$)							D12 ($\times 10^6$)						
3	1.17271	0.59	2.36	16.11	0.59	0.28	0.00	2.45714	3.06	0.24	1.61	2.11	2.78	0.00
5	0.87783	0.05	2.52	12.32	0.01	0.05	0.00	1.55483	1.06	6.56	0.93	0.66	1.03	0.00
7	0.71549	2.66	2.59	65.33	2.57	0.00	0.00	1.15124	3.40	0.44	188.85	2.36	3.33	0.00
10	0.57382	0.97	13.07	37.44	3.08	0.67	0.00	0.85363	4.73	3.66	14.28	2.33	4.12	0.00
12	0.53317	0.10	4.42	51.60	2.61	0.01	0.00	0.73587	3.19	3.67	457.07	1.50	2.93	0.00
15	0.47046	5.71	0.00	65.30	2.75	5.54	1.32	0.62200	2.31	1.04	35.41	1.30	2.28	0.00
17	0.45377	4.79	0.41	48.42	3.86	0.00	0.00	0.56791	2.27	2.70	607.59	1.05	1.55	0.00
20	0.41271	1.31	4.52	69.23	6.37	0.01	0.00	0.51077	2.91	1.70	48.86	0.85	2.72	0.00
	D13 ($\times 10^8$)							D14 ($\times 10^6$)						
3	6.82126	0.35	0.74	51.93	0.28	0.22	0.00	7.73012	0.00	0.56	0.88	1.07	0.00	0.00
5	5.30732	0.21	3.91	95.28	0.00	0.14	0.18	7.09909	0.02	0.47	3.92	1.44	0.01	0.00
7	4.72561	0.24	2.36	118.52	0.00	0.18	0.19	6.66651	0.00	1.48	7.80	8.00	0.00	0.00
10	4.12908	0.00	2.59	50.03	0.33	0.00	0.00	6.15712	0.03	0.90	13.65	10.83	0.00	0.00
12	3.88312	0.16	1.80	165.64	0.00	0.09	0.09	5.94458	0.19	0.93	15.12	10.52	0.00	0.00
15	3.52572	0.72	1.21	192.69	2.08	0.68	0.00	5.63519	0.01	1.78	19.47	11.70	0.00	0.00
17	3.41428	0.00	2.99	191.43	1.04	0.00	0.00	5.49443	0.03	1.77	19.94	17.41	0.00	0.00
20	3.22004	0.49	1.98	220.24	1.71	0.14	0.00	5.35294	0.00	0.98	22.20	20.46	0.00	0.00
	D15 ($\times 10^4$)							D16 ($\times 10^{10}$)						
3	5.26584	0.01	0.22	0.11	0.10	0.00	0.00	2.19334	0.03	0.25	109.71	0.08	0.02	0.00
5	5.25389	0.03	0.49	0.08	0.30	0.00	0.00	1.75209	0.04	0.25	157.66	0.05	0.03	0.00
7	5.24107	0.12	0.46	0.19	0.40	0.00	0.00	1.52476	0.02	0.00	170.02	0.62	0.00	0.02
10	5.22501	0.04	0.77	0.23	0.60	0.00	0.00	1.24970	0.00	0.87	266.32	1.45	0.00	0.00
12	5.21195	0.12	0.67	0.22	0.76	0.08	0.00	1.12905	1.21	0.86	147.42	1.18	0.00	0.00
15	5.19658	0.17	1.16	0.26	1.02	0.00	0.00	0.99651	1.19	1.99	330.68	4.74	0.30	0.00
17	5.18806	0.05	1.08	0.00	0.95	0.00	0.05	0.92565	2.81	4.08	183.29	2.55	0.00	0.00
20	5.17138	0.12	1.15	0.43	1.32	0.00	0.00	0.85090	0.87	0.43	418.83	3.83	0.00	0.00

and 3.5 show that SAMMC was able to improve over the other algorithms and that this improvement is remarkable in most cases.

For further analysis, Figures 3.1 show the objective function values for algorithms SAMMC and KM over a subset of the datasets when the number of clusters varies. Since the SAMMC algorithm is initialized with KM, they also show how much the SAMMC has been able to improve over its initial KM clustering. Figures 3.2 show the objective function values for algorithms SAMMC and KM over the same subset when the number

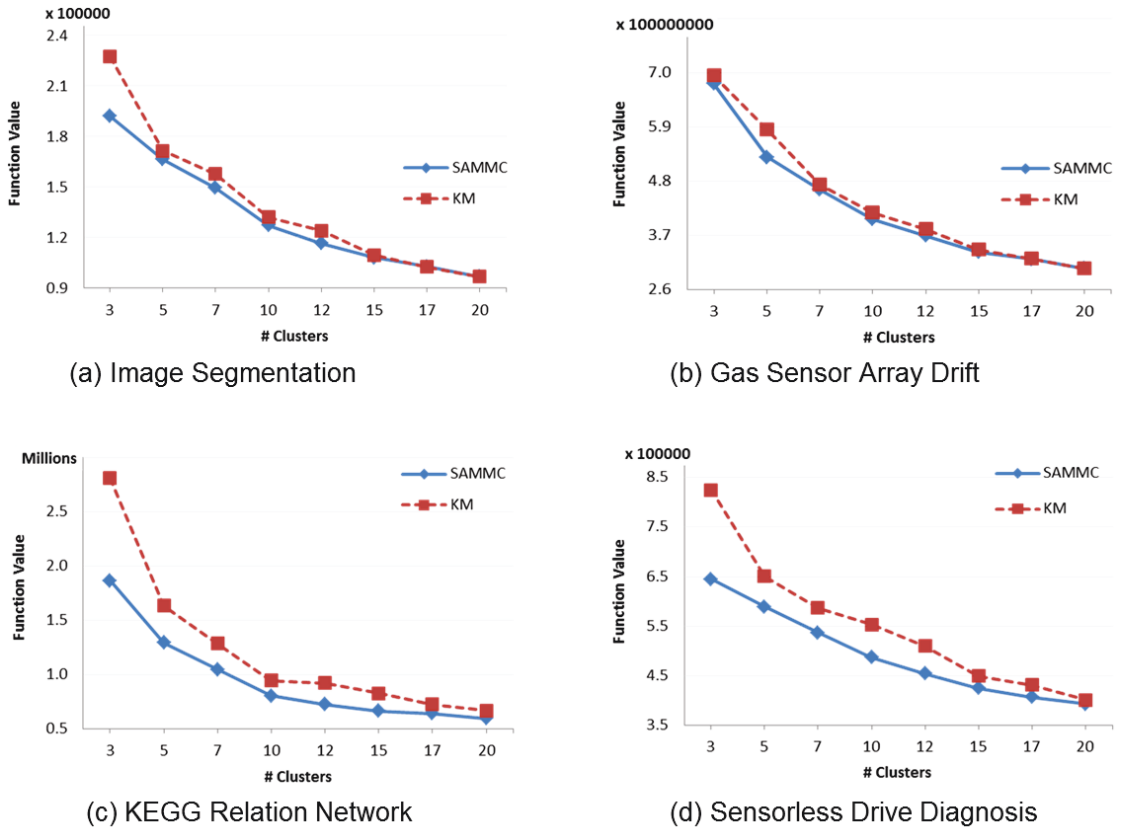


Figure 3.1: Objective function values for SAMMC and KM with varying number of clusters (subset of four datasets).

of iterations varies. As can be seen from these figures, the proposed method significantly decreases the objective function values within the first 500 iterations.

Figure 3.3 shows the variation of the objective function value over different initial solutions. In particular, "SAMMC-lower value" is the average value *minus* the standard deviation over 20 different initial solutions, while "SAMMC-upper value" is the average value *plus* the standard deviation; "km-lower value" and "km-upper value" are the corresponding values for KM. In all experiments, KM is used as the initial solution for SAMMC. Please note that not only the average for KM is always higher, but its variation is much higher.

To analyze the importance of Steps 3. and 4. in the overall SAMMC algorithm, we have removed each step in turn and re-run the algorithm (what is commonly called an *ablation analysis*). We call "SAMMC-pert" the algorithm obtained from SAMMC by removing Step 3. (splitting proposal) and "SAMMC-gmm" the algorithm obtained from SAMMC by removing Step 4. (perturbation proposal). The results show that both steps are important. It should be noted that only a few steps are accepted using ratio (3.4). For

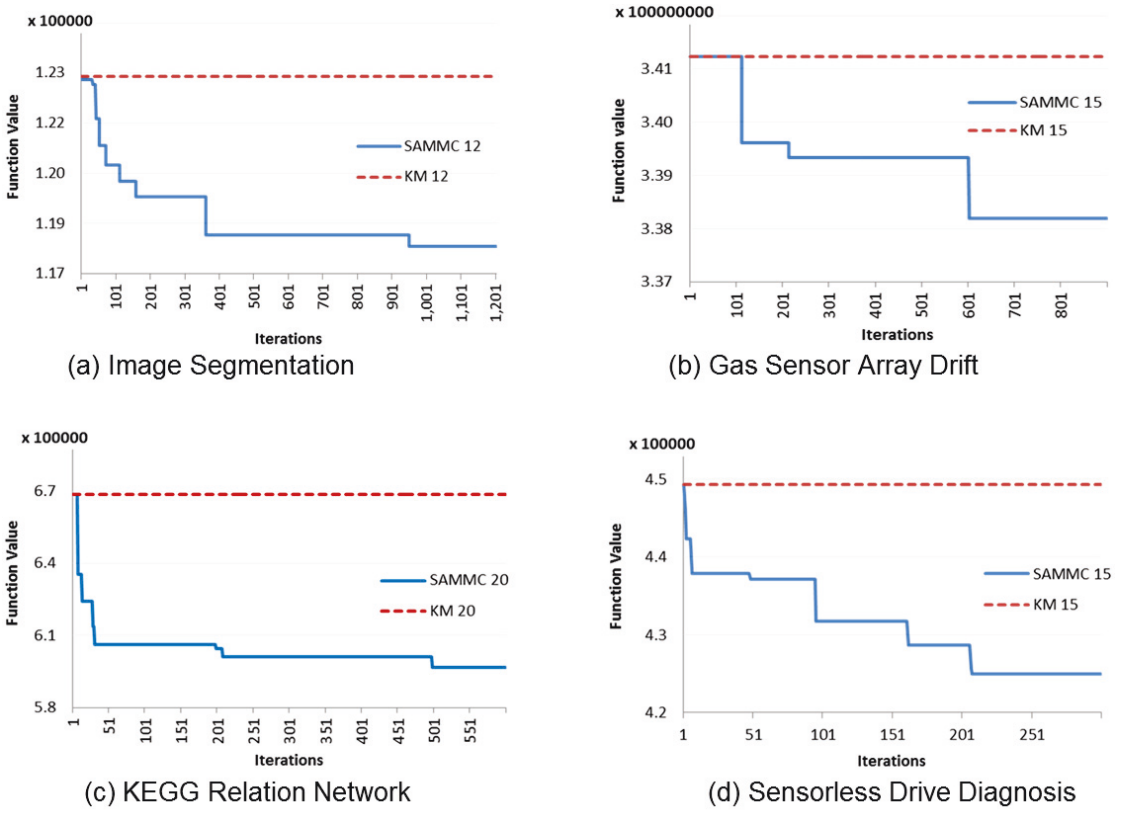


Figure 3.2: Objective function values for SAMMC and KM with varying number of iterations (subset of four datasets); notations "SAMMC k" and "KM k" stand for SAMMC and KM with K clusters.

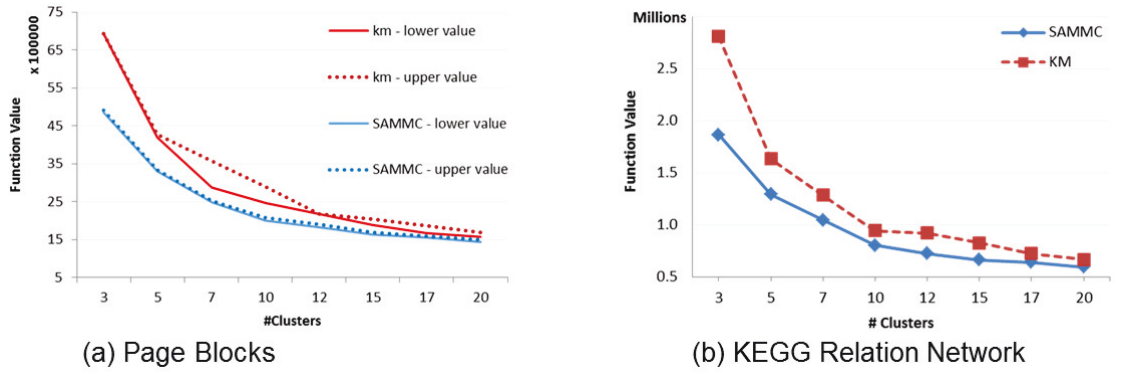
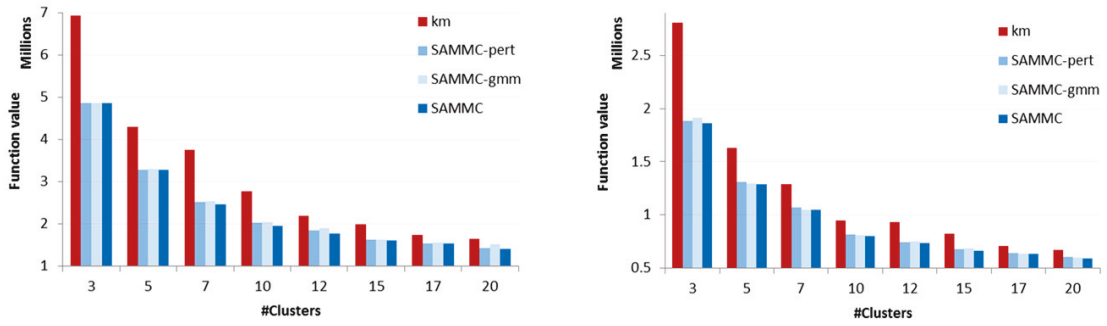


Figure 3.3: Variation of the objective function value over 20 initial solutions. "SAMMC-lower value" and "SAMMC-upper value" are the plot of the average function value of SAMMC minus and plus the standard deviation, respectively; likewise, "km-lower value" and "km-upper value' are the corresponding values for KM.



(a) Page Blocks

(b) KEGG Relation Network

Figure 3.4: Objective function values by removing specific steps from the proposed algorithm (ablation analysis). "SAMMC-pert" and "SAMMC-gmm" are the function values by removing Steps 4. and 3. from SAMMC algorithm, respectively;"km" and "SAMMC" stand for KM and SAMMC algorithms.

example, in Fig 3.4-(a) the average number of steps used by "SAMMC-pert", "SAMMC-pert" and "SAMMC" are 10, 8 and (10,4), respectively, where (10,4) means 10 and 4 successful repeats of Steps 3 and 4, respectively, in "SAMMC".

3.2 An incremental algorithm for clustering document collections

Given a large unlabeled document collection, the aim of this section is to develop an algorithm to distribute this collection into clusters of similar documents. This problem is formulated as an optimization problem and an incremental algorithm is developed for its solution. The use of the incremental approach allows one to design an efficient procedure for finding good starting points for solving nonconvex problem of document clustering. Then we apply the k-means algorithm for solving clustering starting from these points.

The main idea is the following: instead of working with K clusters from the start, we add the clusters one by one in successive iterations. At each iteration, we select the initial position of the added cluster by using a partitioning approach that is described in Section 3.2.2. This approach enjoys a performance guarantee that proves key for the accuracy of the overall algorithm. After the addition of the new initial cluster, a conventional k-means algorithm is called to re-optimize all the clusters (3.11). Then, the algorithm proceeds to the next iteration, until all clusters have been added.

3.2.1 Problem formulation

Given the vector space model, the document vectors may be represented as $x^1, x^2, \ldots, x^M$, with each $x^i \in R^V$. Recall that M is the total number of documents and V stands for the number of unique words in the vector space model. A clustering of the document collection is its partitioning into the disjoint subsets $X^1, X^2, \ldots, X^K$, *i.e.*

$$\bigcup_{j=1}^{K} X^j = \{x^1, x^2, \ldots, x^M\} = X \quad \& X^j \cap X^l = \emptyset, \quad j \neq l.$$

In SKM, the data are projected onto the unit sphere. Dhillon et al. [Dhillon *et al.*2001] have used the popular tf-idf scheme which reads out as (normalized) term frequency-inverse document frequency [Salton and Buckley1988]. The tf-idf normalization implies that $\|x^i\| = 1$, i.e., each document vector lies on the surface of the unit sphere in R^V. The K-clustering (or K-partition) problem is formulated as the following optimization problem:

$$\begin{cases} \min & f_K(c) \\ \text{subject to} & c = (c^1, \ldots, c^K) \in R^{VK}, \end{cases} \tag{3.7}$$

where

$$f_K(c^1, \ldots, c^K) = \sum_{x \in X} \min_{j=1,\ldots,K} d(c^j, x). \tag{3.8}$$

Here, $c^1, \ldots, c^K \in R^V$ are cluster centers and the function $d : R^V \times R^V \to R_+$ is the similarity measure, R_+ is the set of nonnegative numbers.

The function f_K is called the *K-th clustering objective function*. The similarity measure d is defined using the cosine measure, that is for $c, x \in R^V$,

$$d(c, x) = 1 - \cos(x, c) = 1 - \frac{\langle x, c \rangle}{\|x\| \|c\|}, \tag{3.9}$$

where $\langle x, c \rangle$ stands for the inner product of x and c and $\|\cdot\|$ is the Euclidean norm in R^V.

Since all document vectors are normalized it is also required that for each cluster center $c^i \in R^V$ is also normalized that is: $\|c^i\| = 1,\ i = 1, \ldots, K$. In this case one has the following

similarity measure d:

$$d(c,x) = 1 - \langle x, c \rangle. \tag{3.10}$$

Then the K-clustering can be reformulated as the following constrained optimization problem:

$$\begin{cases} \min & f_K(c) \\ \text{subject to} & c = (c^1, \ldots, c^K) \in R^{VK}, \\ & \|c^i\| = 1, i = 1, \ldots, K. \end{cases} \tag{3.11}$$

The problem (3.11) is a nonconvex constrained optimization problem and its objective function is piecewise linear. Due to the minimum operation used in the definition of this function (see (3.8)), it is also nonconvex. Since the document collections usually contain hundreds of thousands of documents, objective function f_K has many local solutions. Furthermore, the typical number of words in these collections is thousands or even tens of thousands. Therefore, Problem (3.11) is a large-scale optimization problem. Finally, the feasible set in this problem is nonconvex and it is a thin set in the V-dimensional space. Such problems are highly challenging not only for global optimization techniques, but also for local optimization methods. In this paper, we use the spherical k-means algorithm as our algorithm of choice.

