Short Text Categorization using World Knowledge

Zur Erlangung des akademischen Grades eines
Doktors der Ingenieurwissenschaften
(Dr.-Ing.)

von der KIT-Fakultät für Wirtschaftswissenschaften
des Karlsruher Instituts für Technologie (KIT)

genehmigte
DISSERTATION

von

Dipl.-Inform. Rima Türker

Tag der mündlichen Prüfung:	03 March 2021
Referent:	Prof. Dr. Harald Sack
Korreferent:	Prof. Dr. Heiko Paulheim

To my parents.

Abstract

The content of the World Wide Web is drastically multiplying, and thus the amount of available online text data is increasing every day. Today, many users contribute to this massive global network via online platforms by sharing information in the form of a short text. Such an immense amount of data covers subjects from all the existing domains (e.g., Sports, Economy, Biology, etc.). Further, manually processing such data is beyond human capabilities. As a result, Natural Language Processing (NLP) tasks, which aim to automatically analyze and process natural language documents have gained significant attention. Among these tasks, due to its application in various domains, text categorization has become one of the most fundamental and crucial tasks.

However, the standard text categorization models face major challenges while performing short text categorization, due to the unique characteristics of short texts, i.e., insufficient text length, sparsity, ambiguity, etc. In other words, the conventional approaches provide substandard performance, when they are directly applied to the short text categorization task. Furthermore, in the case of short text, the standard feature extraction techniques such as bag-of-words suffer from limited contextual information. Hence, it is essential to enhance the text representations with an external knowledge source. Moreover, the traditional models require a significant amount of manually labeled data and obtaining labeled data is a costly and time-consuming task. Therefore, although recently proposed supervised methods, especially, deep neural network approaches have demonstrated notable performance, the requirement of the labeled data remains the main bottleneck of these approaches.

In this thesis, we investigate the main research question of how to perform *short text categorization* effectively *without requiring any labeled data* using knowledge bases as an external source. In this regard, novel short text categorization models, namely, Knowledge-Based Short Text Categorization (KBSTC) and Weakly Supervised Short Text Categorization using World Knowledge (WESSTEC) have been introduced and evaluated in this thesis. The models do not require any hand-labeled data to perform short text categorization, instead, they leverage the semantic similarity between the short texts and the predefined categories. To quantify such semantic similarity, the low dimensional representation of entities and categories have been learned by exploiting a large knowledge base. To achieve that a novel entity and category embedding model has also been proposed in this thesis. The extensive experiments have been conducted to assess the performance of the proposed short text categorization models and the embedding model on several standard benchmark datasets.

Acknowledgements

First of all, I would like to express my deepest gratitude to my supervisor Prof. Harald Sack for giving me the opportunity to complete my Ph.D. under his supervision. I would like to thank him especially for his constant support, patience and encouragement. Despite his busy schedule, he always made time to address my questions. In addition, I would also like to thank Prof. Heiko Paulheim and Prof. York Sure-Vetter for accepting the request to be the reviewer of my thesis.

I would like to thank Dr. Lei Zhang whom I had the opportunity to work with closely. He supported me throughout the journey of my Ph.D., especially in the most stressful and difficult times and for that I am forever grateful.

I was very lucky to be part of an amazing FIZ ISE research team with whom I enjoyed working very much. I would like to thank all my team mates for providing a friendly atmosphere. We have made great memories, which will last forever.

Last but not least, I would like to express my deepest appreciation to my parents, who have always believed in me. Thank you for always being there for me whenever I needed your support despite the long physical distance between us. I dedicate this Ph.D. thesis to my mother, Sevda and my father, Fuad.

Contents

List of Figures

List of Tables

1 Introduction

The World Wide Web (the Web or WWW) is arguably the largest global repository of documents as well as other web sources (e.g., files, images, etc.) with over 1.7 billion websites[1]. The documents, i.e., Web pages and the other Web sources, which constitute a huge global network environment are interlinked by hyperlinks (Jacobs & Walsh, 2004) and identified by Uniform Resource Locators (URLs) (Berners-Lee et al., 1998). This global collection of information covers almost every possible topic of human interest. Today, Google processes roughly over 40,000 search queries every second, more than 3.5 billion searches per day and around 1.2 trillion searches per year worldwide[2]. This fact indicates the usefulness and significance of the Web documents. Also, billions of users[1] access and even contribute to this enormous and global information exchange platform. Therefore, the content of the Web is drastically multiplying every day (Chen et al., 2011).

Moreover, the rapid growth of online platforms such as Really Simple Syndication (RSS) feeds, forums, etc. has given rise to the multiplication of online available short text data. Besides, individuals often share their opinion in the form of a short text over different platforms, such as microblog posts, product reviews and social media texts (Zeng et al., 2018). Hence, it has become crucial to process and extract meaningful information from the available online short text data. However, manually analyzing such a vast amount of data is beyond human capabilities. As a result, the fields of Natural Language Processing (NLP) (Jurafsky & Martin, 2009), which aim to analyze and process natural language documents automatically have evolved rapidly. In other words, NLP concerns to extract meaningful information from a large number of natural language documents to satisfy users' information needs. Some examples of NLP applications are text categorization (Türker et al., 2019), spelling and grammar checking (Zukarnain et al., 2019), auto completion (Tahery & Farzi, 2020), text summarization (Liu & Lapata, 2019), question answering (Lukovnikov et al., 2019), etc. Among these applications, text categorization is the fundamental one, which has been proven to be useful in various applications, such as sentiment analysis, news feed filtering and categorization, spam filtering, etc. (Zeng et al., 2018).

Text categorization is one of the crucial tasks of NLP, which aims to assign one or more predefined categories (e.g., Sports, Economy, Business, etc.) to text documents based on the attributes of the texts. In laymen's terms, the text categorization task concerns labeling text documents according to their content information. The textual data can be anything ranging

[1] https://www.internetlivestats.com/total-number-of-websites/
[2] https://www.internetlivestats.com/google-search-statistics/

from phrases, sentences, paragraphs, or entire documents. Hence, depending on the characteristics of textual data the employed text categorization methods vary. There exist several real-world examples of text categorization applications such as online news platforms, which aim to first correctly categorize news contents in order to provide much more useful news to users according to their personal preference. Another example is from a query-specific document ranking system which concerns finding the most relevant set of documents for a given user query (Ahmad et al., 2019). Therefore, it is also essential to categorize the content of the documents for such systems.

While many text categorization methods (e.g., machine learning-based models) have been proposed and demonstrated notable success in the categorization of long and well-structured documents (e.g., news articles), in the case of short texts their performance significantly falls down (Zeng et al., 2018). The reason here can be attributed to the main characteristic of short texts, i.e., the limited length of the text which is no longer than 200 characters (Song et al., 2014). Furthermore, due to the limited contextual information, which is only a few words (Song et al., 2014), short texts are usually rather ambiguous and sparse. Moreover, unlike other documents, such ambiguity cannot be resolved by relying on the contextual information. Besides, since the short texts are sparse, the feature extraction is rather difficult. Overall, the main attributes of short texts pose several major challenges to conventional text categorization methods.

Due to the general advancement of computational power, recently, several sophisticated deep learning approaches have been proposed for short text categorization (Kim, 2014; Zhang et al., 2015). To alleviate the data sparsity problem, most of these approaches utilize advanced text representation techniques. To achieve that they exploit external knowledge (e.g., a knowledge base) to enrich the text representations. However, although such approaches have demonstrated remarkable performance in this task, they require a significant amount of manually labeled data. Obtaining labeled data is a costly and time-consuming task. Therefore, the requirement of large amounts of labeled data remains the main bottleneck for deep neural network-based approaches (Meng et al., 2018). Especially, if the data to be labeled is of a specific domain (e.g., chemistry, biology, etc.) then only the domain experts can label them. Such labeling of text documents is also a very labor-intensive task because each domain expert has to read each document and label them manually according to their domain knowledge.

Motivated by the stated challenges, this thesis concerns how to perform *short text categorization* effectively *without the requirement of any labeled data*, instead, utilizing world knowledge, such as a knowledge base as an external source.

1.1 Challenges and Tasks

Due to the drastic multiplication of Web content, short text categorization, which is the main focus of this thesis has gained significant attention across academia and industry. However,

two main challenges are faced while performing short text categorization. These challenges and the tasks, which aim to address the challenges are introduced as follows:

- **Challenge 1: Requirement of labeled training data.**

  Over the last few years, with the advent of computational power, researchers have focused on developing deep neural network based methods for the text categorization problem. Further, the proposed approaches have demonstrated extremely superior performance in this task. Despite the success of these models they can easily consume million-scale labeled data (Meng et al., 2018). However, it is hard to obtain training data due to the high cost of human labeling effort (Lu et al., 2014). Therefore, most of the supervised categorization models suffer from the absence of labeled data. This thesis aims to address the labeled data requirement by exploiting a knowledge base as an external source while performing short text categorization. To achieve that the main task is introduced as follows:

  Task 1: Utilizing a knowledge base as an external source.

  Knowledge bases, such as Wikipedia, which contains millions of interlinked entities that are associated with hierarchically related categories are an excellent data source for many natural language processing tasks (Chang et al., 2008; Li et al., 2016c). One of the main concerns of this thesis is leveraging a knowledge base as an external source to overcome the labeled data requirement while performing short text categorization. More specifically, the semantic similarity between the entities present in a short text and predefined categories provides crucial information to derive the category of the text, and such semantic relation can be quantified with the help of an interlinked structure of a knowledge base.

  Besides the requirement of labeled data, another challenge we face while performing short text categorization is stated below:

- **Challenge 2: Limited context and non-standard characteristics.**

  While conventional text categorization methods have demonstrated notable success, they provide substandard performance in the case of short texts. The reason is attributed to the main characteristics of short texts. Unlike structured and long texts (e.g., news documents), short texts do not follow the standard syntax of natural language documents (Wang et al., 2017). Moreover, due to lack of context, short texts are usually ambiguous. Such major characteristics of short texts pose great challenges to the conventional text categorization methods. To overcome these challenges, the

focus of the thesis is on enriching short text representations by exploiting knowledge bases. In order to achieve that the following task is introduced:

Task 2: Enriching text representations by utilizing a knowledge base.

The traditional text categorization models represent texts as a bag-of-words to perform text categorization. Most of these methods ignore the entities and utilize only words present in texts. In the case of short texts, where the context is rather limited and the ambiguity is one of the major problems, such approaches that utilize only words often lead to inaccurate results. Thus, it is indispensable to leverage external sources to obtain more advanced text representations (Wang et al., 2017). In this thesis, in order to enrich the text representations, we focus on exploiting entities present in short texts and their associated categories from the underlying knowledge base.

1.2 Research Questions

The main research question of this thesis is defined as follows:

How to perform short text categorization effectively without requiring any hand-labeled data?

This broad research question is broken down into three specific research questions, each of which entails a combination of aforementioned challenges and tasks. Each of the specific research question will be addressed in the remainder of this thesis.

The following first two research questions are derived from Challenge 1 *"Requirement of labeled training data"* and concerns Task 1 *"Utilizing knowledge bases as an external source"*:

Research Question 1. *How can a knowledge base be utilized to categorize short texts without requiring any labeled training data?*

The entities present in short texts carry vital information about the content of the texts. In fact, most of the time, in the case of short text, entities provide much more information than the words (Wang et al., 2016a). On the other hand, knowledge bases describe real-world entities and their relations, which form a graph structure (Paulheim, 2017). Therefore, linking entities present in short texts to a knowledge base such as Wikipedia as well as mapping predefined categories to Wikipedia categories can help to build a bridge between unstructured data and structured data. Then the interlinked structure of Wikipedia can help to quantify the semantic similarity between the entities within short texts and predefined categories. Such semantic similarity can be leveraged to find the most relevant category for a given short text

without requiring any labeled data. This idea will be investigated in Chapter 4, which is based on the following publication:

Türker, R., Zhang, L., Koutraki, M., & Sack, H. (2019). Knowledge-based short text categorization using entity and category embedding. The Semantic Web-16th International Conference, ESWC 2019, Portoro, Slovenia.

Research Question 2. *How to learn the semantic representation of short texts and predefined categories with the help of a knowledge base?*

The semantic representation of short texts and categories is essential to measure the meaningful similarity between them. Several word embedding models have been proposed to learn the semantic representation of texts such as Skip-gram (Mikolov et al., 2013a). However, such methods provide better text representations, when dealing with longer text, in which case, even if a word is ambiguous, such ambiguity will be handled based on the available context. On the other hand, in the case of short text, where the available context is rather limited and each word obtains significant importance, such approaches often lead to inaccurate representations. Therefore, to be able to calculate semantic similarity between the short texts and the predefined categories we focus on learning the low dimensional representation of entities and categories from a knowledge base by exploiting its interlinked structure. This idea will be investigated in Chapter 4, which is based on the following publication:

Türker, R., Zhang, L., Koutraki, M., & Sack, H. (2019). Knowledge-based short text categorization using entity and category embedding. The Semantic Web-16th International Conference, ESWC 2019, Portoro, Slovenia.

The next research question is derived from Challenge 1 *"Requirement of labeled training data"* and Challenge 2 *"Limited context and non-standard characteristics"* and concerns Task 1 *"Utilizing a knowledge base as an external source"* and Task 2 *"Enriching text representations by utilizing a knowledge base"*:

Research Question 3. *How to combine a knowledge base with a deep neural network to perform short text categorization without requiring any hand-labeled data?*

Supervised categorization methods, especially, recently proposed deep learning approaches have demonstrated superiority for the short text categorization task. There are two main advantages of these methods, first, they provide better categorization performance than the conventional models, second, they significantly reduce the feature engineering efforts (Meng

et al., 2018). However, despite their attractiveness, they require a large amount of labeled training dataset. On the other hand, knowledge graphs are a great data source to generate labeled data, which later can be utilized by the neural networks to perform short text categorization without requiring any manual effort. This idea will be investigated in Chapter 5, which is based on the following publication:

Türker, R., Zhang, L., Alam, M., & Sack, H. (2020). Weakly Supervised Short Text Categorization Using World Knowledge. The 19th International Semantic Web Conference, ISWC 2020.

1.3 Contributions of the Thesis

This section presents the most important contributions of the thesis. Each contribution is the result of the specific research question (cf. Section 1.2) that has been investigated in this thesis. The following contributions are introduced briefly and the details will be discussed in the remainder of the thesis.

Contribution 1. *A new paradigm for short text categorization, based on a knowledge base*

Based on our publication (Türker et al., 2019), a method called Knowledge-Based Short Text Categorization (KBSTC) is presented in Chapter 4. The method does not require any labeled data to perform short text categorization, instead, it utilizes a knowledge base as an external source. More precisely, KBSTC mainly relies on the semantic similarity between the entities present in texts and predefined categories to perform short text categorization. In order to find such semantic similarity, KBSTC exploits a knowledge base link structure. The experiments that have been performed on different datasets show that KBSTC significantly outperforms the categorization approaches which do not require any labeled data, while it comes close to the results of the supervised approaches.

Contribution 2. *A new embedding model to learn the low dimensional representation of entities and categories from a knowledge base*

The proper semantic representation of entities and categories in a common vector space is necessary to be able to calculate the semantic similarity between them. To do so, based on our publication (Türker et al., 2019), a new entity and category embedding model is proposed in Chapter 4. The model first constructs two types of networks, namely, entity-entity and entity-category networks by utilizing the Wikipedia hyperlink structure. The entity-entity network captures the relations between the entities. On the other hand, the entity-category network reflects the relations between the entities and categories. Then the embedding model

utilizes these two networks to generate the low dimensional distribution of the entities and categories. To assess the quality of the proposed entity and category embedding model, its performance has been compared with several different embedding models. The experimental results suggest that the proposed embedding model can capture better semantic relations between the entities and categories.

Contribution 3. *A weakly supervised deep neural short text categorization model*

Based on our publication (Türker et al., 2020), a model called Weakly Supervised Short Text Categorization using World Knowledge (WESSTEC) is presented in Chapter 5. WESSTEC consists of two main modules: (1) Labeled Data Generation module, which is responsible for labeling short text documents by leveraging an external knowledge base and without requiring any manual effort, (2) a deep neural network-based categorization model, which is designed to utilize the generated labeled data by the first module for the training phase. WESSTEC exploits the first module to label short text documents and then extracts different feature sets by utilizing the words and concepts from the labeled documents to train the deep neural network. Finally, the trained model is used to categorize new short text documents. The performance of the model has been evaluated on multiple datasets. The experimental results show that WESSTEC outperforms unsupervised state-of-the-art categorization approaches while it achieves comparable performance to supervised approaches.

1.4 Publications

This section provides our publications as well as a master thesis which was supervised by the author of this thesis.

- **Conference and Workshop Papers**
  - Türker, R., Zhang, L., Alam, M., & Sack, H. (2020). Weakly supervised short text categorization using world knowledge. The 19th International Semantic Web Conference, ISWC 2020.
  - Türker, R., Zhang, L., Koutraki, M., & Sack, H. (2019). Knowledge-based short text categorization using entity and category embedding. The Semantic Web - 16th International Conference, ESWC 2019.
  - Biswas, R., Türker, R., Moghaddam, F. B., Koutraki, M., & Sack, H. (2018). Wikipedia infobox type prediction using embeddings. The 1st Workshop on Deep Learning for Knowledge Graphs and Semantic Technologies (DL4KGS) co-located with the 15th Extended Semantic Web Conference, ESWC 2018.
- **Poster and Demo Papers**

  - Türker, R., Zhang, L., Koutraki, M., & Sack, H. (2018b). "the less is more" for text classification. The 14th International Conference on Semantic Systems, SEMANTiCS 2018.
  - Türker, R., Zhang, L., Koutraki, M., & Sack, H. (2018a). TECNE: Knowledge-based text classification using network embeddings. The 21st International Conference on Knowledge Engineering and Knowledge Management, EKAW 2018.
  - Aras, H., Türker, R., Geiss, D., Milbradt, M., & Sack, H. (2018). Get your hands dirty: Evaluating word2vec models for patent data. The 14th International Conference on Semantic Systems, SEMANTiCS 2018.
  - Türker, R., Koutraki, M., Waitelonis, J., & Sack, H. (2017). Entity suggestion ranking via context hashing. The 16th International Semantic Web Conference, ISWC 2017.
- **Supervised Master Thesis**
  - Alam M., Bie Q., Türker R. & Sack, H. (2020). Entity-based short text classification using convolutional neural networks. International Conference on Knowledge Engineering and Knowledge Management, EKAW 2020.

1.5 Guide to the Reader

This thesis comprises six main chapters. Chapter 1, Chapter 2 and Chapter 3 provide the technical preliminaries and foundations of this thesis. On the other hand, Chapter 4 and Chapter 5 are the core chapters of the thesis, they cover all the research questions introduced in Section 1.2 along with the proposed solutions and the contributions. Finally, Chapter 6 concludes the thesis with a discussion of open issues and possible future directions.

- **Chapter 1** gives the motivation of the work, and introduces the main research question of the thesis which is broken down into three specific research questions. It further summarizes the main contributions and structure of the thesis.

- **Chapter 2** provides the foundations of the thesis. The chapter contains a brief introduction to neural networks, knowledge graphs and semantic measures (e.g., semantic similarity, semantic relatedness). This chapter does not discuss the aforementioned subjects extensively, instead, it aims to provide the necessary background information in the context of this thesis.

- **Chapter 3** introduces the general task of text categorization. Further, it discusses the short text categorization task, which is the main focus of this thesis. Moreover, the chapter also provides state-of-the-art text categorization models and the most com-

monly employed evaluation metrics for text categorization.

- **Chapter 4** proposes a new approach (KBSTC) for the short text categorization problem. The model aims to overcome the labeled data requirement while performing short text categorization by utilizing a knowledge base as an external source. Further, it leverages the semantic similarity between the entities present in a short text and predefined categories to derive the most relevant category for the text. To find the semantic similarity between the entities and categories, this chapter also introduces a new entity and category embedding model.

- **Chapter 5** presents a weakly supervised short text categorization approach (WESSTEC) which does not require any hand-labeled data for the categorization task. The model consists of two main modules, namely, labeled data generation and deep learning-based categorization modules. The method first, labels the unlabeled short text documents with the help of the first module, which is based on a heuristic function, and then the heuristically labeled documents are utilized by the deep neural network model for the categorization task.

