Université de Tunis
Institut Supérieur de Gestion

MÉMOIRE DE MASTER RECHERCHE

Spécialité
Sciences et Techniques de l'Informatique Décisionnelle (STID)

Option
Informatique et Gestion de la Connaissance (IGC)

A word2vec based method for Topic Detection from Twitter

Maha Fraj

Soutenu le 04 décembre 2017, devant le jury composé de:

Rim Faiz	Professeur, Institut Supérieur de gestion de Tunis	Président
Lilia Rejeb	Maitre assistant, Institut Supérieur de gestion de Tunis	Rapporteur
Nadia Essoussi	Professeur, Institut Supérieur de gestion de Tunis	Directeur du mémoire

---

LAboratoire de Recherche Opérationnelle de DÉcision et de Contrôle de Processus (LARODEC)

Résumé:

Twitter constitue l'une des plateformes de microblogging et des services de réseaux sociaux les plus populaires. En effet, l'énorme volume de données générées par les utilisateurs suscite l'intérêt de plusieurs études de recherche. Dans ce mastère, nous sommes intéressés par le domaine de détection de topic qui consiste à identifier et regrouper les tweets liés aux mêmes topics. À cette fin, le clustering est une technique efficace pour grouper les tweets compte tenu de leurs similarités. Cependant, les approches existantes pour la détection de topic ne prennent pas en considération l'aspect sémantique des données. Par conséquent, nous proposons une nouvelle approche qui vise à regrouper les données Twitter en topics tout en préservant les liens sémantiques entre les mots.

Mots-clés: Twitter, word2vec, Clustering du texte, Détection des topics

Abstract:

Twitter is one of the most popular microblogging platforms and social networking services. In fact, the enormous amount of user-generated data rises the interest of multiple research studies. In this Master Thesis, we are interested in the Topic Detection field which consists in identifying and grouping together tweets related to the same topics. To this end, clustering represents an effective technique to sort tweets given their topical similarities. However, the existing approaches for Topic Detection do not consider the semantic aspect of the data. Therefore, we propose a new approach that aims to cluster Twitter data into topics while preserving the semantic links between words.

Keywords: Twitter, word2vec, Text Clustering, Topic Detection

Acknowledgments

I would like to express my sincere gratitude to my advisor Pr. Nadia Essoussi for her support, guidance and valuables advice.

I also would like to express my gratitude towards my co-advisor Mr. Mohamed Aymen Ben HajKacem for his help, motivation and dedication.

I am grateful to all my professors of Institut Superieur de Gestion Tunis and all members of my laboratory LARODEC.

I am also thankful for the continuous support, heartwarming love and care of my beloved family.

Contents

Introduction **1**

Part I : Theoretical Aspects **4**

1 Basic Concepts of Clustering and word2vec **5**

1.1 Introduction 5

1.2 Clustering 5

1.2.1 Fundamental concept of clustering 6

1.2.2 Proximity measures 6

1.2.3 Clustering approaches 8

1.2.4 Text clustering 10

1.3 WORD2VEC model 14

1.3.1 Continuous Bag-Of-Words Model 15

1.3.2 Skip-gram Model 16

1.3.3 word2vec strengths 16

1.4 Conclusion 17

2 Tweets Analysis for Topic Detection **18**

2.1 Introduction 18
2.2 Twitter 19
2.2.1 Definition of a tweet 19
2.2.2 Application domains 20
2.3 Topic detection from Twitter 22
2.3.1 VSM based approaches 22
2.3.2 Latent Semantic Analysis based approaches 23
2.3.3 Latent Dirichlet Allocation based approaches 26
2.3.4 Ontology based approaches 29
2.4 Conclusion 29

Part II : Contributions **31**

3 A word2vec based method for Topic Detection from Twitter **32**
3.1 Introduction 32
3.2 Proposed Approach: A word2vec based method for Topic Detection from Twitter 32
3.2.1 Step 1: Tweets Pre-processing 34
3.2.2 Step 2: Train Skip-gram model over tweets 35
3.2.3 Step 3: Words clustering using K-means 39
3.2.4 Step 4: Tweets representation using word clusters 40
3.2.5 Step 5: Tweets clustering using K-means for topic detection 42
3.3 Conclusion 44

4 Experimental study **45**
4.1 Introduction 45
4.2 Evaluation methodology 45

4.3 Data set description 46

4.4 Experimental environment 46

4.5 Evaluation measures 47

4.6 Experimental results and discussion 48

4.6.1 Comparison of the performance against existing methods . 48

4.6.2 Evaluation against the number of word clusters K 50

4.6.3 Evaluation against the dimension N of Skip-gram vector representations 51

4.6.4 Evaluation against the context window C 52

4.7 Conclusion 53

Conclusion **54**

Appendix **56**

A The learning process of word2vec model **57**

A.1 Introduction 57

A.2 CBOW forward propagation and backpropagation 57

A.3 Skip-gram forward propagation and backpropagation 62

A.4 Conclusion 64

References **65**

List of Figures

1.1 Identification of three clusters in unlabeled data sets 6

1.2 Clustering approaches 9

1.3 term-document matrix 11

1.4 Example of term-document matrix 13

1.5 The word2vec model 14

1.6 The CBOW model 15

1.7 The Skip-gram model 16

2.1 An example of tweet 20

2.2 The SVD decomposition of the term-document matrix 24

2.3 LSA vector representations 25

2.4 The LDA model 27

3.1 Workflow of the W2VTD 33

3.2 Train Skip-gram model over tweets 36

3.3 tweet-cluster matrix 40

4.1 Comparison with existing approaches 50

List of Tables

2.1 Probability distribuion of tweets over topics 27

2.2 Probability distribution of topics over words 28

3.1 Sample of training data with context window $C = 1$ 37

3.2 Example of word clusters 39

3.3 Example of tweet clusters 43

4.1 Data sets description 46

4.2 F-measure results given data set DS1 49

4.3 F-measure results given data set DS2 49

4.4 F-measure results given data set DS3 49

4.5 F-measure results given data set DS4 49

4.6 F-measure against K given DS1 51

4.7 F-measure against K given DS2 51

4.8 F-measure against K given DS3 51

4.9 F-measure against K given DS4 51

4.10 F-measure against N given DS1 52

4.11 F-measure against N given DS2 52
4.12 F-measure against N given DS3 52
4.13 F-measure against N given DS4 52
4.14 F-measure against C given DS1 53
4.15 F-measure against C given DS2 53
4.16 F-measure against C given DS3 53
4.17 F-measure against C given DS4 53

List of Algorithms

1.1 Basic K-means 10
3.1 Tweets Pre-processing 35
3.2 Words clustering using K-means 40
3.3 Tweets representation based on word clusters 42
3.4 Tweets clustering using K-means 43

Introduction

Microblogging is a social network medium where users are able to share brief messages, images, links, etc. Moreover, due to their distinctive aspects such as portability, simplicity and instantaneous characteristic, the number of microblog users and their posts continues to increase. Thus, microblogs are considered an enormous repository of user-generated data and therefore the center of interest of multiple research studies.

Twitter is one of the most popular microblogging platforms. Each day, Twitter users post a great number of messages referred to as tweets which are brief texts restricted to 140 characters per post. Twitter has no restriction on the content of tweets, users can report any type of information from daily life stories to breaking news and emerging events. Furthermore, Twitter provides up to date information regarding different domains such as technology, science, sports, entertainment, etc. Thus, monitoring and analyzing this rich and continuous flow of Twitter data can yield valuable information.

Topic Detection from Twitter is one of the most active research fields. It consists in discovering the topics given a collection of tweets where tweets discussing the same topic are grouped together. Thus, Topic Detection can be considered a clustering problem that aims to produce groups of topically coherent clusters. However, applying clustering algorithms on textual data, such as tweets, requires a preliminary and fundamental phase which is the text representation phase. The latter consists in converting tweets into a structured more manageable form.

Text representation is an essential phase in text clustering that greatly affects the clustering performance. In fact, documents whether they are long documents or microblogs such as tweets are clustered based on their text representations.

The most commonly used technique of text representation is the Vector Space Model (Salton, Wong, & Yang, 1975). However, this model suffers from several limitations such as high dimensionality, sparsity and the inability to preserve the semantic links in the text. To overcome the limitations of the VSM representation, other models were used in the literature such as the vector representation based on the Latent Semantic Analysis (LSA) (Deerwester, Dumais, Furnas, Landauer, & Harshman, 1990) and probabilistic models such as the Latent Dirichlet Allocation model (LDA) (Blei, Ng, & Jordan, 2003). Nevertheless, both LSA and LDA based approaches present a number of limits. While the former suffer from expensive computation, the latter are more suitable for long documents than microblogs.

To deal with these issues, we propose a new **W**ord2**V**ec based method for **T**opic **D**etection, called W2VTD. This method is based on word2vec (Mikolov, Chen, Corrado, & Dean, 2013), a recently introduced neural network model that is able to provide word vector representations in a low dimensional space while preserving the semantic relationships between words. Furthermore, the aim of the proposed approach is to produce semantically and topically homogeneous clusters.

Master Thesis Outline

This Master Thesis is structured into two main parts.

The first part, Theoretical Aspects, is composed of two main Chapters which are the following:

- Chapter 1: It presents the basic concepts of clustering and the word2vec model which represent the fundamental tools of the proposed approach
- Chapter 2: It presents an overview of studies related to Tweets Analysis for Topic Detection.

The second part of this Master Thesis presents our contributions. We divide this part into two Chapters:

- Chapter 3: It details the different steps of the proposed approach.
- Chapter 4: It is dedicated to the experimental study in which we compare W2VTD to other existing methods and we discuss the obtained results.

Finally, our dissertation ends with a conclusion and an Appendix. The conclusion summarizes all the work presented in this report and proposes further works to be done with semantic text clustering.

An Appendix A presenting the learning process of word vector representations of the word2vec model since our approach is based on those word vectors.

Part I

Theoretical Aspects

Part I explains the theoretical aspects of this work. In the first chapter, we present basic concepts of clustering, the K-means algorithm, and we focus on the text representation phase for text clustering. Then, we present the word2vec model for word vector representations. In the second chapter, we investigate in the field of Topic Detection from Twitter and we present several existing approaches.

Chapter 1

Basic Concepts of Clustering and word2vec

1.1 Introduction

The huge amount of data that requires excessive and complex processing resulted in the emerging of several machine learning techniques that aim to extract valuable information. Clustering presents one of the most commonly used techniques for handling different types of data for multiple ends. This chapter presents the fundamental tools required in this master thesis and is organized as follows: Section 1.2 provides an overview of clustering, its concept, the different clustering approaches and finally text clustering including the traditional text representation, Section 1.3 presents a recently introduced neural model for word vector representation which is the word2vec model.

1.2 Clustering

In this section we present the fundamental concept of clustering and its different approaches and we recall the basic K-means algorithm. We then present the concept of text clustering.

1.2.1 Fundamental concept of clustering

Clustering is an unsupervised machine learning technique that consists in dividing unlabeled data objects into groups called clusters, such that the objects within the same group are as similar as possible to one another, and as different as possible to the objects of the other groups as shown in Figure 1.1. Given a data set $X = (x_1, x_2, ..., x_n)$ of N objects where each object x_i can be treated as a vector $x_i = (x_{i1}, ..., x_{id})$ for $(1, ..., d)$ attributes, the goal is to create k clusters while maintaining a high intra-cluster similarity and a low inter-cluster one. Therefore, a proximity measure is needed to detect the degree of similarity between objects.

(a) unlabeled data (b) clustered data

Figure 1.1: Identification of three clusters in unlabeled data sets

1.2.2 Proximity measures

In the clustering process, choosing a proximity measure is an essential task to assign each object in the data set to the appropriate cluster. The goal is to assure a high similarity within the same cluster and a low similarity between disjoint clusters. Therefore, the distance measure can significantly affect the clusters quality. The measures can be sorted into two categories : dissimilarity measures and similarity measures.

Dissimilarity measures

Dissimilarity or distance measures indicates the remoteness between two objects. The less similar the objects are, the more distant they are. Considering two objects x_i and x_l, examples of distance measures (Xu & Wunsch, 2005) are presented next.