3.2.2 Initialisation of the cluster centers

The incremental algorithm proposed in this paper solves the clustering problem gradually by starting with one cluster and adding a new cluster at a time, up to the set number. Hereafter we describe the algorithm for determining the initial position of the cluster added at the K-th iteration.

Assume that the solution $c^1, \ldots, c^{K-1}$, $K \geq 2$ to the $(K-1)$-clustering problem is known. Denote by d^i_{K-1} the distance between x^i, $i = 1, \ldots, M$ and the closest cluster center among $K-1$ centers $c^1, \ldots, c^{K-1}$:

$$d^i_{K-1} = \min \left\{ d(c^1, x^i), \ldots, d(c^{K-1}, x^i) \right\}. \tag{3.12}$$

We will also use the notation d^x_{K-1} for $x \in \{x^1,\ldots,x^M\}$. Consider the following two sets:

$$S_1 = \left\{y \in R^V :\ d(y,x^i) \geq d^i_{K-1},\ \forall i \in \{1,\ldots,M\}\right\}, \tag{3.13}$$

$$S_2 = \left\{y \in R^V :\ \exists i \in \{1,\ldots,M\} \text{ such that } d(y,x^i) < d^i_{K-1}\right\}. \tag{3.14}$$

The set S_1 contains all points $y \in R^V$ which do not attract any point from the set X and the set S_2 contains all points $y \in R^V$ which attract at least one point from X. It is obvious that cluster centers $c^1,\ldots,c^{K-1} \in S_1$. Since the number K of clusters is less than the number of data points in the set X all data points which are not cluster centers belong to the set S_2 (because such points attract at least themselves) and therefore this set is not empty. Note that $S_1 \bigcap S_2 = \emptyset$ and $S_1 \bigcup S_2 = R^V$. This means by taking any point $y \in S_1$ as a starting point for the K-th cluster center will not decrease the value of the clustering function f_K. Therefore, starting points should not be chosen from the set S_1.

Take any $y \in S_2$. Then one can divide the set X into two subsets as follows:

$$\bar{B}_1(y) = \left\{x \in X :\ d(y,x) \geq d^x_{K-1}\right\}, \tag{3.15}$$

$$\bar{B}_2(y) = \left\{x \in X :\ d(y,x) < d^x_{K-1}\right\}. \tag{3.16}$$

The set $\bar{B}_2(y)$ contains all data points $x \in X$ which are closer to the point y than to their cluster centers and the set $\bar{B}_1(y)$ contains all other data points. Since $y \in S_2$ the set $\bar{B}_2(y) \neq \emptyset$. Furthermore, $\bar{B}_1(y) \bigcap \bar{B}_2(y) = \emptyset$ and $X = \bar{B}_1(y) \bigcup \bar{B}_2(y)$.

Define the K-th auxiliary clustering function

$$\bar{f}_K(y) = \sum_{i=1}^{M} \min\{d^i_{K-1}, d(y,x^i)\},\ y \in R^V. \tag{3.17}$$

The difference $z_K(y)$ between the value of the function $\bar{f}_K(y)$ and the value of the $(K-1)$-th clustering function $f_{K-1}(c^1,\ldots,c^{K-1})$ is:

$$z_K(y) = \frac{1}{M} \sum_{x \in \bar{B}_2(y)} \left(d^x_{K-1} - d(y,x)\right) \tag{3.18}$$

which can be rewritten as

$$z_K(y) = \frac{1}{M} \sum_{x \in X} \max \left\{0, d_{K-1}^x - d(y,x)\right\}. \tag{3.19}$$

The difference $z_K(y)$ shows the decrease of the value of the K-th cluster function f_K comparing with the value $f_{K-1}(c^1,\ldots,c^{K-1})$ if the point $(c^1,\ldots,c^{K-1},y)$ is chosen as the cluster center for the K-clustering problem.

If a data point $x \in X$ is the cluster center then this point belongs to the set S_1, otherwise it belongs to the set S_2. Therefore we choose a point y from the set $X \setminus S_1$. We take any $y = x \in X \setminus S_1$, compute $z_K(x)$ and introduce the following number:

$$z_{max}^1 = \max_{x \in X \setminus S_1} z_K(x). \tag{3.20}$$

Let $\gamma_1 \in [0,1]$ be a given number. We compute the following subset of X:

$$\bar{X}_1 = \left\{x \in X \setminus S_1 :\ z_K(x) \geq \gamma_1 z_{max}^1\right\}. \tag{3.21}$$

If $\gamma_1 = 0$ then $\bar{X}_1 = X \setminus S_1$ and if $\gamma_1 = 1$ then the set $\bar{X}_1$ contains data points with the largest decrease z_{max}^1.

For each $x \in \bar{X}_1$ we compute the set $\bar{B}_2(x)$ and its center $c(x)$. We replace the point $x \in \bar{X}_1$ by the point $c(x)$ because the latter is better representative of the set $\bar{B}_2(x)$ than the former. Denote by $\bar{X}_2$ the set of all such centers. For each $x \in \bar{X}_2$ we compute the number $z_K^2(x) = z_K(x)$ using (3.19). Finally, we compute the following number:

$$z_{max}^2 = \max_{x \in \bar{X}_2} z_l^2(x). \tag{3.22}$$

The number z_{max}^2 represents the largest decrease of the values $f_l(c^1,\ldots,c^{l-1},x)$ among all centers $x \in \bar{X}_2$ comparing with the value $f_{K-1}(c^1,\ldots,c^{l-1})$.

Let $\gamma_2 \in [0,1]$ be a given number. We define the following subset of $\bar{X}_2$:

$$\bar{X}_3 = \left\{x \in \bar{X}_2 :\ z_l^2(x) \geq \gamma_2 z_{max}^1\right\}. \tag{3.23}$$

If $\gamma_2 = 0$ then $\bar{X}_3 = \bar{X}_2$ and if $\gamma_2 = 1$ then the set $\bar{X}_3$ contains only centers x with the largest

decrease of the cluster function f_K.

All points from the set $\bar{X}_3$ are considered as starting points for solving problem (3.11). Therefore, their selection guarantees to maximally decrease the objective.

The algorithm for finding the initial cluster centers in solving Problem (3.11) can be summarized as follows:

Algorithm 9: Finding the set of starting cluster centers.

Input: The solution $(c^1,\ldots,c^{K-1})$ to the $(K-1)$-clustering problem.
Output: The set of starting cluster centers for the K-th cluster center.

1. (Initialization). Select $\gamma_1,\gamma_2 \in [0,1]$.
2. Compute z^1_{max} using (3.20) and the set $\bar{X}_1$ using (3.21).
3. Set $C_w = C_w \cup \{C_w^{l+1,1},\ldots,C_w^{l+1,2^l}\}$.
4. Compute z^2_{max} using (3.22) and the set $\bar{X}_3$ using (3.23).

3.2.3 An incremental clustering algorithm and its implementation

In this subsection we present an incremental algorithm for solving Problem (3.11). The steps of the algorithm is as follows:

We call the proposed Algorithm 10 the Spherical Modified Global k-means (SMGKM). The most important and time consuming step in this algorithm is Step 4 where Problem (3.11) is solved starting from many initial cluster centers. For this problem, we use a conventional spherical k-means algorithm which allows us to automatically take into account the constraints of Problem (3.11).

3.2.4 Experimental results

In this section we present numerical results on the evaluation of the proposed method and compare it with the Spherical k-means algorithm (SKM), described in [Dhillon *et al.*2001]. We do not include comparison with hierarchical clustering algorithms as these algorithms have not been widely used in text mining. The reason of using the SKM for comparison is the similarity of the SKM with our proposed method in which both algorithms use spherical space and k-means as the base.

Algorithm 10: An incremental clustering algorithm.

Input: The collection of documents $X = \{x^1, \ldots, x^M\}$.
Output: The set of K cluster centers $\{c^1, \ldots, c^K\}, K > 0$.

1. (Initialization). Compute the center $c^1 \in R^V$ of the set X. Set $l := 1$.
2. (Stopping criterion). Set $l := l + 1$. If $l > K$ then stop. The K-partition problem has been solved.
3. (Computation a set of starting points for the next cluster center). Apply Algorithm 9 to compute the set $\bar{X}_3$ of starting point for the l-th cluster center.
4. (Computation a set of cluster centers). For each $\bar{y} \in \bar{X}_3$ take $(c^1, \ldots, c^{l-1}, \bar{y})$ as a starting point, solve Problem (3.11) and find a solution $(\hat{y}^1, \ldots, \hat{y}^l)$. Denote by $\bar{X}_4$ a set of all such solutions.
5. (Computation of the best solution). Compute

$$f_l^{min} = \min\left\{f_l(\hat{y}^1, \ldots, \hat{y}^l) : \ (\hat{y}^1, \ldots, \hat{y}^l) \in \bar{X}_4\right\}$$

   and the collection of cluster centers $(\bar{y}^1, \ldots, \bar{y}^l)$ such that

$$f_l(\bar{y}^1, \ldots, \bar{y}^l) = f_l^{min}.$$

6. (Solution to the l-partition problem). Set $c^j := \bar{y}^j, \ j = 1, \ldots, l$ as a solution to the l-th partition problem and go to Step 2.

To test and compare the proposed algorithm, we have carried out experiments with six datasets. A brief description of these datasets is given in Table 3.6 and more details of the datasets and preprocessing are given in Table 3.6.

Table 3.6: Dataset summary.

Datasets	M	V
1. Cora	2,240	2,319
2. Associated Press (APress)	2,246	4,994
3. WebKB	4,199	2,153
4. Reuters	12,902	1,313
5. Phone Calls (PCalls)	13,937	2,696
6. 20 Newsgroups (20Newsg)	18,774	3,103

The Cora data set [McCallum *et al.*2000] consists of the abstracts and references of approximately 34,000 computer science research papers; of these, we selected a subset of 2410 papers categorized into one of seven subfields of machine learning.

The Associated Press (APress) collection [Harman1992] contains Associated Press

news stories from 1988 to 1990. The original data includes over 200,000 documents with 20 categories. The sample AP data set from [Blei *et al.*2004], which is sampled from a subset of the TREC AP collection contains 2,246 documents.

The WebKB data set [WebKB] consists of approximately 6000 web pages from computer science departments of various university, divided into seven categories: student, faculty, staff, course, project, department and other. In this paper, we use the four most populous entity-representing categories: student, faculty, course, and project, which all together contain 4,199 pages.

The Reuters data set [Lewis1997] was originally collected by Carnegie Group, Inc. and Reuters, Ltd. in the course of developing the CONSTRUE text categorization system. It consists of 21,578 news stories appeared on the Reuters newswire in 1987. We use a subset of data containing 12,902 documents which are manually assigned to 135 categories.

The 20 Newsgroups dataset (20Newsg) [Rennie2008] contains postings to Usenet newsgroups. The postings are organized by content into 20 different newsgroups with about 1000 messages from each newsgroup and are therefore well suited for text clustering. This collection consists of 18,774 non-empty documents distributed evenly across 20 newsgroups.

Finally, The Phone Calls dataset (PCalls) is from the Transport Accident Commission (TAC) which is a major accident compensation agency of the Victorian Government in Australia. It consists of a collection of 593,433 phone calls from 13,937 single TAC clients recorded by various operators over 5 years. The phone calls are made for different purposes including, but not limited to: compensation payments, recovery and return to work, different type of services, medications and treatments, pain, solicitor engagement and mental health issues.

The following preprocessing steps have been applied to all datasets before their use in the experiments: 1) removal of numbers, punctuation, symbols and "stopwords"; 2) synonyms and misspelled words have been replaced with the base and actual words (for the PCalls dataset); 3) sparse terms (95% sparsity or more) and infrequently occurring words have been removed; 4) we have also removed generic words (for the PCalls dataset) such as names and addresses based on a predefined list. The data have then been projected to a vector space by using the popular tf-idf scheme.

Tables 3.7 and 3.8 show the best objective function value, f_{best}, and relative errors, E_1 and E_2 for the SKM and SMGKM respectively, where the relative error is defined as:

$$E_i = \frac{f_i - f_{best}}{f_{best}} \times 100 \tag{3.24}$$

where f_i is the value of the clustering function obtained by i-th algorithm. In most cases, SMGKM demonstrate better performance, i.e., low values for the objective function in terms of relative errors, and in some cases the differences are significant. When the number of clusters is small, SKM performs slightly better than SMGKM (in some cases) but the differences are not significant.

Table 3.7: The function value and relative errors

K	Associated Press			Cora			WebKB		
	f_{best}	E_1	E_2	f_{best}	E_1	E_2	f_{best}	E_1	E_2
10	1509.66	0.00	0.15	1475.39	0.00	0.06	2607.65	0.00	0.45
12	1481.68	0.00	0.25	1455.01	0.16	0.00	2567.58	0.00	0.27
15	1456.63	0.00	0.26	1423.41	0.00	0.29	2505.60	0.00	0.09
17	1435.98	0.56	0.00	1406.54	0.00	0.21	2455.62	0.34	0.00
20	1411.64	0.66	0.00	1384.11	0.44	0.00	2408.00	0.65	0.00
25	1380.19	0.95	0.00	1351.01	0.49	0.00	2342.28	0.30	0.00
30	1355.94	0.97	0.00	1324.20	0.89	0.00	2285.21	0.25	0.00
35	1337.06	0.90	0.00	1300.97	1.28	0.00	2229.02	0.62	0.00
40	1316.64	0.95	0.00	1279.14	1.61	0.00	2183.48	0.68	0.00
45	1300.28	0.53	0.00	1262.07	1.79	0.00	2143.49	0.68	0.00
50	1286.99	1.03	0.00	1248.15	1.73	0.00	2099.22	1.01	0.00
55	1274.34	0.70	0.00	1235.44	1.95	0.00	2064.24	1.05	0.00
60	1263.40	0.85	0.00	1223.60	2.48	0.00	2035.21	1.16	0.00
65	1254.37	1.06	0.00	1213.49	1.85	0.00	2009.59	1.64	0.00
70	1245.54	0.82	0.00	1204.21	1.96	0.00	1986.34	1.58	0.00
75	1235.14	0.52	0.00	1195.76	2.13	0.00	1960.96	1.67	0.00
80	1227.43	0.78	0.00	1187.62	1.63	0.00	1939.72	1.74	0.00
85	1220.38	0.04	0.00	1180.06	1.67	0.00	1915.72	2.42	0.00
90	1213.26	0.30	0.00	1172.98	1.72	0.00	1896.69	2.43	0.00
95	1203.35	0.00	0.21	1165.74	1.88	0.00	1879.04	2.64	0.00
100	1198.26	0.00	0.13	1159.05	2.01	0.00	1862.89	2.90	0.00

To further compare and assess the quality of the clusters generated by the algorithms, we apply two well-known cluster validity indices: the Dunns and DaviesBouldin validity indices.

Table 3.8: The function value and relative errors

K	Reuters			Phone Calls			20 NewsGroups		
	f_{best}	E_1	E_2	f_{best}	E_1	E_2	f_{best}	E_1	E_2
10	4586.69	0.00	0.06	9493.95	0.00	0.01	12910.54	0.01	0.00
12	4458.53	0.00	0.37	9388.49	0.00	0.16	12772.45	0.00	0.01
15	4289.97	0.00	0.02	9237.82	0.10	0.00	12590.80	0.12	0.00
17	4215.26	0.24	0.00	9144.10	0.53	0.00	12488.07	0.32	0.00
20	4119.90	0.87	0.00	9029.94	0.43	0.00	12367.28	0.15	0.00
25	3984.14	0.69	0.00	8867.60	0.20	0.00	12182.07	0.00	0.09
30	3871.80	1.17	0.00	8729.13	0.23	0.00	12042.95	0.24	0.00
35	3785.61	1.37	0.00	8614.72	0.05	0.00	11898.11	0.00	0.13
40	3719.43	1.41	0.00	8491.54	0.34	0.00	11777.05	0.00	0.27
45	3663.73	0.63	0.00	8385.29	0.23	0.00	11690.82	0.00	0.05
50	3607.86	0.84	0.00	8297.62	0.20	0.00	11559.44	0.13	0.00
55	3555.87	1.20	0.00	8214.63	0.21	0.00	11455.58	0.34	0.00
60	3512.71	1.43	0.00	8139.41	0.15	0.00	11362.46	0.30	0.00
65	3471.35	0.72	0.00	8065.04	0.00	0.06	11282.29	0.30	0.00
70	3431.43	0.95	0.00	7976.79	0.00	0.36	11200.54	0.53	0.00
75	3402.23	1.26	0.00	7931.58	0.00	0.07	11125.10	0.66	0.00
80	3374.66	0.80	0.00	7862.77	0.04	0.00	11044.67	0.60	0.00
85	3344.92	1.07	0.00	7806.99	0.18	0.00	10982.90	0.66	0.00
90	3321.59	1.16	0.00	7756.88	0.14	0.00	10926.60	0.72	0.00
95	3293.53	1.47	0.00	7707.74	0.08	0.00	10874.15	0.49	0.00
100	3271.34	1.15	0.00	7640.79	0.36	0.00	10823.80	0.63	0.00

The Dunn's validity index is defined as

$$I(D) = \max_{i=1,\cdots,K} \left\{ \min_{j=1,\cdots,K; j\neq i} \left\{ \frac{d(c^i, c^j)}{\max_{l=1,\cdots,K} r(c^l)} \right\} \right\} \tag{3.25}$$

where $d(c^i, c^j)$ is the distance between centers c_i and c^j. The $r(c^l)$ is the radius of the l-th cluster center and is defined as

$$r(c^l) = \max_{x \in X^l} \|c^l - x\|, \tag{3.26}$$

where K is the number of clusters. The Dunns cluster validity measure maximizes the inter-cluster distances and minimizes the intra-cluster distances. Therefore, the number of clusters that maximizes $I(D)$ can demonstrate the optimal number of the clusters.