- **Chapter 6** concludes the thesis with a summary of the research questions, main contributions, and an outlook on future research directions.

2 Foundations

This chapter gives an overview of preliminaries and methods that are leveraged to build this thesis. The chapter consists of three main sections, namely, neural networks, knowledge graphs and semantic measures. Section 2.1 introduces the standard and well known neural network architectures such as feedforward, convolutional, recurrent neural networks. Furthermore, text embedding models such as Skip-gram, Doc2Vec, etc. and network embedding models such as LINE, DeepWalk, etc. are also presented in this chapter. Section 2.2 presents a general overview of knowledge graphs, the notable knowledge graphs such as DBpedia, Wikidata, as well as the methods that have been utilized to create the knowledge graphs. Finally, this section is concluded by briefly introducing linked open data and demonstrating its evolution across the years. Section 2.3 is the last section of this chapter, the semantic measures (e.g., semantic relatedness and semantic similarity) are presented in this section. Especially, the semantic similarity has been extensively utilized to build this thesis. The section briefly presents different methods which leverage knowledge graphs, large text corpora, etc., to calculate the semantic similarity.

2.1 Neural Networks

This section first provides a brief history of artificial neural networks. Next, the basic architectures of neural networks (e.g., single layer, multi-layer neural networks, etc.) are discussed. Moreover, the most prominent text embedding and network embedding models and their architectures are presented as well.

2.1.1 Beginnings of Artificial Neural Networks

Alan Turing laid the first foundations of artificial intelligence in the 1950s by publishing a paper titled *"Computing machinery and intelligence"*. In this paper, he introduced the well known *Turing Test* which is designed to test a computer's ability to determine whether it can be regarded as intelligent or not. In the Turing test, a human evaluator, i.e., *referee* tries to have natural language conversations with a person and a computer which is designed to respond like a human. If the referee cannot distinguish between the computer and the person then the machine would pass the Turing test and then the machine could be considered as intelligent. Although this idea was proposed decades ago, today Turing test is still used widely

as a benchmark in artificial intelligence.

The second most important event of artificial intelligence was a *Dartmouth Summer Research Project on Artificial Intelligence* Workshop. The workshop lasted six to eight weeks with brainstorming sessions. In one of the sessions, McCarthy proposed the phrase "artificial intelligence" in 1955. Several notable people in the field of artificial intelligence (e.g., John McCarthy, Marvin Minsky, Julian Bigelow, Donald MacKay, and more besides) participated in the workshop.

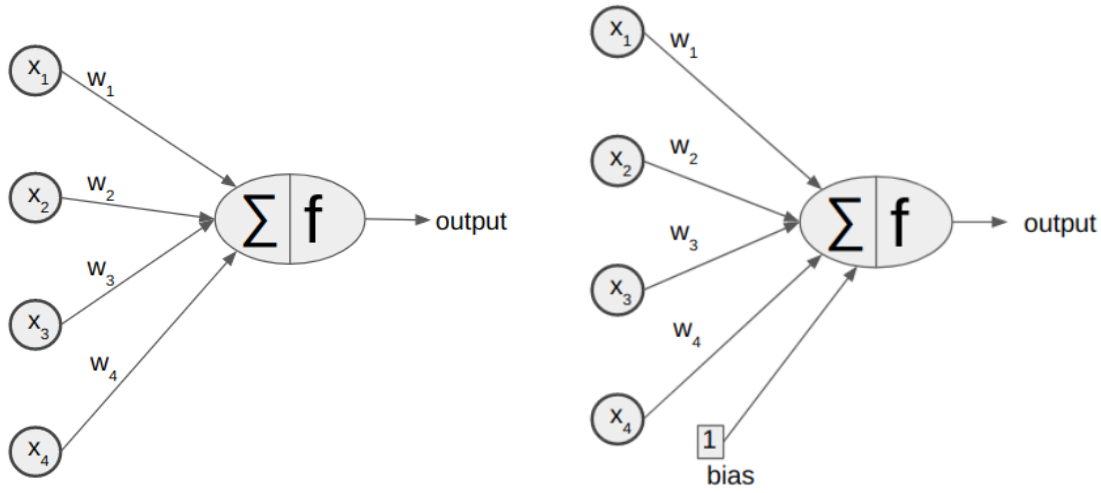
On the other hand, Walter Pitts and Warren McCulloch started the history of neural networks by publishing a paper titled *"A Logical Calculus of Ideas Immanent in Nervous Activity"*[1]. The paper describes the idea of the artificial neural networks and provides the relevant definitions on which we still rely today (Skansi, 2018). In their paper, the neurons were splited into two groups, the first group is called *peripheral afferents*, i.e., input neurons and the second group is the output neurons. Back then the concept of *hidden layer* was not introduced.

Another notable person in the field of neural networks is Frank Rosenblatt, an American psychologist. Rosenblatt discovered the famous *perceptron learning rule* which is still today one of the most widely used learning algorithms (Skansi, 2018). The rule describes how to update the parameters (weights) of a neural network during the training phase. Besides his discovery of perceptron, Rosenblatt also explored several neural network architectures in his book *Principles of Neurodynamics* (Rosenblatt, 1961) and proposed an idea of multilayered networks which are similar to today's conventional convolutional neural networks. His concept of multilayered networks is still considered as a start point for the development of the *deep neural networks*.

The book, written by Marvin Minsky and Seymour Papert in 1969, caused a considerable setback despite the developments in artificial neural networks. In the book, Minsky and Papert tried to prove that perceptrons are simple linear classifiers and have major computational limitations such as XOR function. Further, the book discouraged many people for the further developments of artificial neural networks. However, after that, a positive development, i.e., the discovery of Backpropagation (backward propagation of errors) algorithm, occurred. Although the discovery of Backpropagation by Rumelhart, Hinton, and Williams enabled the community to train a neural network with hidden layers, this discovery was neglected by the community at that time.

In the early 1990s, Support Vector Machines (SVMs) gained significant attention due to their performance and simplicity. People from the artificial intelligence community shifted their focus on SVMs. In the late 1990s, two important improvements occurred which laid the foundations of today's deep neural networks: (1) the invention of the long short-term memory by Hochreiter and Schmidhuber in 1997; (2) the design of the first convolutional neural network, which is called LeNet-5 in 1998 by LeCun, Bottou, Bengio, and Haffner.

[1] `http://www.cs.cmu.edu/~epxing/Class/10715/reading/McCulloch.and.Pitts.pdf`



(a) Perceptron without bias

(b) Perceptron with bias

Figure 2.1: An overview of perceptron architectures.

After that, in 2006, Hinton, Osindero, and Teh published a paper that introduced deep belief networks (DMB). After this paper, a new period of artificial intelligence has begun.

2.1.2 Basic Architecture of Neural Networks

In this section, the underlying architectures of single-layer and multi-layer neural networks are discussed. Single-layer neural networks are also referred to as *perceptrons*, which have a set of inputs and an output layer. The inputs are directly mapped to the output layer, which consists only of a single node. On the other hand, multi-layer neural networks do not have only input and output layers but also hidden layers. These types of neural networks are known as *feed-forward neural networks*.

2.1.2.1 Single-layer Neural Networks: Perceptrons

Perceptrons have the simplest architecture of neural networks. It consists of only an input and output layer. The output layer contains only a single node, which produces the final output value. Figure 2.1 illustrates perceptron architectures with and without a bias neuron. The bias value improves the performance of the network, in other words, it enables the network to make more accurate predictions. Moreover, to incorporate bias into the neural network, an additional input neuron called *bias neuron* which transmits the value 1 to the next layer is added.

The given input features are $x_1, x_2, ..., x_4$ (cf. Figure 2.1) and the input layer simply

transmits each individual feature to the output layer by multiplying them with their corresponding weight. For example, x_1 is multiplied with w_1, x_2 is multiplied with w_2 and so on. In an arbitrary neural network architecture, the input layer does not perform any computation, and thus often, it is not included in the count of the number of layers.

Given a training set, where each instance is of a form ($\overline{X}$, y), and $\overline{X} = [x_1, x_2, ..., x_d]$ denotes an input variable with d-dimensional features and y is the actual label of $\overline{X}$ such that $y \in \{-1, +1\}$. Let $\overline{W} = [w_1, w_2, ..., w_d]$ denote the weight of the edges and b is a bias. Then, the output $\hat{y}$ is computed as follows (Aggarwal, 2018b):

$$\hat{y} = sign\{\overline{W} \cdot \overline{X} + b\} = sign\{\sum_{j=1}^{d} w_j x_j + b\} \tag{2.1}$$

The sign function outputs either +1 or -1 as $y \in \{-1, +1\}$. In other words, the role of the sign function here is an *activation function*. Given a set of inputs to a node, the activation function determines the node's output. Depending on the application at hand, different activation functions such as *sigmoid, ReLU, and softmax* can be utilized (cf. Section 2.1.3). The formalized scenario (cf. Equation 2.1) is appropriate for a binary categorization task, where the possible output of the sign function, i.e., -1 or +1 corresponds to a category label (e.g., spam or not spam).

By the time Rosenblatt proposed the perceptron algorithm, the optimization process of a network was performed heuristically. Therefore, there was not any formal definition of a loss function for perceptrons like we have today for almost all the standard machine learning algorithms. However, the heuristic function's goal was the same as today's loss functions, minimizing the number of miss categorized samples, i.e., minimizing the error rate. However, today several resources related to perceptrons formalize this heuristic learning process. Following the heuristically motivated loss function in the least-squares form with respect to all training instances is given (Aggarwal, 2018b):

$$Minimize_{\overline{W}} L = \sum_{(\overline{X},y) \in D} (y - \hat{y})^2 = \sum_{(\overline{X},y) \in D} (y - sign\{\overline{W} \cdot \overline{X}\})^2. \tag{2.2}$$

The above equation aims to find the optimal weight values $\overline{W}$, which minimizes the Equation 2.2.

Typically, this type of a neural network is trained by feeding each input data instance $\overline{X}$ one by one to create the corresponding prediction $\hat{y}$. Then the weights are updated in each iteration based on the error value $E(\overline{X}) = (y - \hat{y})$ as follows (Aggarwal, 2018b):

$$\overline{W} \Leftarrow \overline{W} \alpha (y - \hat{y}) \overline{X} \tag{2.3}$$

where α denotes the learning rate of the neural network. Finally, the weights are updated iteratively until the convergence is reached.

The introduced perceptron model is a type of a linear classifier which defines a linear hyperplane between the data points. Ideally, the data points belong to the same category fall on one side of the hyperplane and the data points belong to the other category fall on the other side of the hyperplane. Therefore, the perceptron model performs well when the data is linearly separable.

2.1.3 Activation Functions

There exist several activation functions that are designed to be useful for learning a neural network. Often, hidden layers and an output layer utilize different activation functions. The choice of the activation function for a neural network is critical for its performance. For example, the *Rectified Linear Unit* (ReLU) function has become very popular in the last years, due to its effectiveness and efficiency. Most of the standard neural network architectures utilize ReLU for the hidden layers. On the other hand, the choice of the activation function for output layers is determined based on the task at hand. For example, if the task requires a prediction of a probability of a binary class (i.e., 0 or 1), then the *sigmoid* function should be applied to the output node. Another example is, if the neural network is designed for the categorization task (e.g., text categorization), then the required output is a probability distribution over the predicted classes. To obtain such an output, the *softmax* function can be applied to the output layer.

The most widely leveraged activation functions are given as follows:

- **Sigmoid Function:** The sigmoid function maps the output of a node in the range $(0, 1)$, which is a probability value. A well-known example of a sigmoid function is the logistic function. The following formula defines the sigmoid function:

$$\Phi(v) = \frac{1}{1 + e^{-v}} \tag{2.4}$$

  It should be noted that the sigmoid function is mostly utilized to predict the probability of a binary class.

- **Hyperbolic Tangent (Tanh) Function:** The tanh function has a similar graphical shape to the sigmoid function. However, tanh function outputs the value in $[-1, 1]$. The relation between the sigmoid and the tanh is defined as follows:

$$tanh(v) = 2 \cdot sigmoid(2v) - 1 \tag{2.5}$$

  Then the tanh function is defined as follows:

$$\Phi(v) = \frac{e^{2v} - 1}{e^{2v} + 1} \tag{2.6}$$

- **Sign Function:** The sign function or signum function maps the output to +1, 0 or -1 and defined as follows:

$$\Phi(v) = sign(v) = \begin{cases} 1, & \text{if v>0.} \\ 0, & \text{if v=0.} \\ -1, & \text{if v<0.} \end{cases} \tag{2.7}$$

- **Rectified Linear Unit (ReLU) Function:** As stated before, today, the ReLU function is one of the most commonly utilized activation functions. Due to its simplicity, it has replaced sigmoid and tanh in the standard neural network architectures for the hidden layers (Aggarwal, 2018b). The ReLU can be formalized as follows:

$$\Phi(v) = max\{v, 0\} \tag{2.8}$$

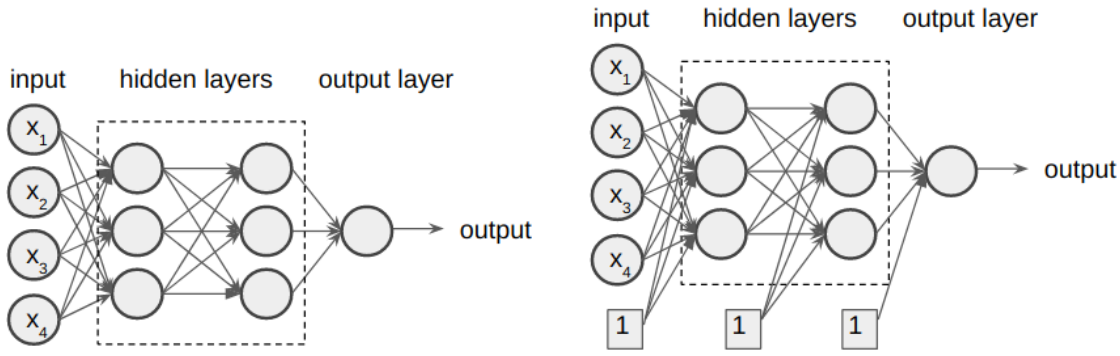
- **Softmax Function:** Often, the softmax function is leveraged in the output layer of a neural network which is designed for a multi-class categorization task. The output layer of such a network consists of multiple output nodes and each of them corresponds to a class (or category). The categorization is performed based on the estimated probability of each class, i.e., the class which provides the highest probability is selected as a relevant class for a given input. Therefore, it is important to convert the output of the last hidden layer into probabilistic values. To do so, the softmax function is utilized. Given a categorization task with k predefined classes, then the output layer should contain k nodes and each output of the node corresponds to a probability of a certain class. The softmax function converts the given vector $v = [v_1, ..., v_k]$ of k real numbers into probability distribution as follows:

$$\Phi(v)_i = \frac{exp(v_i)}{\sum_{j=1}^{k} exp(v_j)} \quad \forall i \in 1, ..., k \tag{2.9}$$

  The softmax function is applied to each element of the vector v. Further, each obtained probability corresponds to the class probability.

2.1.3.1 Feedforward Neural Networks

In the previous section, a single layer neural network, i.e., perceptron, which does not have any hidden layers, is presented. Feedforward neural networks are similar to perceptrons except that they contain additional intermediate layers so-called *hidden layers* between input and output layers (Aggarwal, 2018b). The simple architectures of feedforward neural networks are shown in Figure 2.2. As the name indicates, in feedforward networks, the computations are performed in the forward direction from input to the output.



(a) No bias neurons

(b) With bias neurons

Figure 2.2: An overview of simple feedforward neural network architectures.

To facilitate the discussion, the difference between shallow neural networks and deep neural networks is explained. Shallow neural networks contain input and output layers, and only one hidden layer. On the other hand, deep neural networks contain input and output layers and multiple hidden layers. Hence, the given examples in Figure 2.2 are considered as deep feedforward neural networks.

The default architecture of feedforward neural networks (e.g., Figure 2.2) contain a series of fully connected layers, in other words, each node in one layer is connected to nodes of the subsequent layer. For example, Figure 2.2b is a *3-layer* (with 2 hidden and an output layer) neural network. The feedforward neural networks have a standard architecture. Hence, once the number of layers, nodes in each layer, and the loss function are defined, the rest of the network's architecture is straightforward.

In feedforward neural networks, each connection, i.e., *edge* between a neuron in a layer and another neuron in the subsequent layer has a weight. Each neuron except the ones in the input layer gets as an input of a sum of the multiplications of the inputs from the previous layer and their respective weights. Further, often a bias value is also added to this sum. Then, each neuron applies an activation function to produce an output. Note that the number of units in each layer is referred to as the dimensionality of that layer (Aggarwal, 2018b).

Assume the input is d-dimensional vector $\overline{X}$, where $\overline{X} = [x_1, x_2, ..., x_d]$ and let p_1 denote the number of units in the first hidden layer. W_1 is a matrix such that $W_1 \in \mathbb{R}^{d \times p_1}$ and it contains the weights of the edges between the input layer and the first hidden layer. Similarly, the dimension of the weight matrices in the hidden layers determined by the number of neurons contained in the respective layers. Assume W_r is a weight matrix between r^{th} hidden layer and the $(r+1)^{th}$ hidden layer, then $W_r \in \mathbb{R}^{p_r \times p_{r+1}}$. Finally, assume the output layer contains o nodes, then the final weight matrix is W_{k+1} and $W_{k+1} \in \mathbb{R}^{p_k \times o}$.

Given an input $\overline{X}$ in order to produce the output $\bar{o}$ the following recursive equations

are used as follows (Aggarwal, 2018b):

$$\begin{aligned}
&\bar{h}_1 = \Phi(W_1^T \overline{X}) \\
&\bar{h}_{p+1} = \Phi(W_{p+1}^T \overline{h}_p), \forall p \in \{1, ..., k-1\} \\
&\bar{o} = \Phi(W_{k+1}^T \overline{h}_k)
\end{aligned}$$

where $\bar{h}_1$ is the column vector of an output of the first hidden layer, similarly, $\overline{h}_{p+1}$ is a column vector of an output of $(i+1)^{th}$ layer, Φ is an activation function such as sigmoid, ReLU, etc.

It is often the case that different layers of a network use different activation functions. For example, a simple feedforward neural network, which is designed for a binary categorization task, often uses ReLU in the hidden layers and the sigmoid for the output layer. It should be noted that all units in a particular layer use the same activation function. Further, depending on the application at hand (e.g., categorization or dimensionality reduction), it is possible to easily vary the standard neural network architecture to allow multiple outputs.

The given equations above are the general formulation of the forward operation, which aims to transform the given input feature vectors into the outputs. In other words, similar to the perceptron algorithm, input data are fed into the network one by one or in small batches in order to produce outputs. In perceptron, the training process is straightforward because the optimization is achieved by minimizing the heuristically motivated simple loss function. However, in a multilayer neural network, the loss function is much more sophisticated. The loss function is a combination of all the weights in each layer. The weight values of the multilayer neural network are updated according to the error gradients. In order to compute the gradient of the loss function *backpropagation* algorithm has been widely used. Such training process mainly relies on the update of weights and biases during the training phase with backpropagation. In the following the backpropagation algorithm is briefly explained and more details can be found in (Skansi, 2018).

First, an error function $E(x)$ or cost function which measures the performance of the network is defined as follows (Skansi, 2018):

$$E = \frac{1}{2} \sum_{n \in D} (y^{(n)} - \hat{y}^{(n)})^2 \tag{2.10}$$

where n denotes the training sample, y is the target for the training instance n and the $\hat{y}$ is the prediction, i.e., the output of the model. The error function sums error across all the training samples, then the weights are updated accordingly.

Based on the backpropagation algorithm the derivative of the error function E is taken with respect to w_i (Skansi, 2018):

$$\frac{\partial E}{\partial w_i} = \frac{1}{2} \sum_n \frac{\partial y^{(n)}}{\partial w_i} \frac{dE^{(n)}}{y^{(n)}} \tag{2.11}$$

The updates of weights are proportional to the error derivations and they are added together in all training samples (Skansi, 2018):

$$\triangle w_i = -\eta \frac{\partial E}{\partial w_i} = \sum_n \eta x_i^{(n)} (y^{(n)} - \hat{y}^{(n)}) \tag{2.12}$$

The details of the derivatives are shown in (Skansi, 2018). Finally, the weights are updated according to the formula below (Skansi, 2018):

$$w_{update} = w_{old} - \eta \nabla E \tag{2.13}$$

2.1.4 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are specialized neural networks which aim to process grid-structured data, e.g., image data (2D grid), time-series data (1D grid). The most distinctive feature of convolutional neural networks from other networks is the *convolutional* layer. Typically, a traditional convolutional network architecture contains at least one convolutional layer.