- **Euclidean distance:** this is the most common used distance measure in clustering problems. The distance can get arbitrarily large and is equal to zero if the data points are exactly the same. The Euclidean distance follows this expression:

$$D_E(x_i, x_l) = \Big(\sum_{j=1}^{d} (x_{ij} - x_{lj})^2 \Big)^{\frac{1}{2}} \tag{1.1}$$

- **Minkowski distance:** a generalization of the Euclidean and the Manhattan measures.

$$D_M(x_i, x_l) = \Big(\sum_{j=1}^{d} (x_{ij} - x_{lj})^{\lambda} \Big)^{\frac{1}{\lambda}} \tag{1.2}$$

- **Manhattan distance:** a particular case of the Minkowski distance at $\lambda = 1$

$$D_{Man}(x_i, x_l) = \sum_{j=1}^{d} |x_{ij} - x_{lj}| \tag{1.3}$$

Distance measures are sensitive to data scales and the type of the attributes. When some attributes are not commensurable, the larger ones dominate over the others. In such case, it is best to normalize. The min-max normalization technique can be used as follow:

$$min - maxnormalization = \frac{X - min(X)}{max(X) - min(X)} \tag{1.4}$$

Similarity measures

In opposition to dissimilarity measures, similarity measures determine the degree of likeness between two objects. The more resembling the objects are, the higher the value is. The advantage of these measures is that they scale well to large data sets. Similarity measures can be converted to distance measures as shown in the following examples:

- **Jaccard coeffcient:** (Goshtasby, 2012), calculate similarity as the intersection divided by the union of the objects. The result is bounded between 0 and 1.

$$Sim_J(x, y) = \frac{x^t y}{||x^2|| + ||y^2|| + x^t y} \tag{1.5}$$

The corresponding distance measure is:

$$D_J = 1 - Sim_J \tag{1.6}$$

- **Pearson Correlation coeffcient:**(Goshtasby, 2012), measures the covariance between two variables divided by the product of their standard deviations. The outcome detects linear correlation between objects and rages from -1 to 1. A value of 1 indicates perfect correspondence while -1 is the opposite.

$$SIM_P = \frac{\sum_{j=1}^{d}(x_{ij} - \bar{x_i})(y_{ij} - \bar{y_i})}{\sqrt{\sum_{j=1}^{n}(x_{ij} - \bar{x_i})^2}\sqrt{\sum_{j=1}^{n}(y_{ij} - \bar{y_i})^2}} \tag{1.7}$$

1.2.3 Clustering approaches

Many clustering approaches have been proposed to automatically divide objects into groups of similar data. Typically, clustering algorithms fall into two large classes: hierarchical clustering and partitional clustering as shown in Figure 1.2, however it is notable that other approaches exist, such as density based and graph based, etc.

Hierarchical clustering

This approach consists in breaking the data into a hierarchy of clusters. It consequently allows clusters to have subclusters. The nested clusters are generally represented in a tree like diagram, called a dendrogram, that displays the links between them. The number of clusters is not required in this approach since each level of the dendrogram refers to a specific k.
Hierarchical clustering is achieved through two opposite methods:

- **Agglomerative method:** This is a bottom-up approach. Initially, each object forms an individual cluster. With each step forward, on the basis of a (dis)similarity measure, the closest pairs of clusters are merged into a single cluster that belongs to a higher level in the hierarchy. The agglomeration process stops when a single cluster is reached.
- **Divisive method:** This is a top-down approach. It starts with a single cluster that groups all the data objects. Moving down the hierarchy, it proceeds by recursively splitting clusters until each cluster consists of a singleton or a stopping condition is satisfied.

Partitional clustering

The aim of partitional clustering algorithms is to partition the data set into k disjoint clusters. Therefore, unlike the hierarchical approach, the number of clusters k is required in advance. In order to assign each object to its convenient cluster, the process starts with an initial, usually random, partitioning, then it iteratively reallocates the objects on the basis of a given distance measure until it reaches a stable configuration. Partitional clustering methods are sensitive to the initial partition since it affects considerably the quality of the clusters. As an empirical solution, the algorithm is ran on different initial points. There are different approaches to the partitional clustering, K-means (MacQueen et al., 1967) is the most commonly used one.

(a) Partitional clustering (b) Hierarchical clustering

Figure 1.2: Clustering approaches

K-means

K-means algorithm (MacQueen et al., 1967) is a popular unsupervised learning algorithm for the clustering problem. The main idea is to partition the data set into k mutually exclusive clusters, where k is fixed a priori. K-means is centroid based: given a number of clusters K and a distance measure, the algorithm generates k initial centers from the data set. Each point is assigned to the cluster with the closest centroid. Then, the algorithm updates the centroids as the means of the points in each cluster. The last two step are iterated until the centroids no longer change. The complexity of K-means algorithm is *O(I,n,d,K)*, where *I* is the number of iterations, *n* is the size of the data set, *d* is the number of attributes and *K* is the number of clusters. Basic K-means is shown in Algorithm1.1.

Algorithm 1.1 Basic K-means

1: **input**: a data set X
2: **output**: final partition
3: Select k points at random as initial centers
4: **Repeat**
5: Assign each point to the closest center
6: Recompute the centroid of each cluster
7: **Until** centroids remain stable

1.2.4 Text clustering

Clustering precedes its applicability to multiple tasks related to text domain such as organizing documents into coherent categories, browsing or presenting the overall content of a collection of documents.
Text clustering discovers groups of related documents in large document collections. Its importance has grown significantly over the years as the world moves toward a paperless environment and the Web continues to dominate our lives. Text representation is the fundamental task to transform the text into structured form namely term-document matrix. The most commonly used method for text representation is the Vector Space Model (VSM)(Salton et al., 1975), which will be presented in the following.

The Vector Space Model

The VSM is an algebraic model in which documents and terms are represented as vectors in a multidimensional space. The VSM can be represented as term-document matrix with $V \times M$ shape as presented in Figure 1.3, where $|M|$ is the cardinality of the document space and $|V|$ is the vocabulary size, such as if a document contains a word, its value in the vector w_{ij} is non-zero. In many cases, the values are equal to the Term Frequency-Inverse Document Frequency (tf-idf) weights (Salton & Buckley, 1988).

$$
\begin{array}{c|cccccccc}
 & D_1 & D_2 & D_3 & \dots & D_j & \dots & D_m \\
\hline
Term_1 & w_{11} & w_{12} & w_{13} & \dots & w_{1j} & \dots & w_{1m} \\
Term_2 & w_{21} & w_{22} & w_{23} & \dots & w_{2j} & \dots & w_{2m} \\
Term_3 & w_{31} & w_{32} & w_{33} & \dots & w_{3j} & \dots & w_{3m} \\
\vdots & \vdots & \vdots & \vdots & & & & \vdots \\
Term_i & w_{i1} & w_{i2} & w_{i3} & \dots & w_{ij} & \dots & w_{im} \\
\vdots & \vdots & \vdots & \vdots & & & & \vdots \\
Term_v & w_{v1} & w_{v2} & w_{v3} & \dots & w_{vj} & \dots & w_{vm}
\end{array}
$$

Figure 1.3: term-document matrix

Term Frequency-Inverse Document Frequency

The Term Frequency-Inverse Document Frequency (TF-IDF) (Salton & Buckley, 1988) is a weighting method that is used to score the importance of a word in a document based on the number of times it occurs in the document itself and in the entire collection of documents.
The tf-idf weight is composed by two values:

- **Term Frequency (TF)**: measures how frequently a term appears in a document as follows:

$$tf_{(t,d)} = \frac{\text{Number of times term t appears in a document d}}{\text{Total number of terms in the document d}} \tag{1.8}$$

- **Inverse Document Frequency(IDF)**: measures the relevance of a term in the whole collection of documents, terms that are rare overall in the data set are considered more important.

$$idf_{(t)} = \log(\frac{\text{Total number of documents}}{\text{number of documents containing term t}}) \tag{1.9}$$

The intuition behind the TF-IDF weighting measure is that the importance of a word should increase proportionally to the number of times that word appears in a document. But, for example words such as ”is”, ”the”, ”a” are very common in the English language and tend to appear more often than other words but they carry no meaning in them, therefore their values are offset by their idf weights.

- As a result, the total tf-idf weight is calculated as follows:

$$tf - idf(t, d) = tf_{(t,d)} * idf_{(t)} \tag{1.10}$$

Example 1.1. *Considering the two following texts:*
”Computer Science graduates are disappearing from software development”
”Artificial Intelligence techniques are evolving rapidly”
The size of the vocabulary is equal to 13 referring to the number of distinct words. The tf-idf based VSM representation is a term-document $|13 \times 2|$ *matrix where each row is the vector representation of its corresponding word, and each column is the vector representation of a text. The cells of the matrix are the tf-idf weights.*

	Text 1	Text 2
computer	0.14	0.00
development	0.14	0.00
disappearing	0.14	0.00
from	0.14	0.00
graduates	0.14	0.00
science	0.14	0.00
software	0.14	0.00
are	0.00	0.00
artificial	0.00	0.20
evolving	0.00	0.20
intelligence	0.00	0.20
rapidly	0.00	0.20
techniques	0.00	0.20

Figure 1.4: Example of term-document matrix

Cosine similarity

The cosine similarity is the most adopted measure in document clustering. It is calculated by the cosine of the angle between the vector representations of documents. It ranges from 0 to 1 indicating respectively no similarity and complete similarity. The advantage of the cosine measure is its independence of vector length meaning the similarity of two documents is independent to their lengths. Given two documents d_i and d_j, the cosine similarity is computed as follows:

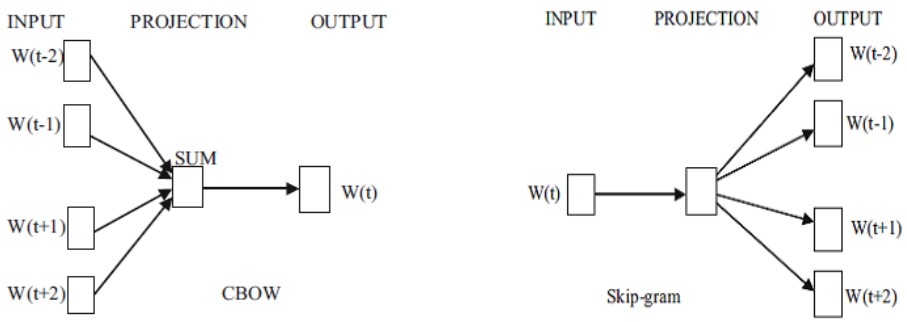
$$\cos\theta = \frac{d_i d_j}{||d_i||||d_j||} \tag{1.11}$$

The corresponding distance measure for the cosine similarity is:

$$D_C = 1 - \cos\theta \tag{1.12}$$

1.3 WORD2VEC model

As it was previously discussed in this Chapter, processing text requires a fundamental step which consists in text representation. To this end, this section presents a recently introduced model for word vector representations i.e. the word2vec model(Mikolov, Chen, et al., 2013; Mikolov, Sutskever, Chen, Corrado, & Dean, 2013). Word2vec was introduced as part of a Google project in 2013 and it is an example of Neural Networks for Language Modeling (NNLM) which are models that are able to provide word vector representations in a relatively low dimensional space while capturing the latent relations between words. Word2vec is a collection of two neural network models, the Continuous Bag-Of-Words (CBOW) and the Skip-gram. Both models were proposed to learn vector representations of words from very large data sets. Each model has three layers: input, hidden and output. Although both models are categorized as deep learning models, they are in fact known as shallow neural networks due to the fact that each contains only one hidden layer. The architecture of word2vec is given in Figure 1.5.



(a) The CBOW model (b) The Skip-gram model

Figure 1.5: The word2vec model (Mikolov, Chen, et al., 2013)

Both CBOW and Skip-gram take as input a large text corpus where each word is represented as a randomly initialized vector and then learn vector representations by predicting a word $w(t)$ given its surrounding context $\{w(t-2), w(t-1), w(t+1), w(t+2)\}$ in the case of the CBOW model, or by predicting the surrounding context of a target word when using the Skip-gram model. In fact, the output layer of both models uses softmax function to produce a probability distribution over words that yields the probability for each word to be the output word. The hidden layer of each model is considered to be the projection layer of the in-

put layer and uses a linear function directly passing its weighted sum of its inputs to the next layer (see Appendix A).

1.3.1 Continuous Bag-Of-Words Model

The first proposed neural network model in (Mikolov, Chen, et al., 2013) is the CBOW model. The aim of CBOW is to predict a target word based on its context. More precisely, the input of this model are the surrounding words (the context) and the output is a single target word. The input layer of the CBOW model is the one-hot encoded vector representations of the C context words, such that $x_k = 1$ and for $k' \neq k$, $x_{k'} = 0$, where k and k' in $\{1, 2, .., V\}$ and V denotes the size of the vocabulary. The input layer and the hidden layer are connected via W, a $V \times N$ weight matrix. Likewise, the hidden layer is connected to the output layer via a $N \times V$ weight matrix W', note that $W' \neq W^T$, and both W and W' are randomly initialized.
To obtain the vector representation of words, the CBOW model learns both weight matrices by forward propagation and backpropagation (see Appendix A), more precisely each row of W would finally correspond to the vector representation of a word in the vocabulary

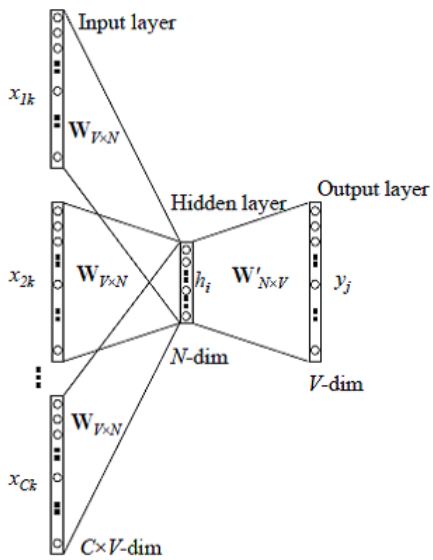


Figure 1.6: CBOW model (Rong, 2014)

1.3.2 Skip-gram Model

The second model proposed in (Mikolov, Chen, et al., 2013) is the Skip-gram model. This model is in a way the reverse of the previous one in a sense that Skip-gram aims to predict the nearby words given a word. Its architecture is shown in Figure 1.7. The Skip-gram model takes as input one-hot encoded V-dimensional vector corresponding to the input word $\mathrm{w_I}$, where $|\mathrm{V}|$ is the size of the vocabulary. Correspondingly to the precedent model, W and W' are two weight matrices that respectively connect the input layer to the hidden layer and the hidden layer to the output layer. By learning W and W', the model provides the vector representations of words in the vocabulary such that each row of W correspond to the vector representation of a word(see Appendix A).