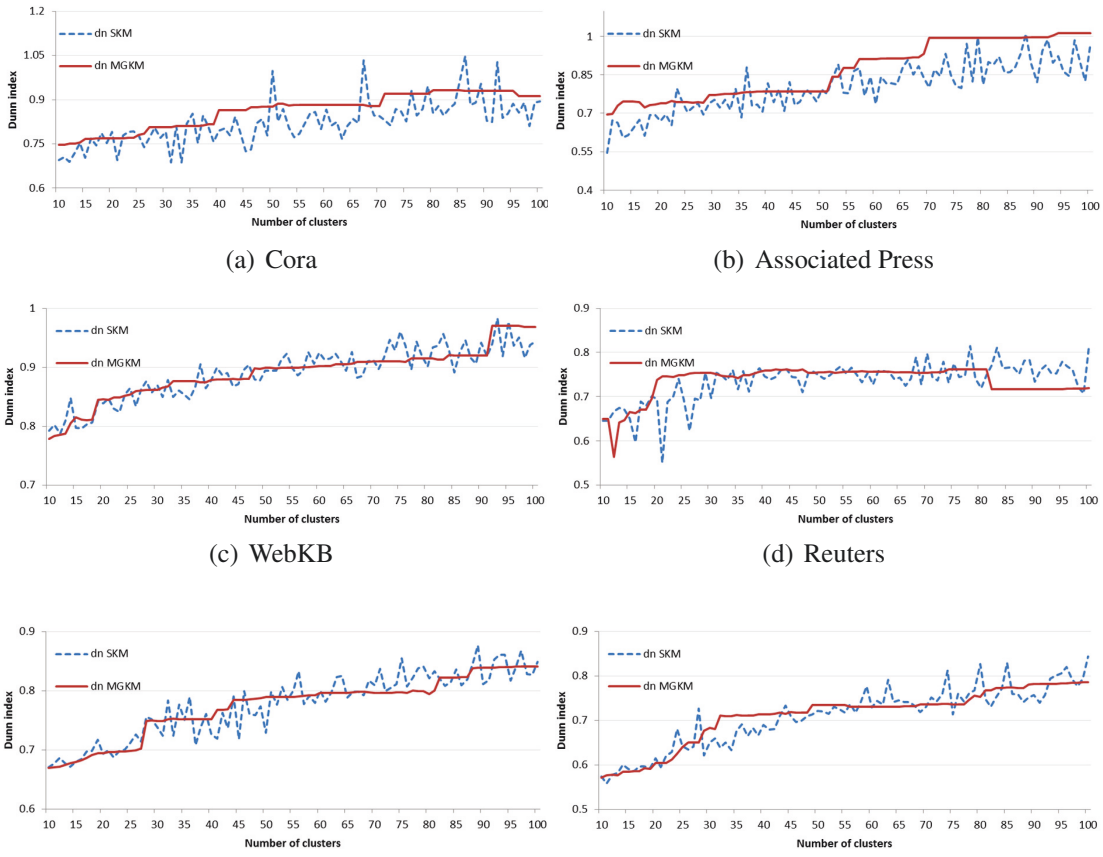
The Davies-Bouldin validity index is a measure of within-cluster to between-cluster

separation

$$I(DB) = \frac{1}{K}\sum_{i=1}^{K} \max_{j=1,\cdots,K;j\neq i} \frac{S_K(X^i)+S_K(X^j)}{d(c^i,c^j)} \tag{3.27}$$

where K is the number of clusters, $S_K(X^l)$ is the average distance of all data points from the cluster X^l to their cluster center c^l and $d(c^i,c^j)$ is the distance between i-th and j-th cluster centers. Smaller values for the $I(DB)$means that clusters are compact and far from each other. Therefore, the smaller $I(DB)$, the better clustering.

Figures 3.5(a) – 3.5(f) display the Dunn's cluster validities for SKM and SMGKM as the number of clusters increases from 10 to 100. Here, the dot lines correspond to the SKM and solid lines to the SMGKM. Terms "dn SKM" and "dn SMGKM" stand for Dunn index values using the SKM and SMGKM, respectively. Graphs for the SMGKM are much more stable than graphs for the SKM when the number of clusters increases. The reason is likely that the SMGKM exploits an incremental scheme which adds one cluster each time, while the SKM calculates all clusters from scratch.



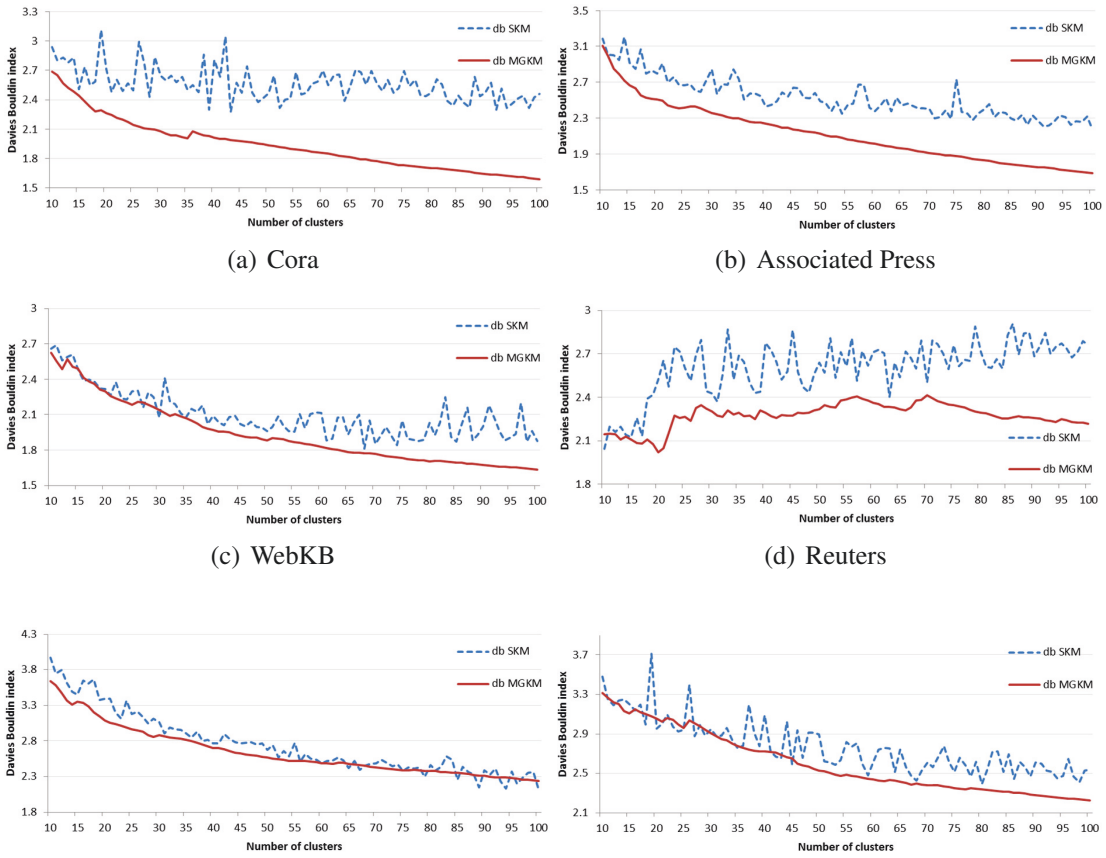
(a) Cora (b) Associated Press

(c) WebKB (d) Reuters

(e) Phone Calls (f) 20 NewsGroups

Figure 3.5: Cluster validity (Dunn) index for datasets 1 – 6

Figures 3.6(a) – 3.6(f) show the Davies-Bouldin indices for SKM and SMGKM as the number of clusters increases. Terms "db SKM" and "db SMGKM" stand for Davies-Bouldin index values using the SKM and SMGKM, respectively. The graph for SMGKM is more stable, confirming stability in Figures 3.5(a) and 3.5(f). These figures also demonstrate significant improvements of SMGKM over the SKM, as the graphs for SMGKM are below (much, when the number of clusters increases) those for SKM in almost all cases.



(a) Cora (b) Associated Press

(c) WebKB (d) Reuters

(e) Phone Calls (f) 20 NewsGroups

Figure 3.6: Cluster validity (Davies Bouldin) for datasets 1 – 6

Table 3.9 reports the optimal number of clusters using the Dunn and Davies-Bouldin measures as well as the total CPU time spent by SKM and SMGKM. K_1^* and K_2^* stand for the optimal number of clusters using the SKM and SMGKM and t_1 and t_2 are the times (cumulative CPU) consumed by SKM and SMGKM, respectively. Despite using a less efficient environment for coding, the times for SMGKM have been lower than those for SKM, except on Cora.

Table 3.9: The optimal number of clusters and cumulative CPU time for computing up to 100 clusters. K_1^* and K_2^* stand for the optimal number of clusters using SKM and SMGKM, respectively, and t_1 and t_2 for the time

dataset	Dunn index		DB index		Total CPU time	
	K_1^*	K_2^*	K_1^*	K_2^*	t_1	t_2
Cora	50	40	40	36	1024	1568
APress	77	70	72	74	2601	2088
WebKB	93	92	68	81	2501	1862
Reuters	29	39	10	20	5327	4745
PCalls	89	90	89	88	15861	11710
NewsG	79	81	68	77	27873	20280

3.3 Conclusion

In this chapter, two methods for clustering have been introduced and discussed. The first method is a simulated annealing maximum-margin clustering under a minimum sum-of-squares objective. The proposed algorithm leverages simulated annealing to escape local minima and a combination of $k-$means++ and SVM to provide high-quality local minima. By exploiting a two-stage organization, the proposed algorithm has been able to mollify the computational complexity of maximum-margin clustering and prove suitable for large-scale data. In the experiments, algorithms including the proposed one, are tested on real-world datasets, including text documents, with number of points ranging from thousands to millions and number of dimensions ranging from a few to thousands. The experimental results have provided clear evidence that the proposed method is able to achieve significant improvements of the objective function in comparison to popular clustering algorithms including k-means++, mini-batch $k-$means++, DPGMM, fuzzy c-means and MMC.

The second method uses an incremental algorithm for document clustering that is capable of finding "deeper" solutions. In the algorithm, a new cluster is added in turn starting from an initial position that is guaranteed to maximally decrease the objective function value. Clustering is performed in a spherical space, meaning that each solution is projected to the unit sphere to mitigate the potential bias from more frequent words.

In the experiments, the proposed method is compared to the spherical k-means algorithm (SKM) that can be regarded as state-of-the-art for document clustering. The results over six challenging text document datasets have shown that the proposed algorithm has

consistently outperformed SKM under a number of clustering indices (objective function value, Dunn and Davies-Bouldin). This gives a ground to believe that the proposed algorithm can prove beneficial for large-scale document clustering applications.

Chapter 4

Taxonomy-Augmented Features for Text Analytics

This chapter describes the main components of our methodology for generating features based on taxonomies, namely i) the approach for generating the hierarchy of word clusters, ii) the approach for extracting taxonomy-based features based on a set of predefined words, and iii) a comparison approach that generates the features directly from the clusters in the hierarchy.

4.1 Hierarchy of word clusters

The work of [Steinbach *et al.*2000] found that "bisecting" *k*-means can produce clusters that are both better than those of standard *k*-means and as good as (or better than) those produced by agglomerative hierarchical clustering. In plain terms, bisecting *k*-means joins *k*-means with divisive hierarchical algorithms and has been successfully applied to document clustering [Steinbach *et al.*2000].

For these reasons, in this chapter we employ a method similar to bisecting *k*-means with two modifications; 1) we use spherical *k*-means instead of standard *k*-means and 2) each cluster is split into two sub-clusters only if the number of its elements exceeds a predefined threshold. A cluster here is a collection of words that are similar to one another within the same cluster and are dissimilar to objects in other clusters. Spherical *k*-means projects data onto the unit sphere and has been shown to be an effective method for

document clustering [Dhillon *et al.*2001]. Building clusters in a hierarchical fashion has the advantages of reducing time complexity, increasing performance as well as implicitly producing a taxonomy of words, where words with similar semantics are expected to appear in the same branch or close branches of the hierarchy. Figure 4.1 illustrates our hierarchical word clusters. All data (word vectors) are, first, partitioned into two clusters, and from the first level down each cluster is iteratively partitioned into two new clusters until the level limit is reached.

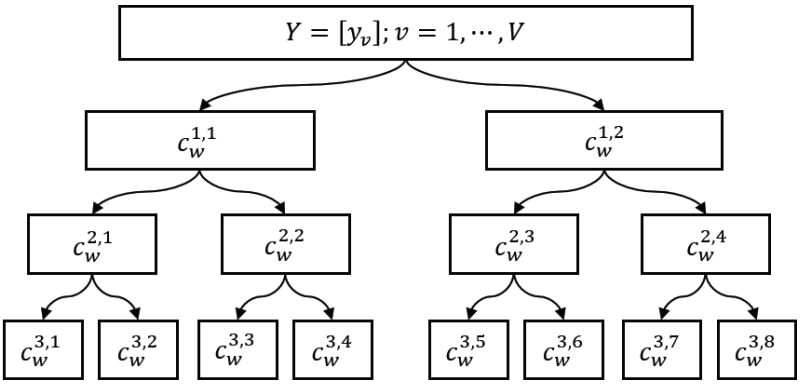


Figure 4.1: Three layers of the hierarchy of word clusters.

Let us note the hierarchy of word clusters as C_w. This expands as $C_w = [C_w^1, \ldots, C_w^L]$, where $C_w^l = \{C_w^{l,1}, \ldots, C_w^{l,2^l}\}$ is the set of clusters in the l-th level. The process of building the hierarchy of clusters is illustrated in Algorithm 11.

Algorithm 11: Computing the hierarchy of clusters in word space

Input: Word vectors $Y = [y_v]$; $v = 1, \ldots, V$; $y_v \in \mathbb{R}^{|W|}$.
Output: Hierarchy of word clusters C_w.

1. Set level number $l = 0$, a threshold δ as lower bound for splitting a cluster and $C_w = \emptyset$.
2. For $i = 1, \ldots, 2^l$:
   2.1. Partition $C_w^{l,i}$ into two new clusters as $C_w^{l+1,2i-1}$ and $C_w^{l+1,2i}$.
3. Set $C_w = C_w \cup \{C_w^{l+1,1}, \ldots, C_w^{l+1,2^l}\}$.
4. If the stopping criteria are satisfied, exit; otherwise, set $l = l + 1$ and repeat from Step 2.

We have two stopping criteria in Step 4.; either when l exceeds an integer value, L (the maximum number of levels in the hierarchy) or when the number of words in a cluster

is less than a predefined threshold, δ. Increasing the number of levels may increase the quality of the word clusters, hence enhancing the quality of the ensuing document vectors; however, it also increases the complexity and therefore a heuristic trade-off is needed. Some of the clusters at the lower levels may have no entities (i.e., they are empty) due to a small number of words ($< \delta$) in the corresponding upper layer.

4.2 Taxonomy-augmented features given the hierarchy of word clusters

A taxonomy can play a key role in document clustering by reducing the large number of features from thousands to a few tens only. The feature reduction process benefits from the taxonomy's semantic relations between words. Our utilization of the taxonomy differs from previous work, since we construct a hierarchy of word clusters and utilize it directly for document representation. In other terms, the documents are projected to a word cluster space by matching their contents to the hierarchy of word clusters. The process of feature extraction from documents using hierarchy of word clusters is given in Figure 4.2.

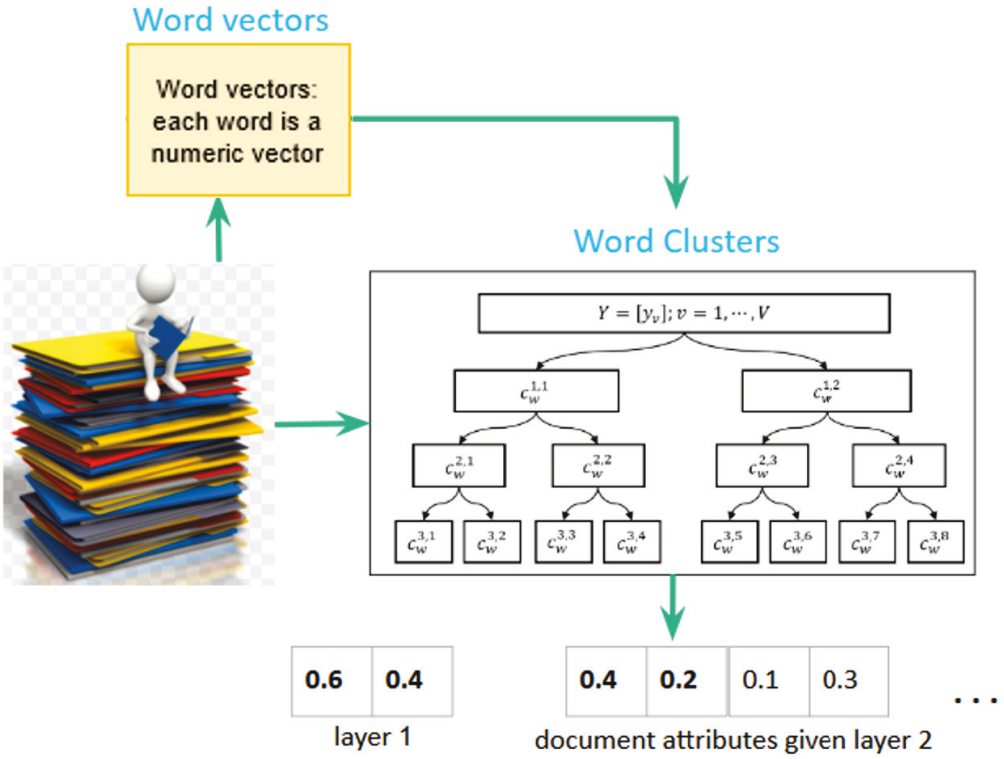


Figure 4.2: The process of extracting features via the hierarchy of word clusters.

Let us assume that $D = [d_1, \ldots, d_M]$ are M documents in word space. Each document can be further noted as $d_i = [d_i^1, d_i^2, ..., d_i^L]$, where L is the number of levels in the hierarchy of word clusters and $d_i^l = [d_i^{l,1}, d_i^{l,2}, ..., d_i^{l,2^l}]$. For simplicity, we store D in a two-dimensional matrix of size $M \times n$, where n is the overall number of clusters in all levels, i.e. $n = \sum_{l=1}^{L} 2^l$.

Algorithm 12: Taxonomy-augmented features

Input: Hierarchy of word clusters C_w.
Output: Document matrix $D \in \mathbb{R}^{M \times N}$

1. Set $D = [d_1, \ldots, d_M] = 0$ where $d_i \in \mathbb{R}^N$, $n = \sum_{l=1}^{L} 2^l$, and L is the number of layers in the hierarchy of word clusters.
2. For each document $d_i, i = 1, \ldots, M$:
   2.1. For each word w in document d_i:
      2.1.1. For level $l = 1, \ldots, L$:
         2.1.1.1. Find cluster index in C_w^l, j, such that $w \in C_w^{l,j}$.
         2.1.1.2. Set $d_i^{l,j} = d_i^{l,j} + x_{iw}$.
   2.2. For $l = 1 \ldots L$:
      2.2.1. Normalize d_i^l to one, i.e. $\|d_i^l\| = 1$.
3. Remove any features from D that have zero value across all documents.

If some of the clusters in the hierachy never appear in any of the documents in the collection, the corresponding features in D will all be zero. Therefore, we remove those features in Step 3. of Algorithm 12. In Step 2.1.1.2., $x_{i,w}$ corresponds to the document-term weight of the i-th document and word w, which is based on tf-idf weighting.