In 1998, the first CNN, i.e., *LeNet-5* was developed by LeCun, Bottou, Bengio and Haffner. This network was trained with MNIST data, which is a large dataset and contains binary images of handwritten digits. Thereafter, these networks have been utilized in a variety of applications. In 2011, convolutional neural networks demonstrated remarkable performance in an image-classification contest. After that they gained significant attention, especially, in the field of image processing. While they can be utilized with various types of data, the majority of the applications have been focused on image data. Some of the examples of applications are image classification, image and video recognition, medical image analysis, etc.

Although convolutional neural networks have been mostly utilized in the field of image processing, applying them to natural language processing tasks has also been explored by the researchers. (Kim, 2014) designed a relatively simple convolutional neural network architecture for a sentence categorization task and the model demonstrated strong empirical performance across several datasets. The proposed model is still one of the most widely used baselines for sentence categorization. After this paper (Kim, 2014), convolutional networks draw more attention from the natural language processing community. Several scientific papers have been published in a similar direction (Peng et al., 2018; Duque et al., 2019; Yao et al., 2019; Conneau et al., 2016).

The convolutional neural networks are similar to the traditional feedforward neural networks in the sense that the information moves in an only forward direction, i.e., from an input layer to hidden layers and finally to an output layer to produce an output. Figure 2.3 illustrates a simple example of a convolutional neural network architecture for a sentence categorization task. Unlike feedforward neural networks, which typically contain one type of a hidden layer, i.e., fully connected layer, convolutional neural networks consist of different

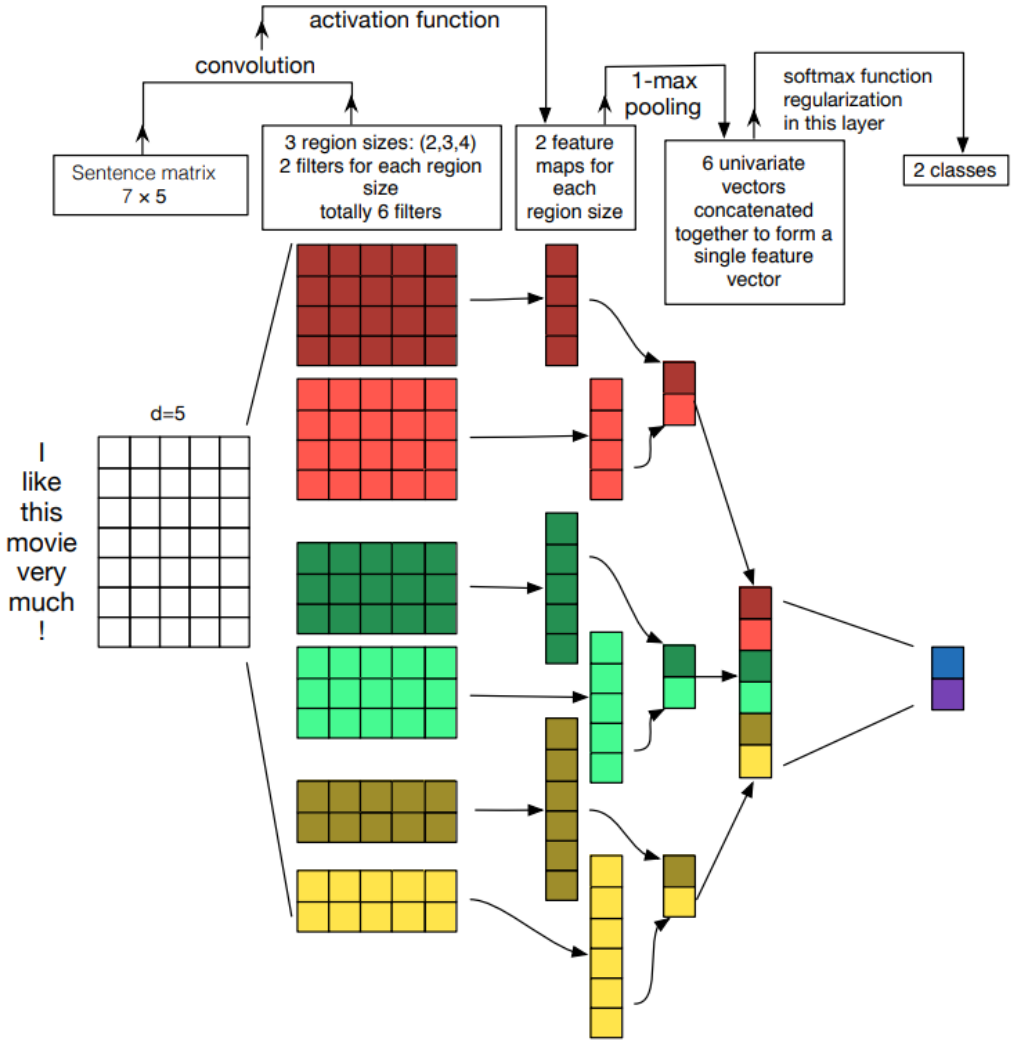


Figure 2.3: A CNN architecture for sentence categorization. The image is extracted from (Zhang & Wallace, 2017).

types of hidden layers. The most common three types of layers, which are contained in standard convolutional neural networks are *convolution*, *pooling*, and *ReLU* layers. ReLU is an activation layer, which is similarly applied as in the traditional neural networks. Besides the aforementioned layers, the final layers of convolutional networks are often fully connected. Further, the last hidden layer is mapped to the output layer to produce the final output of a network. The number of nodes in an output layer is determined in an application-specific way.

Before discussing the layers of convolutional neural networks, we first briefly examine the network's general architecture. The majority of the related literature has mainly focused on analyzing convolutional neural networks in an image categorization setting where the input is image data. However, in this thesis, the focus is short text categorization. Therefore, the model is discussed in the context of text categorization. Figure 2.3 illustrates a simple example of a convolutional neural network architecture which is designed for sentence categorization. Given a piece of text, e.g., a sentence, the goal of the model is to assign a label to it. As already stated, such type of a network accepts grid-like structured data as an input; therefore, each input sentence is converted into a sentence matrix. To this end, first, sentences are tokenized. The vector representation of the tokens is utilized to form the sentence matrix. The rows of the matrix are the token vectors which can be obtained from a pre-trained word embedding model (e.g., Google pre-trained word2vec model). Let d denote the word vectors' dimension and s is the number of tokens present in a given sentence. Then, the dimension of the sentence matrix is $s \times d$. The value of s, i.e., the number of tokens to be considered from an input sentence to form a sentence matrix, is often predefined based on the dataset. Because the dimension of the input matrix specifies the number of neurons in the input layer. Obviously, the length of the sentences in standard datasets vary, and thus the zero-*padding* strategy is used to equalize the dimensions of the input matrices. Given a sentence matrix, the next operation is the convolution, which is performed via *"filters"*, i.e., *"kernels"*. In Figure 2.3 there are 6 different filters with 3 different sizes. This operation aims to produce a feature map from a given input matrix and a kernel. Often, in a convolution layer, the convolution is performed multiple times independently, and each time operation utilizes a different kernel. The main idea is that the convolution operation places the filter to each possible position of the input and for each possible position a dot product between the filter and the matching grid of the input is performed. To be able to perform such a dot product the filter and the input should fully overlap. In other words, the filter should be aligned with the matrix in a way that there should not be any part of the filter which is sticking out from the borders of the matrix. Based on this the possible dot products between the filter and input specify the dimension of the next layer. Note that the size of the filters is determined based on the application at hand. For example, for a sentence categorization task given a sentence matrix where each row is a word vector, then it is reasonable to use filters with a width equals to the dimension of the word vectors. Whereas for image processing the most common filter sizes are 3×3 and 5×5.

After the convolution operation, the obtained feature maps have different dimensions. In order to induce the feature maps to a fixed-length vector and reduce their dimensionality, a max-pooling function is applied to each feature map independently. Thereafter, those vectors are concatenated to form a single dense feature vector. To produce the outputs, the softmax function is applied to this vector. It should be noted that it is possible to feed the dense feature vector through fully connected layers and then the output layer. The architectural design of the network can be easily altered based on the application at hand.

The convolution operation relies on three important ideas that improve the performance of the model (Goodfellow et al., 2016):

- **Sparse interactions.** The size of the filters or kernels is often much smaller than the layers of which the filters are applied. Therefore, the feature maps, which are the convolution operations outputs, have a much smaller size than the inputs. Consequently, fewer parameters need to be stored and processed in the next layers, and thus the efficiency is improved automatically. For example, an input of an image can have millions of pixels that needs be processed. However, with the help of kernels, meaningful features can be detected by utilizing only tens or hundreds of pixels (Goodfellow et al., 2016).

- **Parameter sharing.** As the name indicates, parameter sharing means to exploit the same parameters in multiple operations. For example, in a convolution operation, in order to produce a feature map, a kernel slides over the input of which the filter is applied. In other words, the kernel visits every possible location of the input. However, instead of learning a parameter for each location/pixel (in case of image data) of the input, only one set of parameters, i.e., the kernel's parameters are learned.

- **Equivariant representations.** The idea of parameter sharing enables a layer to be equivariance to translations (Goodfellow et al., 2016). The functions $f(x)$ and $g(x)$ are considered to be equivariant to each other if $g(f(x)) = f(g(x))$. For the sake of simplicity, we give the following example (Goodfellow et al., 2016). Let I be a function that gives the coordinates of a certain feature, e.g., the brightness of a given image. Let g be a function such that $I^{'} = g(I)$ and $I^{'}(x, y) = I(x - 1, y)$ which shifts every pixel of I to one unit right. Then, according to equivariant representations, the output of the following two operations will be the same: (1) first applying the translation function g to I then applying convolution, (2) first applying convolution $I^{'}$ and then applying g.

In the following section, convolutional graph neural networks which are a special type of convolutional neural network are briefly presented.

Convolutional Graph Neural Networks

Convolutional neural networks have demonstrated strong empirical performance on several tasks, such as image recognition, text categorization, etc. Such networks are designed for processing grid-structured data, such as image data. However, there exist many real-world

datasets which are represented in graph forms. For example, social, customer-product, citation networks (Gao et al., 2018). The adaption of convolutional networks to other structured data types, such as graph data is not straightforward. In other words, the traditional convolution operation cannot be directly applied to graph data and there are two main reasons for that (Gao et al., 2018):

- The convolutional operation requires a fixed size of the number of neighbors for each node. However, in graph-structured data the size of the neighbors for each node varies.
- The nodes of a graph should be in an order for the applicability of the convolution operation. However, in generic graphs, there is no ranking function to order the neighbors of nodes.

Due to the aforementioned challenges, the convolutional graph neural networks (ConvGNNs) aim to process graph-structured data by generalizing convolution operation from grid data to graph data. The ConvGNNs find their application in various domains (e.g., computer vision, natural language processing, recommendation systems, chemistry). Moreover, there exists a considerable amount of studies, which aim to apply ConvGNNs to different tasks (Wu et al., 2019).

2.1.5 Recurrent Neural Networks

The aforementioned networks in the previous sections, i.e., feedforward and convolutional neural networks are designed to process data whose attributes do not depend on each other. However, certain data types like text or time-series data sequentially depend on each other. Obviously, text data can be processed by a simple feedforward neural network. Yet, the sequential information of the words is not taken into account by such a network. Recurrent neural networks are specialized for processing sequential data, and they are most commonly used with text data.

Traditional text categorization approaches often, rely on bag-of-words representations of texts. Such methods ignore the sequential information of the words. Bag-of-words representations may work well for the categorization of long documents. However, for more sophisticated tasks (e.g., machine translation, sentiment analysis, etc.) the ordering of the words might be a critical feature. Therefore, in such tasks, recurrent neural networks have been utilized successfully.

Recurrent neural networks have a simple architecture that is derived from conventional feedforward neural networks by adding *recurrent* connections on hidden layers. Although the recurrent networks can be used almost with any sequential data, its application in the text-domain is the most common. In the following, the examples of applications that commonly utilize recurrent neural networks are given (Aggarwal, 2018b):

1. **Language models.** Given a set of history of words with their sequential information,

language models aim to predict the next word in that sequence. This is a typical scenario of standard language models, which find their application in various areas of text mining and information retrieval.

2. **Auto regressive analysis.** Auto regressive analysis tasks aim to learn the next element of a given real-valued time-series.

3. **Machine translation.** Given a sentence as an input, machine translation aims to translate the sentence into the desired language. In such a scenario the input and the output are sentences.

4. **Text categorization.** Text categorization aims to assign one or more predefined categories to a given text. In the case of text categorization, the input is a piece of text and the output is a vector of class probabilities.

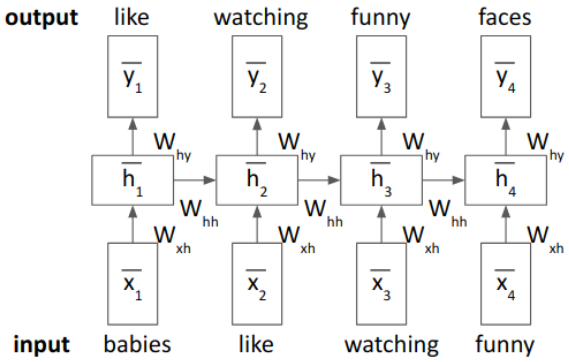


Figure 2.4: An overview of a simple recurrent neural network architecture for language modeling.

The simple recurrent network representation is shown in Figure 2.4. This architecture is particularly designed for language modeling, which aims to predict the next word, given the history of the words. The main characteristic of the network is the presence of shared weight matrices, i.e., W_{xh}, W_{hh}, W_{hy} in the temporal layers of the network. Each time-stamp, i.e., the position in the sequence (starts at 0 or 1, and increases by 1) has an input, output, and hidden unit. First, each word from a given sequence is one-hot encoded and then fed one at a time to the neural network. In other words, each word is fed to the network at the relevant time-stamp. In the given example of a language model, the output is a vector of probabilities where each dimension corresponds to a certain word.

Given an input vector at time *t* (e.g., one hot encoded t^{th} word of the sequence) is $\overline{x}_t$,

the hidden state is $\overline{h}_t$ and the output $\overline{y}_t$ which is the predicted probabilities of the $(t+1)^{th}$ word. The hidden state can be formulated as follows (Aggarwal, 2018b):

$$\overline{h}_t = f(\overline{h}_{t-1}, \overline{x}_t) \tag{2.14}$$

where $\overline{h}_{(t-1)}$ is the hidden vector at time $(t-1)$, input $\overline{x}_t$ and output $\overline{y}_t$ are d-dimensional vector of a vocabulary of size d. Further, the function f utilizes weight matrices and activation functions to compute all the hidden states. Although each hidden state is updated at each timestamp, the weight values remain the same over all the timestamps. Equation 2.15 can be expanded to include also the output as follows (Aggarwal, 2018b):

$$\begin{aligned} \overline{h}_t &= tanh(\overline{W}_{xh}\overline{x}_t + \overline{W}_{hh}\overline{h}_{t-1}) \\ \overline{y}_t &= W_{hy}\overline{h}_t \end{aligned} \tag{2.15}$$

where $tanh$ denotes an activation function, W_{xh} is input-hidden matrix and W_{hh} is a hidden-hidden matrix (cf. Figure 2.4), and $\overline{y}_t$ is an output.

The given architecture in Figure 2.4 can easily be modified in order to be employed for other applications. In other words, depending on the application at hand input and output units can easily be adapted. For example, for a sentiment analysis task which is a special type of a text categorization application, the input is sequential data and the output corresponds to the category of the given sequence.

In the following section, long short-term memory networks which are a special type of recurrent neural network are briefly presented.

Long Short-Term Memory Networks

Due to temporal layers, recurrent neural networks can be very deep and thus training becomes a challenging task. Further, as stated before the parameter matrices of the recurrent network are shared among the different layers of the network. Such characteristics of the network can harm the optimization process. This problem referred to as *vanishing and exploding gradients*, which is one of the most common problems that recurrent neural networks face (Aggarwal, 2018a).

Long Short-Term Memory (LSTM) networks are designed to address the problem of vanishing and exploding gradients by changing the recurrence conditions of the hidden states. To do so, there exist an additional hidden vector so called *cell state* and denoted by $\bar{c}_t{}^{(k)}$ which is of a size p. The cell state is responsible for keeping the information (at least a part of it) from the previous stages. To achieve that a smooth way of updating the cell states over time is applied. By doing so, the persistence in information storage which prevents the vanishing and exploding gradient problems is achieved.

The parameter update operation leverages 4 different p dimensional vectors, i.e., $\bar{i}$ *(input)*, $\bar{f}$ *(forget)*, $\bar{o}$ *(output)*, $\bar{c}$ *(new content of the cell state)*.

To compute the hidden state vector $\bar{h_t}^{(k)}$ and the cell vector $\bar{c_t}^{(k)}$ first the the intermediate vectors are computed as follows:

$$\begin{bmatrix} \bar{i} \\ \bar{f} \\ \bar{o} \\ \bar{c} \end{bmatrix} = \begin{pmatrix} sigm \\ sigm \\ sigm \\ tanh \end{pmatrix} W^{(k)} \begin{bmatrix} h_t^{-(k-1)} \\ h_{t-1}^{-(k)} \end{bmatrix}$$

where "$sigm$" denotes a sigmoid operation. The vector $\bar{c}$ is the newly proposed contents of the cell state.

After calculating the four intermediate vectors, with the weigh matrices $W^{(k)}$ for the kth layer, then the equation that updates the cell state defined as follows (Aggarwal, 2018b):

$$\bar{c_t}^{(k)} = \bar{f} \odot \bar{c}_{t-1}^{(k)} + \bar{i} \odot \bar{c}$$

where $\odot$ denotes element-wise product of vectors.

Finally, the equation which is applied to update the hidden states is defined as follows (Aggarwal, 2018b):

$$\bar{h_t}^{(k)} = \bar{o} \odot tanh\bar{c}_t^{(k)}$$

2.1.6 Text Embedding Models

The idea of enabling intelligent systems to understand natural language has been a goal of various fields of artificial intelligence (Srinivasan, 2017). Such systems require natural language data to be transformed into a specific format to be able to process them. One of the most common ways is to represent texts in a form of a sparse and high dimensional discrete vectors, e.g., bag-of-words, one-hot encoding, etc. However, these types of representation models have three main disadvantages:

1. The dimension of the vectors is high, which is determined by the unique number of words present in vocabulary.
2. To generate text vectors the semantic relation between the words is not taken into account.
3. Such models often lead to inaccurate results on new and rare words.

Due to these challenges, recently word embedding models have emerged as an important research field (Almeida & Xexéo, 2019). These models aim to generate a low dimensional

vector representation of words or phrases by preserving syntactic and semantic relationships of the words. The models are trained to learn the vector representation of words in a way that semantically similar words should have similar representations. Hence, such words are placed close to each other in a vector space.

The embedding models have been extensively leveraged in a variety of natural language processing tasks, e.g., text categorization, question answering, machine translation, semantic analysis. Moreover, it is also common to exploit word vectors from embedding models to simply construct document vectors. Subsequently, the generated document vectors can also be utilized in a wide range of natural language processing tasks such as query-specific document ranking, document similarity calculation, document categorization.

Beside the word embedding models, there is a considerable research body on *document embedding* models which aim to generate the distributed representation of texts, i.e., documents, paragraphs, sentences. The basic idea of these models is utilizing syntactic and semantic information of words contained in documents to generate document vectors.
In the following, we give an overview of the most prominent word and document embedding models:

- **Skip-gram (Mikolov et al., 2013a).** The Skip-gram is one of the models of word2vec which is a group of word embedding models (i.e., Skip-gram and Continuous bag-of-words) (Mikolov et al., 2013a). The method learns the low dimensional representation of words while capturing syntactic and semantic word relationships from a given large corpus. The word vectors are computed by leveraging 2-layer simple feedforward neural networks. Training Skip-gram is very efficient. In other words, depending on the size of the corpus and the parameters, the training phase can only take a couple of hours. Moreover, the model can be easily adapted to obtain domain-specific word or phrase representations. For example, for a patent document categorization task, the word vectors can be generated by training the model with a relevant large patent text corpus. It should be noted that the Skip-gram model designed to be trained in an unsupervised manner.