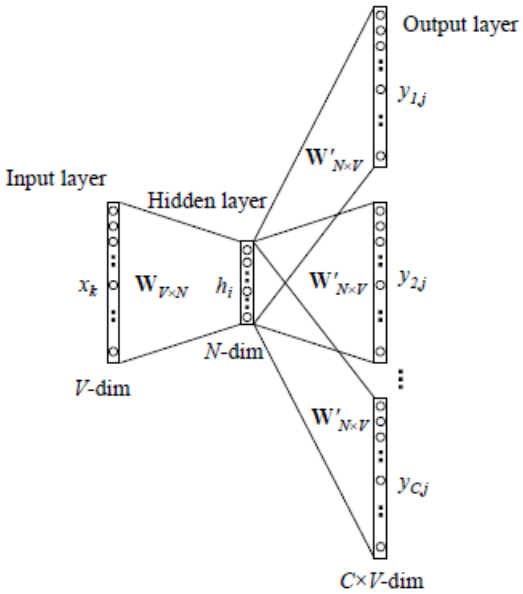


Figure 1.7: The Skip-gram model(Rong, 2014)

1.3.3 word2vec strengths

Although word2vec model architecture is simple comparing to classic neural networks with multiple layers and a non-linear hidden layer, it was proved however

capable of providing remarkable results. In fact, word2vec yields word vectors that encode the semantics of words and conserve the relationships between them by preserving the contextual information of each word. Empirically, words that are semantically close have close vector representations. Moreover, word2vec manages to capture syntactic and semantic regularities, such as male-female, verb tenses and country-capital (Mikolov, Chen, et al., 2013). Furthermore, the generated word vectors tend to obey the laws of analogy. For example, if we consider the analogy *"woman is to queen as man is to king"*, word2vec is capable of representing this analogy by applying simple algebraic operations with the vector representation of words:

$$vec_{king} - vec_{man} + vec_{woman} = vec_{queen}$$

Even more, word2vec model is able to answer questions such as *"What is the word that is similar to small in the same sense as biggest is similar to big?"* as follows:

$$X = vec_{biggest} - vec_{big} + vec_{small}$$

Word2vec can yield the correct answer by looking for the word vector closest to X using cosine distance which yields as a result the word *"smallest"*.

1.4 Conclusion

In this chapter, we presented the basic concepts on which our work is founded. We first provided an overview of clustering and its different approaches where we gave the example of K-means algorithm. We then focused on the concepts of text clustering and the traditional vector representation of text. Lastly, we presented the word2vec model and we focused on its ability of learning word vector representations while preserving the semantic relations between words. Considering the interesting characteristics of word2vec, we propose a solution for topic detection based on the vector representations yielded by this model.

In our work, we aim to handle Twitter data. Therefore, the next chapter will introduce Twitter and present a literature review of monitoring and analyzing Twitter data in general, and topic detection from Twitter in particular.

Chapter 2

Tweets Analysis for Topic Detection

2.1 Introduction

Microblogging has become, in recent years, a universally important source of information as well as an efficient mean of communication. The user base increases exponentially due to the availability of Internet access and communication devices. Microbloggers constantly express their daily thoughts, opinions and topic of interests (music, movies, sports...) and post news, events and status (Java, Song, Finin, & Tseng, 2007).
Today, Twitter is the most popular microblog, it provides rich and continuous flow of user-generated data. Such collection of data presents a profound field for research and exploitation. Moreover, microblogs are considered a particular type of textual data. In this work, we are interested in clustering Twitter data in order to detect topics. To this end, the related concepts and existing methods are presented in this chapter. Section 2.2 introduces Twitter, the particularity of its posts and its different application domains, Section 2.3 presents existing approaches that have been conducted to perform topic detection from twitter data.

2.2 Twitter

Twitter is the largest microblogging platform, its number of users and posts continue to increase at a rapidly accelerating rate counting more than 300 million[1] monthly active users that post 500 million posts per day. Twitter users shares personal facts and opinions that vary from political to religious to cultural. Furthermore, Twitter provides up to date information covering multiple areas in different domains like technology, science, sports, entertainment, etc. Moreover, people tend to reach out more to their Twitter accounts when big events take place such as elections, the World cup, Olympic Games and festivals. Twitter grants several means to assure the interaction and the communication aspects, users can have followers and can follow other accounts to receive their tweets on their timeline. In addition, Twitter provides different options: the Retweet option allows the user to republish other users tweets in order to share its content. Thus, information can be spread rapidly and attain wider public. In addition, Twitter enable users to reply to each other. A Reply starts with "@username" to indicate that the tweet is a response to the original post. Microblogs ,as their name indicates, are short in length and due to their free format, Internet users gravitates more and more towards them abandoning traditional blogs.

2.2.1 Definition of a tweet

Twitter posts are referred to as tweets: short text restricted to 140 characters (Java et al., 2007). This length constraint makes users thrifty with words. Therefore, posts are brief and concise. Tweets can contain external links, media files, mentions (user name preceded by @) and hashtags (words preceded by #) that usually indicate the general context of the tweet or a trendy topic. Aside from the length restriction, the content of tweets have no grammatical and linguistic limitations. Tweets are characterized by a randomness in word usage, they are full of jargon and can contain mixed languages and misspellings. Besides, users usually employ informal vocabulary and abbreviations and can post irrelevant content making Twitter a challenging medium to work with. To this end, a large amount of work has been conducted to process and analyze Twitter data.

[1] `https://www.statista.com/statistics/282087/number-of-monthly-active-twitter-users/`

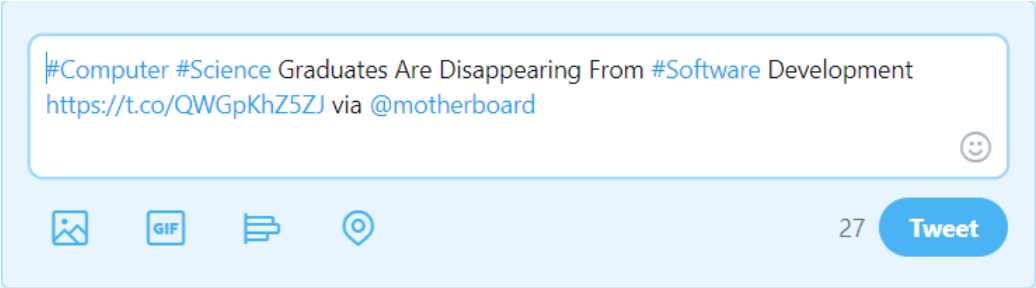


Figure 2.1: An example of tweet

2.2.2 Application domains

The user base of Twitter gathers all different kind of users (regular users, news reporters, politicians, celebrities, etc). Such diversity provides rich and valuable data for multiple applications such as sentiment analysis, news extraction, event detection and topic detection.

Sentiment analysis

Sentiment analysis, also known as opinion mining, tries to distinguish between different sentiments that are explicitly or implicitly expressed in texts, blogs, microblogs, etc. Twitter posts are rich with data that displays users preferences, point of views and sentiments towards products, public figures, events, etc. Therefore, multiple studies have been conducted on these microblogs data to perform sentiment analysis. In (Pak & Paroubek, 2010), tweets are classified into three classes that express different sentiments: positive, negative and neutral. In (Davidov, Tsur, & Rappoport, 2010), the authors exploited both hashtags and emoticons in tweets to identify different types of human emotions. (Jiang, Yu, Zhou, Liu, & Zhao, 2011) incorporated content features (words, emoticons, punctuation), lexicon features and syntactic features to perform sentiment analysis.

News extraction

Twitter is considered a strong competitor to traditional news media. Users posts cover breaking and up-to-the-minute news, such asset has become the interest of

several research studies. (Sankaranarayanan, Samet, Teitler, Lieberman, & Sperling, 2009) proposed TwitterStand, a news system based on tweets content such as geo-localization, hashtags, users interactions, etc. Similarly, (Phuvipadawat & Murata, 2010) exploited hashtags and retweets to collect, group, rank and track breaking news. (Zhao et al., 2011) carried out a comparison study between Twitter's content and other news medium and concluded that Twitter and traditional media cover a large range of similar news. (Dilrukshi, De Zoysa, & Caldera, 2013) developed a system to extract different news (crime, economy, health, sports, politics,...) related to a given country for a given time. (Jin, Cao, Zhang, Zhou, & Tian, 2017) inspected visual and statistical features of images content in microblogs to automatically distinguish real news from fake ones.

Event detection

In contrast to traditional media, Twitter is able to cover real-time events. An event can be defined as something that happens at a precise time and place (Allan, Papka, & Lavrenko, 1998). Twitter users act as sensors in that they immediately post and provide the time and region of numerous events they are part of or they simply witness. Hence, multiple event detection systems were built based on users tweets, for example the earthquake reporting system proposed in (Sakaki, Okazaki, & Matsuo, 2010), Twevent system in (C. Li, Sun, & Datta, 2012), ET system by (Parikh & Karlapalem, 2013), TEDAS system that focused on crime and disaster related Tweets using keywords and meta information provided in posts, such as location and timestamps (R. Li, Lei, Khadiwala, & Chang, 2012), (Corney, Martín, & Göker, 2014) focused on detecting sport related events .(Zhang et al., 2015) proposed an approach that detects events and predicts their popularity using users profiles information (e.g. number of followers).

Topic detection

Beyond breaking news and latest events, Twitter users share a variety of information that covers more general topics. In fact, Twitter represent a practical and useful medium for its users to keep track of topics related to their interests whether it concerns politics, sports, technology, etc.
In this master thesis, we are interested in the field of Topic Detection from Twitter data. Therefore, we present the different approaches that aim to discover topics

from tweets in the next section.

2.3 Topic detection from Twitter

Topic detection consists in grouping together tweets that discuss similar topics. Therefore, we look at it the as a clustering problem. In fact, tweets represent the unlabeled data which we like to divide into groups, such that each group represent tweets that are topically related. Consequently, applying clustering algorithms on Twitter data requires a preliminary and fundamental phase in which tweets are converted into a structured more manageable form. Therefore, we categorize the existing approaches of topic detection based on their models of representation.

2.3.1 VSM based approaches

In order to cluster tweets into different topics, the following approaches uses the VSM with tf-idf weights that we formerly presented in Chapter 1. (Tsur, Littman, & Rappoport, 2013) proposed an algorithm that clusters tweets into different topics. The approach is based on splitting the task of clustering into two phases: batch clustering and online clustering. The former consists in applying K-means on virtual documents where each is a concatenation of tweets sharing a specific hashtag. The latter relies on the results obtained in the batch phase to cluster a stream of tweets. (Rangrej, Kulkarni, & Tendulkar, 2011) conducted a comparative study between K-means, SVD and Affinity Propagation algorithms to evaluate their performances on Twitter data. (Rosa, Shah, Lin, Gershman, & Frederking, 2011) carried experiments on Twitter data using both unsupervised and supervised methods in order to group tweets into different topics. They concluded that K-means based on VSM failed to provide semantically coherent results.
Despite the efficiency of VSM in text representation, it however suffers from various limitations. First, VSM generates a high dimensional representation of data to which machine learning algorithm, such as clustering, are very sensitive. Second, the yielded representation is significantly sparse meaning it contains a great number of zero entries. Third, although the tf-idf weighting is able to give an idea of a words relative importance in a document, it does not however provide any insight into words semantic relations. To deal with these issues, several approaches have been proposed such as Latent Semantic Analysis based approaches, Latent Dirich-

let Allocation based approaches and Ontologie based approaches, which will be presented in the following sections.

2.3.2 Latent Semantic Analysis based approaches

To subdue the semantic loss caused by the VSM representation, various research studies adopted the Latent Semantic Analysis (LSA)(Deerwester et al., 1990). We first elucidate the LSA based model and then we present a number of LSA based approaches for topic detection from Twitter and discuss their limits.

The LSA based model

LSA is a method for discovering hidden concepts or latent semantic in text based on correlations between terms. LSA assumes that words that are close in meaning will occur in similar pieces of text (passage). For text representation, LSA first builds a $A_{V \times M}$ term-document matrix as illustrated in Figure 1.3. LSA then uses Singular Value Decomposition (SVD), a mathematical matrix decomposition technique such that:

$$A = U \Sigma V^T \tag{2.1}$$

where U is a $|v \times r|$ matrix with orthonormal columns, V is a $|v \times m|$ matrix with orthonormal columns and Σ is a $|r \times r|$, where r represent the number of hidden concepts, Σ is diagonal matrix with the entries sorted in decreasing order. The entries of the Σ matrix are the singular values, and the U and V matrices are the left and right singular vectors, corresponding to term and document vectors as shown in Figure 2.2.