4.3 Taxonomy-augmented features given a set of words

In this section, we propose an alternative approach for representing the documents for clustering. In this approach, the documents are projected to a space of predefined words, and the feature dimensionality is the same as the size of this set of words. This approach is particularly suitable for short documents, where the total number of words per document is limited; typically, say, less than 100 words. For such short documents, the BoW features result in an extremely sparse matrix. In order to alleviate this issue, the representation of short documents must be enriched with information from the semantics of the words. The correlation between words and its use for measuring short-text similarities have been

studied in the literature, for example, in [Seifzadeh *et al.*2015]. Based on [Seifzadeh *et al.*2015], if two short segments do not have any common words, but words from the first segment appear frequently with words from the second segment in other documents, this implies that these segments are semantically related, and any measure of their similarity should be high.

Our proposal differs from previous works, including [Seifzadeh *et al.*2015], since we build features by combining the documents' content, the hierarchy of word clusters, and a set of predefined words. An intuition of the approach can be provided in these terms: for every word in a given document and for every level in the hierarchy, we find the cluster where the word belongs. We then retrieve all the predefined words that belong to that same cluster, and we increment their counters. The final counters of the predefined words are used as the feature vector to represent the document. In other terms, the predefined words can be seen as the "representative elements" of the clusters they belong to, and their counters are incremented every time a document's word falls in their cluster. To describe the process precisely, let us assume that $D = [d_1, \ldots, d_M]$ is the document matrix, where $d_i \in \mathbb{R}^n$ and n is the number of the predefined words. The process of extracting features from documents considering a set of predefined words is given in Figure 4.3, and the steps and details for generating the document features are described by Algorithm 13.

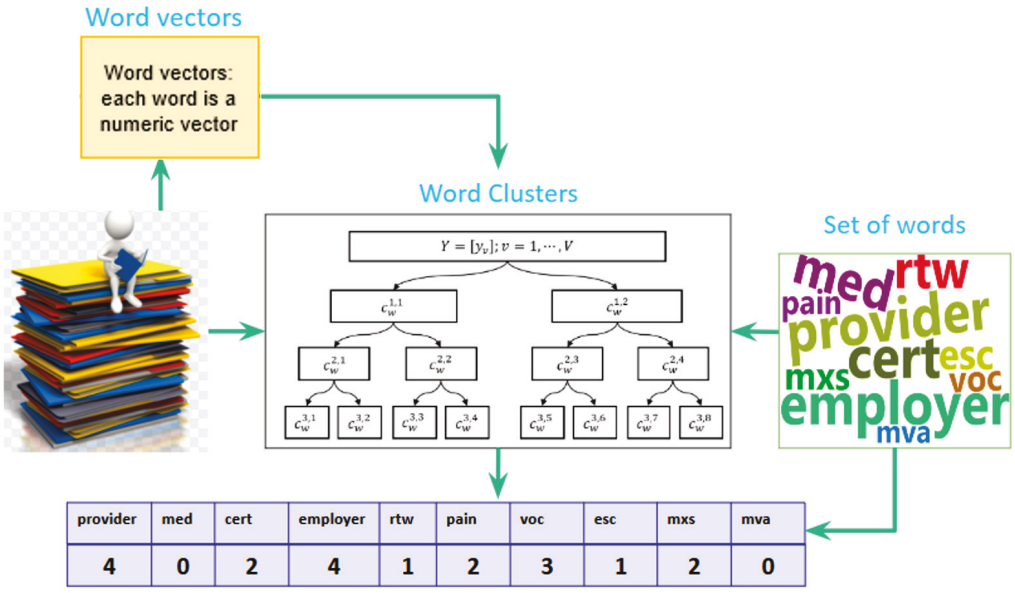


Figure 4.3: The process of extracting features via the hierarchy of word clusters and a set of predefined words.

The number of the indices in $I(l, j)$, Step 3., depends on how many of the predefined

Algorithm 13: Taxonomy-augmented features given a set of predefined words

Input: Hierarchy of word clusters C_w, set of predefined words $S = \{w_1, \ldots, w_n\}$.
Output: Documents $D \in \mathbb{R}^{M \times n}$

1. Set $D = [d_1, \ldots, d_M] = 0$, where $d_i \in \mathbb{R}^n$ and n is the size of the set of predefined words. Set $W_s = [w_s^1, \ldots, w_s^L] = \left[[w_s^{1,1}, w_s^{1,2}], \ldots, [w_s^{L,1}, \ldots, w_s^{L,2^L}]\right]$, and $w_s^{l,j} = \emptyset$.
2. For each word w in S :
   2.1 For level $l = 1, \ldots, L$:
      2.11 Find index of cluster in C_w^l, j, such that $w \in C_w^{l,j}$.
      2.12 Set $w_s^{l,j} = w_s^{l,j} \cup w$.
3. For each document $d_i, i = 1, \ldots, M$:
   3.1 For each word w in document d_i :
      3.11 For level $l = 1, \ldots, L$:
         3.111 Find cluster index in C_w^l, i.e. j such that $w \in C_w^{l,j}$.
         3.112 Retrieve all words of $w_s^{l,j}$ with corresponding indices in S, $I(l, j)$.
         3.113 Set $d_i^{I(l,j)} = d_i^{I(l,j)} + x_{i,w}$.

words are in the cluster of word w (from zero to, potentially, n). Rather than simple counters, the document features d_i^j add up the tf-idf weight of word w, $x_{i,w}$.

4.4 Taxonomy-augmented features for document clustering

In this section, we use two algorithms, described in previous sections, to solve clustering problem. The main benefits of these algorithms are using small number of features, while clustering quality remains in a good mode. To this aim, we use the three-step process. First, a word embedding technique is used to convert each distinct word to a vector of $|W|$ dimensions. Second, the word vectors are partitioned into a hierarchy of clusters, simply referred to as "word clusters", via a hierarchical clustering algorithm. Third, the individual documents are projected onto the word clusters to produce their taxonomy-based feature vectors.

To test and compare the proposed models, we have carried out experiments with three, diverse datasets. The first (*PCalls*) consists of phone call transcripts recorded by the

Transport Accident Commission (TAC), the accident compensation agency in Victoria, Australia. It contains a total of 59,048 transcripts from phone calls from 8,000 single clients, transcribed by various operators. The second (*WebKB*) is a classic clustering benchmark consisting of web pages from computer science departments of universities, [WebKB]. The last (*Reuters*) is a dataset of news stories published on the Reuters newswire in 1987. The main statistics of these datasets are given in Table 4.1. For more details, please refer to [Bagirov *et al.*2018] and references therein.

Table 4.1: Dataset summary.

Datasets	Dataset name	M	#words after pre-processing
D1	PCalls	8,000	6,684
D2	WebKB	4,199	2,153
D3	Reuters	12.902	1,313

The following preprocessing steps have been applied to each dataset before its use in the experiments: 1) removal of numbers, punctuation, symbols and "stopwords"; 2) replacement of synonyms and misspelled words with the base and actual words (only for the PCalls dataset); 3) removal of sparse terms (keeping 95% sparsity or less) and infrequently occurring words; 4) removal of uninformative words such as people names and addresses.

We have employed two document vector methods as the baselines for comparison; 1) the well-known tf-idf and 2) doc2vec which is based on the average of word embeddings. As word embeddings, we have used GloVe for its reported strong performance in a variety of tasks [Pennington *et al.*2014]. To learn the embeddings, we have used the following settings: the dimensionality was set to 200; the context window size was set to 12; and the number of training epochs was set to 1,000. For clustering, we have used two very popular algorithms, k-means++ and PAM (partitioning around medoids). In Algorithm 13, we set $n = 100$.

We compare the effectiveness of the model based on two complementary measures: connectivity and silhouette. The connectivity captures the "degree of connectedness" of the clusters and it is measured in terms of how many nearest neighbors of any given sample belong to other clusters. The connectivity value ranges between 0 and infinity and should be minimized. The silhouette measures the compactness and separation of the clusters and it ranges between -1 (poorly clustered observations) and 1 (well clustered

observations). To compute these measures, we have used clValid, the R package for cluster validity [Brock *et al.*2008].

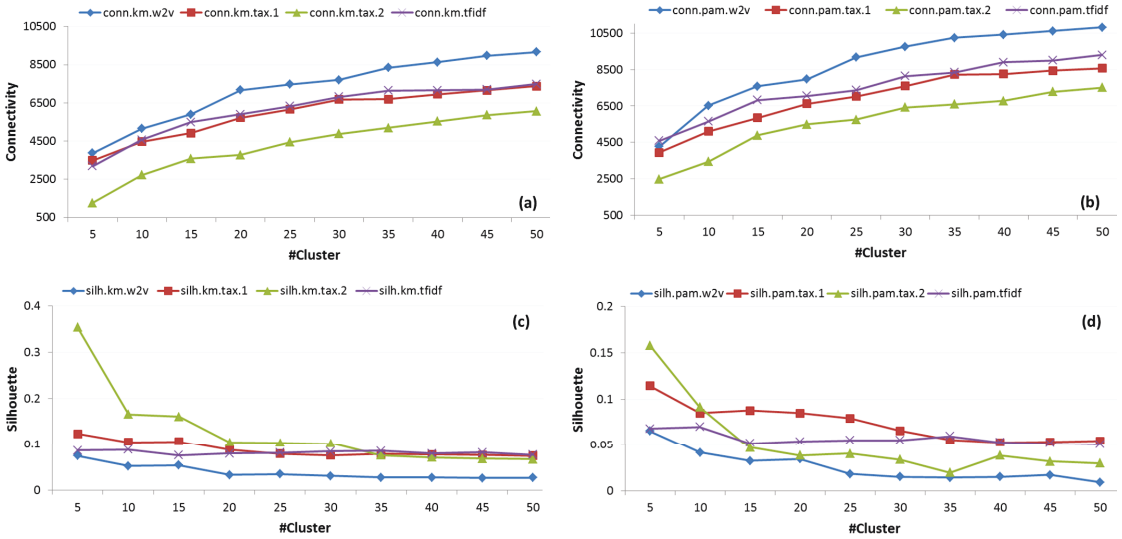


Figure 4.4: Connectivity and silhouette measures of all models for the PCalls dataset.

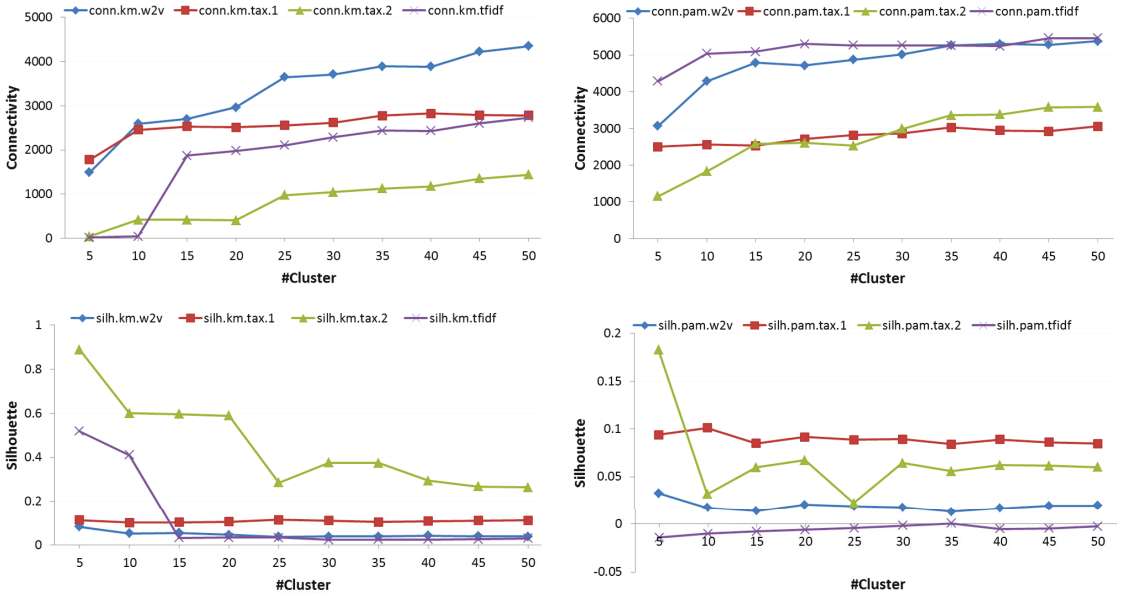


Figure 4.5: Connectivity and silhouette measures of all models for the WebKB dataset.

Figures 4.4 – 4.6 show the connectivity and silhouette measures for PCalls, WebKB and Reuters as a function of the number of the clusters. In these figures, "tfidf" stands for the BoW model, "w2v" for doc2vec, and "tax.1" and "tax.2" for our proposed models from Algorithms 12 and 13, respectively. In turn, "km" and "pam" stand for *k*-means++ and PAM, respectively. These results show that the proposed taxonomy-augmented models, "tax.1" and "tax.2", have performed better than the other two models over all datasets

for a large majority of cluster numbers and for both clustering algorithms. Among the conventional models, the performance of BoW is generally better than that of doc2vec. We do not formally compare the time complexity of the models, but we note that BoW is the most time-consuming due to its large number of features. Based on our experiments, it is approximately 10 times slower than the other models.

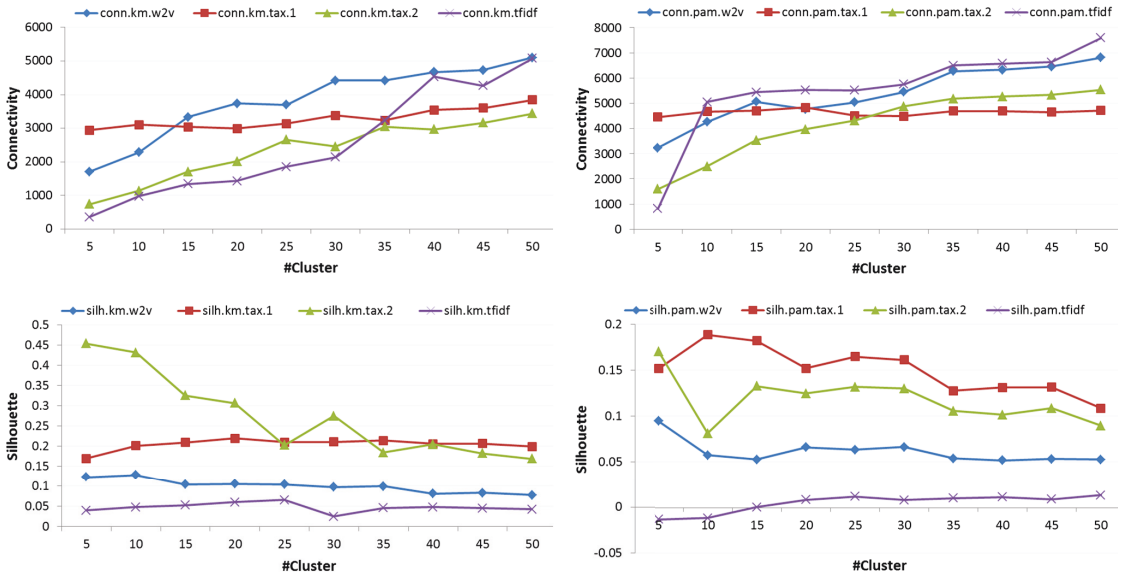


Figure 4.6: Connectivity and silhouette measures of all models for the Reuters dataset.

4.5 Taxonomy-augmented features for document classification

Document classification (or categorisation) is likely the most widespread automated task on textual data, with a vast array of applications such as sentiment analysis, ad targeting, spam detection, client relationships, risk assessment and medical diagnosis. A class is a subset of documents which are, in some sense, similar to one another and different from those of other classes, and the goal is to assign text documents to a predefined set of classes. Document classification has been extensively studied, especially since the emergence of the Internet where documents are typically created by an unverified variety of authors and with little metadata.

In this section, we use algorithm 13 to classification and semantic analysis to demonstrate its effectiveness for these tasks. We also leverage sets of predefined "words of interest" for the specific organization to greatly decrease the number of features to a very

manageable size, equal to the size of the set of predefined words. The feature extraction algorithm uses a three-step process: first, a word embedding technique is used to convert each distinct word to a vector of $|W|$ dimensions. Second, the word vectors are partitioned into a hierarchy of clusters, simply referred to as "word clusters", via a hierarchical clustering algorithm. Third, the individual documents are projected onto a space of predefined words whose size is equal to the size of this set.

To test and compare the proposed approach, we have carried out experiments with textual data from an accident support agency, the Transport Accident Commission (TAC) of the Victorian Government in Australia. The data consist of phone calls between the agency's clients and its claim managers, annotated by the managers into transcripts. Phone calls for single clients take place over time, so the data are suitable for monitoring the clients' progress, such as return to work and physical and emotional recovery. The phone calls typically cover a wide range of topics: for example, some are related to the client's health and recovery, while others are related to payments and compensation.

To conduct experiments on classification, we have merged all the phone call transcripts of each individual client into single documents and created four datasets (D1-4) based on the number of words per document and the number of documents itself. Datasets D1 and D2 consist, respectively, of 2,000 and 5,000 short-length documents, ranging from 20 words to 100 words per document. The TAC experts have indicated that such short-length documents are likely to come from clients with fewer phone calls overall and more rapid recovery. Datasets D3 and D4 are similar, but with larger-size documents ranging from 100 to 5,000 words each. The four datasets have been manually annotated by the TAC experts into a binary classification problem using labels "MH" and "NO MH" (label "MH" is used by the TAC to identify clients with a variety of mental health issues). All datasets are balanced in terms of number of samples per class.

The following preprocessing steps have been applied to each phone call before its use in the experiments: 1) removal of numbers, punctuation, symbols and "stopwords"; 2) replacement of synonyms and misspelled words with the base and actual words; 3) removal of sparse terms (keeping 95% sparsity or less) and infrequently occurring words; 4) removal of uninformative words such as people names and addresses.

To learn the word embeddings, we have used the following settings: the dimensionality

was set to 100; the context window size was set to 12; and the number of training epochs was set to 1,000. In Algorithm 13, we have set $n = 100$.

For the experiments on document classification we have employed two document features as the baselines for comparison: 1) the well-known tf-idf and 2) doc2vec which is based on the average of word embeddings. We have also used Algorithm 12 as another method for comparison. As word embeddings, we have used GloVe due to its reported strong performance in a variety of tasks [Pennington *et al.*2014]. For classification, we have compared three popular classifiers: 1) eXtreme Gradient Boosting (XGBoost) [Chen and Guestrin2016], 2) a support vector machine (SVM) with a radial basis function kernel, and 3) a random forest. We have used 10-fold cross-validation to report the accuracy: first, the dataset is randomly shuffled and split into 10 folds; then, in turn, each fold is used as the test set and the remaining nine as training set. All codes, including the packages for the classifiers, have been implemented in the R language in a Windows environment.