  Figure 2.5 depicts the overview of the Skip-gram model. For the sake of simplicity let $w(t-2), w(t-1), w(t), w(t+1), w(t+2)$ be a sequence of words from a given corpus. Given a center word $w(t)$ from a sequence, the model tries to predict the probability for every word in the vocabulary of being the nearby, i.e., surrounding context, word of $w(t)$. According to Figure 2.5, $w(t)$ is a center word and its surrounding context words are $w(t-2), w(t-1), w(t+1), w(t+2)$. The number of surrounding context words for each center word is determined by a parameter called *window size*. In this example, the window size is equal to 2 as there are 2 left and 2 right context words for $w(t)$.

  The goal of training the Skip-gram model is to compute word vectors that are applicable to predict the surrounding words of a given center word. Then the objective of Skip-

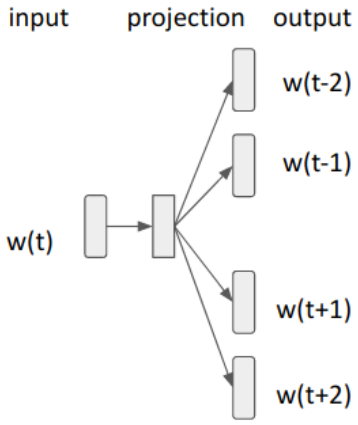


Figure 2.5: An overview of Skip-gram model architecture.

gram model is to maximize the average log probability of a given sequence of words $w_1, w_2, w_3, ..., w_T$ as follows:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} \log p(w_{t+1}|w_t) \tag{2.16}$$

where c is the size of the training context and specified by the window size. The bigger the window size is more the training samples are.

The probability $p(w_{t+1}|w_t)$ defined by using softmax function:

$$p(w_O|w_I) = \frac{exp({v'_{w_O}}^T v_{w_I})}{\sum_{w=1}^{W} exp({v'_w}^T v_{w_I})} \tag{2.17}$$

where v_w is input and v'_w is output vector representations of w, and W is the number of all the unique words in the vocabulary. Computing such an equation for each training sample is a very expensive task. Therefore, instead of calculating full softmax, hierarchical softmax, which is computationally much more efficient is used. Unlike softmax, which evaluates $|W|$ output nodes to find the probability distribution, hierarchical softmax evaluates only approximately $\log_2(W)$ nodes. The output layer, i.e., the last layer is represented as a binary tree in hierarchical softmax. Each leaf node corresponds to a word w, $w \in W$ and each node in the tree represents the relative probability of its child nodes. The hierarchical softmax defined as follows:

$$P(w|w_I) \prod_{j=1}^{L(w)-1} \sigma([n(w,j+1) = ch(n(w,j))] \cdot {v'_{n(w,j)}}^T v_{w_I}) \tag{2.18}$$

where $n(w,j)$ is j-th node on the path from the root to w, $L(w)$ denotes the length of this path and $\sigma(x) = 1/(1+exp(-x))$. Given the equation of hierarchical softmax the cost of computing $P(w|w_I)$ is on average not greater than $\log W$.

Furthermore, the authors adapt Noise Contrastive Estimation (NCE) (Gutmann & Hyvärinen, 2012) method as an alternative to hierarchical softmax. Noise contrastive estimation is based on the assumption that an ideal model should be capable of differentiating data from the noise through the logistic regression model. Then for the Skip-gram model given $P(w_O|w_I)$ the goal is to distinguish between the target word w_O and the negative samples which are generated randomly. For each data sample, there are k negative samples. Then the objective of negative sampling defined as follows:

$$\log\sigma({v'_{w_O}}^T v_{w_I}) + \sum_{i=1} \mathbb{E}_{w_i \sim P_n(w)} \left[\log\sigma(-{v'_{w_O}}^T v_{w_I}) \right] \tag{2.19}$$

where $P_n(w)$ is a noise distribution, from which the negative samples are generated. $P_n(w)$ is a parameter and the authors show that the uniform distribution of it performs the best for several tasks.

The Skip-gram model requires large corpora for learning the vector representation of words. The frequency of words has a vital impact on the quality of the vectors. In a very large corpus, usually, the most frequent words are the least informative words (e.g., "the", "a", "an", "up"). To address this problem a simple subsampling approach has been defined as:

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}} \tag{2.20}$$

where $f(w_i)$ is the frequency of the word w_i and t is a chosen threshold, around 10^{-5}.

Overall, the Skip-gram model has been the base of many embedding models such as DeepWalk, node2vec, Doc2Vec, etc. In addition, it is still one of the most standard baselines for many word, document and network embedding models.

- **Continuous Bag-of-Words (CBOW) (Mikolov et al., 2013a).** The continuous Bag-of-Words (CBOW) is another model of word2vec. The model aims to learn the low dimensional representation of words from given large text corpora. Figure 2.6 illustrates the overview of CBOW. The model is very similar to the Skip-gram model, however the inputs and outputs are reversed. CBOW also computes the word vectors by exploiting a simple feedforward neural network.

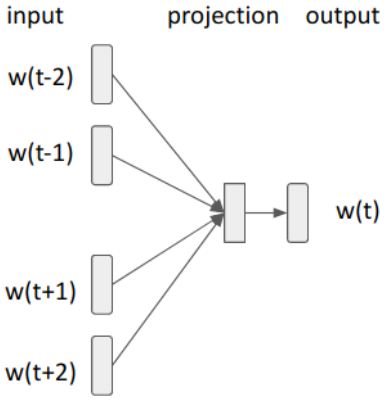


Figure 2.6: An overview of CBOW model architecture.

Let $w(t-2), w(t-1), w(t), w(t+1), w(t+2)$ be a sequence of words from a given corpus (cf. Figure 2.6). Given context words $w(t-2), w(t-1), w(t+1), w(t+2)$ CBOW trained to be applicable for predicting the center, i.e., target word $w(t)$. Similar to Skip-gram model, the number of context words are determined by the given window size. The objective of CBOW model is to maximize the average log probability of a given sequence of words $w_1, w_2, w_3, ..., w_T$ as follows:

$$\frac{1}{T}\sum_{t=1}^{T} \log p(w_t|w_{t-c}...w_{t+c}). \tag{2.21}$$

where c is window size.

The probability $p(w_t|w_{t-c}...w_{t+c})$ can be calculated with the softmax function as follows:

$$p(w_t|w_{t-c}...w_{t+c}) = \frac{exp(\bar{v}^T v'_{w_t})}{\sum_{w=1}^{W} exp(\bar{v}^T v'_w)} \tag{2.22}$$

where W is the entire vocabulary of the given corpus, v'_w is the vector of word w and $\bar{v}$ is the average vector of all the context words.

- **Doc2Vec (Le & Mikolov, 2014).** Doc2Vec model extends Skip-gram model in order to obtain the latent representation of texts, i.e., sentences, paragraphs, documents. Unlike Skip-gram, which learns only the vector representation of words from a large corpus, Doc2Vec learns vector representation of words as well as documents in which the words present. Doc2Vec is an unsupervised algorithm which is trained to be useful for predicting the words within documents to learn the document representations. Therefore,

semantically similar documents that share many common words expected to be located close to each other in a common vector space. Figure 2.7 illustrates the overview of

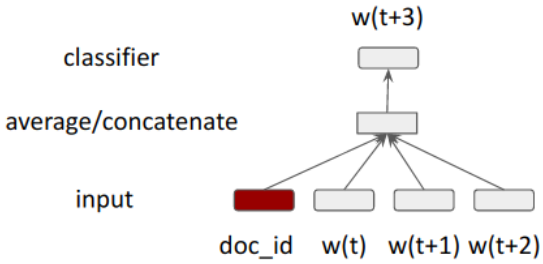


Figure 2.7: An overview of Doc2Vec model architecture.

Doc2Vec architecture. The document vector D is concatenated (or averaged) with the word vectors W from the document and the model predicts the next word from the given context.

The model is inspired by the CBOW architecture, which is trained to predict context words from a given center word. Similar to CBOW, the Doc2Vec model is also trained to predict the next word from a piece of given context information. However, in Doc2Vec the inputs are not only words but also a document vector, which is treated similarly as the word vectors. Every word and document from the given corpus mapped to a unique vector. After the training, word and document vectors can be used in a wide range of natural language processing tasks such as text categorization, question answering, text summarization, etc. Since the architecture of Doc2Vec is very similar to Skip-gram, the technical details are omitted.

- **BERT (Devlin et al., 2019).** Bidirectional Encoder Representations from Transformers (BERT) is an embedding model, which utilizes both right and left context jointly to pre-train deep bidirectional representations. The pre-trained model then can be easily fine-tuned to create a language representation model for a wide range of natural language processing tasks. The implementation of BERT is much more sophisticated than the previous embedding models , therefore, the technical details of the model are skipped in this section.

  Input to BERT is a sequence of words which can be a single sentence or a pair of sentences. The framework consists of two main steps, namely, pre-training and fine-tuning. During the pre-training phase the model is trained with two unsupervised tasks, namely, masked language modeling and next sentence prediction. The masked language modelling randomly masks some tokens of the input, then it tries to predict those masked tokens. In this task, the output layer leverages a softmax function over the entire vocabulary. Finally, the model tries to predict the masked inputs instead of an entire input.

The second task of pre-training is binarized next sentence prediction (NSP). The goal of this task is to train the model to learn the connection between two sentences. The dataset for this task is easily generated from a given corpus as follows:

Given a sentence A and if sentence B is the actual next sentence that comes right after A then it would be labeled as IsNext and the randomly selected sentences from the corpus would be labeled as NotNext.

The second component of the framework is fine-tuning. After the pre-training phase, all the parameters of the model can be fine-tuned with labeled data. In other words, depending on the application at hand different types of inputs and outputs can be plugged into BERT in order to fine-tune all the parameters. For example, for a question answering task, inputs, and outputs, i.e., question-passage pairs are fed into BERT for fine-tuning. Note that for each application, such as question answering, named entity recognition there are different fine-tuned models.

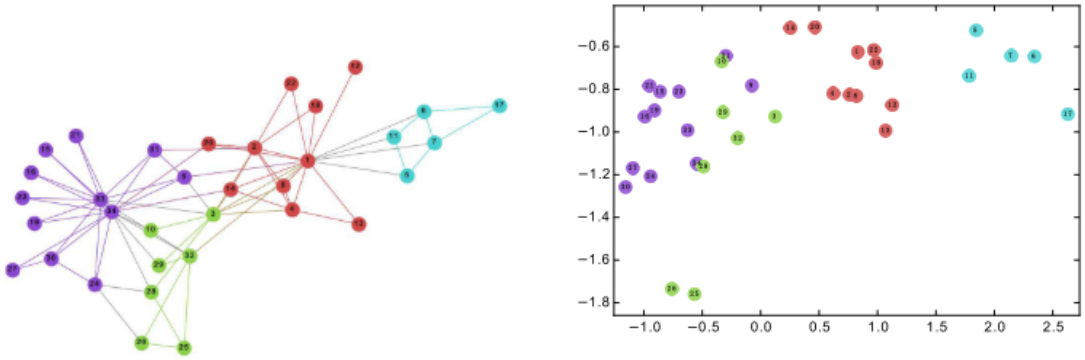
2.1.7 Network Embedding Models

Networks are powerful graph-based structures for modeling large-scale data. Many real-world systems form a structure of a network such as information networks, social networks, etc. Moreover, networks are great data sources for a variety of tasks such as link prediction, node categorization, node clustering, etc. However, the raw representation of network elements such as nodes, edges, etc. cannot be directly processed by the machine learning systems. There exist traditional explicit network representation methods such as adjacency matrices; however, they seem to be relatively inefficient, especially in large-scale networks. Hence, effective and efficient representation of networks has emerged as an important field of research. To facilitate the discussion first the formal definition of a network embedding is given as follows (Arsov & Mirceva, 2019):

Definition 2.1.1. (Network Embedding):
A network embedding function $f : v \rightarrow r_v$ which maps each vertex $v \in V$ to d dimensional vector in $\mathbb{R}^d$, r_v is the d dimentional vector representation of v.

Recently, several network embedding models have been designed to generate the low dimensional vector representation of nodes of a network. These models aim to preserve the network structure while learning the distributed representation of nodes in a low dimensional space. For example, if any two nodes in a network have a strong connection, then these two nodes are located closely in the vector space. Because the distance between two vectors in a vector space quantifies the measure of similarity between them.

On the other hand, the similarity between the nodes can be defined in an application-specific way. For example, in a social network, if two nodes share several common neighboring nodes, then those nodes can be considered similar. In that case, they should be placed



(a) Karate network

(b) The distributed representation of the karate network

Figure 2.8: An example of a network embedding model. The image is extracted from (Perozzi et al., 2014)

closely in the vector space. Figure 2.8 illustrates an example of an embedding model. Given karate network (Figure 2.8a) as an input, the similar nodes (which share the same color) placed close to each other in the embedding space (Figure 2.8b).

There are two main goals that network embedding models aim to achieve while representing the nodes in a low dimensional space (Cui et al., 2019):

1. Reconstruction of the original network should be possible from the learned vector space, in other words, the distance between the nodes that connected in the original network should be small in the learned vector space.

2. The learned embedding space should be applicable to original network inference tasks, such as identifying important nodes.

Network embedding models adopt different approaches, i.e., matrix factorization, random walk, deep neural networks, etc. in order to transfer the nodes into their low dimensional representation. In the following the most commonly used methods by the network embedding models are given:

- **Matrix Factorization.**
  Network embedding models aim to find the low dimensional representation of nodes of a given network. Matrix factorization methods are common ways to achieve this purpose. Such methods accept large network topology matrices, where each row and

column corresponds to a node as an input (Cui et al., 2019). Each entry in such matrices indicates the relation between the corresponding nodes. Then given a matrix M, the matrix factorization can be defined as follows:

$$\min_{W,C} \|M - W^T C\|. \tag{2.23}$$

where W and C are two matrices and have lower ranks than M. Overall, the matrix factorization operation, given a matrix M aims to find W and C matrices.

- **Random Walk.**
  Random walks have been used in a variety of applications such as in recommendation systems as a similarity measure (Perozzi et al., 2014). In the context of network embeddings, random walk models are being utilized to produce arbitrary paths from a given network. By doing so, neighborhood information of vertices can be extracted from the network. Network embedding models, which exploit random walk techniques mainly rely on the neighborhood information of vertices to generate vector representation of vertices. This idea highly relates to a neural language model by regarding a vertex as a word and a random walk as a sentence.

- **Neural networks.**
  Neural networks especially deep neural networks are also commonly exploited by several network embedding models (Devlin et al., 2019; Perozzi et al., 2014; Tang et al., 2015b). Section 2.1 provides a general overview of neural networks.

The network embedding models that have been employed in this thesis work are presented as follows:

- **DeepWalk (Perozzi et al., 2014).** The DeepWalk network embedding model aims to learn distributed representations of vertices in a given network by considering the neighborhood relations of vertices. To this end, the model attempts to conduct two main tasks:
  1. The model generates random walks over the network,
  2. The representation of each vertex is generated based on the Skip-gram model (cf. Section 2.1.6) by utilizing the generated paths (from the 1st step).

  More precisely, DeepWalk relates the distribution of each vertex present in random walks to the distribution of words that appear in a piece of text (Cui et al., 2019). Motivated by this assumption, DeepWalk adapts the language model, i.e., Skip-gram to update the representation of each vertex generated by the random walks. Given a graph the random walk generator randomly samples a vertex v which is considered as a root of the random walk W. Then the walk uniformly samples the neighbor of the

last visited vertex iteratively until the maximum length of the walk t is reached. The maximum length of the walks (t) is an input parameter. Likewise, the number of walks at each vertex is also an input parameter.
The Skip-gram model iterates over the collection of vertices that appear within the given window size. In Section 2.1.6 the Skip-gram model is introduced, therefore, in this section, the technical details of Skip-gram are skipped.

- **LINE (Tang et al., 2015b).** Large-scale Information Network Embedding (LINE) learns the latent representations of vertices of an information network. The network could be undirected, directed, and/or weighted. As the name implies, the model is capable of scaling to very large information networks. Figure 2.9 illustrates an example of a simple information network. The thickness of the edges between vertices indicates the strongness of the connections.

  LINE aims to optimize an objective function, which preserves the local structure, i.e., *first-order proximity* as well as the global structure, i.e., *second-order proximity* of a given network.

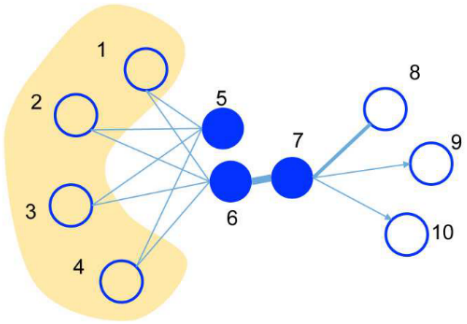


Figure 2.9: An example of an information network. The image is extracted from (Tang et al., 2015b).

First-order proximity is defined between two nodes that are connected by an edge (Cui et al., 2019). For example, in Figure 2.9 the edge between vertex 6 and 7 is thicker therefore, the connection is stronger. According to the first-order proximity, these nodes should be close to each other in the vector space. On the other hand, the first-order proximity between the vertex 5 and 6 is zero as there is no edge between them. To model the first order proximity between vertices v_i and v_j the following joint probability defined as:

$$p_1(v_i, v_j) = \frac{1}{1 + exp(-\vec{u}_i^T \cdot \vec{u}_j)} \tag{2.24}$$

where $\vec{u}_i$ ($\vec{u}_j$) is the vector representation of node v_i (v_j), respectively. In addition, its empirical probability can be defined as $\hat{p_1}(v_i, v_j) = \frac{w_{ij}}{W}$, where $W = \sum_{(i,j)\in E} w_{ij}$, E is the set of edges between nodes in the network, and w_{ij} is the weight of the edge (i, j). In order to preserve the first-order proximity, the model aims to minimize the KL-divergence between the two distributions $p_1(v_i, v_j)$ and $\hat{p_1}(v_i, v_j)$. By omitting some constants, the final goal is to minimize the following objective function:

$$O_1 = -\sum_{(i,j)\in E} w_{ij} \log p_1(v_i, v_j) \tag{2.25}$$

On the other hand, the second-order proximity is determined between the two nodes in a network by considering their common (shared) nodes. In other words, two nodes that share the same neighbors are considered to be similar according to the notation of second-order proximity. For example, the node 5 and 6 in Figure 2.9 should be placed closely as there are several common neighbors between them. To model the second-order proximity, for each edge (i, j), the conditional probability is defined as follows:

$$p_2(v_j|v_i) = \frac{exp(-\vec{u}_j^T \cdot \vec{u}_i)}{\sum_{k=1}^{|V|} exp(-\vec{u}_k^T \cdot \vec{u}_i)} \tag{2.26}$$

where V is the set of nodes connected with v_i in the network. The empirical probability of $p_2(v_j|v_i)$ can be defined as $\hat{p_2}(v_j|v_i) = \frac{w_{ij}}{d_i}$, where d_i is the out-degree of v_i. In order to preserve the second-order proximity, the conditional distribution $p_2(v_j|v_i)$ is made close to $\hat{p_2}(v_j|v_i)$ based on the KL-divergence over the entire set of nodes in the network, such that the model minimizes the following objective function:

$$O_2 = -\sum_{(i,j)\in E} w_{ij} \log p_2(v_j|v_i) \tag{2.27}$$

In order to keep both first-order and second-order proximities for each node, two LINE models are trained. The first LINE model is trained by preserving the first-order proximity, and then another LINE model is trained by preserving the second-order proximity. Finally, concatenating the embeddings of both models yields a final embedding for each node.

- **PTE (Tang et al., 2015a).** PTE aims to learn the distributed representation of texts in a semi-supervised manner. It leverages labeled as well as unlabeled data to learn the representation of documents. Further, unlike other embedding models such as Skip-gram, which do not use any labeled data and are generalizable for a variety of tasks, PTE is designed to be utilized by a particular task. More precisely, the obtained vector

representations from PTE can be easily fine-tuned for a certain task with a small set of labeled dataset. Given a large text corpus PTE first represents co-occurrence information between words-words, words-documents, and words-labels by generating three different networks as follows:

- Word-word network reflects the co-occurrence information of words in the same context (i.e., context window). The weight of each edge of this network is determined by the number of times co-occurrence of the two words in the same context windows.