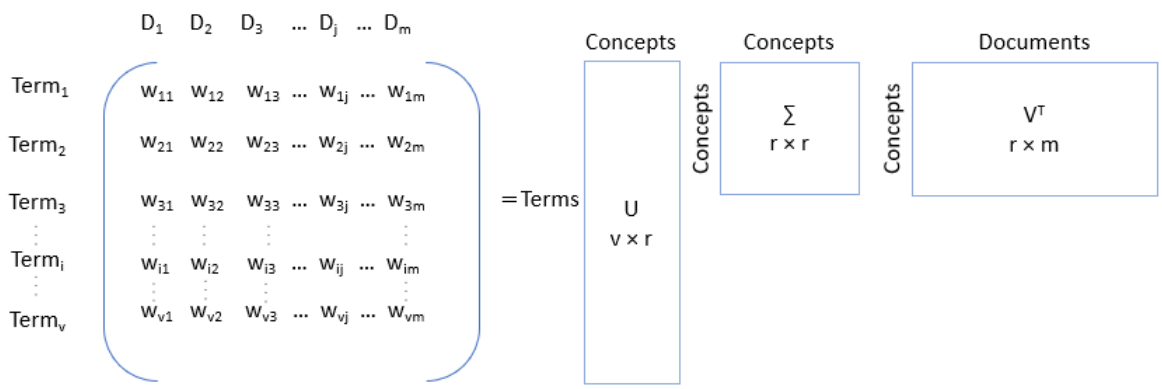


Figure 2.2: The SVD decomposition of the term-document matrix

Each term and document are expressed as a vector where each element in a vector gives the degree of participation of the document or term in the corresponding concept. LSA uses a truncated SVD to reduce the dimensionality, keeping only the k largest singular values and their associated vectors such as:

$$A_k = U_k \Sigma_k V_k^T \tag{2.2}$$

A_k is considered the best possible approximation to the matrix $A_{v \times m}$ by eliminating the zero vector and singular values that are relatively small. A_k, U_k, Σ_k and V_k^T are latent semantic representations of $A_{v \times m}$ in a latent semantic space.

Example 2.1. *We consider the two following tweets:*
"Computer Science graduates are disappearing from software development"
"Artificial Intelligence techniques are evolving rapidly"
The corresponding term-document matrix is given in Figure 1.4. Suppose we set $k = 2$*, the result of the SVD of the matrix (Eq 2.2) is the following:*

$$U_2 = \begin{bmatrix} -0.20 & 0.19 \\ -0.36 & -0.05 \\ -0.36 & -0.05 \\ -0.36 & -0.05 \\ -0.36 & -0.05 \\ -0.36 & -0.05 \\ -0.36 & -0.05 \\ -0.36 & -0.05 \\ -0.04 & 0.43 \\ -0.04 & 0.43 \\ -0.04 & 0.43 \\ -0.04 & 0.43 \\ -0.04 & 0.43 \end{bmatrix} \quad \Sigma_2 = \begin{bmatrix} 5.39 & 0.00 \\ 0.00 & 4.56 \end{bmatrix} \quad V_2 = \begin{bmatrix} -0.99 & -0.12 \\ -0.12 & 0.99 \end{bmatrix}$$

The vector representations of the words and the tweets are given respectively by the matrices U_2 and V_2^T. In fact, we scale the coordinates of these vectors by multiplying with the corresponding singular values of Σ_2 and thus the word vectors are the rows of $U_2\Sigma_2$ whereas the document vectors are represented by the columns of $Sigma_2 V_2^T$ as shown in Figure 2.3.

	concept 1	concept 2
computer	-1.68	-0.28
development	-1.98	0.24
disappearing	-1.68	-0.28
from	-1.98	-0.24
graduates	-1.68	-0.28
science	-1.98	-0.24
software	-1.68	-0.28
are	-1.11	0.87
artificial	-0.24	1.98
evolving	-0.20	2.34
intelligence	-0.24	1.98
rapidly	-0.20	2.34
techniques	-0.24	1.98

(a)

	concept 1	concept 2
Tweet 1	-5.35	-0.65
Tweet 2	-0.55	4.53

(b)

Figure 2.3: (a) Word vector representation, (b) Tweet vector representation

LSA based approaches

(Mathioudakis & Koudas, 2010) proposed TwitterMonitor, a system based on LSA that aims to cluster together words that are strongly correlated to trendy words .i.e words that show a high rate in the stream. The obtained cluster represent the trendy topics. (Nur'aini, Najahaty, Hidayati, Murfi, & Nurrohmah, 2015) presented an approach that combines both LSA technique and K-means clustering algorithm to cluster tweets into topics.
LSA-based methods enables the discovery of latent semantics in the documents. However, these methods suffers form a number of limitations such as the difficulty of interpreting the results and the high computational cost due to the matrices multiplication.

2.3.3 Latent Dirichlet Allocation based approaches

To overcome the sparsity limit of the VSM and the tf-idf weighting, several approaches were proposed using the Latent Dirichlet Allocation (LDA) (Blei et al., 2003) in order to detect topics from Twitter. In this section, we first explain the LDA model and then we present a number of LDA based approaches and examine their limitations.

The LDA model

The LDA (Blei et al., 2003) is a probabilistic topic model that represents each document as a probability distribution over a number of topics while each topic itself is a probability distribution over a number of words. Unlike other models, LDA do not represent words by weights such as their frequencies in documents, instead each word is represented as a probability distribution.
A corpus is a collection of M documents $D = d_1, d_2, \cdots, d_M$, where each document is a set of N words $W = w_1, w_2, \cdots, w_v$. Figure 2.4 shows the graphical model of LDA such that LDA follows a generative process:

- Choose $\theta \sim Dir(\alpha)$
  For each word w :
- Choose a topic $z_n \sim Multinomial(\theta)$

- Choose a word $w_n \sim Multinomial(\beta_k)$, where k is the number of topics.

α is the parameter of the Dirichlet prior on the per-document topic distribution.
β is the parameter of the Dirichlet prior on the per-topic word distribution.
θ_m is the topic distribution of document d_m
z_v is the topic for the vth word of document d_m.
And Dirichlet distribution is computed as follows:

$$p(\theta|\alpha) = \frac{\Gamma(\sum_{i=1}^{k} \alpha_i)}{\prod_{i=1}^{k} \Gamma(\alpha_i)} \theta_1^{\alpha_1 - 1} \cdots \theta_k^{\alpha_k - 1} \tag{2.3}$$

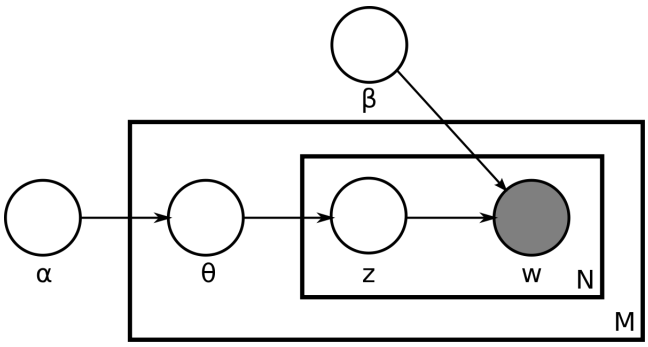


Figure 2.4: The LDA model (Blei et al., 2003)

Example 2.2. *We take the example of the two tweets in Example2.1:*
"Computer Science graduates are disappearing from software development"
"Artificial Intelligence techniques are evolving rapidly"
Suppose we want to discover 2 hidden topics, then each tweet is represented as a probability distribution over the 2 topics, while each topic is a distribution over words. The result of the LDA are shown in Table 2.1 and Table 2.2.

Table 2.1: Probability distribuion of tweets over topics

	Topic 1	Topic 2
Tweet 1	0	1
Tweet 2	1	0

Table 2.2: Probability distribution of topics over words

Word	Topic 1	Topic 2
are	0.0136	0.2258
computer	0.0136	0.1182
development	0.0136	0.1182
disappearing	0.1506	0.0107
from	0.0136	0.1182
graduates	0.0136	0.1182
science	0.0136	0.1182
software	0.0136	0.1182
artificial	0.1506	0.0107
evolving	0.1506	0.0107
intelligence	0.1506	0.0107
rapidly	0.1506	0.0107
techniques	0.1506	0.0107

LDA based approaches

Several approaches were proposed in order to detect topics in microblogs data using the LDA model. (Hong & Davison, 2010) inspected Twitter to accomplish two tasks, the first is predict popular tweets i.e tweets that have a potential to be retweeted and the second is to categorize users given the topics they discuss (art, politics, fashion, sports, technology, etc). Two models were considered, the LDA model and its extension: the Author-Topic Model (ATM) (Rosen-Zvi, Griffiths, Steyvers, & Smyth, 2004). Both models were trained on aggregated messages in order to predict popular tweets and group users and their corresponding posts into topical categories. To aggregate messages, three schemes were proposed: the MSG scheme aggregates messages generated by the same user, the USER scheme trains LDA on aggregated user profiles and the TERM scheme aggregates messages given a specific term. The USER pooling gave better results than the other two. Similarily, (Mehrotra, Sanner, Buntine, & Xie, 2013) compared five pooling schemes in order to extract topics from Twitter using the LDA model. (Yan, Guo, Lan, & Cheng, 2013) presented BTM a topic model based on biterms (i.e.,

unordered co-occurring word pairs) rather than words. Their approach provided more coherent topics than those discovered by the LDA model. (Yan, Guo, Lan, Xu, & Cheng, 2015) proposed an extension of BTM that exploited bursty biterms (word pairs that occur more frequently together in a limited time span) in order to detect bursty topics.
LDA-based models are computationally expensive and are unable to preserve semantic relations between words. LDA follows the assumption that a topic may be identified by the common co-occurrence of words. Such assumption is true when processing long documents, but fails when we consider short and noisy data such as tweets.

2.3.4 Ontology based approaches

Ontology based approaches relies on external knowledge bases to uncover the topics and hidden semantics in Twitter data. (Michelson & Macskassy, 2010) proposed Twopics, an entity-based approach that aims at discovering the topics of interest for particular users. Specifically, the retrieved entities are linked to Wikipedia pages and topics are discovered from the tree structure of the taxonomy of Wikipedia. In a similar approach, (Genc, Sakamoto, & Nickerson, 2011) mapped Twitter messages into their most similar Wikipedia pages and calculated semantic distances between messages based on the distances between their closest Wikipedia pages. (Vicient & Moreno, 2015) proposed an unsupervised method to perform semantic clustering of hashtags to discover the main topics of tweets. The approach links hashtags to WordNet synets and Wikipedia to measure their semantic similarity and then applies hierarchical clustering.
Ontology based approaches have shown better results in terms of semantic treatment of data. However, these methods requires external knowledge and their performance depends on the availability of domains.

2.4 Conclusion

In this chapter, we gave an overview of Twitter, we pointed out the characteristics of tweets and the numerous applications derived from Twitter data. We then presented various approaches for topic detection from Twitter and we discussed their limits consisting of high dimensional representations and the inability to pre-

serve the semantic links between words. To this end, the next chapter introduces our proposed approach which presents a novel representation method and aims to perform topic detection from Twitter data.

Part II

Contributions

Part II presents the contributions of this work. This part focuses on presenting our newly developed method for Topic Detection from Twitter. Chapter 3 details the different steps of the proposed approach. Chapter 4 presents the experimental study.

Chapter 3

A word2vec based method for Topic Detection from Twitter

3.1 Introduction

Topic detection from Twitter remains an active field of research due to the challenging aspect of processing tweets. As expressed in the previous chapter, existing methods suffers from a considerable number of limitations related to tweet representation such as sparsity, high dimensionality and the inability to capture the semantics behind words. Such limits undoubtedly and severely affect the quality of detected topics. To deal with this issue, we propose in this chapter a new **W**ord2**V**ec based method for **T**opic **D**etection, called W2VTD. This method presents a new vector representation of tweets for a better clustering results.

3.2 Proposed Approach: A word2vec based method for Topic Detection from Twitter

The main purpose of W2VTD is to produce semantically coherent clusters of tweets for topic detection based on a new vector representation. More precisely, we aim to overcome several limitations of word/document vector representations consisting of sparsity, high dimensionality and loss of semantics. As shown in Figure 3.1, the proposed approach consists of five steps. Given a collection of

tweets, we first perform a pre-processing step that consists in reducing the noise in the data by applying stop words removal along with additional cleansing. Then, we train the Skip-gram model of word2vec over the tweets in order to obtain word vector representations while maintaining the semantics relations between words. Thereafter, we apply K-means algorithm over the word vectors to get word clusters. The next step consists in building a new representation model of tweets based on the word clusters. Finally, we run the K-means algorithm for a second time to obtain clusters of topically similar tweets. In the following, each step will be detailed separately for better understanding of the different aims of the proposed approach.

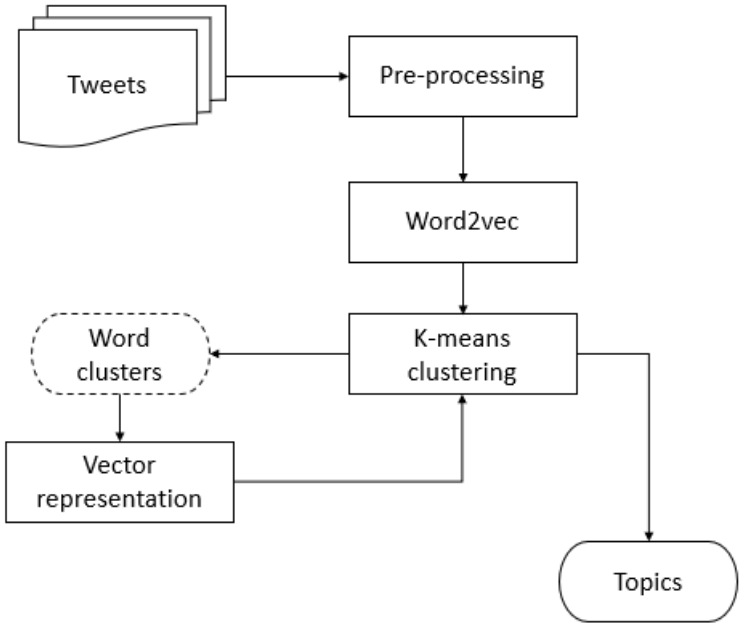


Figure 3.1: Workflow of the W2VTD

3.2.1 Step 1: Tweets Pre-processing

The raw content of tweets is extremely noisy given the fact that it has no grammatical and linguistic limitations. Therefore, in order to reduce the amount of noise, the data is subjected to various pre-processing steps.