Figure 4.7 shows the average accuracies from the 10-fold cross-validation for datasets D1-4. Notations "xgb", "svm" and "rf" stand for classifiers XGBoost, SVM with RBF kernel and random forest, respectively. Notations "w2v", "tfidf", "tax.1" and "tax.2" stand for feature extraction models doc2vec, BoW with tf-idf weighting, taxonomy-augmented algorithm 12 and taxonomy-augmented algorithm 13, respectively.

Based on Figure 4.7, the model based on predefined words, "tax.2", performs better than the others in most cases, followed by "tax.1". The "tfidf" model performs very similarly to "tax.2" using "xgb"; however, it fails to produce accurate predictions with the other two classifiers, in particular with "svm". Amongst the classification methods, "xgb" performs the best, followed by "rf" and "svm", respectively.

In Figure 4.8 we also use the average of the absolute deviations (AVDEV) to explore the variability of the 10-fold cross-validation accuracies around their means. Figure 4.8 shows that the AVDEV for all methods are mostly in a very similar range, with the exception of "tfidf" using "rf" in D1 and D4 where the AVDEV are, undesirably, much higher.

We have not conducted a formal complexity analysis for the methods. However, we have noted that classification takes a very similar time with the different feature vectors, with the exception of "tfidf" that is considerably slower. If we include the time for feature extraction, "w2v" becomes the fastest, followed by "tax.1", "tax.2" and "tfidf", respec-

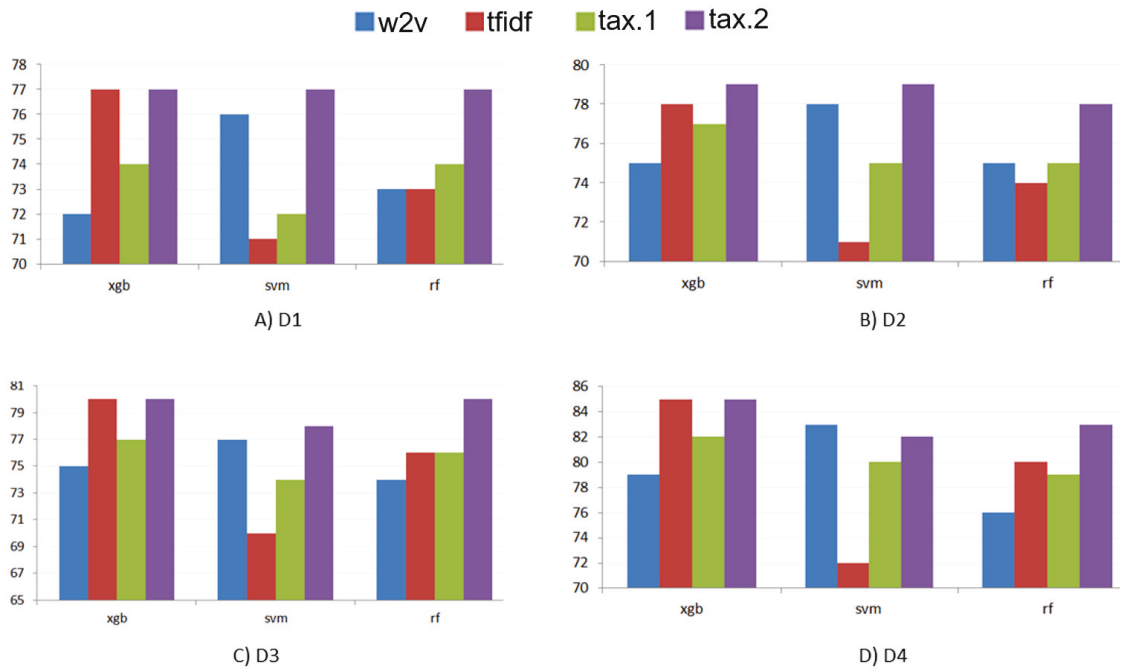


Figure 4.7: Average accuracy from 10-fold cross-validation. The horizontal axis maps the classifier and the coloured bars represent the feature vectors.

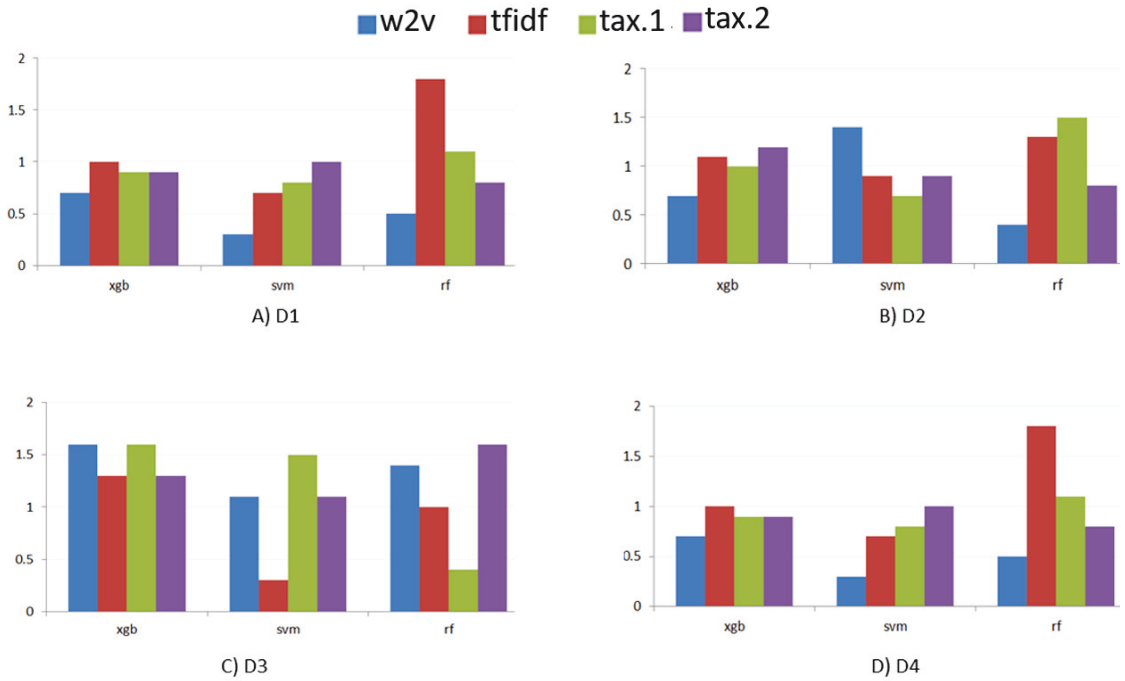


Figure 4.8: Average of the absolute deviations (AVDEV) of accuracies from their mean. Horizontal axis shows classification methods and vertical axis is the AVDEV value.

tively. Model "tfidf" is much slower due to its much larger dimensionality, in particular in conjunction with "svm" classification.

4.5.1 Semantic analysis

In this section, we illustrate the use of the taxonomy-augmented features of Algorithm 13 for the "semantic analysis" of TAC clients. The data for each client consist of several phone calls, where each phone call has been represented by a set of predefined words using Algorithm 13. In this section, we use $S = \{family, pain, provider, recovery, stress, upset\}$ as predefined words. In this way, each phone call is represented as a distribution over chosen words, which in turn are represented as distributions over the entire vocabulary. The concept looks very similar to that of *topic modelling*, where each document is represented as a distribution over topics and each topic is a distribution over the vocabulary. However, the two models differ notably in that the topics are latent variables without an a-priori semantic, whereas our predefined words are observed and can be chosen by experts.

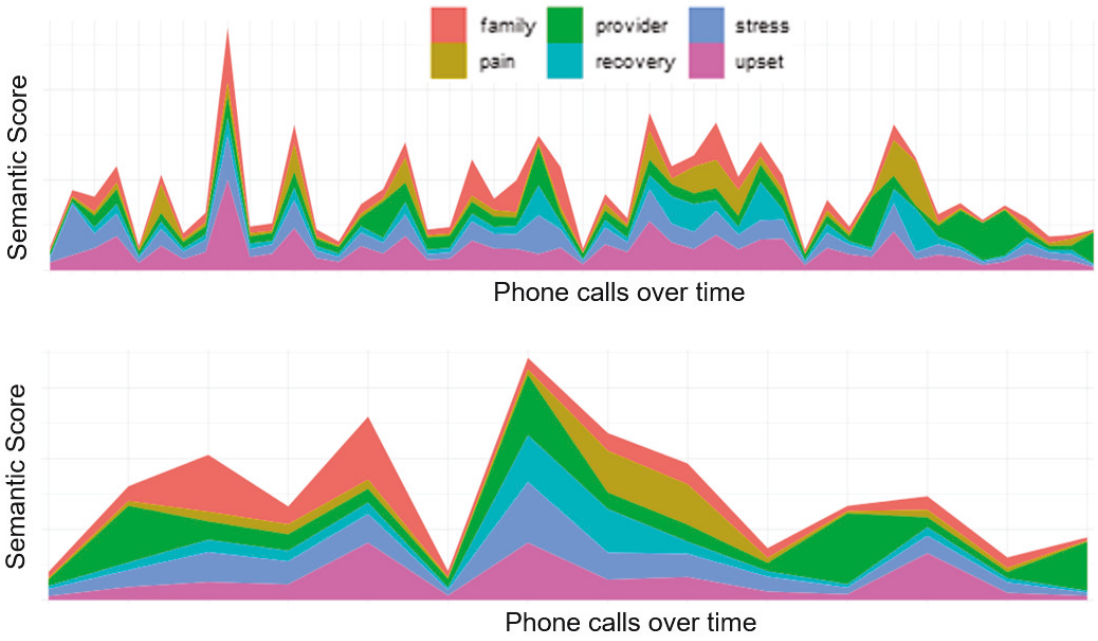


Figure 4.9: Evolution of the chosen features in the phone calls of two randomly-selected clients. The semantic scores are computed by Algorithm 13.

Figure 4.9 shows the semantic analysis of two clients as plots of the six chosen features along successive phone calls. The first client recorded a total of 48 phone calls over 38 months and the second client 14 over 29 months (a few phone calls have been removed because the number of words after preprocessing had fallen below a minimum set threshold). This figure should be regarded just as an illustrative example as any set of words or reasonable size can be used in this analysis.

4.6 Conclusion

In this Chapter, the application of two taxonomy-augmented feature extraction approaches has been examined in a variety of important document tasks such as classification, clustering, and semantic analysis. The approaches addresses two urgent challenges of conventional document representations, namely 1) their large number of features, and 2) the dismissal of the word ordering in the formation of the features. By amending these two shortcomings, the proposed models have proved able to provide a more compact and semantically-meaningful document representation and improve the tasks' performance.

In an original set of experiments on document classification, we have compared the proposed approach with two well-known methods, BoW/tf-idf and doc2vec, over four phone call datasets from an accident support agency. The results have shown that the proposed models have achieved better average accuracy in the large majority of cases, while their absolute deviations from the averages have kept almost the same. These improvements confirm the results obtained by the proposed models in experiments on document clustering. In addition, we have illustrated the usefulness of the proposed feature vector for the monitoring of individual progress over time.

Chapter 5

Topic Modelling

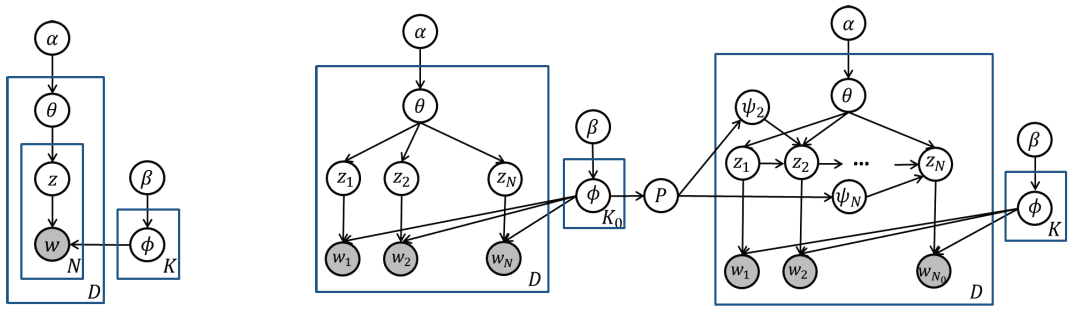
Topic models are unsupervised models to automatically discover the topics discussed in a collection of documents. Most topic modelling techniques follow the LDA principles and meet its main assumptions: 1) each topic is a mixture of words and 2) each text document is a mixture of topics. In this section, we present two novel approaches for topic modelling. The first approach is based on LDA and extends it by utilising prior knowledge and using a Markov process for topic generation. In the second approach, the procedure for creating the topics is cluster-based and is completely different from that of LDA and other mainstream topic models in the literature. However, the two main stated assumptions still apply to the proposed models.

5.1 A semi-supervised Markov topic model

In this section, we propose a topic model approach that follows the LDA principles and uses a Markov process for topic learning. Topic models can generate topics which are incoherent by human. To deal with this, we present a topic model which uses knowledge from prior topics. This prior knowledge can be extracted from an initial topic model. More precisely, we introduce a two-staged semi-supervised Markov topic model in which in the first stage a topic model will produce prior knowledge such as topic terms and the topical co-occurrence matrix. These outputs are used to build a final model in which topic assignment follows a Markov chain process.

The graphical representation of the model is displayed in Fig. 5.1(b). As shown in this

figure, it consists of two stages; i) prior topics and ii) final topic model. The aim of the first stage is to determine a subset of prior topics to calculate words' relations upon. More precisely, a transition matrix for the topic assignment in the second stage is calculated by looking at both subsequent words in the prior topics and co-occurrences of words in the collection. The frequent co-occurring words in the collection and in prior topics represent the more topically-related words - this is the main idea behind the proposed topic model, called semi-supervised hidden Markov topic model, or SHMTM, hereafter. In addition, the prior topics will also be used to extract a low-dimensional vocabulary for the final model.



(a) LDA (b) SHMTM

Figure 5.1: Graphical models of (a) LDA model and (b) the proposed model, SHMTM. The number of prior topics in SHMTM is K_0 ($K_0 \geq K$) and the vocabulary size is V_0 ($V_0 \leq V$).

5.1.1 Initial process

Frequently co-occurring words in collections can be much more topically-related than words without significant co-occurrence. To calculate words relations, we first extract a topic model with a high number, K_0, of topics using the LDA model. For each pair of words in the vocabulary, or in a lower-dimensional vocabulary using only the top words, we calculate the probability of words co-occurring in the collection given the prior topics. The words relations give us a measure to find how two subsequent words are topically related. This measure, called a "transition matrix" herewith for analogy with Markov chains, is used by the final topic model for topic assignment. The initialization procedure can be summed up as Algorithm 14.

The number of prior topics, K_0, is equal to or greater than the final number of topics, and there is no specific criterion for an optimal number of prior topics. K_0 can be considered as

Algorithm 14: Initial topic model for calculating transition matrix.

Input: Text corpus, number o f initial topics K_0.
Output: Transition matrix P.

1. Set the number of initial topics, K_0.
2. Find K_0 topics using the LDA model.
3. Select the top n_k words of each topic with high probabilities and set $W_0 = \{w_i^k\}; i = 1, \ldots, n_k; k = 1, \ldots, K_0$.
4. Calculate a transition probability matrix, $P = [P_{i,j}]_{i,j=1}^{V_0}$, between top words of each topic, where V_0 is the size of set W_0.

an upper bound for the possible topics and it may depend on the data set and application; we fix $K_0 = 200$. The number of top words n_k is also fixed to 20 based on the fact that only a few words from each topic have significant impact on the topics. The term $P_{i,j}$ in Step 4., is proportional to the probabilities of the i-th and j-th words in the collection where w_i and w_j are both top words in the same topic; they are calculated as:

$$P_{i,j} = \sum_k p(w_j^k, w_i^k) p(w_j^k), \tag{5.1}$$

where $p(w_j^k, w_i^k)$ is the probability of co-occurrence of words w_j and w_i in the collection (or related external data), and $p(w_j^k)$ is the probability (weight) of word w_j in the k-th topic. We calculate the score for any two words in the vocabulary by investigating the top n_k words of each topic. If two words do not appear as top words of any topics, a weight of zero will be assigned to the corresponding score, meaning that those words are not topically related (or independent given the topic). In contrast, if two words appear in many topics and frequently co-occur in the collection, they are more likely to be dependent words or topically-related words.

To leverage (5.1) as a transition probability matrix, one need to scale each row of P to interval $[0, 1]$. The diagonal of this matrix is also set to 1 to ensure that two (subsequent) equal words are assigned to the same topic. As P is not symmetric, i.e. $P_{i,j} \neq P_{j,i}$, one can consider the mean of P and the transpose of P, which is used in this paper. It is noted that only top words of prior topics are used to construct a transition matrix as well as a low-dimensional vocabulary for the final model.

5.1.2 Generative model

We propose a topic model where the topic assignment is formed in a semi-supervised way, i.e. based on a transition matrix obtained in the previous step. The central idea is that highly topically-related, subsequent words are generated from the same topics. However, the extent to which subsequent words are topically related is decided based on a Markov chain process. A transition matrix is first constructed using both prior topics and word co-occurrences in the collection, and topic assignment is then decided using the transition matrix and a random number drawn from a binomial distribution. The steps of the generative model are given in Algorithm 15.

Algorithm 15: Semi-supervised hidden Markov topic model (SHMTM).

Input: Text documents D and number of topics K.
Output: Topic-word matrix ϕ and document-topic matrix θ.

1. For each topic k in $\{1,\ldots,K\}$.
   1.1. Draw a distribution over words $\phi_k \sim Dir_{W_0}(\eta)$.
2. For each document d :
   2.1. Draw a vector of topic proportions $\theta_d \sim Dir_K(\alpha)$.
   2.2. For $n = 1$:
      2.2.1. Draw a topic assignment $z_{d,1} \sim Mult_K(\theta_d)$.
      2.2.2. Draw a word $w_{d,1} \sim Mult_{W_0}(\phi_{z_{d,1}})$.
   2.3. For $n = 2,\ldots,N_d$:
      2.3.1. Draw a binomial random number, $b_n \sim binomial(P_{n-1,n})$.
      2.3.2. If $b_n = 0$, set $z_{d,n} = z_{d,n-1}$, else draw a new topic $z_{d,n} \sim Mult_K(\theta_d)$.
      2.3.3. Draw a word $w_{d,n} \sim Mult_{W_0}(\phi_{z_{d,n}})$.

As shown in Step 2.3.1. of Algorithm 15, the topic of each word is decided based on two factors: i) the transition matrix and ii) the binomial distribution. The topic of the first word in each document follows the standard LDA topic assignment, i.e., it is generated from a Dirichlet distribution. The model can be regarded as semi-supervised overall since the process is a Markov chain employing prior knowledge.