- Word-document network encodes the co-occurrence information of words in documents. The weight of an edge between a word and document defined by the number of times the word present in the document.

- Word-label network reflects the information of word co-occurrences in a category level. The weight between the word w_i and category c_j is w_{ij} and it is defined as: $w_{ij} = \sum_{(d:l_d=j)} n_{di}$ where ld is a category label of document d and n_{di} is the term frequency of word w_i.

- The combination of word-word, word-document, and word-label networks constitutes the heterogeneous text network.

The heterogeneous text network is being exploited by PTE to learn the latent representation of words while preserving the second-order proximity (cf. Section 2.1.7).

The overall heterogeneous network consists of three homogeneous networks, i.e., word-word, word-document and word-label networks. To model the second-order proximity of a homogeneous network, for each edge (v_i, v_j), the conditional probability $p(v_j|v_i)$ is defined as follows (Tang et al., 2015b):

$$p(v_j|v_i) = \frac{exp(-\vec{u}_j^T \cdot \vec{u}_i)}{\sum_{v_k \in V} exp(-\vec{u}_k^T \cdot \vec{u}_i)}\,, \tag{2.28}$$

where V is the set of vertices connected with v_i in the network, $\vec{u}_i$, $\vec{u}_j$ and $\vec{u}_k$ are the vectors of vertices v_i, v_j and v_k, respectively. The empirical probability of $p(v_j|v_i)$ can be defined as $\hat{p}(v_j|v_i) = \frac{w_{ij}}{d_i}$, where d_i is the out-degree of v_i and w_{ij} is the weight of the edge (v_i, v_j).

In order to preserve the second-order proximity, the conditional distribution $p(v_j|v_i)$ is made close to $\hat{p}(v_j|v_i)$ based on the KL-divergence over the entire set of vertices in the

network, to this end, the model minimizes the following objective function:

$$O = -\sum_{(v_i,v_j)\in E} w_{ij} \log\left(p(v_j|v_i)\right), \tag{2.29}$$

The embedding of the individual word-word, word-document and word-label networks are learned simultaneously by minimizing the following objective function:

$$O_{pte} = O_{ww} + O_{wd} + O_{wl}, \tag{2.30}$$

where O_{ww}, O_{wd} and O_{wl} are the objective functions defined in Equation (2.29) for the homogeneous word-word, word-document and word-label networks, respectively. To optimize the objective function in Equation 2.30, the edges are firstly collected from these three homogeneous networks as three sets, one for word-word edges, one for word-document edges and the other for word-label edges, and then in each training iteration, edges are sampled from each set to update the model. Readers can refer to (Tang et al., 2015a; Tang et al., 2015b), for the detailed optimization process.

After learning the word vectors, the vector representation of an arbitrary text (e.g., phrase, sentence, paragraph) d, which consists of words $w_1, w_2, w_3, ..., w_n$, defined as follows:

$$\vec{d} = \frac{1}{n}\sum_{i=1}^{n} \vec{u}_i. \tag{2.31}$$

- **RDF2Vec (Ristoski & Paulheim, 2016).** RDF2Vec aims to learn the distributed representations of entities from RDF graphs. Similar to DeepWalk, RDF2Vec converts the RDF graph into a set of sequences. To do so, it leverages graph walk techniques, i.e., breadth-first algorithm and Weisfeiler-Lehman Subtree RDF graph kernels (de Vries, 2013; de Vries & de Rooij, 2015). The Weisfeiler-Lehman Subtree graph kernel has been proposed for a graph comparison task. Basically, it computes the number of subtrees that are shared between two graphs in order to determine the similarity. The authors (de Vries & de Rooij, 2015) have modified the method to be applicable to RDF graphs. RDF2Vec leverages this model to generate a set of entity sequences. The distribution of each entity present in sequences can be related to the distribution of the words appearing in sentences. Further, the generated sequences are used to train neural language models, i.e., Skip-gram and Continues bag-of-words.

  The proposed model differs from the DeepWalk mainly in two aspects:

  1. RDF2Vec is specifically designed for RDF graphs, which are directed and edge labeled, such as knowledge bases.

2. The generated entity vectors are task-independent, in other words, the latent representation of the entities can be utilized in different tasks with different datasets.

- **Joint Embedding of Hierarchical Categories and Entities (Li et al., 2016c).** The embedding model aims to embed entities and hierarchically related categories from large knowledge bases into a common vector space. Further, the method designed to be useful for capturing semantic relatedness between entities and categories by integrating structural knowledge from knowledge bases. (Li et al., 2016c) propose two different embedding models, namely, Category Embedding (CE) model and Hierarchical Category Embedding (HCE) model to reflect the semantic relatedness between entities and categories.

  – **Category Embedding (CE).** The model extends the already existing entity embedding approach (Hu et al., 2015) by additionally including category information. Similar to DeepWalk, and RDF2Vec, the proposed approach is also based on the Skip-gram embedding model. The originated work (Hu et al., 2015) forms an entity context by acquiring all the entities from the entire article (e.g., Wikipedia article) that describes the entity. The model improves this approach by integrating category information from a knowledge base to the entity context. Given a target entity, the model is designed to predict its context entities and its associated categories from the knowledge base.

    The probability of target-context entity pair (e_t, e_c) defined as follows:

$$P(e_c|e_t) = \frac{e_t \cdot e_c}{\sum\limits_{e \in E} exp(e_t \cdot e)} \tag{2.32}$$

    Then, the entity and the category vectors are learned by maximizing the following average log probability:

$$L = \frac{1}{|D|} \sum_{(e_c, e_t) \in D} [\log P(e_c|e_t) + \sum_{c_i \in C(e_t)} \log P(e_c|c_i)]. \tag{2.33}$$

    where D is the set of all entity pairs, c_i is a category vector and $C(e_t)$ is the associated categories of entity e_t.

  – **Hierarchical Category Embedding (HCE).** This model extends the CE model by including the hierarchy information of the categories into semantic space. The model assumes that if a category is placed close to an entity in the semantic space then its ancestor categories should also be placed close to the entity. Further,

the ancestor categories are also weighted based on the distance between the target entity and the ancestor category. In other words, the categories that are close to the target entity based on the category hierarchy (e.g., directly associated categories) should be weighted more. Based on these assumptions the following weighted-average log probability is proposed:

$$L = \frac{1}{|D|} \sum_{(e_c,e_t)\in D} [w_i \text{log} P(e_c|e_t) + \sum_{c_i \in A(e_t)} \text{log} P(e_c|c_i)]. \tag{2.34}$$

where $A(e_t)$ denotes the ancestor categories of e_t, w_i is the weight of each category and $w_i \alpha \frac{1}{l(c_c,c_i)}$ where $l(c_c, c_i)$ is the average number of steps to reach c_c from c_i. The weight value reflects the relevancy between the category and its ancestor categories. The closer a categories to its ancestor category, the more relevant they are considered to be.

Both CE and HCE adapts the same optimization approach, i.e., stochastic gradient descent of Skip-gram as follows:

$$\begin{aligned} L = & \sum_{(e_c,e_t)\in D} [\text{log}\sigma(e_c \cdot e_t) + \sum_{c_i \in A(e_t)} w_i \text{log}\sigma(e_c \cdot c_i)] + \\ & \sum_{(\grave{e}_c,e_t)\in \grave{D}} [\text{log}\sigma(-\grave{e}_c \cdot e_t) + \sum_{c_i \in A(e_t)} w_i \text{log}\sigma(-\grave{e}_c \cdot c_i)] \end{aligned} \tag{2.35}$$

where $\grave{D}$ is the negative sample pairs.

The authors evaluated both embedding models, i.e., category embedding and hierarchical category embedding, in the context of concept categorization and dateless categorization. The hierarchical category embedding method has provided better performance in both tasks.

2.2 Knowledge Graphs

This section provides a general overview of knowledge graphs. The section contains three main subsections. In the first subsection, formal definitions of knowledge graphs and relevant details are given. In the second subsection, the most prominent open knowledge graphs are introduced. Finally, the last subsection briefly discusses the Linked Open Data and its evolution.

2.2.1 Definitions and Preliminaries

Knowledge graphs have been utilized in many information systems which require to leverage a structured, diverse, large-scale collection of data (Paulheim, 2017; Hogan et al., 2020), e.g., the Google Knowledge Graph plays a very important role for improve the search engine results. Besides the industry, knowledge graphs have also been extensively utilized in various research areas such as artificial intelligence. Hence, there has been a considerable amount of study on knowledge graphs, and many scientific literature have been published in this direction (Hogan et al., 2020; Paulheim, 2017; Ehrlinger & Wöß, 2016; Färber et al., 2015).

Given that knowledge graphs are drawing more attention especially after the announcement of the Google Knowledge Graph, different definitions have been proposed to describe them (Hogan et al., 2020). Here, some of the most recent and prominent definitions are presented as follows:

Definition 2.2.1. (Knowledge Graph):
"A knowledge graph is a semi-structured data model characterized by three components: (i) a ground extensional component, that is, a set of relational constructs for schema and data (which can be effectively modeled as graphs or generalizations thereof); (ii) an intensional component, that is, a set of inference rules over the constructs of the ground extensional component; (iii) a derived extensional component that can be produced as the result of the application of the inference rules over the ground extensional component (with the so-called "reasoning" process)." (Bellomarini et al., 2019)

Definition 2.2.2. (Knowledge Graph):
"A knowledge graph mainly describes real world entities and their interrelations, organized in a graph; defines possible classes and relations of entities in a schema; allows for potentially interrelating arbitrary entities with each other; covers various topical domains." (Paulheim, 2017)

Definition 2.2.3. (Knowledge Graph):
"A knowledge graph acquires and integrates information into an ontology and applies a reasoner to derive new knowledge." (Ehrlinger & Wöß, 2016)

Definition 2.2.4. (Knowledge Graph):
Different than the aforementioned definitions, Färber et al. (2018) gives the formal definition of a knowledge graph as an RDF graph. *"An RDF graph consists of a set of RDF triples (cf. Definition 2.2.5), where each RDF triple* (s,p,o) *is an ordered set of the following RDF terms: a subject* $s \in U \cup B$*, a predicate* $p \in U$*, and an object* $U \cup B \cup L$*. An RDF term is either a URI* $u \in U$*, a blank node* $b \in B$*, or a literal* $l \in L$*.* U*,* B *and* L *are pairwise disjoint".* (Färber et al., 2018)

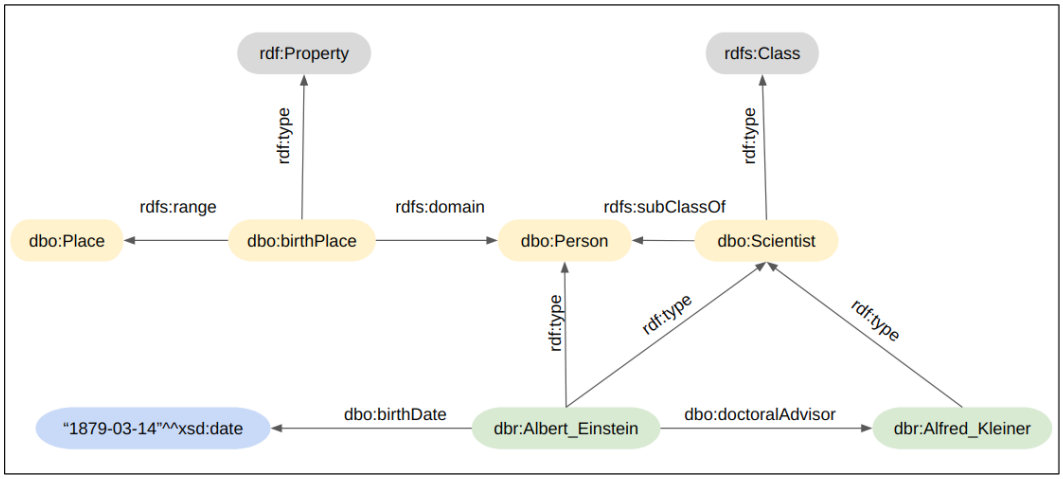


Figure 2.10: An example of a simple RDF graph

Furthermore, in the *Semantic Web* community the term knowledge graph is also used to refer to the Semantic Web Knowledge Bases such as Wikidata, DBpedia, YAGO, etc. (Paulheim, 2017). To facilitate the discussion, first we give the definition of the Semantic Web which was proposed by Tim Berners-Lee et al. as "*A new form of web content that is meaningful to computers*", in 2001. Since then the fundamentals of the Semantic Web have been based on the idea of representing knowledge in a structured and machine understandable way. Today, knowledge bases are one of the most essential components of the Semantic Web. They employ a graph-based form to represent knowledge, which can be domain specific such as facts about chemical interactions or domain independent. The facts in knowledge bases are modeled by directed edge-labeled graphs as shown in Figure 2.10. Such graphs consist of a set of nodes such as `dbr:Albert_Einstein, dbo:Scientist` and directed-labeled edges between those nodes such as `rdf:type, rdfs:subClassOf`.

The Semantic Web knowledge bases follow standardized data modeling formats which facilitate the knowledge exchange. These standards include but not limited to *Resource Description Framework* (RDF), *Resource Description Framework Schema* (RDFS) which is an extension of the RDF and *Web Ontology Language* (OWL). The specifications of the data modeling formats are published by World Wide Web Consortium[2] (W3C) which is an organization publishes the main international standards for the World Wide Web.

The RDF standards allow to define different types of nodes in a knowledge base namely, *resources* (entities on the Web) which could be anything like people, location, documents, etc., *literals* that allow representing data type values such as dates, integers, etc. and

[2] `https://www.w3.org/standards/`

blank nodes. The RDF resources are identified by unique *Internationalized Resource Identifiers* (IRIs) (Dürst & Suignard, 2005) that allow identification of entities on the Web. However, blank nodes are not assigned to any identifier and thus they do not carry any additional information within a knowledge base. Instead, blank nodes can only indicate an existence of a thing. The representation of knowledge in knowledge bases is based on the idea of making statements about the resources in the form of RDF triples (`subject, predicate, object`), where the subject could be an entity or a blank node, whereas the object could be an entity or a blank node or a literal. Note that literals can only be in an object position.

Definition 2.2.5. (RDF triple):
"Given a set of URIs U, a set of blank nodes B, and a set of literals L, an RDF triple is represented as $t = (s, p, o) \in (U \cup B) \times U \times (U \cup B \cup L)$*, where* s *is a subject,* p *is a predicate and* o *is an object"* (Alam, 2015).

Definition 2.2.6. (Triple Pattern):
"Let U, B, L be disjoint infinite sets of URIs, blank nodes, and literals, respectively. Let V be a set of variable such that $V \cap (U \cup B \cup L) = \emptyset$*. A triple pattern defined as a 3-tuple* $(s, p, o) \in (U \cup V) \times (U \cup V) \times (L \cup U \cup V)$*, where the components subject, predicate and object correspond to RDF terms or variables"* (Deibe, 2018).

Definition 2.2.7. (RDF Graph):
"A finite set of RDF triples is called as RDF Graph G such that $G = (V, E)$*, where V is a set of vertices and E is a set of labeled edges"* (Alam, 2015).

Some of the RDF triple examples from Figure 2.10 are `(dbr:Albert_Einstein rdf:type dbo:Person)` and `(dbr:Alfred_Kleiner rdf:type dbo:Scientist)`. Note that RDF's namespace abbreviated by `"rdf:"`[3]. There exist several serialization syntaxes to convert RDF graphs into machine readable forms such as N-triples[4], Turtle[5], etc. Listing 2.1 is the turtle serialization of the RDF graph, which is depicted in Figure 2.10. Listing 2.1 shows that the RDF vocabulary has been used to define resources (e.g., `dbr:Albert_Einstein rdf:type dbo:Scientist`) and predicates (e.g., `dbo:birthPlace rdf:type dbo:Property`).

[3] `http://www.w3.org/1999/02/22-rdf-syntax-ns#`
[4] `https://www.w3.org/TR/n-triples/`
[5] `https://www.w3.org/TR/turtle/`

Listing 2.1: RDF document in turtle serialization

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix dbo: <http://dbpedia.org/ontology/> .
@prefix dbr: <http://dbpedia.org/resource/> .

dbo:birthPlace   rdfs:range      dbo:Place .
dbo:birthPlace   rdfs:domain     dbo:Person .
dbo:birthPlace   rdf:type        rdf:Property .

dbo:Scientist    rdf:type        rdfs:Class .
dbo:Scientist    rdf:subClassOf  dbo:Person .

dbr:Albert_Einstein     dbo:birthDate     "1879-03-14"^^xsd:date .
dbr:Albert_Einstein     rdf:type          dbo:Person .
dbr:Albert_Einstein     rdf:type          dbo:Scientist .
dbr:Albert_Einstein     dbo:doctoralAdvisor     dbr:Alfred_Kleiner .
```

Another particular RDF vocabulary is RDFS which enables the specification of schema knowledge (Hitzler et al., 2010). RDFS's namespace abbreviated by `"rdfs:"`[6]. For example, the definition of domain and range of properties (e.g., (`dbo:birthPlace rdfs:domain dbo:Person`), (`dbo:birthPlace rdfs:range dbo:Place`)) and hierarchical relationships, i.e., sub classes (e.g., `dbo:Scientist rdfs:subClassOf dbo:Person`), sub properties and more besides (Hitzler et al., 2010). Due to the capability of specifying such schema knowledge, RDFS can also be considered as an *ontology language*. To facilitate the discussion first define the term *ontology*. According to (Gruber et al., 1993) ontology can be defined as follows:

Definition 2.2.8. (Ontology):
From a computer science point of view there exist different definitions for an ontology. Originally, (Gruber et al., 1993) defined ontology as *"explicit specification of a conceptualization"*. Thereafter, an ontology defined as *"formal specification of a shared conceptualization"* by (Borst et al., 1997). Finally, (Studer et al., 1998) combined these two definitions and defined an ontology as *"formal, explicit specification of a shared conceptualization"*. Today, the latter definition is the one which is most commonly accepted and used.

In this definition, *conceptualization* implies that existing of an abstract model of a cer-

[6] `http://www.w3.org/2000/01/rdf-schema#`

tain domain which contains identified relevant concepts and their relations. *Explicit* denotes that meaning of all concepts must be defined explicitly. *Shared* stands for consensus about the ontology and *formal* refers to machine readability and understandability.

Figure 2.10 illustrates that the simple ontologies can be modeled by RDF(S), however, for modeling more complex representations of knowledge the Web Ontology Language (OWL) is being used. OWL is based on formal logic and supports much greater expressiveness than RDF(S). Further, it allows logical reasoning on the knowledge and thus enables to deduce implicit knowledge which is not explicitly modeled. There exist different flavours of OWL2 which is the last version OWL and based on description logic, i.e., OWL2 EL, OWL2 RL, OWL2 QL, OWL2 DL, OWL2 Full. Each of these species of OWL2 has different level of expressivity. OWL2 DL is more expressive than OWL2 EL, OWL2 RL and OWL2 QL and less expressive than OWL2 Full. In other words, OWL2 DL is the expressive and decidable version of OWL. Therefore, in this section we focus on OWL2 DL.

OWL2 standards which are specified by World Wide Web Consortium allow to define class expressions (e.g., conjunction, disjunction), properties, class and property axioms, and facts. Moreover, OWL supports RDF syntax therefore, similar to RDFS documents, each OWL document in RDF syntax is also an RDF document. OWL's name space is abbreviated by "owl:" [7]. Further, the OWL axioms expressed in terms of classes, individuals and properties as follows:

- **Classes.** OWL classes are similar to RDFS classes. There exist two predefined classes in OWL namely, `owl:Thing` whose members are all individuals and `owl:Nothing` which is an empty class, i.e., it does not contain any individual. Definition of a class as follows:
  - `:Wine rdf:type owl:Class.`

- **Individuals.** OWL individuals can be defined in two different ways, namely via class membership which is similar to RDFS class instances and without a class membership.
  - `:RoseWine rdf:type :Wine .`
  - `:Beer rdf:type owl:NamedIndividual .`

- **Properties.** OWL allows definition of two types of properties, i.e., object properties and datatype properties and they are similar to RDFS properties.
  - **Object Properties.** Object Properties of OWL have classes as domain and range.