• Stop words removal

When processing text, we notice the so-called stop words, which are very common words and appear more frequently than others. Example of these words in English language are ”the”, ”and”, ”to”, ”it”. Such words offer no additional value in deciphering what a text is about. Therefore, stop words removal consists in eliminating those frequent terms which are mostly prepositions, articles, pronouns, etc. Traditionally, stop words are chosen from a fixed list, which is constructed from lexical information. This has led to the appearance of different stop words lists which vary in size. In our work, we used the list provided in (Buckley, Salton, Allan, & Singhal, 1995) which is 571 words in length. In addition, we used a Twitter customized stop words list that groups commonly used abbreviations in tweets such as ”lol”, ”yay”, ”thnks”, etc.

• Short length words

In addition to stop words, tweets can include short words that provide little value into identifying the topic. Moreover, the excessive use of abbreviations and short words is likely an indicator of poor content. Therefore, such words can be removed from the corpus without affecting the process of detecting topics in tweets. In our work, to avoid severe filtering we discard words that do not exceed 2 characters in length.

• Additional pre-processing

Along with stop words removal and filtering short words, all the words are lower cased to ensure that same words are identified only once. Moreover, punctuation, numbers, the hashtag symbol ”#” and extra white space are filtered out. Additionally, mentions (@username) and URLs that figure in tweets are discarded in order to focus on the topics discussed in the actual tweets. Furthermore, tweets that appear multiple times are considered as spams, thus duplicated tweets are removed.

Example 3.1. *Considering the following tweet:*

"#Computer #Science Graduates Are Disappearing From #Software Development https://t.co/QWGpKhZ5ZJ via @motherboard"

Once pre-processing is done, the tweet becomes as follows:

"computer science graduates disappearing software development"

After removing stop words, URLs and short length words, we retain the most important words in the tweet. This step does not only eliminate words that provide no value to detecting the topic of tweets, but also reduces the dimensionality of the vector representations of both words and tweets. This latter ensures faster processing and lower the sparsity. The pseudo-code of this step is presented in Algorithm 3.1.

Algorithm 3.1 Tweets Pre-processing

1: **input**: T_C : corpus of tweets
2: **output**: T'_C: corpus of pre-processed tweets
3: **Begin**
4: $T'_C \leftarrow$ removeUsers(T_C)
5: $T'_C \leftarrow$ removeURLs(T'_C)
6: $T'_C \leftarrow$ lowerCase(T'_C)
7: $T'_C \leftarrow$ removePunctuation(T'_C)
8: $T'_C \leftarrow$ removeNumbers(T'_C)
9: $T'_C \leftarrow$ removeWhitespace(T'_C)
10: $T'_C \leftarrow$ removeStopwords(T'_C)
11: $T'_C \leftarrow$ removeShortwords(T'_C)
12: **End**

3.2.2 Step 2: Train Skip-gram model over tweets

The second step in our approach consists in training the Skip-gram model of word2vec over the data. More precisely, once the tweets are pre-processed as described in Step 1, we feed our corpus of tweets as input to the neural model to learn and produce word vector representations as illustrated in Figure 4.1. It is important to note that this step produces vector representations that preserve

the semantic links between words in our corpus. Hence, this representation can contribute eventually to identifying semantically coherent topics.

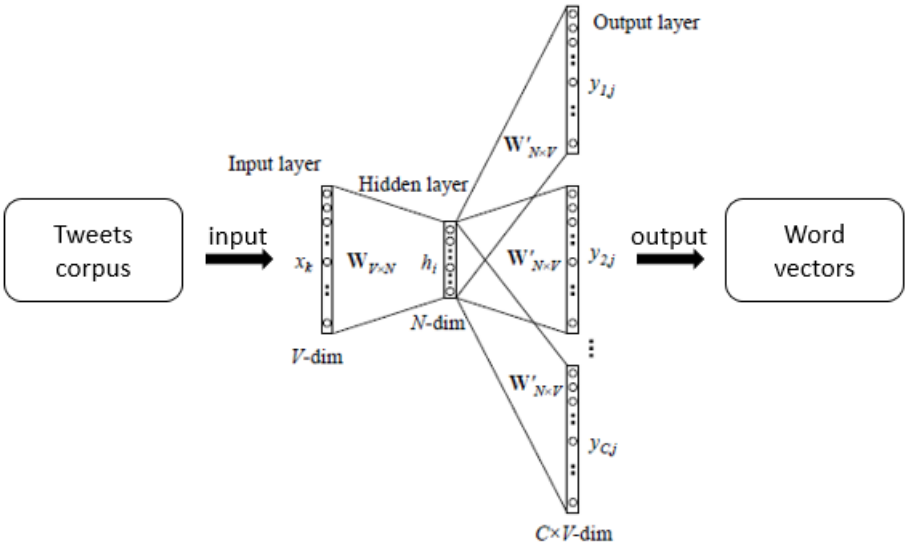


Figure 3.2: Train Skip-gram model over tweets

The Skip-gram model takes each tweet as a training sample to learn the word vector representations. More precisely, the distinct words in our data form the vocabulary V and for each word w form V, Skip-gram will provide its vector representation by predicting the context words of w. First, each word is inputted as a V-dimensional one-hot vector, and the context size C which indicates the number of context words and the dimension of the vector representations N are parameters that are fixed empirically. Then, Skip-gram will learn the weights of W and W′ given a number of iteration. The final update of W constitute the new vector representations, where each row of W corresponds to a vector representation of a word in the vocabulary. We illustrate this step by an example.

Example 3.2. *Considering the same tweet of the same example 3.1:*
"computer science graduates disappearing software development"
If we fix the context window C to 1 (one word behind and one word ahead), we can represent the sample as shown in Table 3.1.

Table 3.1: Sample of training data with context window $C = 1$

Input	Context[1]	Context[2]
computer	<>	science
science	computer	graduates
graduates	science	disappearing
disappearing	graduates	software
software	disappearing	development
development	software	<>

Every word in the vocabulary is represented by a V-dimensional one-hot encoded vector. The number of distinct words in our vocabulary is $V = 6$, thus each word in our sample is represented as follows:

computer	*[1 0 0 0 0 0]*
science	*[0 1 0 0 0 0]*
graduates	*[0 0 1 0 0 0]*
disappearing	*[0 0 0 1 0 0]*
software	*[0 0 0 0 1 0]*
development	*[0 0 0 0 0 1]*

The input layer and the hidden layer are connected by a $V \times N$ matrix W, and the hidden layer is connected to the output layer by a $N \times V$ matrix W'. Both matrices are randomly initialized. For this example, we fix $N = 3$

$$\mathrm{W} = \begin{bmatrix} -0.10 & 0.92 & 0.43 \\ 0.14 & -0.45 & 0.51 \\ -0.24 & 0.33 & 0.51 \\ 0.54 & -0.20 & 0.87 \\ -0.68 & 0.20 & 0.42 \\ 0.38 & 0.81 & 0.67 \end{bmatrix}$$

$$\mathrm{W}' = \begin{bmatrix} 0.11 & 0.88 & -0.40 & 0.75 & 0.91 & 0.25 \\ -0.32 & -0.12 & 0.05 & 0.25 & -0.13 & 0.42 \\ -0.04 & 0.27 & 0.71 & -0.03 & 0.54 & 0.33 \end{bmatrix}$$

If we consider that "computer" is the input word, then the model will learn the vector representations by predicting the words in the context of "computer" as presented in Table 3.1. The learning is done by following the forward propagation and backpropagation steps explained in Appendix A.
First, the model computes the output of the hidden layer:

$$[1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0] \times \begin{bmatrix} -0.10 & 0.92 & 0.43 \\ 0.14 & -0.45 & 0.51 \\ -0.24 & 0.33 & 0.51 \\ 0.54 & -0.20 & 0.87 \\ -0.68 & 0.20 & 0.42 \\ 0.38 & 0.81 & 0.67 \end{bmatrix} = [-0.10 \quad 0.92 \quad 0.43]$$

The computed result is propagated to the output layer as follows:

$$[-0.10 \quad 0.92 \quad 0.43] \times \begin{bmatrix} 0.11 & 0.88 & -0.40 & 0.75 & 0.91 & 0.25 \\ -0.32 & -0.12 & 0.05 & 0.25 & -0.13 & 0.42 \\ -0.04 & 0.27 & 0.71 & -0.03 & 0.54 & 0.33 \end{bmatrix}$$

$$= [-0.32 \quad -0.08 \quad 0.39 \quad 0.14 \quad 0.02 \quad 0.50]$$

Since the output layer in Skip-gram model is a softmax layer that computes the probability for each word in the vocabulary to be the context word, we have as result:

$$[0.10 \quad 0.13 \quad 0.21 \quad 0.17 \quad 0.15 \quad 0.24]$$

The actual context word of "computer" is the word "science" represented as [0 1 0 0 0 0], Skip-gram will compute the prediction error by subtracting the vector [0 1 0 0 0 0] from the probability vector. The error is back propagated through the model to update the weights of W *and* W' *(see Appendix A). This process is repeated given a chosen number of iteration and the final weights of* W *constitute the vector representations of all the words in the vocabulary, where each row of* W *corresponds to a vector representation of a word*

3.2.3 Step 3: Words clustering using K-means

This step takes as input the word vector representations provided by the previous step to perform words clustering. In fact, given that the vector representations are able to capture the semantic of each word given its context, we apply clustering over the words vectors using the K-means algorithm in order to identify different groups of words which fit together semantically.
Table 3.2 shows an example of different clusters that gathers within them words that are semantically linked. Also, we notice that a word that occurs in different forms such as ”development” and ”developer”, ”voting” and ”voter” belong to the same cluster. This aspect is ensured by the vector representations obtained by the Skip-gram model and it spare us the stemming in the pre-processing step. The pseudo-code of this step is given in Algorithm 3.2.

Table 3.2: Example of word clusters

Cluster 1	Cluster 2	Cluster 3	Cluster 4
music	democracy	financial	data
pop	voting	product	cloud
rock	freedom	trends	storage
hit	justice	investing	development
disco	protest	entrepreneur	application
dance	congress	profit	computing
radio	voter	traders	developer
songs	supporters	benefiting	database
artists	policies	consumption	mining
festivals	campaigns	stock	applications

Algorithm 3.2 Words clustering using K-means

1: **input**: *Vec* : the vector representations of words; K: number of clusters
2: **output**: *Cluster*: final partition of word clusters
3: **Begin**
4: Initialize randomly K centers from *Vec*
5: **While**(not converged) **Do**
6: Assign each word to the closest center using the cosine similarity
7: Update the centers based on the new partitioning
8: **End while**
9: **End**

3.2.4 Step 4: Tweets representation using word clusters

In the traditional vector representations (VSM), the different words of the vocabulary represent the features. However, such representation suffers from high dimensionality given that the size of the vocabulary can be significantly large. Thus, we propose in this step a new tweets representation using the words clusters provided from the previous step. In our approach, the words clusters constitute our new features, thus the number of features is reduced to the number of words clusters. Therefore, our approach reduces the dimension of the vector representations which results in a new VSM based on word clusters as shown in Figure 3.3.

$$
\begin{array}{c|ccccc}
 & C_1 & C_2 & C_3 & \cdots\cdots & C_k \\
\hline
\text{Tweet 1} & w_{11} & w_{12} & w_{13} & \cdots\cdots & w_{1k} \\
\text{Tweet 2} & w_{21} & w_{22} & w_{23} & \cdots\cdots & w_{2k} \\
\text{Tweet 3} & w_{31} & w_{32} & w_{33} & \cdots\cdots & w_{3k} \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
\text{Tweet m} & w_{m1} & w_{m2} & w_{m3} & \cdots\cdots & w_{mk}
\end{array}
$$

Figure 3.3: tweet-cluster matrix

Furthermore, each cluster groups semantically related words, thus if a word does not appear in a specific tweet T but is semantically linked to words occurring in that tweet, then their weights will be incorporated in the overall weight of the corresponding cluster in the tweet. To this end, we must change the TF-IDF weighting schema as follows:

$$w_{ij} = \frac{tf_{ij} \times idf_i}{\sum\limits_{\mathrm{w} \in \mathrm{T_j}} tf_{ij} \times idf_i} \tag{3.1}$$

where w_{ij} is the weight of the word cluster C_i in the tweet T_j, tf_{ij} is the frequency of C_i in T_j that is the sum of the tf weights of words wo belonging to cluster C_i and tweet T_j as expressed in Eq3.2, idf_i is the idf weighting of C_i which is the sum of the idf weights of words w in C_i (Eq 3.3). The denominator is for normalization.

$$tf_{ij} = \sum_{l=1}^{n} tf(\mathrm{w_l} \in \mathrm{cluster_i}, T_j), \quad i \in \{1, \cdots, k\}; j \in \{1, \cdots, m\} \tag{3.2}$$

$$idf_i = \sum_{l=1}^{n} idf(\mathrm{w_l} \in \mathrm{cluster_i}) \quad i \in \{1, \cdots, k\} \tag{3.3}$$

The new vector representations of tweets brings multiple improvements to the traditional VSM. In fact, our approach does not only overcome the dimensionality problem but it is also able to hold the semantics by using the word clusters. Moreover, the yielded representation based on the new weighting is notably less sparse than the traditional representation. The pseudo-code of this step is given in Algorithm 3.3.