5.1.3 Experiments

We use the point-wise mutual information (PMI score) to evaluate the proposed model using: 1) a low-dimensional vocabulary of top words from prior topics and 2) the whole vocabulary of the collection. According to [Newman *et al.*2011, Arora *et al.*2012a], the PMI score is a suitable measure for comparing topics' quality and coherence. For ease of reference, hereafter we refer to the SHMTM using the top words of prior topics in the dictionary as SHMTM-top, the SHMTM using all words of the document collection for the dictionary as SHMTM-all, the LDA with top words of prior topics as LDA-top, and the LDA with all words of the collection as LDA-all. All algorithms were implemented in Python 3.5. The PMI score is proportional to the probability of co-occurrence of words normalized by the probabilities of individual words [Newman *et al.*2011], and is expressed as:

$$PMI(K) = \frac{1}{45}\sum_{i<j} PMI(w_i, w_j),\ i, j \in \{1, \ldots, 10\}, \tag{5.2}$$

where

$$PMI(w_i, w_j) = \log \frac{P(w_i, w_j)}{P(w_i)P(w_j)} \tag{5.3}$$

where $P(w_i, w_j)$ is the probability of co-occurring i-th and j-th words in the collection, and 45 is the number of distinct possible word pairs in the top 10 words. The above equation, (5.2), measures the PMI score of the k-th topic, and the PMI score of K topics is simply the average of their PMI score values. Figure 5.2 shows how words are topically related for some words using 200 prior topics on the phonecalls data set. The higher the value, the more topically related. Based on this table, "return to work" (rtw) is highly related to other words including "recovery" and "progress"; hence, their topic assignments will very likely be the same.

Figures 5.3 and 5.4 display the PMI scores for LDA and SHMTM using the phonecalls and filenotes data sets, respectively. The higher the PMI score, the more the agreement with human-judged topic coherence [Newman *et al.*2011]. Based on the PMI scores with the phonecalls data set, SHMTM performs better than LDA, particularly when the number of topics is greater than 20. In addition, the PMI scores with the filenotes data set show improvement in most cases. These figures also illustrate the variation of the PMI score

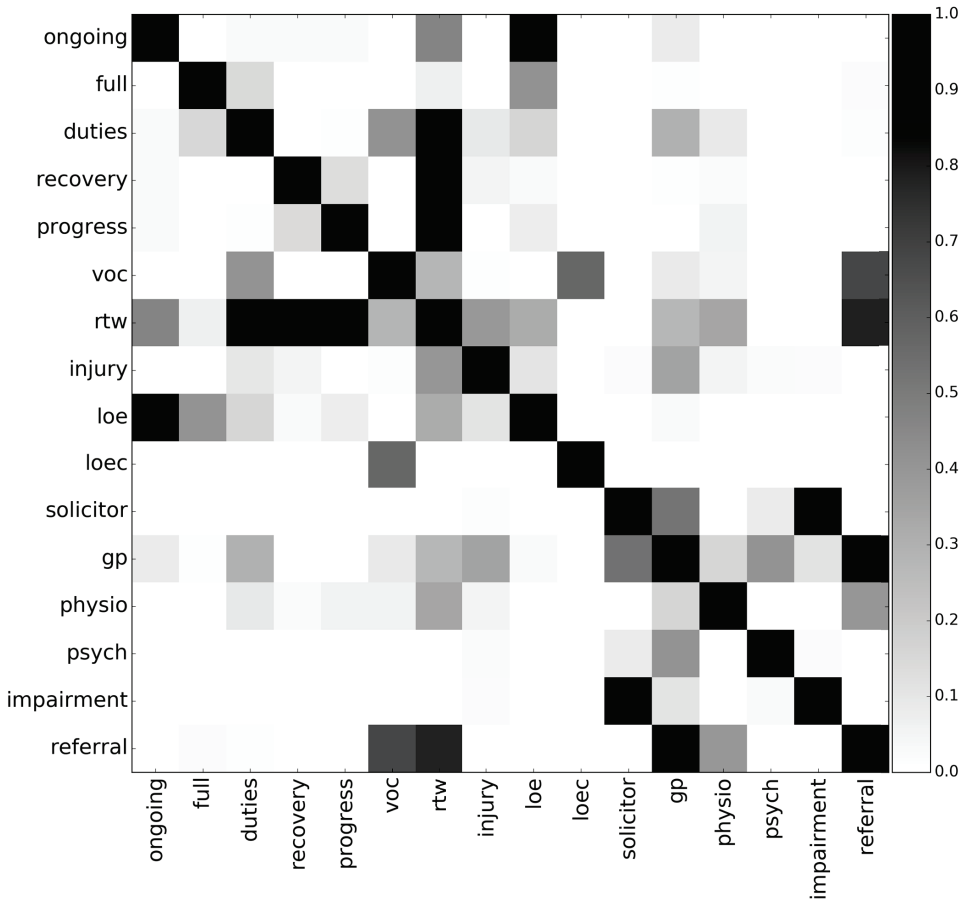


Figure 5.2: A sample set of words and their relations in the transition matrix. The relations have been created using 200 prior topics on the phonecalls data set.

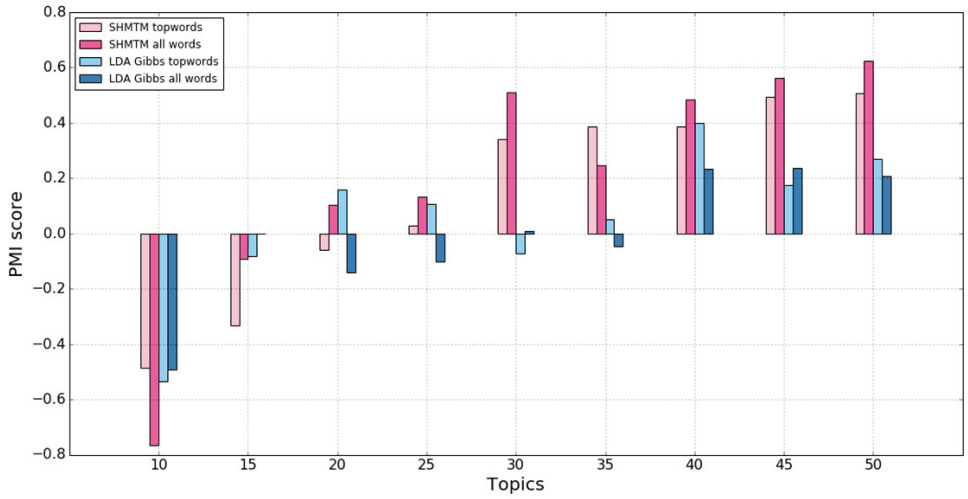


Figure 5.3: PMI score with the phonecalls data set for the SHMTM model using 1) the top words of prior topics (SHMTM topwords) and 2) all the words in the collection (SHMTM all words), and for the LDA model with Gibbs sampling using 3) the top words only (LDA Gibbs topwords) and 4) all the words in the collection (LDA Gibbs all words)

with the number of topics. With the phonecalls data set, the PMI scores get more stable when the number of topics gets to around 30; for the filenotes data set, around 20.

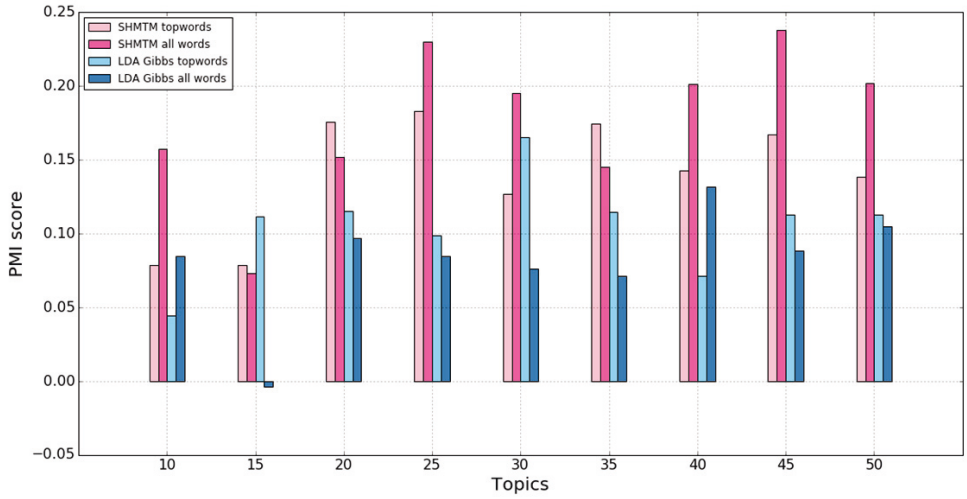


Figure 5.4: PMI score with the filenotes data set for the SHMTM model using 1) the top words of prior topics (SHMTM topwords) and 2) all the words in the collection (SHMTM all words), and for the LDA model with Gibbs sampling using 3) the top words only (LDA Gibbs topwords) and 4) all the words in the collection (LDA Gibbs all words)

Table 5.1: Top words from four topics for SHMTM using only the top words or the whole dictionary with the phonecalls data set

SHMTM-top			
gp	**work**	**voc**	**taxi**
loec	**rtw**	**employment**	**travel**
report	**back**	**job**	**work**
solicitor	pain	appointment	**drive**
psych	**recovery**	**voc_provider**	approval
attend	**treatment**	**work**	appointment
treatment	**physio**	**job_seeking**	**physio**
impairment	gp	pay	**driving**
appointment	weeks	**income**	**transport**
decision	long	attend	**assist**
SHMTM-all			
solicitor	**rtw**	**voc**	**taxi**
loec	**treatment**	**job**	**travel**
impairment	**recovery**	**work**	**work**
report	holistic	**employment**	**ankle**
provider	followup	**voc_provider**	rtw
gp	**back**	**job_seeking**	**drive**
assessment	time	loec	**foot**
decision	**physio**	service	**unable**
psych	gp	**voc_service**	approval
cease	**work**	**provider**	**physio**

Tables 5.1 and 5.2 present the top words for four topics (out of a total of 35) of each model with the phonecalls data set. Although some topics of the different models look similar in terms of top words, words of topics in SHMTM-all and SHMTM-top are more

Table 5.2: Top words from four topics for LDA using only the top words or the whole dictionary with the phonecalls data set

LDA-top			
gp	**rtw**	**voc**	**taxi**
report	**treatment**	loec	**travel**
solicitor	assist	**job**	medcert
impairment	**recovery**	**voc_provider**	**work**
injury	time	**employment**	appointment
back	support	capacity	review
med	left	**work**	**drive**
decision	holistic	**job_seeking**	**hospital**
psych	gp	service	rtw
time	issue	**voc_service**	**transport**
LDA-all			
solicitor	pain	loec	**taxi**
pay	**treatment**	**voc**	**travel**
impairment	**rtw**	**employment**	service
back	psych	income	home
report	**recovery**	**job**	approval
psych	week	**voc_provider**	appointment
cso	weeks	**work**	**assist**
followup	gp	capacity	**drive**
retraining	**back**	**job_seeking**	support
jme	holistic	cease	weeks

topically-related than from the other two models. For example, words "training" and "retraining" appear as top words in the first topic of LDA-all, but the topic also contains unrelated words such as "solicitor", "loec" (loss of earning compensation) and "impairment" as other top words. This is also shown by Fig. 5.3, where the SHMTM-all model proves to be the most accurate among all models (for numbers of topics greater than 20), followed by SHMTM-top, LDA-top and LDA-all.

5.2 Cluster based topic learning

In this section we present a novel topic learning approach that is heavily reliant on clustering. Figure 5.5 shows a synopsis of the proposed approach which consists of the following steps:

1. (Preprocessing) Includes removal of stop-words, sparse terms and common names.
2. (Topic-words matrix) Using a sample set of n-grams from the text corpus, a word embedding and a clustering approach, calculates the topic-word matrix.

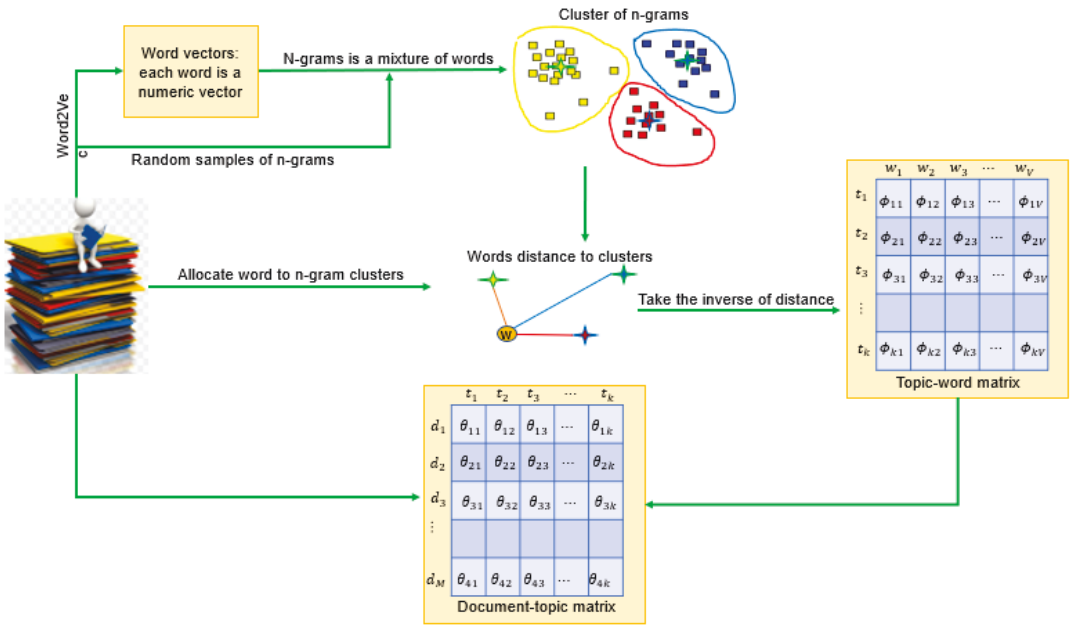


Figure 5.5: Overall process of the cluster-based topic learning.

3. (Document-topic matrix) Determines the document-topic matrix by scanning the text corpus and sampling from the topic-word matrix.

Our assumptions for topic-word and document-topic are the same as the LDA model, i.e. each topic is a mixture of words, and each document is a mixture of topics. However, the topic-word distribution does not need to follow a Dirichlet distribution. The same applies to the document-topic distribution, i.e. it may not necessarily follow the Dirichlet distribution.

5.2.1 Topic-word learning

Hereafter we describe an approach for calculating the topic-word matrix using *n*-grams. In the LDA, the topic-word distribution follows a Dirichlet distribution, and the ensuing problem is that the word semantics is not taken into account. To amend this issue, we use *n*-grams and word embeddings to incorporate word semantics in the model. To do so, word vectors are calculated using a word embedding technique such as Word2Vec on large-scale data (they can be external data). A random sample set of *n*-grams form the text corpus is selected and is partitioned into K clusters using, for example, the k-means algorithm. As each *n*-grams is also a set of words, each *n*-gram can be represented as an average vector with the same dimension as the individual words' vectors. The closeness of the vectors to the clusters' centers is then calculated and is used as cluster-word or

topic-word matrix. The idea behind this is that each n-gram can carry only one topic; therefore each cluster (a collection of n-grams) represents only one topic. The closeness of the vectors to the centers represents the topic-word matrix; therefore each topic is a mixture of words. Below we describe the steps of the algorithm to calculate the topic-word matrix.

Algorithm 16: Topic-word matrix.

Input: External text corpus, text documents X and number of topics K.
Output: Topic-word matrix ϕ.

1. (Word vector representation) Compute word vectors by using a word embedding technique on a large collection of documents.
2. (Build n-grams) Select r random samples of n-grams from the text corpus. Convert the n-grams to vectors of the same dimension as the individual words' vectors using a weighted average of the corresponding word vectors.
3. (Clusters of n-grams) Cluster the n-grams with a clustering algorithm such as k-means.
4. (Topic-word mixture) Given $\phi = [\phi_1, \cdots, \phi_k] = [\phi_{ij}]$ where ϕ_{ij} is the weight of ith word in jth cluster, the lth topic is calculated as:

$$\phi_l = (\phi_{1l}, \phi_{2l}, \cdots, \phi_{Vl})^T, \;\; \phi_{vl} = \|w_v - c^l\|$$

where $\|.\|$ is the Euclidean norm, w_v the v-th word and c^l is the center of the lth cluster calculated in Step 3..

5.2.2 The document-topic matrix

To determine the document-topic matrix or distribution, one can use matrix factorization by considering the fact that $X = \theta\phi$, where X is the document-term matrix (e.g. tf-idf), ϕ the topic-word weights (or topics) described in 5.2.1, and θ the document-topic matrix. In a real situation the equality condition is mitigated to $X \approx \theta\phi$, and it is equivalent to minimizing the following criteria:

$$\min_{\theta} \|X - \theta\phi\| \quad \text{s.t.} \quad \theta \geq 0. \tag{5.4}$$

Adding a regulariser, Eq. (5.4) is equivalent to minimising the following:

$$f(\theta) = \|X - \theta\phi\|_2^2 + \lambda\|\theta\|_2^2. \tag{5.5}$$

where λ is a scalar parameter. Although minimisation of (5.4) is similar to non-negative matrix factorization (NMF), [Arora *et al.*2012b], for topic modelling, our proposal only deals with the document-term matrix, θ; thus, it is different from NMF.

One can solve the optimization problem of (5.5) based on an optimization approach. An alternative to the optimization procedure is to use a heuristic semi-supervised approach, where sampling from the topic-word matrix can be used for determining the document-topic matrix. In both methods, the aim is to find the document-topic matrix, θ. The quality of the solution can be examined using the objective function (5.5). The process can be progressively repeated by iteratively increasing the number of topics and calculating the document-topic matrix each time, until no progress is achieved.

The heuristic approach consists of simply scanning all the documents and assigning topics to words by sampling from the topic-word matrix. This is the approach we have used to date; using an incremental optimization procedure for solving (5.5) is our future work. The main steps of the algorithm for determining the document-topic matrix are as follows:

Algorithm 17: Document-topic matrix.

Input: Text documents D and topic-word matrix ϕ.
Output: Document-topic matrix $\theta = [\theta_j^i],\ i = 1:m,\ j = 1:K$.

For each document d in D:

(a) For each word w in d:
- Sample r random topics given the word weights in vector ϕ^w.
- Select a topic, l, for word w as that with the highest frequency among r topics.
- Set $\theta_l^d = \theta_l^d + 1$.

(b) normalize θ^d to add up to one.

In Algorithm 17, θ^d corresponds to the weight vector of document d, θ_l^d to the weight of topic l in document d, and ϕ^w to the topic vector for word w, i.e. $\phi = [\phi^1, \cdots, \phi^V]^T$.