```
* :pairWith rdf:type owl:ObjectProperty ;
            rdfs:domain owl:Thing ;
            rdfs:range :Seafood .
```

[7] <http://www.w3.org/2002/07/owl#>

  - **Datatype Properties.** Datatype Properties of OWL have datatypes as range.

```
* :color rdf:type owl:ObjectProperty ;
         rdfs:domain owl:Thing ;
         rdfs:range xsd:date .
```

Obviously, it is also possible to define much more expressive facts with OWL such as logical class constructions, property restrictions, etc. More details about OWL can be found in (Hitzler et al., 2010).

2.2.2 Open Knowledge Graphs

Open knowledge graphs are openly available and freely usable. There are different ways of constructing open knowledge graphs, i.e, they can be curated by a small group of people or crowd sourced by a large group of people or created by utilizing automatic or semi-automatic methods (Paulheim, 2017). In the following, the examples of knowledge bases which have been built by different methods are given:

- **Cyc and OpenCyc (Lenat, 1995).** Cyc is one of the oldest curated knowledge bases of common sense which is developed and maintained by the CyCorp company starting in 1984. The artificial intelligent project Cyc, represents millions of common sense facts. For example:
  - "You have to be awake to eat.",
  - "You cannot remember events that have not happened yet.".

  Cyc aims to be served as a structured data source especially to the artificial intelligence based applications such as information retrieval, speech recognition, etc. (Lenat, 1995) A small version of Cyc, so called OpenCyc has been made publicly available in order to serve as a great data source, especially, for artificial intelligence researchers. Further, there used to be an existing endpoint to OpenCyc, which was also linked to other Linked Open Data[8] datasets such as DBpedia. The 2012 version of OpenCyc consist of approximately 2.5 million facts around 120,000 instances. The schema of OpenCyc contains around 45,000 hierarchly related type information and 19,000 possible relations (Paulheim, 2017). The CyCorp company, the developer of OpenCyc, has stopped its support for the knowledge base since March 2017.

- **WordNet (Fellbaum, 1998).** WordNet is a large lexical knowledge base which represents semantic relations between words for the English language. It is created in the Cognitive Science Laboratory of Princeton University starting in 1985. WordNet

[8] https://lod-cloud.net/

includes the lexical categories of different types of words, i.e., nouns, verbs, adjectives and adverbs. The words that are from the same lexical category collected into sets of synonyms, called *synsets*. Synsets are interlinked by the semantic links such as hyperonymy, meronymy. This knowledge graph can be navigated easily with the browser. WordNet is publicly available and can be downloaded freely. The 3.0 version of WordNet consists of in total 155,287 unique strings, 206,941 word-sense pairs and 117,659. The unique strings include 117,798 nouns with 82,115 synsets, 11,529 verbs with 13,767 synsets, 21,479 adjectives with 18,156 synsets, 4,481 adverbs with 3,621 synsets.

- **Freebase (Bollacker et al., 2008).** Freebase is a public knowledge base which is created through crowd sourcing by the announcement of American software company Metaweb in March 2007. In contrast to other curated knowledge bases, e.g., Cyc, which required more than 900 person years to create (Paulheim, 2017), Freebase provided an interface to the public editors. Thereby, the editors could directly include new facts to the knowledge base by editing structured data with the provided schema templates. Besides the contribution of the editors, the knowledge from Wikipedia has been integrated to Freebase (Färber et al., 2015). In 2010, Freebase acquired by Google and on August 31, 2016, it was completely shut down. The last version of Free-base consist of more than 3 billion facts about around 50 million entities. Further, its schema contains 27,000 entity types and 38,000 relations (Paulheim, 2017).

- **Wikidata (Vrandecic & Krötzsch, 2014).** Wikidata is a collaboratively edited knowledge base by the community and it is operated by the Wikimedia foundation. Wikidata has been extensively exploited as a main data source by a wide range of information system applications, especially by the Semantic Web community. Unlike aforementioned knowledge bases, Wikidata does not only contain facts but also the source of the facts so that the validity of facts can be approved. Moreover, after the shut down of Freebase, its data moved to Wikidata. To date, Wikidata contains roughly 83,343,004 instances[9] and 1,036,340,466 million statements[10]. Its schema defines roughly 7,450 relations[11].

- **DBpedia (Auer et al., 2007).** DBpedia is one of the most popular knowledge graphs in the Linked Open Data cloud (Färber et al., 2015). Unlike the aforementioned knowledge bases, DBpedia has been created by automatically extracting structured and multilingual knowledge from Wikipedia, i.e., Wikipedia infoboxes. The types of infoboxes in Wikipedia are mapped to the DBpedia ontology (i.e., DBpedia Classes),

[9] https://www.wikidata.org/wiki/Wikidata:Main_Page
[10] http://tools.wmflabs.org/wikidata-todo/stats.php
[11] https://www.wikidata.org/w/index.php?title=Special:ListProperties

further, the attributes of these infoboxes correspond to the properties in DBpedia ontology (Paulheim, 2017). Similar to Wikidata, DBpedia has been widely utilized in various research fields of the Semantic Web. Today, the most recent version of the DBpedia, i.e., DBpedia 2016-10 contains 6.6M entities[12], 13 billion facts (RDF triples), and its ontology schema comprises 760 classes, 1,622 datatype properties as well as 1,105 object properties.

- **YAGO (Suchanek et al., 2007).** Similar to DBpedia, YAGO is also created by extracting knowledge from Wikipedia. In contrast to DBpedia, YAGO comprises information not only from Wikipedia (e.g., infoboxes, categories) but also WordNet (e.g., synsets, hyponymy), GeoNames (Färber et al., 2015; Fabian et al., 2007). Further, DBpedia creates different knowledge bases for each language edition of Wikipedia (e.g., English, German, etc.) that are linked to each other whereas YAGO contains knowledge from various Wikipedia language editions. The version 3 of YAGO contains around 24 million facts about 4.5 million entities in 10 languages. The schema comprises 488,496 classes and 77 manually defined relations (Mahdisoltani et al., 2015).

- **NELL (Carlson et al., 2010).** Never Ending Language Learning[13] (NELL) project have been developed at Carnegie Mellon University, Pittsburgh, Pennsylvania since January 2010. Unlike DBpedia and YAGO which utilizes semi-structured content (e.g, Wikipedia Infoboxes) as a base, NELL designed to exploit unstructured data to extract meaningful knowledge. In other words, the project attempts to conduct two main tasks everyday: (1) it tries to extract facts from text found in Web pages, (2) it aims to improve the first step i.e., fact extracting capability, so that next time it can extract more accurate facts from the Web. Each extracted fact has a weight which reflects the confidence of it. The system is still running today and has been enhancing its knowledge base since the time project has been developed. To date, NELL has collected over 50 million candidate facts by reading the Web, and 2,810,379 of these facts are in high confidence. On NELL website, only the facts that has the high confidence are displayed.

So far the well-known open knowledge bases have been presented. Most of the open knowledge bases contain links to other RDF data sources which enables users to navigate from one data source another one by following RDF links. Hence, the Semantic Web aims to make meaningful links between publicly available RDF data sources (e.g, DBpedia, Wikidata, etc.) to enable persons and machines to explore *Linked Open Data* freely. In the following let us briefly discuss the phrase *Linked Open Data*.

[12] `https://wiki.dbpedia.org/develop/datasets/dbpedia-version-2016-10`
[13] `http://rtw.ml.cmu.edu/rtw/`

2.2.3 Linked Open Data

Linked Open Data (LOD) refers to publicly available RDF datasets which are identified via URI and accessible via HTTP. Further, those RDF datasets are also linked to other datasets via URI.

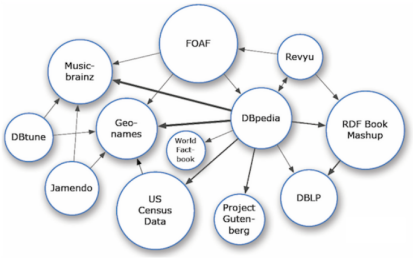
The principles of Linked Data were first outlined by Berners-Lee in 2006:

- Use URIs as names for things.
- Use HTTP URIs so that people can look up those names.
- When someone looks up a URI, provide useful information, using the standards (e.g. RDF).
- Include links to other URIs, so that they can discover more things.

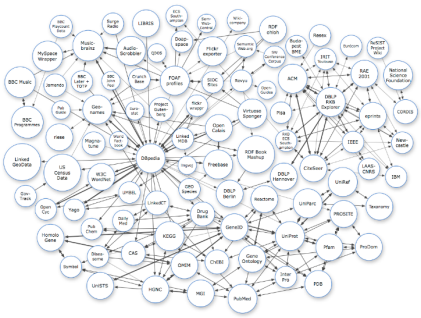
The four rules provide broad guidance on which publishing RDF data can be based. LOD can be accessed using traditional HTML browsers and users can easily explore and navigate between different data sources.

The network of publicly available RDF datasets that are in concordance with linked data principles is growing more and more everyday. As of April 2017, the number of publicly available RDF datasets is 9,960 and it contains roughly more than 149 billion facts and more than around 800 million links. The visualization of interlinked RDF datasets which is referred to as *Linked Open Data Cloud*[14] has been created by Max Schmachtenberg & Cyganiak. Figure 2.11 illustrates the evolution of the LOD cloud from 2007 its beginning until 2019. Each node in the networks represents one dataset. As of March 2019, the huge network consists of 1,239 datasets including 16,147 links. Further, each color of the datasets, i.e., nodes, from 2014, 2017 and 2019 represents a different domain. For example, the green nodes from 2017 represent datasets from linguistics domain. Moreover, DBpedia has been one of the most popular datasets in these LOD clouds across the years.

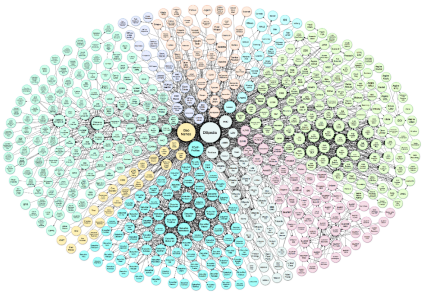
[14] `https://lod-cloud.net/`



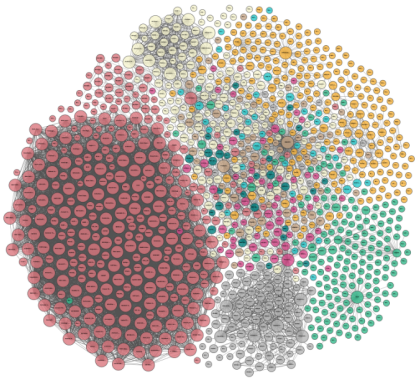
May 2007 (12 datasets)



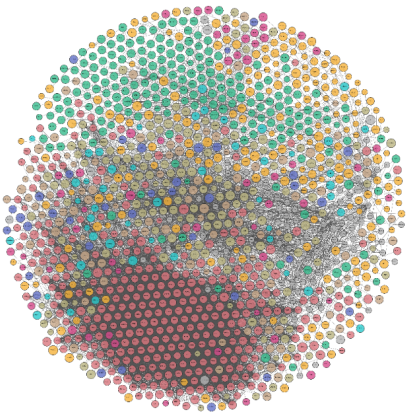
July 2009 (95 datasets)



August 2014 (570 datasets)



August 2017 (1,163 datasets)



May 2020 (1,255 datasets)

Figure 2.11: The evolution of the LOD cloud

2.3 Semantic Measures

Semantic measures are leveraged by a wide range of fields such as Natural Language Processing, Semantic Web, Bioinformatics, etc. The main topic of this thesis is short text categorization, which is one of the fundamental tasks of natural language processing, and thus in this section semantic measures in the context of natural language processing applications are presented.

Semantic measures are theoretical tools, which aims to quantify the semantic relation between elements, e.g., words, sentences, documents (Harispe et al., 2013). The formal definition of semantic measures is given below.

Definition 2.3.1. (Semantic Measures):
$\sigma_k : E_k \times E_k \rightarrow \mathcal{R}$
where E_k *is the set of elements,* $k \in K$ *and* K *is various types of elements that are comparable semantically, e.g.,* $K = words, concepts, sentences, texts, ...$ *and* $\mathcal{R} = \{[0,1], \mathbb{R}^+, \{a, b, c, ...\}\}$ (Harispe et al., 2013).

Although there exist several notations that are associated with semantic measures, *semantic relatedness* and *semantic similarity* are the ones most commonly utilized in natural language processing applications such as text categorizatio.

Definition 2.3.2. (Semantic Relatedness):
"Strength of the semantic interactions between two elements without restriction regarding the types of semantic links considered" (Harispe et al., 2013).

Definition 2.3.3. (Semantic Similarity):
"Specializes the notion of semantic relatedness, by only considering taxonomical relationships in the evaluation of the semantic strength between two elements" (Harispe et al., 2013).

(Gomaa et al., 2013) have introduced semantic similarity with the help of *corpus-based* and *knowledge-based* algorithms:

- **Corpus-based similarity:** In order to measure the corpus-based similarity of language units, large corpora are exploited. The most popular techniques of corpus-based similarity are as follows:
  - **Latent Semantic Analysis (LSA) (Landauer & Dumais, 1997):** This method relies on the assumption that similar words tend to occur in similar pieces of texts. Hence, to generate the document vectors, the frequency of words contained in the documents is considered. In other words, a large matrix where each row represents a term and each column corresponds to a document from a given corpus is constructed by considering word occurrences in documents. Subsequently, Singular Value Decomposition (SVD) is applied to the matrix, to reduce the dimensionality

while preserving the similarity among the documents. Then the similarity between the documents is quantified by applying cosine similarity to the document vectors.

- **Explicit Semantic Analysis (ESA) (Gabrilovich et al., 2007):** Similar to LSA, ESA constructs document vectors by leveraging a large corpus, i.e., English Wikipedia. Each document is transformed into a weighted vector of Wikipedia entities, i.e., concepts. In other words, first, the words from a given corpus are converted into a vector where each dimension corresponds to a TF-IDF value between the word and a Wikipedia article. Then the document vectors are formed by utilizing the vectors of the words present in the documents.
- **Normalized Google Distance (NGD) (Cilibrasi & Vitanyi, 2007)**: In contrast to LSA and ESA which utilize large corpora, NGD exploits the number of hits returned by the Google search engine to determine the semantic similarity between two words. The semantically similar words tend to be close based on the NGD, whereas the dissimilar words tend to be far away from each other. The NGD between two terms, i.e., x and y defined as follows:

$$sim(x,y) = \frac{max\{\log f(x), \log f(y)\} - \log f(x,y)}{\log M - min\{\log f(x), \log f(y)\}} \tag{2.36}$$

where $f(x)$ is the number of hits for the search term x, similarly, $f(y)$ is the number of hits for the search term y. In addition, $f(x,y)$ is the number of web pages on which both x and y occur, and M is the total number of web pages that have been searched by Google.

- **Knowledge-based similarity:** The knowledge-based similarity between the language units is determined by exploiting information obtained from large knowledge bases, e.g., WordNet (cf. Section 2.2.2). The measures of knowledge-based semantic similarity are categorized into two groups (Gomaa et al., 2013):
  1. based on information content,
  2. based on path length.

Moreover, (Resnik, 1995) has defined the knowledge-based semantic similarity, with the help of the notion of information content. The method utilizes a taxonomy structure, i.e., *is-a relations* to derive the semantic similarity between concepts. The model first associates each concept in the taxonomy with a probability. To this end, the model defines a probability function p such that $p : C \rightarrow [0,1]$, for any concept $c \in C$ of the taxonomy, $p(c)$ is the probability of encountering an instance of concept c. Finally, the semantic similarity between arbitrary two concepts is defined as follows (Resnik, 1995):

$$sim(c_1, c_2) = \min_{c \in S(c_1, c_2)} [-\log p(c)] \tag{2.37}$$

where $-\log p(c)$ is the information content of the concept c defined by (Ross, 1976) and

$S(c_1, c_2)$ is the set concepts subsume both c_1 and c_2.

Moreover, (Waitelonis et al., 2015) have proposed *connectedness weighting* to quantify the semantic similarity and relevancy between an entity within a document D and the document D. The method first collects all the entities from D and the entities (from the underlying knowledge graph), which are connected at least two entities from the document. The function $rel(e_i, e_j)$ is defined to obtain a graph between the collected entities, and it returns true if there is a link between e_i and e_j. Given $e_i \in D$ let E_i denote the set of entities that e_i has a link to and F_i denote a set of entities to which e_i indirectly connected:

$$E_i = \{e \in D | rel(e, e_i)\} \quad \text{and} \quad F_i = \{e \in D | \exists x : rel(e, x) \wedge rel(x, e_i)\}$$

Then the connectedness is defined as follows:

$$cn(e_i, d) = 1 + (|E_i| + |F_i|) \times \frac{|D|}{n_d}, \text{where} \quad n_d = \sum_{e_j \in D} |E_j| + |F_j|. \tag{2.38}$$

On the other hand, (Nies et al., 2014) have proposed three different methods, i.e., ontology-based, link-based, shared-links-based to measure the semantic similarity between the documents. The proposed measures, first, link the named entities present in a text to an external knowledge base, i.e., DBpedia, and then exploit these entities to derive the similarity between the documents. The proposed similarity measures are as follows (Nies et al., 2014):

- **ontology-based:** The similarity between two entities is determined by the number of edges in the shortest path between the entities in the underlying ontology schema.
- **link-based:** The similarity between two entities is derived based on the in/direct connections from the underlying knowledge base.
- **shared-links-based:** As the name indicates the similarity between two entities is quantified based on the number of shared connections.

Furthermore, there has been a considerable amount of studies on the distributed representation of language units (cf. Section 2.1.6). Several embedding models have been proposed to construct the vector representation of words, entities, sentences, paragraphs, etc. Today, such embedding models are extensively utilized due to their effectiveness and efficiency. Therefore, to quantify the semantic similarity between the elements of the semantic spaces has become a crucial task to be achieved. One of the most common ways of measuring the similarity between two vectors from a common multidimensional space is comparing the distance between them. In other words, the vectors that are closely placed in the vector space considered to be similar. In order to measure the distance between two vectors, several different metrics have been utilized widely.

The most prominent vector similarity measures are presented as follows:

- **Cosine Similarity:** The cosine similarity is one that most commonly used to measure the similarity between two vectors of language units, such as word vectors, document vectors, query vectors, etc. The measure is based on the cosine angle between the two vectors. Hence, if any two vectors roughly pointing to the same direction then they are considered to be similar. Given vectors A and B the cosine similarity is calculated as follows:

$$sim(A, B) = \frac{A \cdot B}{\|A\|\|B\|} \tag{2.39}$$

- **Euclidean Distance:** The Euclidean distance is derived by measuring a regular straight line which connects the vectors. The smaller the distance, the greater is the similarity between the vectors. The Euclidean distance is calculated with the following formula:

$$sim(A, B) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + ... + (a_n - b_n)^2} \tag{2.40}$$

  where A, B are vectors and the Cartesian coordinates of A, B are $A = (a_1, a_2, ..., a_n)$ and $B = (b_1, b_2, ..., b_n)$, respectively.

- **Manhattan Distance:** The Manhattan distance is measured by the sum of the absolute differences of coordinates of given two data points. The formula of the Manhattan distance is as follows:

$$sim(A, B) = \|A - B\| = \sum_{i=1}^{n} |a_i - b_i| \tag{2.41}$$

  where A, B are vectors and $A = (a_1, a_2, ..., a_n)$, $B = (b_1, b_2, ..., b_n)$.

2.4 Summary

This chapter has presented a brief overview of the general neural network models (e.g., feedforward, convolutional neural networks, etc.), notable knowledge graphs (e.g., DBpedia, Wikidata, etc.) and semantic measures (e.g., semantic similarity, semantic relatedness, etc.). The neural networks have been extensively leveraged to build this thesis. Therefore, in the next sections, how these models are utilized to build more sophisticated approaches is

explained. Furthermore, as stated before, knowledge bases have been utilized as a main data source for the proposed models in this thesis (cf. Section 4 and Section 5). Therefore, in the next sections, more details about how the knowledge graphs are being utilized are given. In the last section of this chapter, semantic measures such as semantic relatedness and semantic similarity have been presented. The approaches proposed in this thesis (cf. Section 4 and Section 5) extensively leverage semantic similarity measure, and thus several different ways of calculating semantic similarity have been discussed in this chapter.

3 Text Categorization

This chapter aims to give general background information about text categorization. The chapter consists of six sections. Section 3.1 introduces an arbitrary-length text (e.g., news article) categorization task, its application scenarios and related studies on this task. Section 3.2 describes a short text (e.g., news title) categorization task which is the main topic of this thesis and provides related studies on this task. Extracting meaningful features from text documents is a crucial task for text categorization. In order to achieve that text preprocessing techniques are applied as the first step to text documents by natural language processing applications. In this regard, Section 3.3 and Section 3.4 present the most prominent text preprocessing techniques and feature extraction methods, respectively. Section 3.5 provides a brief description of the machine learning algorithms which are commonly used for the text categorization task. Finally, the chapter is concluded with Section 3.6 which presents the most commonly used evaluation metrics (e.g., accuracy, precision, recall, etc.) for text categorization models.

3.1 Arbitrary-length Text Categorization

The available text data on the Web is growing more and more every day. According to IBM[1], 80% of the data on the Web is unstructured, and the most common form of these unstructured data is text (e.g., phrases, sentences, paragraphs, documents). It is essential to obtain meaningful insights from such data. To this end, the most important steps are learning to process and understand the data (Sarkar, 2016). Hence, as a fundamental step, *text categorization* is a crucial task, which enables us to automatically organize such a vast amount of text data.

Definition 3.1.1. (Text Categorization):
Given an input text t and n predefined labels $L = l_1, l_2, .., l_n$, the output of the text categorization system is the most relevant label $l_i \in L$ for the given text t, i.e., computing the label function $f_{cat}(t) = l_i$, where $l_i \in L$.

Due to the availability of text data on the Web, the text categorization task has gained significant attention across industry and academia. Hence, the problem of text categorization has been extensively studied in various domains such as data mining, machine learning, information retrieval, etc. It is one of the most fundamental tasks in natural language processing.

[1] https://www.ibm.com/blogs/watson/2016/05/biggest-data-challenges-might-not-even-know/