Algorithm 3.3 Tweets representation based on word clusters

```
input: C : words clusters; T: corpus of tweets
output: Vec_T: the vector representations of tweets
Begin
For each tweet T_i Do
    For each cluster C_i Do
    For each word in C_i Do
        Compute the weight w_ij of C_i with regards to T_j with equation 3.1
    End For
    End For
End For
End
```

3.2.5 Step 5: Tweets clustering using K-means for topic detection

The aim of the last step of our approach is to cluster tweets into different topics. To this end, once the new tweets representation is provided by the previous step, we apply K-means algorithm over the vector representations using the cosine similarity. The desired output of this step is semantically coherent clusters of tweets (topics) which is ensured by the past steps of our approach. We also note that running K-means over the new tweet vector representations is considerably less computationally expensive than applying K-means on the traditional VSM because our new vector representations has a smaller dimension. An example of tweet clusters based on their topic similarity is shown in Table 3.3. The pseudo-code of this step is given in Algorithm 3.4

Table 3.3: Example of tweet clusters

Topics	**Tweets**
topic 1	*#Computer #Science Graduates Are Disappearing From #Software Development https://t.co/QWGpKhZ5ZJ via @motherboard* *How is the computer science path different from other routes in to software development?* *I think the university route is valid. But none of the game development degrees. Computer science with a CG.* *And tech! Huge demand for engineering or computing graduates.*
topic 2	*I had this crazy nightmare last night, USA lost to Trinidad and Tobago and didnt qualify for the 2018 World Cup.* *Bosnian fans burned the pitch after their team failed to qualify for the #World #Cup!* *The commentators go insane as Panama qualify to the World Cup for the first time in history.* *UNREAL SCENES: Panama make it to their first ever World Cup.*
topic 3	*Spain gives Catalan leader a deadline of Monday to clarify whether he declared independence.* *Spain Congress applauds PM when defending repression of voters by occupation police in #Catalonia* *Spain congressman wondering how Spain diplomacy has not prevented international media showing police repression in Catalonia.* *In Spain dozens have been fined for their tweets about Government repression.*

Algorithm 3.4 Tweets clustering using K-means

1: **input**: Vec_T : the vector representations of tweets; K': number of clusters
2: **output**: $Cluster'$: final partition of clusters
3: **Begin**
4: Initialize randomly K' centers from Vec_T
5: **While**(not converged) **Do**
6: Assign each tweet to the closest center using the cosine similarity
7: Update the centers based on the new partitioning
8: **End while**
9: **End**

3.3 Conclusion

In this chapter, we have presented our proposed approach of tweet clustering for topic detection. We have detailed the steps of the approach and its ability to overcome sparseness, high dimensionality and the loss of semantics and we associated an illustrative example to each step. Furthermore, We have shown how the word2vec model could be used to provide high quality word vector representations which eventually yielded semantically coherent clusters. In the next chapter, we will conduct an experimental study to evaluate the performance of our approach in comparison to other existing methods.

Chapter 4

Experimental study

4.1 Introduction

In order to evaluate the performance of the proposed approach, we carry out a number of experiments. The effectiveness is measured in terms of the clustering accuracy in comparison with existing approaches presented in the previous chapter and also with regards to different parameters. This Chapter is organized as follows: Section 4.2 describes the evaluation methodology that will be followed in the experiments, Section 4.3 is a description of the data sets used for evaluation, Section 4.4 describes the environment where we performed our experiments, Section 4.5 presents the evaluation measures and Section 4.6 provides the experimental results.

4.2 Evaluation methodology

In order to evaluate the performance of our approach, we conduct several experiments with regards to the collected data sets. First, we carry out a comparison with existing approaches: the traditional VSM with K-means algorithm and an LDA based approach. The purpose behind this comparison is to evaluate the effect of the different vector representations of each approach on the clustering results. Then, we evaluate the clustering performance of the proposed approach with regards to different parameters, the dimension of vector representations of

Skip-gram, the number of word clusters and the size of the context window to study their impact.

4.3 Data set description

To evaluate the performance of our approach, we collected tweets using Twitter REST API which enables the access to Twitter data. However, REST APIs have a strict rate limit on the number of request per user given a time limit and the number of tweets per request. Due to API limitations on number of requests the data gathering process lasted several days.
With the number of collected tweets, we were able to create four data sets of different sizes, each contains respectively 20,000; 30,000; 40,000 and 50,000 tweets. Furthermore, to evaluate the quality of the clustering, we manually annotated the tweets into K different topics, creating a gold standard classes. Same for each data set, there are 7 topics forming the gold standard classes: Politics, Technology, Sports, Fitness, Movies, Music and Entertainment. The summary of datasets we used are presented in Table 4.1.

Table 4.1: Data sets description

Data set	Size (number of tweets)	Topics
DS1	20,000	7
DS2	30,000	7
DS3	40,000	7
DS4	50,000	7

4.4 Experimental environment

Data collections and experiments were performed on standard PC computer equipped with Intel Core i7-4510U CPU up to 3.1 GHz, and 8GB of RAM and 1T of disk running Windows 10. All algorithms are implemented in R using the RStudio environment.

4.5 Evaluation measures

In our evaluation, we consider the accuracy of the outputted clusters as the main clustering performance criterion. To this end, we adopt the F-measure (Larsen & Aone, 1999), a commonly used measurement in conjunction with its two basis metrics, recall and precision to measure how well the produced clustering solutions C_j match the gold standard classes G_i.

Precision

The Precision measure (Larsen & Aone, 1999) of a cluster is defined as the ratio of the number of correctly clustered tweets in the cluster to the total number of tweets in the cluster (whether correct or not). Thus, with respect to a golden standard class G_i , the precision of a cluster C_j is calculated as follows:

$$Precision(G_i, C_j) = \frac{N_{ij}}{|C_j|} \tag{4.1}$$

where N_{ij} is the number of members of class G_i in cluster C_j, that is the tweets judged to be of topic t given G_i and C_j, $|C_j|$ is the number of tweets in C_j. The higher the value of precision is, the better the clustering is.

Recall

The Recall mearsure (Larsen & Aone, 1999) of a cluster is defined as the ratio of the number of correctly clustered tweets that are in the cluster to the number of correct tweets that should be in the cluster (i.e., that are in the gold standard class G_i). So the recall of a cluster C_i with respect to a class G_i is calculated as follows.

$$Recall(G_i, C_j) = \frac{N_{ij}}{|G_i|} \tag{4.2}$$

F-measure

The F-measure (Larsen & Aone, 1999) combines both precision and recall and it is defined as a weighted mean of precision and recall as follows:

$$F - measure(G_i, C_j) = 2 * \frac{Precision(G_i, C_j) \times Recall(G_i, C_j)}{Precision(G_i, C_j) + Recall(G_i, C_j)} \quad (4.3)$$

The overall F-measure is computed as follows:

$$F - measure(C) = \sum_{G_i \in G} \frac{|G_i|}{|m|} max_{C_j \in C}\{F - measure(G_i, C_j)\} \quad (4.4)$$

where m is the total number of tweets.

4.6 Experimental results and discussion

We start in this section by evaluating the performance of W2VTD compared to existing approaches such as, the LDA based method and K-means based on the traditional VSM. Then, we evaluate the impact of parameters on the performance of the proposed approach. the evaluation is carried based on different parameters, the number of word clusters K, the dimension of Skip-gram vector representations N, and finally the size of the context window C.

4.6.1 Comparison of the performance against existing methods

To investigate the ability of our approach in yielding high quality clusters, we carry out a comparison with other existing approaches, the LDA based method and K-means based on the traditional VSM. We ran the different approaches on the earlier described data sets of tweets: DS1, DS2, DS3 and DS4. The obtained results regarding each data set are given respectively in Tables 4.2, 4.3,4.4 and 4.5.

Given the results, the proposed approach outperformed both LDA based method and Kmeans based on VSM with higher precision, recall and F-measure scores. Furthermore, we can observe that the LDA based method results did not exceed a precision of 0.628, a recall of 0.633 and an F-measure of 0.615. This poor performance validates the inabilityof LDA to deal with short documents such as tweets. Although W2VTD and K-means based on VSM use the same clustering algorithm, our approach was however able to produce more coherent clusters. For example, the precision of W2V is 13% higher for DS2 and DS4 and 15% for DS1. This is explained by the ability of proposed vector representations to preserve the semantics behind words.

Table 4.2: F-measure results given data set DS1

	DS1		
	Precision	Recall	F-measure
LDA	0.610	0.527	0.565
VSM + K-means	0.692	0.671	0.681
W2VTD	0.853	0.812	0.832

Table 4.3: F-measure results given data set DS2

	DS2		
	Precision	Recall	F-measure
LDA	0.628	0.602	0.615
VSM + K-means	0.698	0.601	0.646
W2VTD	0.825	0.715	0.766

Table 4.4: F-measure results given data set DS3

	DS3		
	Precision	Recall	F-measure
LDA	0.514	0.491	0.502
VSM + K-means	0.783	0.643	0.706
W2VTD	0.807	0.732	0.768

Table 4.5: F-measure results given data set DS4

	DS4		
	Precision	Recall	F-measure
LDA	0.563	0.633	0.596
VSM + K-means	0.757	0.718	0.737
W2VTD	0.889	0.842	0.865

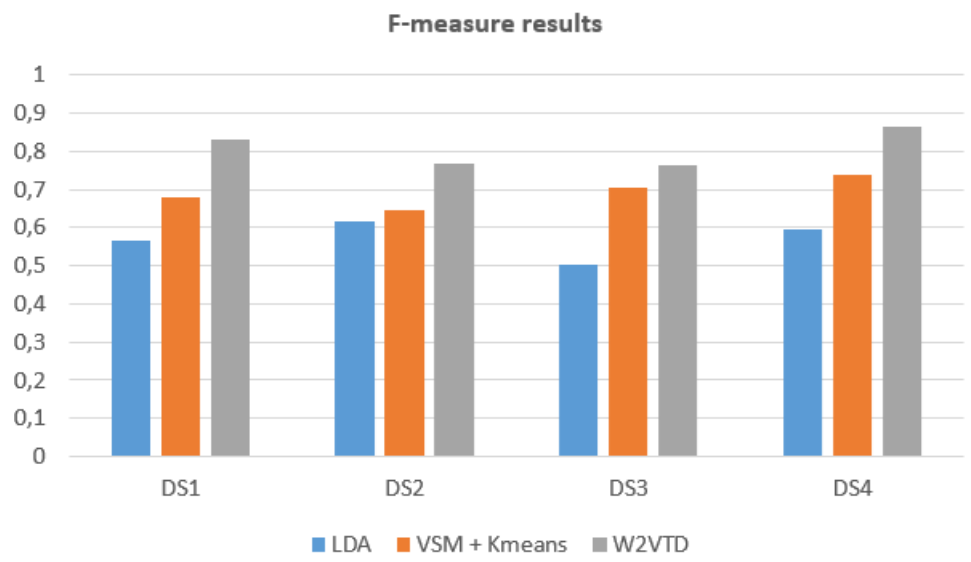


Figure 4.1: Comparison with existing approaches

4.6.2 Evaluation against the number of word clusters K

Considering that the word clusters provided by Step 3 in the proposed approach constitute the new features for the tweets vector representations, we are interested in evaluating the impact of varying the number of word clusters K. The results are given in Tables 4.6, 4.7, 4.8 and 4.9, where each table represents the result given a data set.

As the results shows, W2VTD performs differently considering different K for each data set. For example, for data set DS1 with 20.000 tweets, the precision, the recall, and the F-measure are higher when $k = 60$ scoring respectively 0.848, 0.812 and 0.830. On the other hand, for data set DS4 with a number of 50.000 tweets, the results are higher when $K \geq 100$ with a precision of 0.889 , a recall 0.842 of and an F-measure of 0.865. Thus, we deduct that the impact of the number of word clusters K varies with the size of the data set.