5.2.3 Evaluation of the proposed method for document classification

The data sets used in our implementation are from the TAC, an accident compensation agency of the Victorian Government in Australia. They consist of phone calls between

TAC clients and staff to address challenges and issues of the clients. Each observation might contain several phone calls related to the same client. The data is annotated with a class label of "yes" or "no" depending on weather it contains a challenge or not. We have used different subsets of the data by random sampling, and trained various classifiers. We have used 10-fold cross validation on each subset. A brief description of the data sets is available in Table 5.3. Data sets D1 and D2 consist of short-length documents, ranging from 20 words to 100 words per document, while data sets D3 and D4 are larger-size documents ranging from 100 to 5,000 words each. M_1 and M_2 are the number of samples from classes "yes" or "no", respectively.

Table 5.3: Data set summary.

Datasets	Dataset name	M_1	M_2
D1	Short PCalls 1	1,789	2,887
D2	Short PCalls 2	3,098	5,000
D3	Long PCalls 1	1,789	2,887
D4	Long PCalls 2	3,098	5,000

The data have required pre-processing including: transforming to lowercase letters, removing stop words and common words, and replacing synonyms. For example, "return to work" and "back to work" are treated as the same words and need to be converted to a single word, say for instance "rtw".

We compare the proposed method with three baselines; 1) the tf-idf features, 2) the weighted averages of word vectors and 3) features obtained from the LDA model. We have used extreme gradient boosting (xgb) as a classifier [Chen and Guestrin2016].

Figures 5.6 and 5.7 show the average accuracy and its standard deviation over the 10-fold cross validation for the short-length documents of data sets D1 and D2. Figures 5.8 and 5.9 show the same quantities for data sets D3 and D4. In these figures, "dtm" is the document-term matrix, i.e. the tf-idf features, "doc2vec" the weighted averages of word vectors, "lda" the features obtained from the LDA model and "tm_cluster" the features obtained by the proposed algorithm.

It can be observed from these figures that the standard deviations of all models are in a similar range, and the differences do not seem consistent. For example, "lda" has a lower standard deviation than the other techniques with data sets D3 and D4, while it

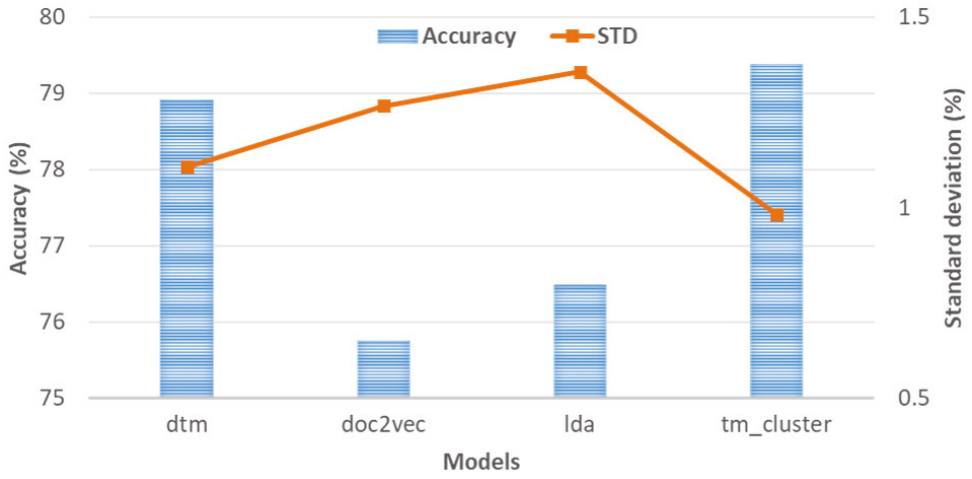


Figure 5.6: Accuracy and standard deviation (%) of models for D1 using xgb classification. The bar is for the accuracy and the line with dots shows the standard deviation.

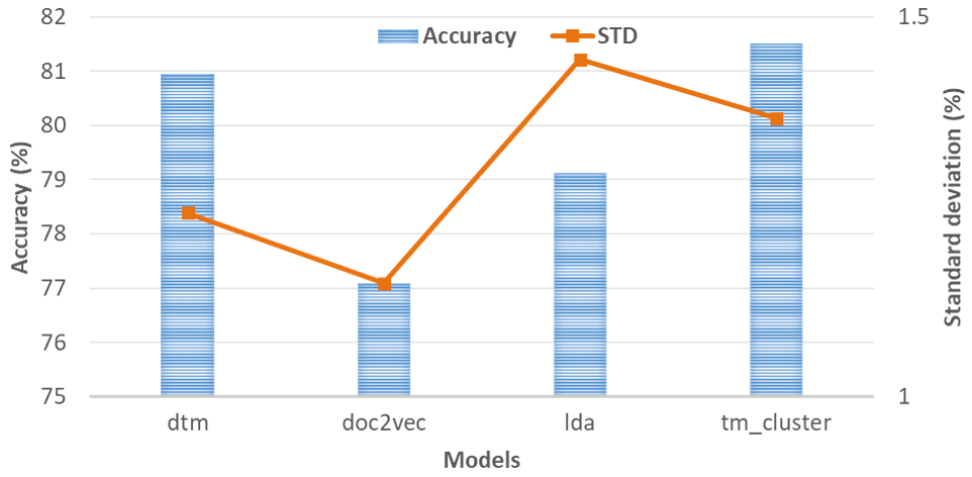


Figure 5.7: Accuracy and standard deviation (%) of models for D2 using xgb classification. The bar is for the accuracy and the line with dots shows the standard deviation.

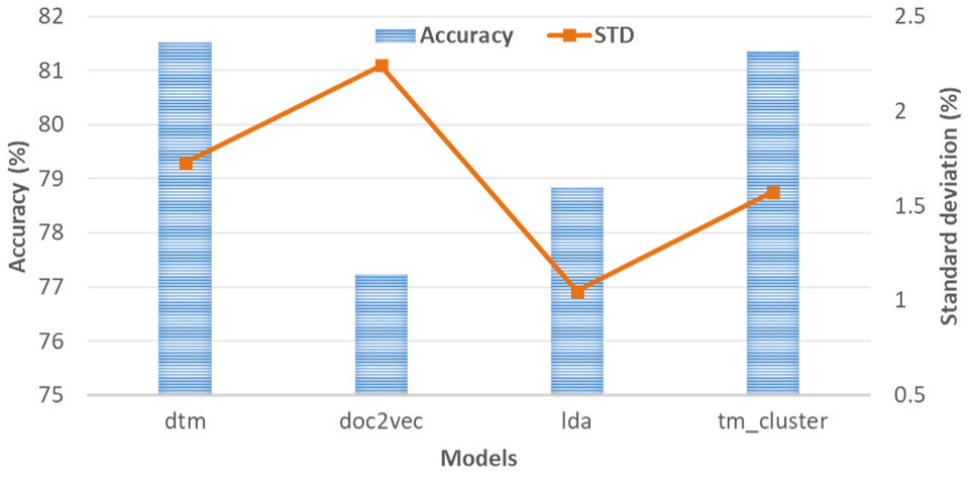


Figure 5.8: Accuracy and standard deviation (%) of models for D3 using xgb classification. The bar is for the accuracy and the line with dots shows the standard deviation.

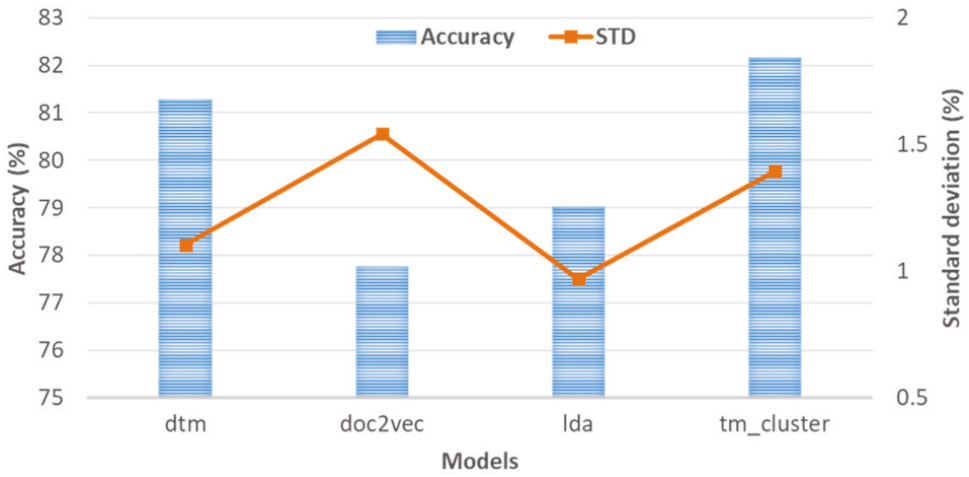


Figure 5.9: Accuracy and standard deviation (%) of models for D4 using xgb classification. The bar is for the accuracy and the line with dots shows the standard deviation.

has a higher standard deviation with data sets D1 and D2. Similar observations can be addressed to the other models. In terms of accuracy, the proposed method performs better than the others overall (particularly for data sets D1, D2 and D4), followed by "dtm" and "lda", respectively. The "dtm" model produces very close results to the proposed model. We also note that the accuracy results for large-sized documents are, in general, slightly better than those for short-length documents. However, this might also depend on the number of instances (the more the samples, the better the accuracy).

5.3 Conclusion

Two algorithms have been developed in this Chapter. The first is a semi-supervised hidden Markov topic model using prior knowledge. The model consists of two stages where the prior knowledge is accounted for in the first stage, and coherent topic models are produced by the final model. In the model's assumptions, the topic of a word not only follows a Dirichlet distribution, but also depends on the topic of the previous word. The topic dependency is decided using a binomial distribution and a transition probability matrix obtained by an initial modelling of prior topics. Experiments on real text documents (clients phone calls and file notes) from the TAC have given evidence to improvements of the PMI score measure and topic coherence.

The second method uses n-grams for topic generation. A random sample of n-grams are created from large corpus and are partitioned into clusters using k-means. Each cluster is a collection of n-grams, so it is assumed as one topic. The distances of the words to

the clusters' centers are considered as the topic-word matrix. The document-topic matrix is, then, computed by sampling from the topic-word matrix. The proposed features (the document-topic matrix) have been evaluated via classification on four datasets from the TAC, giving evidence to the effectiveness of the proposed document-topic matrix over BoW, Doc2Vec (Word2Vec) and document-topic features from LDA.

Chapter 6

Conclusion

This thesis has presented word embedding-based techniques for text clustering and topic modelling as two of the most important tools in text analytics and natural language processing. One of the main focuses of this research has been incorporating word semantics into account as well as addressing the curse of dimensionality.

Text clustering and topic modelling are very close subjects, since the aim of both is to assign documents to clusters. Topic modelling can be considered as soft clustering, where each document is a mixture of topics. I have tried to find relationships between them and introduce algorithms that use the benefits of both. All methodologies developed here have been examined on real case scenarios of an industry partner in the healthcare domain. For clustering, I have developed three methodologies: 1) a maximum-margin clustering approach for large-scale data, 2) an incremental document clustering technique and 3) taxonomy-augmented based techniques. In the first methodology, a classic clustering framework under the Euclidean norm has been approached by a novel algorithm that leverages a) simulated annealing to escape local minima and b) a combination of $k-$means++ and SVM to provide effective clustering. By exploiting a two-stage organization, the proposed algorithm has demonstrated to be able to mollify the computational complexity of maximum-margin clustering and be suitable for large-scale data. In the experiments, we have tested it on real-world datasets with number of points ranging from thousands to millions. Therefore, we can argue that it suits application to text document collections. The experimental results clearly demonstrate that the proposed method produces significantly improved results in comparison with de-facto standard clustering algo-

rithms such as k-means++, mini-batch $k-$means++, a Dirichlet process Gaussian mixture model and fuzzy c-means. In the second methodology, we have proposed an algorithm – a modified global k-means – to improve the quality of the local solution for clustering document collections in a reasonable amount of time without having to resort to parallel processing. Thus, our main emphasis has been on a trade-off between high accuracy and efficiency, with modest main memory consumption. The preliminary results are promising and further investigation is under evaluation. The last algorithm has been evaluated for classification as well as semantic analysis for the industry partner. In the semantic analysis, the industry partner can monitor clients' behaviour with respect to their health conditions over time. For topic modelling, I have developed two techniques: 1) a semi-supervised hidden Markov topic model which takes word relations into account and 2) a cluster-based topic modelling, where topics are constructed using n-grams. The semi-supervised hidden Markov topic model has been designed to use prior knowledge. The model consists of two stages, where prior knowledge is accounted for in the first stage and used by the final model to produce human-judged coherent topics. The topic of a word not only follows a Dirichlet distribution assignment, but it also depends on the topic of the previous word. The topic dependency is decided using a binomial distribution and a transition probability matrix obtained from the prior topics. Experiments on real text documents (clients phone call notes) from the Transport Accident Commission have demonstrated an improvement of the PMI-Score measure and topic coherence. The second model uses a clustering technique to partition n-grams, and then, find the topic-word distribution.

Overall, we have ground to believe that this thesis has provided a tangible contribution to the areas of text document clustering and topic modelling, especially with emphasis on the targeted application. We look forward to expanding the presented work in future contributions.

Bibliography

[Aljalbout *et al.*, 2018] Elie Aljalbout, Vladimir Golkov, Yawar Siddiqui, and Daniel Cremers. Clustering with deep learning: Taxonomy and new methods. *CoRR*, abs/1801.07648, 2018.

[Alshari *et al.*, 2017] Eissa M. Alshari, Azreen Azman, Shyamala Doraisamy, Norwati Mustapha, and Mustafa Alkeshr. Improvement of sentiment analysis based on clustering of Word2Vec features. In *Proceedings - International Workshop on Database and Expert Systems Applications, DEXA*, 2017.

[Arora *et al.*, 2012a] Sanjeev Arora, Rong Ge, Yoni Halpern, David M. Mimno, Ankur Moitra, David Sontag, Yichen Wu, and Michael Zhu. A practical algorithm for topic modeling with provable guarantees. *CoRR*, abs/1212.4777, 2012.

[Arora *et al.*, 2012b] Sanjeev Arora, Rong Ge, and Ankur Moitra. Learning topic models – going beyond svd. In *Proceedings of the 2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, FOCS '12, pages 1–10, Washington, DC, USA, 2012. IEEE Computer Society.

[Arthur and Vassilvitskii, 2007] D. Arthur and S. Vassilvitskii. K-means++: the advantages of careful seeding. In *In H. Gabow (Ed.) Proc. of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA07)*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[Bagirov and Yearwood, 2006] A.M. Bagirov and J. Yearwood. A new nonsmooth optimization algorithm for minimum sum-of-squares clustering problems. *European Journal of Operational Research*, 170(2):578–596, 2006.

[Bagirov *et al.*, 2011] A.M. Bagirov, J. Ugon, and D. Webb. Fast modified global k-means algorithm for incremental cluster construction. *Pattern Recognition*, 44(4):866–876, April 2011.

[Bagirov *et al.*, 2018] A. Bagirov, S. Seifollahi, M. Piccardi, E. Zare, and B. Kruger. SMGKM: An efficient incremental algorithm for clustering document collections. *CICLing 2018*, 2018.

[Bagirov, 2008] A.M. Bagirov. Modified global k-means algorithm for minimum sum-of-squares clustering problems. *Pattern Recognition*, 41(10):3192–3199, 2008.

[Bai *et al.*, 2013] L. Bai, J. Liang, C. Sui, and Ch. Dang. Fast global k-means clustering based on local geometrical information. *Information Sciences*, 245:168 – 180, 2013.

[Banerjee *et al.*, 2005] A. Banerjee, I.S. Dhillon, Ghosh J., and S. Sra. Clustering on the unit hypersphere using von mises-fisher distributions. *Journal of Machine Learning Research*, 6:1345–1382, 2005.

[Beel *et al.*, 2016] Joeran Beel, Bela Gipp, Stefan Langer, and Corinna Breitinger. Research-paper recommender systems: A literature survey. *Int. J. Digit. Libr.*, 17(4):305–338, November 2016.

[Bellegarda, 2005] J. R. Bellegarda. Latent semantic mapping [information retrieval]. *IEEE Signal Processing Magazine*, 22:70–80, 2005.

[Blei and Lafferty, 2006] D.M. Blei and J.D. Lafferty. Dynamic topic models. In *International Conference on Machine Learning (ICML)*, 2006.

[Blei and Lafferty, 2007] D.M. Blei and J.D. Lafferty. A correlated topic model of science. *Statistics*, 1(1):17–35, 2007.

[Blei and McAuliffe, 2008] D.M. Blei and J. McAuliffe. Supervised topic models. In *Advances in Neural Informormation Processing Systems*, volume 20, pages 121–128. MIT Press, Cambridge, MA, 2008.

[Blei *et al.*, 2003] D.M. Blei, A.Y. Ng, and Jordan MI. Latent dirichlet allocation. *Machine Learning Research*, 3:1107–1135, 2003.

[Blei *et al.*, 2004] D. Blei, T. Griffiths, M. Jordan, and J. Tenenbaum. Hierarchical topic models and the nested chinese restaurant process. In *Advances in neural information processing systems*, volume 16, 2004.

[Brock *et al.*, 2008] G. Brock, V. Pihur, S. Datta, and S. Datta. clValid: An R package for cluster validation. *Journal of Statistical Software*, 25:1–22, 2008.

[Buckley and Lewit, 1985] Chris Buckley and Alan F. Lewit. Optimizations of inverted vector searches. *SIGIR '85*, pages 97–110, 1985.

[Chen and Guestrin, 2016] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM.

[Chen and Liu, 2014] Z. Chen and B. Liu. Topic modeling using topics from many domains, lifelong learning and big data. In *Proc. of the 31st International Conference on Machine Learning (ICML)*, pages 703–711, 2014.

[Cheng, 2008] Y. Cheng. Ontology-based fuzzy semantic clustering. In *Proceedings - 3rd International Conference on Convergence and Hybrid Information Technology, ICCIT 2008*, volume 2, pages 128–133, 2008.

[Cutting *et al.*, 1992] Douglass R. Cutting, David R. Karger, Jan O. Pedersen, and John W. Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '92, pages 318–329, New York, NY, USA, 1992. ACM.

[Das *et al.*, 2015] R. Das, M. Zaheer, and C. Dyer. Gaussian LDA for topic models with word embeddings. In *Proceedings ACL 2015*, pages 795–804, 2015.

[Davies and Bouldin, 1979] David L. Davies and Donald W. Bouldin. A cluster separation measure. *IEEE Trans. Pattern Anal. Mach. Intell.*, 1(2):224–227, February 1979.

[Dean and Ghemawat, 2008] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, January 2008.

[Deerwester *et al.*, 1990] S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *J Am Soc Inf Sci*, 41(6):391, 1990.