The main goal of the categorization task is to assign a category label to an instance of text data for which the label is unknown. It should be noted that in a *multi label categorization* task there is no limitation on the number of assigned labels. In such a task the system can assign one or more categories to a given text instance. The discussion of this task is omitted as it is out of the scope of this thesis. In this thesis we focus on *multi class categorization* task where the goal is to assign only one label from a predefined set of labels to an instance of a text. The general categorization task assumes that each category label is a predefined categorical value, e.g., *Sports*, *Science*, *Politics*, etc.

There exist three main types of text categorization methods, namely, *manual*, *rule based* and *machine learning based*. Although manually labeled documents by the experts are often highly accurate, it is the most expensive method among the mentioned approaches. Especially, if the text to be labeled is of a specific domain only several domain experts can categorize such data. Further, manually labeling text documents is also a very time consuming task. Therefore, today such a method is the least desired approach for the text categorization task. Rule-based systems, on the other hand, leverage user defined linguistic rules to perform categorization. These systems are built and maintained manually. Further, the goal of the designed rules is utilizing the semantic content of the documents to assign relevant categories. Each rule consists of a pattern and the corresponding category (Chakravarthy et al., 2008). It is often the case that the rules are designed based on the domain of the datasets, and thus in the case of a change of the domain all the rules might need to be re-designed. Hence, such a process still requires huge amounts of manual effort. Finally, the machine learning based systems, in contrast to the aforementioned systems, rely on the past observations, i.e., training set to make predictions about the labels of unseen data. Machine learning systems, first, utilize a training set which is a labeled dataset to learn the association between the predefined labels and the content of the documents. In the second step, the trained models are utilized to make predictions about labels on unseen documents. Although machine learning-based systems seem to be the most efficient and effective, they require large amounts of labeled data for the training phase. As mentioned before, obtaining labeled data is an expensive and time-consuming task. However, despite the problem of labeled data requirements, machine learning-based systems are the most common text categorization methods.

An overview of processes that are involved in a machine learning based text categorization task is shown in Figure 3.1. The traditional supervised approaches have two main steps, i.e., training and testing. In the training phase, the labeled documents are first preprocessed (cf. Section 3.3), i.e, cleaned from the noise and then the features are extracted (cf. Section 3.4) from the documents. The categorization algorithm (cf. Section 3.5) is trained by exploiting the features and the labels. Thereafter, each given unlabeled document, i.e., a test sample, is first preprocessed and its features are extracted. Finally, the trained model is used to label the unlabeled documents by utilizing the extracted features. Each process is explained in detail in the reminder of this chapter.

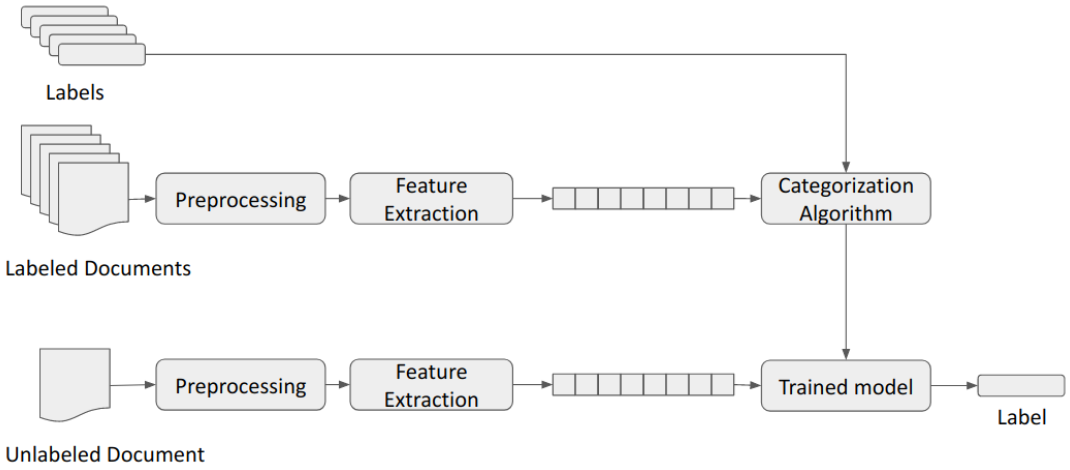


Figure 3.1: An overview of a text categorization task

Moreover, text categorization has been exploited in various domains and has a wide variety of application scenarios, for instance:

- **Information retrieval.** Text categorization techniques have been extensively exploited by information retrieval systems as well as search engines (Kowsari et al., 2019). The main task of information retrieval systems is to retrieve all the relevant documents to a given user query while retrieving minimum irrelevant documents. One of the most common problems such systems face is efficiency. Often, very large corpora are being utilized as a basis for information retrieval systems. Hence, finding the most relevant documents can be a very time consuming task. Therefore, text categorization techniques are applied to such a vast amount of documents to enhance the search results by reducing the number of irrelevant documents that are being retrieved.

- **Document organization.** As stated before, text categorization techniques have gained significant attention across the industry and academia. The main reason for this fact is their capability of organizing a vast amount of text data effectively. There exists a variety of text categorization techniques. They can be exploited for the document organization task in many domains, e.g., a large collection of patent documents, Web content, digital libraries, academic articles. In addition, it has become essential for many companies to organize and analyze business information like legal documents, emails, Web pages, etc. in order to make decisions within companies.

- **Sentiment analysis.** The sentiment analysis task utilizes text categorization techniques

to identify the expressed opinion/emotion (e.g., positive or negative or neutral) within a text. Today, especially, businesses adopt sentiment analysis systems to understand the social sentiment of their brands, products, etc. Thereafter, according to the social sentiment of customers, businesses adapt their products.

- **Spam filtering.** The goal of the spam filtering systems, as the name indicates, is to decide whether a given email is a spam or not. It is important and desirable for many organizations to determine the spam, i.e., unwanted, and/or virus-infected emails in order to avoid them from getting into email inboxes. Most of the business email networks and university networks use spam filtering models to protect their email systems from the many possible risks. The spam filtering systems extensively utilize binary text categorization techniques in order to decide whether an email is a spam or not.

In the following section, the related studies for arbitrary-length document categorization, which utilize traditional machine learning techniques are presented.

Related Work on Arbitrary-length Text Categorization

As stated before, text categorization is a well studied problem for several decades. In fact, in the 80s the most popular way of performing text categorization was rule based (Sebastiani, 2002). In other words, first, a set of rules (if-then rules) for each category defined manually by the experts. Thereafter, the documents were categorized based on the rules (Sebastiani, 2002). However, since the early 90s, the application of machine learning models for the text categorization task has become very popular (Aggarwal & Zhai, 2012).

Traditional machine learning based approaches often rely on vector space document representations (e.g, Bag-of-Words (BoW), Term Frequency-Inverse Document Frequency (TF-IDF)) for constructing documents' features. Subsequently, machine learning methods such as Decision Trees, Support Vector Machines (SVMs), K-nearest neighbor, etc. are applied to text categorization problem by utilizing the extracted features.

Decision trees and rule based models find their applications in a wide range of text categorization tasks (Su & Zhang, 2006; Li & Jain, 1998; Cohen & Singer, 1999). Moreover, probabilistic models and naïve Bayes have also been exploited successfully (Sahami et al., 1998; Koller & Sahami, 2007). In fact, in the text categorization domain the naïve Bayes might be the most commonly utilized generative model (Aggarwal & Zhai, 2012).

SVMs were first proposed by (Cortes & Vapnik, 1995) and demonstrated remarkable success in the problem of text categorization. Further, (Joachims, 1997) showed that text data is well suited for SVMs to achieve notable success. The SVMs were first applied to text data by (Joachims, 1997; Joachims, 1998). Today, they are still one of the most widely leveraged baseline models, in fact, even very sophisticated deep learning models often struggle to outperform their performance. In addition to the SVMs, which are linear categorizers,

logistic regression has also been applied to the text categorization problem successfully (Ifrim et al., 2008; Genkin et al., 2007; Komarek & Moore, 2003).

Besides the aforementioned traditional machine learning models, recently several deep learning methods have been proposed. They have demonstrated remarkable performance in text categorization as well as many other natural language processing tasks. Due to their effectiveness, today the most successful text categorization methods are mostly based on deep learning approaches.

The recurrent neural networks consider the sequential information of data for the categorization task and they have been utilized most commonly with textual data. Moreover, it has been proven by several studies (Wang et al., 2019; Zhou et al., 2015; Liu et al., 2016) that recurrent neural networks can easily be exploited for text categorization. (Wang et al., 2015) and (Liu et al., 2016) apply recurrent neural networks for sentiment analysis of tweets, and (Teng et al., 2016) improves the proposed model by exploiting more sophisticated recurrent neural network architecture for the same task. (Zhang et al., 2016) combines a recurrent neural network and convolutional neural network for sentence and document modeling.

Besides the recurrent neural networks, recently, convolutional neural networks have also been utilized for text categorization. Several convolutional neural network variations have been studied for sentence categorization and sentiment analysis (Collobert et al., 2011; Kalchbrenner et al., 2014; Lei et al., 2015). (Zhang et al., 2015) proposes a deep learning architecture where character-level features are exploited. In other words, it is the first study that has applied convolutional neural networks only on character level features. The model first extracts character level features and these features then fed into a convolutional network. The model outperforms the baselines such as bag-of-words based models and word-based convolutional neural networks on most of the datasets. Due to its performance, the method has become very popular among researchers. (Conneau et al., 2017) have proposed a sophisticated neural network architecture so-called a very deep convolutional neural network by exploiting up to 29 convolutional layers. The model utilizes character level features to perform categorization. Further, it is the first approach which applied very deep convolutional neural networks to the text categorization task. The reported results show that the model is capable of obtaining the state-of-the-art results on several public datasets. Recently, (Yao et al., 2019) have proposed a graph convolutional network-based model for text categorization. The approach first builds a word-document graph, based on corpus-based word co-occurrence. Further, the model considers the text categorization problem as a node categorization problem. Given document labels, the model then jointly learns the low dimensional representation of both words and documents. The experimental results demonstrate that the model outperforms numerous baselines.

3.2 Short Text Categorization

The growth of Web e-commerce and online platforms has given rise to the multiplication of online available short text data. Hence, it has become crucial to perform short text categorization for a wide range of applications, such as sentiment analysis, spam filtering, etc. However, due to the main characteristics of short text, the traditional text categorization methods perform poorly when they are directly applied to short texts.

The main characteristic of short text is the limited length which is no longer than 200 characters (Song et al., 2014). In other words, unlike arbitrary text documents, the short text consists of a dozen words to a few sentences (Song et al., 2014). Some of the short text examples are search snippets, tweets, news feeds, short messages, online chat logs, forum messages, etc.

Besides its limited length, short texts have other unique characteristics:

- **Sparseness** (Song et al., 2014): As stated before, the most distinct feature of short texts is the insufficient text length which consists of dozens of words. Due to the limited text length, extracting meaningful features from short texts is much more challenging in comparison to arbitrary text documents.
- **Malformed** (Sakor et al., 2019): Short texts are often not structured well, i.e., the meaning of the texts are not explicitly expressed, they are incomplete.
- **Non-standardability** (Song et al., 2014): The short text often contains non-standard terms as well as many misspellings.
- **Ambiguity** (Wang et al., 2017): Due to polysemes and typos within the short texts, they are usually very ambiguous. Further, due to the insufficient context, the ambiguities often cannot be resolved based on the available context.

Short text categorization is similar to arbitrary text categorization (cf. Definition 3.1.1), which aims to assign a category to a given short text. However, due to the aforementioned characteristics of short text, conventional text categorization models which rely on *vector space model* cannot perform well on short text documents. For example, given a sentence:

`"Tiger lost the US Open."`

The given sentence[2] is an ambiguous short text. Humans can categorize this text correctly to the category `"Sports"` without the need of any training process or any labeled data. However, most of the categorization algorithms fail to categorize this sentence correctly due to its shortness, sparsity and ambiguity. In addition, the words inside the text such as

[2] http://apps.yovisto.com/labs/ner-benchmarks/data/kore50-nif-2014-11-03.ttl

`Tiger`, `lost`, `US` and `Open` are very ambiguous. For example, `Tiger` might refer to different concepts such as an animal[3], a place[4], a movie[5], etc. These different interpretations of a word confuse the algorithms to assign a correct category to this sentence.

One of the main reasons that humans can categorize such an ambiguous and short text, is the capability of considering surface forms and their referred entities in the given text. Knowing that `Tiger` refers to a golf player "Tiger Woods[6]" and `US Open` refers to "United States Open Championship[7]" yields a correct categorization. Nonetheless, almost all the standard text categorization approaches consider words instead of entities present in the text (Zhang et al., 2015). Therefore, designing a successful short text categorization model requires an additional effort. Unlike arbitrary text categorization models, short text categorization approaches adopt more sophisticated text representation methods. In other words, due to the sparsity, short text categorization models often, enrich the text representations by capturing more semantic and syntactic information from the documents. Further, several methods utilize external knowledge such as knowledge bases to enhance the semantic representation of the texts.

In the following section, the related studies for short text categorization, which utilize traditional machine learning techniques are presented.

Related Work on Short Text Categorization

As stated before, due to the sparsity issue of short texts, traditional text categorization methods often cannot be directly applied to short text. Hence, there exists a considerable amount of research body that focuses on alleviating the data sparsity problem of short text. To do so, several studies (Wang et al., 2017; Chen et al., 2019; Wang et al., 2016b) utilize external data sources in order to enrich short text representations.

(Wang et al., 2017) propose a convolutional neural network-based model which leverages external sources, further, it combines *implicit* as well as *explicit* representations. The explicit approaches, model the text according to the traditional natural language processing steps, such as part of speech tagging, conceptualization, labeling, etc. On the other hand, the implicit representation methods model the text by leveraging neural language models, such as Skip-gram. In other words, the implicit representation methods try to obtain the low dimensional distributed representation of short texts. The proposed method, first, conceptualizes the short texts by utilizing an external knowledge base. Then, the short texts are transformed into vectors by obtaining the vector representation of entities and words from pre-trained embed-

[3]https://en.wikipedia.org/wiki/Tiger
[4]https://en.wikipedia.org/wiki/Tiger,_Georgia
[5]https://en.wikipedia.org/wiki/Tiger_(2017_film)
[6]https://en.wikipedia.org/wiki/Tiger_Woods
[7]https://en.wikipedia.org/wiki/2018_U.S._Open_(golf)

ding models. Finally, a convolutional neural network is trained by feeding the embedding of the short texts. The performance of the proposed model is compared with several baselines including a combination of BoW and traditional SVM, and more sophisticated deep neural networks (Kim, 2014; Zhang et al., 2015). The model significantly outperforms all the baselines on several benchmarks. Moreover, similar to (Wang et al., 2017), yet, a more sophisticated approach proposed by (Chen et al., 2019). The model combines the implicit and explicit text representations by leveraging knowledge bases, i.e., YAGO and Freebase. The method utilizes an attention mechanism in order to assign more weights to the relevant concepts with respect to the entire concept set. Further, the model employs also a self-attention mechanism to obtain short text representations using word-level and character-level features. The model has trained by leveraging both short text representations and concept representations and outperformed all the baselines. Moreover, (Wang et al., 2016b) proposed deep learning-based approaches for short text categorization. The results of these approaches have been compared with traditional supervised methods, such as SVM, multinomial logistic regression, etc. The authors showed that in most of the cases their approach achieved superior results. To overcome the sparsity problem of short text documents (Chen et al., 2011) proposed a method that enhances the short text representations by leveraging topics at multiple granularities. Similarly, in order to expand the semantic representation of short text documents, (Phan et al., 2008) proposed an approach to discover hidden topics based on an external Wikipedia corpus using LDA.

While performing well in practice, the aforementioned approaches, i.e., traditional machine learning based as well as deep learning approaches have both training and testing phases. For the training phase, they require a million scales of labeled data. As stated before, obtaining training data is a time-consuming and costly task. Further, their performances highly depend on the size of training data, its distribution, and the chosen hyperparameters. Last but not least, these approaches might face an efficiency problem, i.e., the training and testing phase might be slow depending on the complexity of the method as well as the size of the dataset.

3.3 Text Preprocessing

Text data, especially short texts are often, very noisy, i.e., they contain unnecessary characters, words, misspellings, etc. Extracting meaningful features becomes a challenging task from such data. Moreover, the feature extraction techniques cannot be properly applied to the noisy texts. Hence, the noise contained within texts can have adverse effects on the performance of the applications. Therefore, text cleaning and preprocessing have become a crucial step for many natural language processing applications. Depending on the dataset and the application at hand different text cleaning and preprocessing techniques can be utilized.

The most common text preprocessing techniques are presented as follows:

- **Text tokenization:** Text tokenization is a process of splitting a piece of into smaller parts which are called tokens. Two types of tokenization methods, i.e., sentence and word tokenizations are most commonly exploited. The sentence tokenization aims to split a paragraph or a document into its sentences. On the other hand, word tokenization aims to break the sequence of text into its word tokens.

  For example, given a sentence:
  `"Berlin is the capital of Germany"`, and its word tokens are `{"Berlin", "is", "the", "capital", "of", "Germany"}`.

- **Noise removal:** Text documents often contain unnecessary characters (e.g., hashtags, emojis, etc.) which could harm the feature extraction process and consequently, the categorization process. Therefore, most of the natural language processing applications remove special characters and punctuations from the text before processing or extracting any features from it.

- **Case conversions:** Text documents might contain inconsistent capitalization of words. This can be a critical problem, especially for the text categorization systems. For example, "car" and "Car" refer to the same thing. Therefore, it would be inappropriate to consider them as two different words in the feature extraction process. Therefore, converting all the words to a lower case can help to achieve consistency in terms of capitalization form of the words. However, this process can be problematic for the interpretation of some words, for instance, "US" refers to "United States of America" and "us" is a pronoun (Kowsari et al., 2019). Moreover, for different languages such as German, it might be even better to keep the capitalization form of the words due to the nature of the language. Overall, based on the characteristics of a corpus (e.g., the language of the carpus) the case conversion can be applied.

- **Removing stop-words:** Text documents often include many stop words such as "of", "the", "in", etc., which do not contribute to the performance of the categorization process. Therefore, many natural language processing applications remove stop words before they start processing the text data. Further, another benefit of removing stop words is that it helps to reduce the size of a dataset, which is critical when dealing with a large scale corpus.

- **Stemming and Lemmatization:** Text documents could contain a different form of the same words, such as "works" and "worked". Stemming and lemmatization techniques aim to reduce the morphological variation of words. While the goal of stemming is to reduce the inflected words into their base form, lemmatization tries to bring words down to their lemma form, i.e., dictionary form. Although stemming and lemmatization seem

to be closely related, there is a key difference between them. In contrast to lemmatization, which aims to correctly identify the meaning of a word based on its context, the stemmer works on a single word without considering its context. Further, in comparison to lemmatizations, stemmers are much easier to implement and they are faster. For example, given words "closed" and "closely" will be both converted to "close" after applying a standard stemming algorithm. On the other hand, an example of lemmatization is a word "better" has "good" as its lemma.