Table 4.6: F-measure against K given DS1

	DS1		
	Precision	Recall	F-measure
k=60	0.848	0.812	0.830
k=80	0.839	0.794	0.816
k=100	0.853	0.812	0.832
k=120	0.805	0.801	0.803

Table 4.7: F-measure against K given DS2

	DS2		
	Precision	Recall	F-measure
k=60	0.845	0.775	0.808
k=80	0.899	0.896	0.898
k=100	0.825	0.715	0.766
k=120	0.808	0.776	0.791

Table 4.8: F-measure against K given DS3

	DS3		
	Precision	Recall	F-measure
k=60	0.853	0.772	0.814
k=80	0.805	0.835	0.820
k=100	0.807	0.732	0.768
k=120	0.802	0.762	0.782

Table 4.9: F-measure against K given DS4

	DS4		
	Precision	Recall	F-measure
k=60	0.854	0.773	0.811
k=80	0.834	0.759	0.795
k=100	0.889	0.842	0.865
k=120	0.828	0.789	0.808

4.6.3 Evaluation against the dimension N of Skip-gram vector representations

The dimension of word vector representation N is a parameter that needs to be fixed in order to learn vector representations with the Skip-gram model as explained in Step 2 of our approach. The words are then clustered based on their vector representations to form the new features of the new representation based on word clusters. Therefore, it is important to study the impact of varying the vectors dimension N on the clustering performance.
The obtained results are given in Table 4.10, 4.11, 4.12 and 4.13. We notice that when choosing a low dimensionality for example$N = 50$, W2VTD give lower precsion, recall and F-measure, independently of the data set size. However, when $100 < N < 200$ W2VTD yields better results. Hence, we can conclude that keeping a reasonably large word vector representation ensures a better preservation of the semantic links between the different words in the vocabulary.

Table 4.10: F-measure against N given DS1

	DS1		
	Precision	Recall	F-measure
N=50	0.810	0.793	0.801
N=100	0.853	0.812	0.832
N=200	0.827	0.789	0.808

Table 4.11: F-measure against N given DS2

	DS2		
	Precision	Recall	F-measure
N=50	0.815	0.765	0.789
N=100	0.825	0.715	0.766
N=200	0.796	0.819	0.807

Table 4.12: F-measure against N given DS3

	DS3		
	Precision	Recall	F-measure
N=50	0.732	0.702	0.717
N=100	0.807	0.732	0.768
N=200	0.862	0.779	0.818

Table 4.13: F-measure against N given DS4

	DS4		
	Precision	Recall	F-measure
N=50	0.799	0.734	0.766
N=100	0.889	0.842	0.865
N=200	0.809	0.808	0.809

4.6.4 Evaluation against the context window C

As discussed in Chapter 1, the Skip-gram models learns the word vector representations by predicting the context words given an input word, i.e. the words in its surroundings. Thus, the size of the context window C needs to be fixed. Moreover, this parameter holds within it the contextual information which is the words semantics. Therefore, we study the effect of varying the window size C on the clustering performance.
Since tweets are short text with a restricted number of characters which evidently leads to a limited number of words, and considering the pre-processing step of text, we kept C between 2 and 6.
Tables 4.14, 4.15, 4.16 and 4.17 shows the obtained results. From these tables, we notice that varying this parameter yields better results in case of a larger data set. For example, the precision takes 0.837 for DS1 while it takes 0.889 for DS4. However, the overall results remains within close intervals. Intuitively, we state that this parameter could have a more important impact in case of longer documents.

Table 4.14: F-measure against C given DS1

	DS1		
	Precision	Recall	F-measure
C=2	0.829	0.797	0.813
C=4	0.853	0.812	0.832
C=6	0.838	0.733	0.782

Table 4.15: F-measure against C given DS2

	DS2		
	Precision	Recall	F-measure
C=2	0.847	0.763	0.803
C=4	0.825	0.715	0.766
C=6	0.869	0.801	0.834

Table 4.16: F-measure against C given DS3

	DS3		
	Precision	Recall	F-measure
C=2	0.791	0.809	0.800
C=4	0.807	0.722	0.768
C=6	0.849	0.751	0.797

Table 4.17: F-measure against C given DS4

	DS4		
	Precision	Recall	F-measure
C=2	0.840	0.762	0.799
C=4	0.889	0.842	0.865
C=6	0.837	0.772	0.803

4.7 Conclusion

In this chapter, we have conducted several experiments regarding different aspects and data sets. First, we compared our approach with two existing ones. The obtained results from this comparison shows that our approach outperforms the other two. Then, we studied the impact of varying different parameters on the clustering performance. The results shows that the overall clustering quality remains high. As a conclusion, the proposed approach is able to provide high quality and coherent clusters within the domain of topic detection.

Conclusion

Throughout this Master Thesis, we have investigated in the field of Topic Detection from Twitter that consists in identifying and grouping together tweets discussing the same topic. In fact, Topic Detection is a clustering problem that aims to produce clusters that are topically homogeneous. However, clustering text data such as tweets necessarily demands a text representation phase in order to convert the data into a structured more manageable form.

Actually, text representation is a fundamental step that affects significantly the performance of the clustering algorithm in yielding coherent topics. Hence, multiple studies have proposed different representation models. However, the existing approaches suffer from a number of limitations such as high dimensionality, sparsity and the loss of words semantics.

To overcome the limitations of existing approaches, we have proposed in this Master Thesis a new approach for Topic Detection called W2VTD that is able to provide semantically coherent clusters. W2VTD is based on a recently introduced neural network called word2vec that is able to generate word vector representations in a low dimension while preserving the semantic links between words.

The proposed approach takes as input a corpus of tweets and follows five steps. First of all, W2VTD performs a pre-processing step on the corpus by removing stops words and any information that has no value in identifying the topic. Secondly, we train the Skip-gram model of word2vec over the tweets in order to obtain word vector representations while maintaining the semantics relations between words. Thirdly, W2VTD applies the K-means algorithms over the word vectors to produce word clusters that are semantically coherent. The next steps consists in building a new representation model of tweets based on those word

clusters. Finally, the tweets are clustered based on their new vector representations using the K-means algorithm. The final output of W2VTD is topically coherent clusters of tweets.

In order to evaluate the performance of the proposed approach, we have carried out multiple experiments in which we have compared W2VTD with other existing approaches for Topic Detection given different data sets of tweets.

In this Master Thesis, we have focused on the text representation phase of clustering. Although this phase has a great impact on the overall clustering performance, the algorithm itself can be inspected in order to improve the outputted results. An extension of W2VTD could use a heuristic to fix the initial number of clusters. Furthermore, in our work we handled tweets, their short length holds to the assumption that each tweet discusses one topic only. However, with the intention to extend our approach to handle longer documents where each document can exhibit several topics, we can consider a fuzzy clustering algorithm.

Appendix

In this Appendix, we will present the learning process of the Continuous Bag Of Words model and the Skip-gram model for producing word vector representations

Appendix A

The learning process of word2vec model

A.1 Introduction

In this appendix we detail the learning process of the Continuous Bag of Words (CBOW) model and the Skip-gram model of word2vec. Both models learn the word vector representations by forward propagation and backpropagation.

A.2 CBOW forward propagation and backpropagation

The input of the CBOW model are the surrounding words (the context) and the output is a single target word. For a better understanding of the CBOW model, we assume that the context consists of only one word. The model is consequently represented as shown in Figure A.1

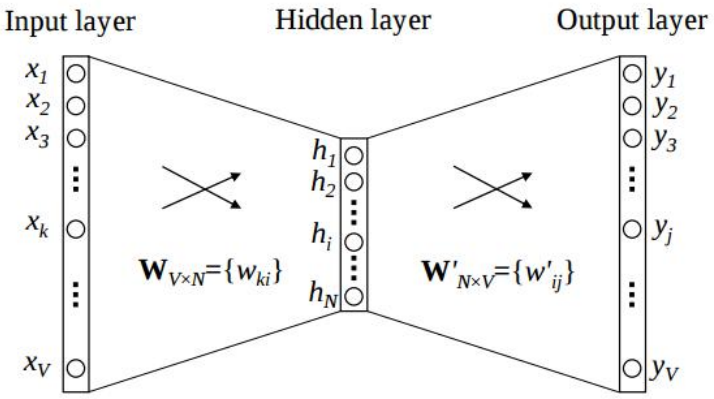


Figure A.1: CBOW model with a single-word context (Rong, 2014)

The input layer of the simplified CBOW model is the one-hot encoded vector representation of the context word, such that $x_k = 1$ and for $k' \neq k$, $x_{k'} = 0$, where k and k' in $\{1, 2, .., V\}$ and V denotes the size of the vocabulary. The input layer and the hidden layer are connected via W, a $V \times N$ weight matrix. Likewise, the hidden layer is connected to the output layer via a $N \times V$ weight matrix W', note that $W' \neq W^T$, and both W and W' are randomly initialized.
To obtain the vector representation of words, the CBOW model learns both weight matrices by forward propagation and backpropagation.

Forward propagation

The first step is to compute the output of the hidden layer, such that:

$$\mathrm{h} = \mathrm{W}^{\mathrm{T}}\mathrm{x} \tag{A.1}$$

Since x is non zero only at x_k, then $\mathrm{W}^{\mathrm{T}}\mathrm{x}$ will be the kth row of W denoted $\mathrm{v}_{\mathrm{w_I}}$ which is in fact the vector representation of the input context word w_I, therefore we have:

$$\mathrm{h} = \mathrm{W}^{\mathrm{T}}\mathrm{x} = \mathrm{v}_{\mathrm{w_I}} \tag{A.2}$$

The next step is to compute the inputs to the output layer given W', such that:

$$\mathrm{u} = \mathrm{W}'^{\mathrm{T}}\mathrm{h} \tag{A.3}$$

Let $\mathrm{v}'_{\mathrm{w_j}}$ be the jth column of the matrix W', which is also the vector representation of the word w_j of the vocabulary, then for every element u_j of u we have:

$$u_j = \mathrm{v}'_{\mathrm{w_j}}{}^{\mathrm{T}}\mathrm{h} \tag{A.4}$$

Finally, to compute the output y_j of the output layer, the softmax function is applied to obtain the posterior distribution of words as follows:

$$y_j = p(w_j|w_I) = \frac{exp(u_j)}{\sum_{j'=1}^{V} exp(u_{j'})} \tag{A.5}$$

Substituting (A.28) and (A.4) into (A.5), we obtain:

$$p(w_j|w_I) = \frac{exp(\mathrm{v}'_{\mathrm{w_j}}{}^{\mathrm{T}}\mathrm{v}_{\mathrm{w_I}})}{\sum_{j'=1}^{V} exp(\mathrm{v}'_{\mathrm{w_{j'}}}{}^{\mathrm{T}}\mathrm{v}_{\mathrm{w_I}})} \tag{A.6}$$

$\mathrm{v_w}$ and v'_{w} are two different vector representation of the word w, while the first is provided by a row of W, the second is a column of W'.

Learning the weight matrices with backpropagation

In this phase, we train the CBOW to learn the weight matrices W and W'. The aim of the training is to improve the prediction of the output word w_O (denote its index in the output layer as $j*$), in other words maximize eq A.5, such that:

$$max\, p(w_O|w_I) = max\, y_{j*} \tag{A.7}$$

$$= max\, log\, y_{j*} \tag{A.8}$$

$$= max\, log\, \frac{exp(u_{j*})}{\sum_{j'=1}^{V} exp(u_{j'})} \tag{A.9}$$

$$= u_{j*} - log \sum_{j'=1}^{V} exp(u_{j'}) \tag{A.10}$$

Maximizing $p(w_O|w_I)$ is also minimizing a loss function E, such that:

$$E = -\, log\, p(w_O|w_I) \tag{A.11}$$

• Updating the hidden-output layer weights

The first step is to derive the update equation of the weights between hidden and output layers with respect to u_j as follows:

$$\frac{\partial E}{\partial u_j} = y_j - t_j := e_j \tag{A.12}$$

where $t_j = 1$ if $j = j*$ otherwise $t_j = 0$. This derivative is none other than the prediction error e_j of the output layer.
Next we derive E with respect to the output weight w'_{ij} to obtain the gradient on the hidden → output weights.

$$\frac{\partial E}{\partial w'_{ij}} = \frac{\partial E}{\partial u_j} \cdot \frac{\partial u_j}{\partial w'_{ij}} = e_j \cdot h_i \tag{A.13}$$

where h_i is the output of the ith node of the hidden layer.
To obtain the weight updating equation for hidden → output weights, stochastic gradient descent is applied:

$$w'^{(new)}_{ij} = w'^{(old)}_{ij} - \eta \cdot e_j \cdot h_i \tag{A.14}$$

or

$$\mathrm{v}'^{(\mathrm{new})}_{\mathrm{w_j}} = \mathrm{v}'^{(\mathrm{old})}_{\mathrm{w_j}} - \eta \cdot e_j \cdot \mathrm{h} \qquad \texttt{for}\ j = 1, \cdots, V \tag{A.15}$$

• Updating the input-hidden layer weights

After updating the equations for W', the next step is to update W by computing the derivative of E given the output of the hidden layer as follows:

$$\frac{\partial E}{\partial h_i} = \sum_{j=1}^{V} \frac{\partial E}{\partial u_j} \frac{\partial u_j}{\partial h_i} = \sum_{j=1}^{V} e_j \cdot w'_{ij} := \mathrm{EH_i} \tag{A.16}$$

EH, a N-dim vector, is the sum of the output vectors of all words in the vocabulary, weighted by their prediction error.

Next,the derivative of E with regards to $W = \{w_{ki}\}$ is computed.
Considering

$$h_i = \sum_{k=1}^{V} x_k \cdot w_{ki} \tag{A.17}$$

we then have:

$$\frac{\partial E}{\partial w_{ki}} = \frac{\partial E}{\partial h_i}\frac{\partial h_i}{\partial w_{ki}} \tag{A.18}$$

$$= \mathrm{EH_i} \cdot x_k \tag{A.19}$$

$$\frac{\partial E}{\partial \mathrm{W}} = \mathrm{x} \cdot \mathrm{EH} \tag{A.20}$$

Equation A.20 yields a $V \times N$ matrix with only one non zero row since x is one-hot vector. Thus, the update equation of W, using stochastic gradient descent is the following:

$$\mathrm{v}_{\mathrm{w_I}}^{(\mathrm{new})} = \mathrm{v}_{\mathrm{w_I}}^{(\mathrm{old})} - \eta \mathrm{EH} \tag{A.21}$$

Heretofore, we presented the learning process of the CBOW model to produce vector representation of words with only one word in the context, we now edefine some equations for the case of multi-word context CBOW model.