[Devlin *et al.*, 2018] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[Dhillon and Sra, 2005] Inderjit S. Dhillon and Suvrit Sra. Generalized nonnegative matrix approximations with bregman divergences. In *Proceedings of the 18th International Conference on Neural Information Processing Systems*, NIPS'05, pages 283–290, Cambridge, MA, USA, 2005. MIT Press.

[Dhillon *et al.*, 2001] S. Dhillon, J. Fan, and Y. Guan. Efficient clustering of very large document collections. In C. Kamath V. Kumar R. Grossman and R. Namburu, editors, *Data Mining for Scientific and Engineering Applications*. Kluwer Academic Publishers, Oxford, 2001.

[Duchi, 2007] J. Duchi. Derivatieves for linear algebra and optimisation. Available in web page <URL: `http://www.cs.berkeley.edu`>, 2007.

[Elsayed *et al.*, 2015] A. Elsayed, H.M.O. Mokhtar, and O. Ismail. Ontology based document clustering using Mapreduce. *International Journal of Database Management Systems*, 7(2):1–12, 2015.

[Erra *et al.*, 2015] U. Erra, S. Senatore, F. Minnella, and G. Caggianese. Approximate tf-idf based on topic extraction from massive message stream using the gpu. *Information Sciences*, 292:143–161, 2015.

[Fodeh *et al.*, 2011] S. Fodeh, B. Punch, and P.-N. Tan. On ontology-driven document clustering using core semantic features. *Knowledge and Information Systems*, 28(2):395–421, 2011.

[Friedman, 1997] J.H. Friedman. On bias, variance, 0/1-loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery*, 1(1):55–77, 1997.

[Gieseke *et al.*, 2009] F. Gieseke, T. Pahikkala, and O. Kramer. Fast evolutionary maximum margin clustering. In *Proc. of International Conference on Machine Learning (ICML)*, pages 361–368, 2009.

[Gong *et al.*, 2017] Hongyu Gong, Jiaqi Mu, Suma Bhat, and Pramod Viswanath. Prepositions in context. *CoRR*, abs/1702.01466, 2017.

[Gong *et al.*, 2018] Hongyu Gong, Suma Bhat, and Pramod Viswanath. Embedding syntax and semantics of prepositions via tensor decomposition. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 896–906, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[Gopalan and Sankaranarayanan, 2011] R. Gopalan and J. Sankaranarayanan. Max-margin clustering: Detecting margins from projections of points on lines. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[Görür and Rasmussen, 2010] D. Görür and C.E. Rasmussen. Dirichlet process gaussian mixture models: Choice of the base distribution. *Journal of Computer Science and Technology*, 25:653 – 664, 2010.

[Griffiths and Steyvers, 2004] T.L. Griffiths and M. Steyvers. Finding scientific topics. In *Proc. Natl. Acad. Sci.*, volume 101, pages 5228–5235, 2004.

[Gruber *et al.*, 2007] A. Gruber, Y. Weiss, and M. Rosen-zvi. Hidden topic markov models. In *Proc. of the 11th International Conference on Artificial Intelligence and Statistics (AISTATS-07)*, pages 163–170, 2007.

[Gui *et al.*, 2014] Jie Gui, Zhenan Sun, Jun Cheng, Shuiwang Ji, and Xindong Wu. How to estimate the regularization parameter for spectral regression discriminant analysis and its kernel version? *IEEE Trans. Cir. and Sys. for Video Technol.*, 24(2):211–223, February 2014.

[Gui *et al.*, 2016] Jie Gui, Tongliang Liu, Dacheng Tao, Zhenan Sun, and Tieniu Tan. Representative vector machines: A unified framework for classical classifiers. *IEEE Transactions on Cybernetics*, 46:1877–1888, 2016.

[Harman, 1992] D. Harman. Overview of the first text retrieval conference (trec-1). In *Proc. of the First Text Retrieval Conference (TREC-1)*, pages 1–20. DIANE Publishing, 1992.

[Harris, 1954] Z.S. Harris. Distributional Structure. *Word*, 10 (2-3):146 – 162, 1954.

[Hofmann, 1999] T. Hofmann. Probabilistic latent semantic indexing. In *Proc. of the 22nd Annual International SIGIR Conference on Research and Development in Information Retrieval*, 1999.

[Hotho *et al.*, 2003] A. Hotho, S. Staab, and G. Stumme. Ontologies improve text document clustering. In *Third IEEE International Conference on Data Mining*, pages 541–544, 2003.

[Hu *et al.*, 2008] Y. Hu, J. Wang, N. Yu, and X. S. Hua. Maximum margin clustering with pairwise constraints. In *Proc. of the 8th IEEE international Conference on Data Mining (ICDM)*, pages 253–262, 2008.

[Jain *et al.*, 1999] A.K. Jain, M.N. Murty, and P.J. Flynn. Data clustering: A review. *ACM Comput. Surv.*, 31(3):1248–1255, 1999.

[Karnin *et al.*, 2012] Z.S. Karnin, E. Liberty, S. Lovett, R. Schwartz, and O. Weinstein. Unsupervised svms: On the complexity of the furthest hyperplane problem. In *COLT 2012 - The 25th Annual Conference on Learning Theory*, pages 2.1–2.17, 2012.

[Kim *et al.*, 2015] J. Kim, F. Rousseau, and M. Vazirgiannis. Convolutional sentence kernel from word embeddings for short text categorization. In *Proceedings EMNLP 2015*, number September, pages 775–780, 2015.

[Kim *et al.*, 2017] Han Kyul Kim, Hyunjoong Kim, and Sungzoon Cho. Bag-of-concepts. *Neurocomput.*, 266(C):336–352, November 2017.

[Kingma and Welling, 2014] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[Kirkpatrick *et al.*, 1983] S. Kirkpatrick, C.D. Gelatt-J, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[Kogan, 2007] J. Kogan. *Introduction to Clustering Large and High-dimensional Data*. Cambridge University Press, 2007.

[Koller and Sahami, 1997] Daphne Koller and Mehran Sahami. Hierarchically classifying documents using very few words. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, pages 170–178, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[Kowalski, 1997] G. Kowalski. *Information Retrieval Systems Theory and Implementation*. Kluwer Academic Publishers, 1997.

[Kusner *et al.*, 2015] M.J. Kusner, Y. Sun, N.I. Kolkin, and K.Q. Weinberger. From word embeddings to document distances. In *Proceedings - ICML*, volume 37, pages 957–966, 2015.

[Lacoste-Julien *et al.*, 2009] S. Lacoste-Julien, F. Sha, and M. Jordan. Disclda: Discriminative learning for dimensionality reduction and classification. In *Advances in Neural Information Processing Systems*, volume 21, pages 897–904, 2009.

[Le and Mikolov, 2014] Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32, pages 1188–1196, 2014.

[Lenc and Král, 2017] L. Lenc and P. Král. Word embeddings for multi-label document classification. In *Proceedings of Recent Advances in Natural Language Processing*, pages 431–437, 2017.

[Lewis, 1997] D.D. Lewis. Reuters-21578 text categorization collection distribution 1.0, `http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html`. 1997.

[Li *et al.*, 2009] Y.-F. Li, I.W. Tsang, J.T.-Y. Kwok, and Z.-H. Zhou. Tighter and convex maximum margin clustering. In *Proc. of the 12th international conference on artificial intelligence and statistics*, pages 344–351, 2009.

[Li *et al.*, 2017] Lian-sheng Li, Sheng-jiang Gan, and Xiang-dong Yin. Feedback recurrent neural network-based embedded vector and its application in topic model. *EURASIP Journal on Embedded Systems*, 2017(1):1–6, 2017.

[Lichman, 2001] M. Lichman. UCI machine learning repository. Available in web page <URL: `http://archive.ics.uci.edu/ml`>, University of California, Irvine, School of Information and Computer Sciences, 2001. (April 8th, 2016).

[Likas *et al.*, 2003] A. Likas, N. Vlassis, and J. Verbeek. The global k-means clustering algorithm. *Pattern Recognition*, 36(2):451 – 461, 2003.

[Lilleberg *et al.*, 2015] J. Lilleberg, Y. Zhu, and Y. Zhang. Support vector machines and word2vec for text classification with semantic features. *Proceedings of IEEE 14th International Conference on Cognitive Informatics & Cognitive Computing (ICCI*CC)*, pages 136–140, 2015.

[Locatelli, 2000] M. Locatelli. Simulated annealing algorithms for continuous global optimization: convergence conditions. *Journal of Optimization Theory and Applications*, 104:121 – 133, 2000.

[McCallum *et al.*, 2000] A. McCallum, K. Nigam, J. Rennie, and K. Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval Journal*, 3:127–163, 2000.

[Mikolov *et al.*, 2013a] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *Arxiv*, pages 1–12, 2013.

[Mikolov *et al.*, 2013b] Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. Exploiting similarities among languages for machine translation. *CoRR*, abs/1309.4168, 2013.

[Moseley and Wang, 2017] B. Moseley and J.R. Wang. Approximation bounds for hierarchical clustering: Average linkage, bisecting K-means, and local search. Number Nips, pages 3097–3106, 2017.

[Newman *et al.*, 2011] D. Newman, E. V. Bonilla, and W. Buntine. Improving topic coherence with regularized topic models. In *Proc. of the 24th International Conference on Neural Inform. Processing Syst.*, pages 496–504, 2011.

[Ordin and Bagirov, 2015] B. Ordin and A.M. Bagirov. A heuristic algorithm for solving the minimum sum-of-squares clustering problems. *Journal of Global Optimization*, 61:341–361, 2015.

[Papadimitriou *et al.*, 2000] C.H. Papadimitriou, P. Raghavan, H. Tamaki, and S. Vempala. Latent semantic indexing: A probabilistic analysis. In *Journal of Computer and System Sciences*, volume 61, pages 217–235, 2000.

[Peng *et al.*, 2015] Xi Peng, Shaoting Zhang, Yu Yang, and Dimitris N. Metaxas. Piefa: Personalized incremental and ensemble face alignment. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, pages 3880–3888, Washington, DC, USA, 2015. IEEE Computer Society.

[Peng *et al.*, 2018a] Hao Peng, Jianxin Li, Yu He, Yaopeng Liu, Mengjiao Bao, Lihong Wang, Yangqiu Song, and Qiang Yang. Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, pages 1063–1072, Republic and Canton of Geneva, Switzerland, 2018. International World Wide Web Conferences Steering Committee.

[Peng *et al.*, 2018b] Xi Peng, Rogerio S. Feris, Xiaoyu Wang, and Dimitris N. Metaxas. Red-net: A recurrent encoder—decoder network for video-based face alignment. *Int. J. Comput. Vision*, 126(10):1103–1119, October 2018.

[Pennington *et al.*, 2014] J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings EMNLP 2014*, pages 1532–1543, 2014.

[Peters *et al.*, 2018] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[Pham *et al.*, 2015] Hieu Pham, Thang Luong, and Christopher Manning. Learning distributed representations for multilingual text sequences. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 88–94, Denver, Colorado, June 2015. Association for Computational Linguistics.

[Qimin *et al.*, 2015] C. Qimin, G. Qiao, W. Yongliang, and W. Xianghua. Text clustering using vsm with feature clusters. *Neural Computing and Applications*, 26(4):995–1003, 2015.

[Recupero, 2007] D.R. Recupero. A new unsupervised method for document clustering by using WordNet lexical and conceptual relations. *Information Retrieval*, 10(6):563–579, 2007.

[Rennie, 2008] J. Rennie. The 20 newsgroups data set. Available in web page <URL: `http://qwone.com/ jason/20Newsgroups`>, 2008.

[Robertson and Walker, 1994] S. E. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '94, pages 232–241, New York, NY, USA, 1994. Springer-Verlag New York, Inc.

[Salton and Buckley, 1988] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manage.*, 24(5):513–523, August 1988.

[Salton and McGill, 1983] G. Salton and M.J. McGill. *Introduction to Modern Retrieval*. McGraw-Hill Book Company, New York, 1983.

[Sculley, 2010] D. Sculley. Web-scale k-means clustering. In *Proc. of the 19th international conference on World wide web*, pages 1177–1178, 2010.

[Seifollahi *et al.*, 2014] S. Seifollahi, P.A. Dowd, and C. Xu. An enhanced stochastic optimization in fracture network modelling conditional on seismic events. *Computers and Geotechnics*, 16:85–95, 2014.

[Seifollahi *et al.*, 2017a] S. Seifollahi, A. Bagirov, R. Layton, and I. Gondal. Optimization based clustering algorithms for authorship analysis of phishing emails. *Neural Processing Letters*, 46(2), 2017.

[Seifollahi *et al.*, 2017b] Sattar Seifollahi, Massimo Piccardi, and Ehsan Zare Borzeshi. A semi-supervised hidden markov topic model based on prior knowledge. In *Data Mining - 15th Australasian Conference, AusDM 2017, Melbourne, VIC, Australia, August 19-20, 2017, Revised Selected Papers*, pages 265–276, 2017.

[Seifollahi *et al.*, 2018] Sattar Seifollahi, Massimo Piccardi, Ehsan Zare Borzeshi, and Bernie Kruger. Taxonomy-augmented features for document clustering. In *Data Mining - 16th Australasian Conference, AusDM 2018, Bahrurst, NSW, Australia, November 28-30, 2018, Revised Selected Papers*, pages 241–252, 2018.

[Seifzadeh *et al.*, 2015] S. Seifzadeh, A.K. Farahat, M.S. Kamel, and F. Karray. Short-text clustering using statistical semantics. *Proceedings WWW 2015*, pages 805–810, 2015.

[Singh and Mukerjee, 2015] Pranjal Singh and Amitabha Mukerjee. Words are not equal: Graded weighting model for building composite document vectors. *CoRR*, abs/1512.03549, 2015.

[Srivastava and Sutton, 2017] Akash Srivastava and Charles A. Sutton. Autoencoding variational inference for topic models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[Steinbach *et al.*, 2000] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *KDD workshop on text mining*, volume 400, pages 1–2, 2000.

[Tang *et al.*, 2014] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings ACL*, pages 1555–1565, 2014.

[Teh *et al.*, 2006] Y.W. Teh, D. Newman, and M. Welling. A collapsed variational bayesian inference algorithm for latent dirichlet allocation. In *Advances in Neural Inform. Processing Syst.*, pages 1353–1360, 2006.

[Valizadegan and Jin, 2006] H. Valizadegan and R. Jin. Generalized maximum margin clustering and unsupervised kernel learning. In *B. Schölkopf, J. Platt and T. Hoffman (Eds.), Advances in Neural Information Processing Systems 19, (2007)*, pages 1417–1424. MIT Press, Cambridge, 2006.

[Van-Rijsbergen, 1989] C.J. Van-Rijsbergen. *Information Retrieval*. Buttersworth, London, second edition, 1989.

[Wang and Manning, 2012] Sida Wang and Christopher Manning. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 90–94, Jeju Island, Korea, July 2012. Association for Computational Linguistics.

[Wang *et al.*, 2008] C. Wang, D.M. Blei, and D. Heckerman. Continuous time dynamic topic models. In *Proc. of the 24th Conference on Uncertainty in Artificial Intelligence*, pages 579–586, 2008.

[Wang *et al.*, 2009] F. Wang, X. Wang, and T. Li. Maximum margin clustering on data manifolds. In *International Conference on Data Mining*, 2009.

[Wang *et al.*, 2016] P. Wang, B. Xu, J. Xu, G. Tian, C.L. Liu, and H. Hao. Semantic expansion using word embedding clustering and convolutional neural network for improving short text classification. *Neurocomputing*, 174:806–814, 2016.

[WebKB,] WebKB. Available electronically at `http://www.cs.cmu.edu/~WebKB`.

[Wei *et al.*, 2007] L. Wei, D. Blei, and A. McCallum. Nonparametric bayes pachinko allocation. In *Proc. of the 23rd Conference on Uncertainty in Artificial Intelligence*, pages 243–250, Vancouver, BC, Canada, 2007.

[Winkler *et al.*, 2011] R. Winkler, F. Klawonn, and R. Kruse. Fuzzy c-means in high dimensional spaces. *International Journal of Fuzzy System Applications*, 1(1):1–16, 2011.

[Wu and Stathopoulos, 2014] Lingfei Wu and Andreas Stathopoulos. Primme_svds: A preconditioned SVD solver for computing accurately singular triplets of large matrices based on the PRIMME eigensolver. *CoRR*, abs/1408.5535, 2014.

[Xu and Schuurmans, 2005] L. Xu and D. Schuurmans. Unsupervised and semi-supervised multi-class support vector machines. In *Proc. of the 20th National Conference on Artificial Intelligence*, pages 192–199, Pittsburgh, PA, 2005.

[Xu *et al.*, 2005] L. Xu, J. Neufeld, B. Larson, and D. Schuurmans. Maximum margin clustering. In *In L. K. Saul, Y. Weiss and L. Bottou (Eds.), Advances in neural information processing systems*, number 17, pages 1537 – 1544, 2005.

[Xuerui and McCallum, 2006] X. Xuerui and A. McCallum. Topics over time: a non-markov continuous-time model of topical trends. In *Proc. of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 424–433, 2006.

[Xun *et al.*, 2017] G. Xun, V. Gopalakrishnan, F.M.Y. Li, J. Gao, and A. Zhang. Topic discovery for short texts using word embeddings. *Proceedings - IEEE International Conference on Data Mining, ICDM*, pages 1299–1304, 2017.

[Yang *et al.*, 2016] Jianwei Yang, Devi Parikh, and Dhruv Batra. Joint unsupervised learning of deep representations and image clusters. *CoRR*, abs/1604.03628, 2016.

[Zamir *et al.*, 1997] Oren Zamir, Oren Etzioni, Omid Madani, and Richard M. Karp. Fast and intuitive clustering of web documents. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, KDD'97, pages 287–290. AAAI Press, 1997.

[Zhang *et al.*, 2007] K. Zhang, I.W. Tsang, and J.T. Kwok. Maximum margin clustering made practical. In *Proc. of the 24th International Conference on Machine Learning (ICML)*, pages 1119–1126, Corvallis, OR, 2007.

[Zhang *et al.*, 2015] D. Zhang, H. Xu, Z. Su, and Y. Xu. Chinese comments sentiment classification based on word2vec and SVMperf. *Expert Systems with Applications*, 42(4):1857–1863, 2015.

[Zhang *et al.*, 2018] Yue Zhang, Qi Liu, and Linfeng Song. Sentence-state LSTM for text representation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 317–327, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[Zhao *et al.*, 2008] B. Zhao, F. Wang, and C. Zhang. Efficient multiclass maximum margin clustering. In *Proc. of the 25th international conference on Machine Learning (ICML)*, pages 1248–1255, 2008.

[Zhao *et al.*, 2009] B. Zhao, J. Kwok, F. Wang, and C. Zhang. Unsupervised maximum margin feature selection with manifold regularization. In *Computer Vision and Pattern Recognition*, 2009.