To compute the output of the hidden layer, the CBOW model takes the average of the vectors of the input context words, and use the product of W and the average vector as the output such as:

$$\mathrm{h} = \frac{1}{C} W^T (\mathrm{x}_1 + \mathrm{x}_2 + \cdots + \mathrm{x}_\mathrm{C}) \tag{A.22}$$

$$= \frac{1}{C} (\mathrm{v}_{\mathrm{w}_1} + \mathrm{v}_{\mathrm{w}_2} + \cdots + \mathrm{v}_{\mathrm{w}_\mathrm{C}}) \tag{A.23}$$

where C is context size, $\mathrm{w}_1, \cdots, \mathrm{w}_\mathrm{C}$ are the words the in the context, and v_w is the input vector of a word w.

The redefinition of the loss function is as follows:

$$E = -\log p(w_O|w_{I,1}, \cdots, w_{I,C}) \tag{A.24}$$

$$= -u_{j*} + \log \sum_{j'=1}^{V} \exp(u_{j'}) \tag{A.25}$$

$$= -\mathrm{v'_{w_O}}^{\mathrm{T}} \cdot \mathrm{h} + \log \sum_{j'=1}^{V} \exp(\mathrm{v'_{w_O}}^{\mathrm{T}} \cdot \mathrm{h}) \tag{A.26}$$

The update equation for the hidden → output weights stay the same as that for the single word context model (see Eq A.15). But the update equation for input→hidden weights is redefined as follows:

$$\mathrm{v}_{\mathrm{w_{I,c}}}^{(\mathrm{new})} = \mathrm{v}_{\mathrm{w_{I,c}}}^{(\mathrm{old})} - \frac{1}{C}\eta \mathrm{EH} \qquad \texttt{for}\, c = 1, \cdots, C \tag{A.27}$$

where $v_{w_{I,c}}$ is the input vector of the $c^t h$ word in the input context.

A.3 Skip-gram forward propagation and backpropagation

Forward propagation

Similarly to the CBOW model, the hidden layer nodes simply pass the weighted sum of inputs to the next layer,we recall the definition of h below:

$$\mathrm{h} = \mathrm{W}^{\mathrm{T}}\mathrm{x} = \mathrm{v}_{\mathrm{w_I}} \tag{A.28}$$

Likewise, the inputs to each of the output nodes is computed by the weighted sum of its inputs. Therefore, the input to the jth node of the cth output word is

$$u_{c,j} = u_j = \mathrm{v'_{w_j}}^{\mathrm{T}} \cdot \mathrm{h} \qquad \texttt{for}\, c = 1, 2, \cdots, C \tag{A.29}$$

where $\mathrm{v'_{w_j}}$ is the output vector of the jth word $\mathrm{w_j}$ in the vocabulary.
Unlike the CBOW model, the output layer of Skip-gram returns multiple outputs $\{y_1, \cdots, y_C\}$ by computing C multinomial distributions given an input word w_I.

$$p(w_{c,j} = w_{O,c}|w_I) = y_{c,j} = \frac{\exp(u_{c,j})}{\sum_{j'=1}^{V} \exp(u_{j'})} \tag{A.30}$$

Learning the weight matrices with backpropagation

To learn the weight matrices, a loss function is defined as follows:

$$E = -\log p(w_{O,1}, w_{O,2}, \cdots, w_{O,C}|w_I) \tag{A.31}$$

$$= -\log \prod_{c=1}^{C} \frac{\exp(u_{c,j*c})}{\sum_{j'=1}^{V} \exp(u'_j)} \tag{A.32}$$

$$= -\sum_{c=1}^{C} u_{j*c} + C \cdot \sum_{j'=1}^{V} \exp(u_{j'}) \tag{A.33}$$

where $j*_c$ is the index of the actual c^{th} output context word in the vocabulary.

• Updating the hidden-output layer weights

This step consists in computing the derivative of E with respect to $u_{c,j}$ to obtain the prediction error.

$$\frac{\partial E}{\partial u_{c,j}} = y_{c,j} - t_{c,j} := e_{c,j} \tag{A.34}$$

EI, a V -dimensional vector, denotes the sum of prediction errors over all context words, such that:

$$\mathrm{EI_j} = \sum_{c=1}^{C} e_{c,j} \tag{A.35}$$

Next, the derivative of E on the weight matrix $\mathrm{W}' = w'_{ij}$ is computed as follows:

$$\frac{\partial E}{\partial w'_{i,j}} = \sum_{c=1}^{C} \frac{\partial E}{\partial u_{c,j}} \cdot \frac{\partial u_{c,j}}{\partial w'_{i,j}} = \mathrm{EI_j} \cdot h_i \tag{A.36}$$

Using gradient descent, the update equation for the hidden →output matrix W' is:

$$w'^{(new)}_{ij} = w'^{(old)}_{ij} - \eta \cdot \mathrm{EI} \cdot h_i \tag{A.37}$$

or

$$\mathrm{v}'^{(\mathrm{new})}_{\mathrm{w_j}} = \mathrm{v}'^{(\mathrm{old})}_{\mathrm{w_j}} - \eta \cdot e_j \cdot \mathrm{EI} \cdot \mathrm{h} \qquad \texttt{for}\ j = 1, 2, \cdots, V \tag{A.38}$$

• Updating the input-hidden layer weights

The derivative of the update equation for the input → hidden matrix is similar to Eq (A.16), except that the prediction error e_j is replaced with $\mathrm{EI_j}$. Thus, the update equation is identical to Eq(A.21) and is as follows:

$$\mathrm{v}_{\mathrm{w_I}}^{(\mathrm{new})} = \mathrm{v}_{\mathrm{w_I}}^{(\mathrm{old})} - \eta \cdot \mathrm{EH} \tag{A.39}$$

where EH is an N-dim vector, each element of which is defined as:

$$\mathrm{EH_i} = \sum_{j=1}^{V} \mathrm{EI_j} \cdot w'_{ij} \tag{A.40}$$

A.4 Conclusion

In this Appendix, we have detailed the learning process of the CBOW model and the Skip-gram model in order to produce word vector representations.

References

Allan, J., Papka, R., & Lavrenko, V. (1998). On-line new event detection and tracking. In *Proceedings of the 21st annual international acm sigir conference on research and development in information retrieval* (pp. 37–45). ACM.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, *3*(Jan), 993–1022.

Buckley, C., Salton, G., Allan, J., & Singhal, A. (1995). Automatic query expansion using smart: Trec 3. *NIST special publication sp*, 69–69.

Corney, D., Martín, C. J., & Göker, A. (2014). Spot the ball: Detecting sports events on twitter. In *Ecir* (pp. 449–454). Springer.

Davidov, D., Tsur, O., & Rappoport, A. (2010). Enhanced sentiment learning using twitter hashtags and smileys. In *Proceedings of the 23rd international conference on computational linguistics: posters* (pp. 241–249). Association for Computational Linguistics.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American society for information science*, *41*(6), 391.

Dilrukshi, I., De Zoysa, K., & Caldera, A. (2013). Twitter news classification using svm. In *Computer science & education (iccse)* (pp. 287–291). IEEE.

Genc, Y., Sakamoto, Y., & Nickerson, J. (2011). Discovering context: classifying tweets through a semantic transform based on wikipedia. *Foundations of augmented cognition. Directing the future of adaptive systems*, 484–492.

Goshtasby, A. A. (2012). Similarity and dissimilarity measures. In *Image registration* (pp. 7–66). Springer.

Hong, L., & Davison, B. D. (2010). Empirical study of topic modeling in twitter.

In *Proceedings of the first workshop on social media analytics* (pp. 80–88). ACM.

Java, A., Song, X., Finin, T., & Tseng, B. (2007). Why we twitter: understanding microblogging usage and communities. In *Proceedings of the 9th webkdd and 1st sna-kdd 2007 workshop on web mining and social network analysis* (pp. 56–65). ACM.

Jiang, L., Yu, M., Zhou, M., Liu, X., & Zhao, T. (2011). Target-dependent twitter sentiment classification. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1* (pp. 151–160).

Jin, Z., Cao, J., Zhang, Y., Zhou, J., & Tian, Q. (2017). Novel visual and statistical image features for microblogs news verification. *IEEE Transactions on Multimedia*, *19*(3), 598–608.

Larsen, B., & Aone, C. (1999). Fast and effective text mining using linear-time document clustering. In *Proceedings of the fifth acm sigkdd international conference on knowledge discovery and data mining* (pp. 16–22).

Li, C., Sun, A., & Datta, A. (2012). Twevent: segment-based event detection from tweets. In *Proceedings of the 21st acm international conference on information and knowledge management* (pp. 155–164). ACM.

Li, R., Lei, K. H., Khadiwala, R., & Chang, K. C.-C. (2012). Tedas: A twitter-based event detection and analysis system. In *Data engineering (icde), 2012 ieee 28th international conference on* (pp. 1273–1276). IEEE.

MacQueen, J., et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th berkeley symposium on mathematical statistics and probability* (Vol. 1, pp. 281–297).

Mathioudakis, M., & Koudas, N. (2010). Twittermonitor: trend detection over the twitter stream. In *Proceedings of the 2010 acm sigmod international conference on management of data* (pp. 1155–1158). ACM.

Mehrotra, R., Sanner, S., Buntine, W., & Xie, L. (2013). Improving lda topic models for microblogs via tweet pooling and automatic labeling. In *Proceedings of the 36th international acm sigir conference on research and development in information retrieval* (pp. 889–892). ACM.

Michelson, M., & Macskassy, S. A. (2010). Discovering users' topics of interest on twitter: a first look. In *Proceedings of the fourth workshop on analytics for noisy unstructured text data* (pp. 73–80). ACM.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Dis-

tributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111–3119).

Nur'aini, K., Najahaty, I., Hidayati, L., Murfi, H., & Nurrohmah, S. (2015). Combination of singular value decomposition and k-means clustering methods for topic detection on twitter. In *Advanced computer science and information systems (icacsis), 2015 international conference on* (pp. 123–128). IEEE.

Pak, A., & Paroubek, P. (2010). Twitter as a corpus for sentiment analysis and opinion mining. In *Lrec* (Vol. 10).

Parikh, R., & Karlapalem, K. (2013). Et: events from tweets. In *Proceedings of the 22nd international conference on world wide web* (pp. 613–620). ACM.

Phuvipadawat, S., & Murata, T. (2010). Breaking news detection and tracking in twitter. *Web Intelligence and Intelligent Agent Technology (WI-IAT), IEEE/WIC/ACM International Conference*, *3*, 120-123.

Rangrej, A., Kulkarni, S., & Tendulkar, A. V. (2011). Comparative study of clustering techniques for short text documents. In *Proceedings of the 20th international conference companion on world wide web* (pp. 111–112).

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

Rosa, K. D., Shah, R., Lin, B., Gershman, A., & Frederking, R. (2011). Topical clustering of tweets. *Proceedings of the ACM SIGIR: SWSM*.

Rosen-Zvi, M., Griffiths, T., Steyvers, M., & Smyth, P. (2004). The author-topic model for authors and documents. In *Proceedings of the 20th conference on uncertainty in artificial intelligence* (pp. 487–494). AUAI Press.

Sakaki, T., Okazaki, M., & Matsuo, Y. (2010). Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on world wide web* (pp. 851–860). ACM.

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information processing & management*, *24*(5), 513–523.

Salton, G., Wong, A., & Yang, C.-S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, *18*(11), 613–620.

Sankaranarayanan, J., Samet, H., Teitler, B. E., Lieberman, M. D., & Sperling, J. (2009). Twitterstand: news in tweets. In *Proceedings of the 17th acm sigspatial international conference on advances in geographic information systems* (pp. 42–51). ACM.

Tsur, O., Littman, A., & Rappoport, A. (2013). Efficient clustering of short messages into general domains. In *Icwsm.*

Vicient, C., & Moreno, A. (2015). Unsupervised topic discovery in micro-

blogging networks. *Expert Systems with Applications*, *42*(17), 6472–6485.

Xu, R., & Wunsch, D. (2005). Survey of clustering algorithms. *IEEE Transactions on neural networks*, *16*(3), 645–678.

Yan, X., Guo, J., Lan, Y., & Cheng, X. (2013). A biterm topic model for short texts. In *Proceedings of the 22nd international conference on world wide web* (pp. 1445–1456). ACM.

Yan, X., Guo, J., Lan, Y., Xu, J., & Cheng, X. (2015). A probabilistic model for bursty topic discovery in microblogs. In *Aaai* (pp. 353–359).

Zhang, X., Chen, X., Chen, Y., Wang, S., Li, Z., & Xia, J. (2015). Event detection and popularity prediction in microblogging. *Neurocomputing*, *149*, 1469–1480.

Zhao, W., Jiang, J., Weng, J., He, J., Lim, E., Yan, H., & Li, X. (2011). Comparing twitter and traditional media using topic models. *European Conference on Information Retrieval, Springer*, 338–349.