3.4 Feature Extraction

There exist a vast amount of available text data on the Web. Hence, both industry and academia desire to be able to automatically process and get meaningful insights from such data. Several applications such as information retrieval, search engines, text categorization, text summarization, machine translation, etc. utilize text data as the main data source. However, most of the systems including text categorization models cannot process the text data in a raw form, i.e., a sequence of words. Therefore, to be able to process text data, the essential step is converting it into numerical features, i.e., vector representations. The features of text data refer to the properties of text which are unique and measurable (Sarkar, 2016). The vector representation of the data should reflect the attributes of the text. There exist several feature extraction techniques which help to build feature vectors from texts that are intended to be useful for the subsequent application, such as text categorization, information retrieval, etc.

The *Vector Space Model (a.k.a Term Vector Model)* is an extensively utilized model for transforming text documents into vectors. The vector space model can be defined as follows:

Definition 3.4.1. (Vector Space Model):
Let d denote a document such that $d \in D$, where D is a given corpus and n is the total number of distinct terms in the corpus D. Then the vector space can be denoted as $VS = W_1, W_2, ...W_n$ and then d will have a n dimensional vector $d = w_1, w_2, ...w_n$ where w_i denotes the weight of the term i in d.

There are several ways of computing each dimension value, i.e., the weight of each term, such as term frequency, term frequency-inverse document frequency, etc. Following the most common feature extraction models from texts are presented:

- **Bag-of-Words (BoW):** As the name indicates, a bag-of-words model represents a text as a set, i.e., a bag of its words. The bag-of-words model does not consider the sequential and semantic information of the words and extracts only the uni-grams from texts. Following an example is given.

Given a sentence:
`"Berlin is the capital of Germany and Ankara is the capital of Turkey."`

The BoW representation of the sentence is
`BoW={"Berlin":1, "is":2, "the":2, "capital":2, "of":2, "Germany":1, "and":1, "Ankara":1, "Turkey":1}.`

After transforming texts into their bag-of-words representations, several measures can be calculated to transform the bag-of-words into feature representations. One of the most common ways of calculating features from bag-of-words is the term frequency. According to the vector space model, the above BoW representation is converted into a feature vector as follows:

`BoW=[1, 1, 1, 2, 1, 2, 2, 2, 1]`

Each dimension of the vector corresponds to a term and its weight is the number of occurrences of the term within the corresponding document.

- **Term Frequency-Inverse Document Frequency (TF-IDF):** TF-IDF is another very popular feature extraction model. The main essence of the model is to convert documents into vectors to reflect the importance of each term for each document from a given corpus. Moreover, TF-IDF is a type of a term weighting factor which is extensively utilized in vector space models. TF-IDF model transforms the documents into high dimensional vectors where each dimension corresponds to a certain term from the vocabulary. The weight of each term, i.e., each dimension value, computed to be useful for measuring the importance of a term for a document. In order to quantify the importance of a term for a given document the multiplication of the term frequency and inverse document frequency are considered and they are defined as follows:

Definition 3.4.2. (Term Frequency):
The term frequency of a term refers to the number of times the term occurs in a certain document. In other words, the simple count of occurrence of a term in the document.

Definition 3.4.3. (Inverse Document Frequency):
The inverse document frequency quantifies the importance of a term in a corpus by considering its frequency of occurrences across the documents.

The articles, i.e., "a", "an", "the" may appear much more frequently than the many other

terms in the documents. However, they do not carry any important information for distinguishing the documents from each other. Therefore, the inverse document frequency intends to decrease the weight of the terms that appear very frequently while increasing the weight of the terms that appear rarely in the corpus. The inverse document frequency is calculated as follows:

$$-\log\frac{n_t}{N} \tag{3.1}$$

where N denotes the total number of text documents in the given corpus and n_t is the number of documents in which the term t appears. Finally, the weight w_t of a term t in document d according to TF-IDF is calculated as follows:

$$w_{t,d} = \begin{cases} tf(t,d) \cdot -\log\frac{n_t}{N} & \text{if } tf(t,d) > 0. \\ 0, & \text{otherwise.} \end{cases} \tag{3.2}$$

Note that there are several variations of TF-IDF weighting schema. Formula 3.2 is one of the simplest ways of calculating the TF-IDF weight of a term.

Embedding models: Embedding models have become very popular techniques for constructing feature sets from text documents. Several embedding methods such as Word2Vec, Doc2Vec, etc. have been proposed to generate the low dimensional representation of language units, i.e., terms, phrases, documents. In comparison to the vector space model, embedding models are much more sophisticated. Often, they rely on neural networks, matrix decomposition (cf. Section 2.1.6 and Section 2.1.7) to generate the low dimensional vector representations. While BoW and TF-IDF methods which leverage only the word occurrence information, embeddings capture both syntactic and semantic word relations. In other words, BoW and TF-IDF ignore the semantic relatedness between the words. Moreover, the embedding models generate a low dimensional representation of language units, however, the dimension of the vector space models is often equal to the unique number of words present in a given large corpus. In other words, features that are generated based on BoW have a much higher number of dimensions. Therefore, the categorization task is much more efficient with the embedding models. Given that embedding models have become very popular and demonstrated notable success in many natural language applications in the last years, recently proposed almost all the text categorization approaches rely on embedding models. Since the embedding models are presented in Section 2.1.6 and Section 2.1.7, their technical details are omitted in this section.

3.5 Text Categorization Algorithms

This section provides a brief description of the machine learning algorithms, which have been commonly used for the text categorization task. The approaches that are presented in this

section have a training and testing phase. The training phase aims to train a machine learning model by utilizing a labeled dataset, i.e., a training set. The model learns the association between the attributes of documents and the labels. Thereafter, the model is leveraged to predict the labels of unseen data, i.e., test documents, and this process is referred to as a test phase.

3.5.1 Decision Trees and Random Forest

The essence of a decision tree is to split a given dataset in a hierarchical, i.e., tree-like fashion. The division is achieved based on numerous decisions, i.e., split conditions. In other words, the model aims to split a given dataset into regions, where each of them is associated with a class label. Moreover, random forest models are the more robust and solid implementation of decision trees. In the next two subsequent sections, both models are presented briefly.

3.5.1.1 Decision Trees

Decision trees, as the name implies, have a tree-like structure. They split the data in a top-down fashion where the root represents the data space. Each branch of a tree presents splitting condition(s) and the leaves of the tree are the class labels. The split conditions are similar to the feature selection criteria (e.g., looking for specific terms) in text categorization (Aggarwal, 2018a). The split criteria which is based on a single condition called *"univariate split"*, whereas the one, which is based on multiple attributes called *multivariate split*. Decision trees can be successfully applied to various domains such as text categorization. In the case of text categorization, split criteria can correspond to a frequency of one or more words present in given texts or presence or absence of certain words (Aggarwal, 2018a). However, it is a well-known fact that decision trees are prone to *over fitting*. The decision trees are built based on the training datasets. In other words, the split conditions are determined based on the training sets. Hence, it is possible to construct a tree in a very accurate way where each leaf contains data points from a single class, i.e., without any misclassified instance. Such type of a decision tree provides 100% accuracy on the training set. However, it performs very poorly on a test set. The reason for a substandard performance here can be attributed to the extreme way of constructing the decision tree. This type of a problem is tackled by adopting a *pruning* process for the lower level of the tree. However, in this way, the leaves of the tree may contain multiple class labels (Aggarwal, 2018a). Once a decision tree is built, the prediction phase is very straightforward. For each test sample, starting from the root, the branch to follow is determined based on the split criteria. This process is repeated in a top-down fashion until one leaf node is reached.

3.5.1.2 Random Forests

Random forests are more robust and effective implementation of decision trees. As the name implies, random forests adopt a randomized feature selection process for the tree constructions. Further, in contrast to decision trees, random forests construct multiple trees by randomizing feature selection process. Then for each test point, the constructed trees are utilized to make a prediction. The obtained predictions from each tree are then averaged, i.e., the number of times each predicted class is counted, to produce the final result. In this way, i.e., randomizing the tree construction and exploiting multiple trees for the prediction task, improves the performance of categorization.

3.5.2 Naïve Bayes

A naïve Bayes is a probabilistic model which is designed based on the well known Bayes theorem. To facilitate the discussion, first, the definition of a Bayes theorem is given as follows:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{3.3}$$

The naïve Bayes algorithm has been extensively studied and is still one of the standard baselines for text categorization. The model is a probabilistic generative model, which relies on the idea that a given text corpus is generated from a mixture of different predefined categories. The generative process is defined as follows (Aggarwal, 2018a):

1. Select *r*th class C_r with a probability $\alpha_r = P(C_r)$,
2. Generate the document according to the probability distribution of C_r.

The training and test sets are considered to be formed according to this generative model. Obviously, only the training data are utilized to estimate the parameters of the probabilistic model. Thereafter, the model is leveraged to estimate the probability of generation of each test document from each predefined category. Finally, each test sample is labeled with the category that provides the highest probability of generating the document. Note that the generative process utilizes Bernoulli or multinomial distributions most commonly.

The Bernoulli Model: The Bernoulli model considers only the absence and presence of terms within documents, it ignores the term frequencies. Therefore, a document vector is a sparse binary vector and each dimension of it corresponds to a term. According to the Bernoulli model the probability of generating a document D from a class C_r defined as follows:

$$P(D|C_r) = \prod_{t_j \in d} p_j^{(r)} \prod_{t_j \notin d} (1 - p_j^{(r)}) \tag{3.4}$$

where $p_j^{(r)}$ is the probability of the jth term t_j present in the document is generated from rth class. Given the formula above, the naïve Bayes model assumes that the presence or absence of the different terms is conditionally independent with respect to the given class. However, such an assumption is not true in a real-world setting. Hence, the term *naïve* is used to refer this model.

The training phase aims to estimate the model parameters, namely, $p_j^{(r)}$ and α_r. On the other hand, in the testing phase, these parameters are utilized to predict the label of a given test document.

Multinomial Model: In contrast to the Bernoulli model, the multinomial model utilizes term frequencies to model probability distributions. However, the training and the test phase of the model are very similar to the Bernoulli model. According to the multinomial model the probability of $P(D|C_r)$ defined as follows:

$$P(D|C_r) = \prod_{j=1} (q_{jr})^{d_j} \tag{3.5}$$

where q_{jr} is the parameter of the model which corresponds the fractional presence of the term t_j in the rth class and D is a document and its vector of frequency given by $(d_1...d_n)$. Similar to the Bernoulli model, the training phase aims to estimate the model parameters, i.e., q_{jr} and α_r.

3.5.2.1 *k*-Nearest Neighbors

The k-nearest neighbors algorithm assigns labels to unlabeled instances based on their similarities to training samples. The model does not have any training phase, instead, it converts each training and test instance into feature vectors. This enables model to calculate the similarity between the training and test samples.

The model first identifies the k-nearest neighbors of a given test instance based on similarity. After collecting k neighbors, the category which has the majority of the neighbors is returned as the most relevant to the given test sample. The basic implementation of the model with n training samples requires n comparisons, i.e., similarity computation, for each test instance to find the k-nearest neighbors. Thus, the efficiency of the model becomes an issue. This complexity problem can be tackled by using an inverted index (Aggarwal, 2018a) data structure model.

In order to compute the vector similarities, often, cosine similarity is utilized. Moreover, k is a parameter and its value can be determined empirically depending on the characteristics of the dataset. In other words, different values of k is leveraged by the training set and the one which provides the highest accuracy is used for the categorization task. The alternative way of finding the optimal value of k is by utilizing a validation set in a leave-one-out manner.

The k nearest neighbor algorithm have been extensively utilized as a baseline for the text categorization tasks. However, when the value of k is set to 1 then the model becomes less robust and can easily over fit. However, if the size of the training set is sufficiently large and representative then the model can still perform well. The alternative way of avoiding error that occurs when k is set to 1 is utilizing *weighted nearest neighbors* approach (Aggarwal, 2018a).

3.5.3 Support Vector Machines

A Support Vector Machine (SVM), in the case of binary categorization aims to split data points which belong to two categories based on a margin. The model first creates two hyperplanes so called *margin hyperplanes* symmetrically on both sides of the decision boundary. The hyperplanes are generated in a way that most data points lie on either side of the hyperplanes, however, on the correct side. Figure[8] 3.2 illustrates the simple SVM which is trained with samples from two categories. The decision surface $w \cdot x - b = 0$ lies in the middle of two

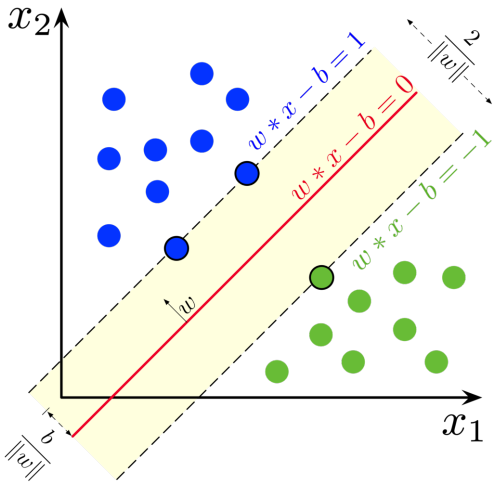


Figure 3.2: An example of an SVM trained with samples from two classes

hyperplanes and it has the equal distance to each of the *margin hyperplanes* where $w \cdot x - b = 1$ and $w \cdot x - b = -1$. Note that the points that lie between the margins are penalized and this region reflects the uncertainty points called bounded support vectors. Moreover, two hyperplanes could also touch one or more training data points as shown in Figure 3.2 and such data points called as free support vectors. The common variation of SVM objective

[8] https://en.wikipedia.org/wiki/Support_vector_machine#/media/File:SVM_margin.png

function is as follows:

$$MinimizeJ = \frac{1}{2}\|\overline{W}\|^2 + C \cdot \sum_{i=1}^{n} \xi_i \tag{3.6}$$

where C is chosen empirically, i.e., the value that maximizes the accuracy. The points that violate the margin penalized by C and their amount represented by a slack variable ξ. Therefore, the main goal of the training function is to minimize each ξ. In other words, the objective function J aims to minimize the slack variable ξ as it represents the amount that margin rules are violated.

3.5.4 Logistic Regression:

Logistic regression is a probabilistic model, which predicts the probabilities of predefined categories for each given test instance. As the name indicates, it uses a logistic function to model the binary categorization task. Given a training set which is in a form of $(x_1, y_1)...(x_n, y_n)$ where $y \in \{1, -1\}$, then the objective function of logistic regression is defined as follows:

$$MinimizeJ = \sum_{i=1}^{n} \log[1 + exp\{-y_i(\overline{W} \cdot \overline{x_i})\}] + \frac{\lambda}{2}\|\overline{W}\|^2 \tag{3.7}$$

Similar to the presented machine learning models, the training phase tries to estimate the parameters to be able to make predictions for the given test set. Further, the parameters of the model are learned by exploiting the stochastic gradient descent model (Aggarwal, 2018a). Note that there are two variants of the prediction task, namely, deterministic and probabilistic. Often, the probabilistic prediction is utilized. Typically, logistic regression considers that the dependent variable y is generated from a probability distribution which is parameterized by sigmoid of $\overline{W} \cdot \overline{x_i}$. Depending on the predicted target variable, it is possible to use a different type of distribution.

3.6 Evaluation Methods

The evaluation of text categorization methods is a crucial task to assess the performance of the systems. The existing techniques designed to evaluate the text categorization systems in terms of effectiveness and efficiency. Further, such techniques allow the comparison of a system with other text categorization methods. The effectiveness of a text categorization approach denotes the ability to categorize the documents into their correct labels. On the other hand, the efficiency refers to the execution time as well as the requirement of external resources (e.g., CPU, memory) for building and running the system. There are several metrics such as *accuracy*, *precision*, *recall*, etc. to evaluate the effectiveness of the systems. Most of the evaluation metrics are based on the elements of a *confusion matrix*. Table 3.1 illustrates a

general structure of a confusion matrix which comprises "true positives (TPs)", "true negatives (TNs)", "false positives (FPs)", "false negatives (FNs)". The confusion matrix allows to visualize the performance of a method. Note that the illustrated confusion matrix is suitable for binary categorization, where there are only two classes, namely, positive and negative.

		Predicted	
		Negative	Positive
Actual	Negative	True Negative (TN)	False Positive (FP)
	Positive	False Negative (FN)	True Positive (TP)

Table 3.1: An example of a confusion matrix

Following the elements of the confusion matrix are explained briefly:

- **True positives:** The collection of data samples that are correctly categorized as positive.
- **True negatives:** The collection of data samples that are correctly categorized as negative.
- **False positives:** The collection of data samples that are incorrectly categorized as positive.
- **False negatives:** The collection of data samples that are incorrectly categorized as negative.

As stated before, most of the text categorization models exploit elements of the confusion matrix to evalute the performance of the system by calculating the standard metrics. The most prominent text categorization evaluation metrics are as follows:

- **Accuracy:** The accuracy is the most basic and commonly used measure for evaluating the performance of the categorization model and defined as follows:

$$accuracy = \frac{(TP + TN)}{(TP + FP + FN + TN)}$$

- **Precision and Recall:** The precision measures the correctness while recall measures the completeness of the system. They are calculated in the following ways:

$$precision = \frac{TP}{(TP + FP)}$$

$$recall = \frac{TP}{(TP + FN)}$$

- **F-1 score:** The F-1 score measures the performance of the model by leveraging precision and recall. In other words, it is defined as the harmonic mean of precision and recall, and calculated as follows:

$$F_1 = \frac{(2 \times precision \times recall)}{(precision + recall)}$$

- **Macro Average and Micro Average:** Macro average method calculates the precision, recall and F-1 score for each category, i.e., positive and negative, separately. Thereafter, it takes the average of each metric in the following ways:

$$macroPrecision = \frac{1}{C} \frac{\sum_{i=1}^{C} TP_i}{\sum_{i=1}^{C} TP_i + FP_i}$$

$$macroRecall = \frac{1}{C} \frac{\sum_{i=1}^{C} TP_i}{\sum_{i=1}^{C} TP_i + FN_i}$$

$$macroF_1 = \frac{(2 \times macroPrecision \times macroRecall)}{(macroPrecision + macroRecall)}$$

Micro average method on the other hand, aggregates all the TPs, FPs, TNs, FNs from each class to calculate the metrics in the following ways:

$$microPrecision = \frac{\sum_{i=1}^{C} TP_i}{\sum_{i=1}^{C} TP_i + FP_i}$$

$$microRecall = \frac{\sum_{i=1}^{C} TP_i}{\sum_{i=1}^{C} TP_i + FN_i}$$

$$microF_1 = \frac{(2 \times microPrecision \times microRecall)}{(microPrecision + microRecall)}$$

- **Receiver Operating Characteristic (ROC) Curve:** In comparison to previously introduced evaluation metrics, the ROC curve has been used less frequently for the evaluation of the text categorization algorithms. As the name indicates, it is a curve shaped graph illustrates the performance of the categorization algorithm at all the categorization thresholds. The curve has two parameters, namely, True Positive Rate (TPR) and False Positive Rate (FPR). Moreover, the ROC curve is created by piloting the TPR against FPR at different threshold settings.

  The parameter of the curve is calculated as follows:

  - **True Positive Rate (TPR):** It is same as the recall calculation as follows:

$$TPR = \frac{TP}{TP + FN} \tag{3.8}$$

  - **False Positive Rate (FPR):** It is defines as follows:

$$TPR = \frac{FP}{FP + TN} \tag{3.9}$$

  A typical ROC curve[9] looks like as follows:

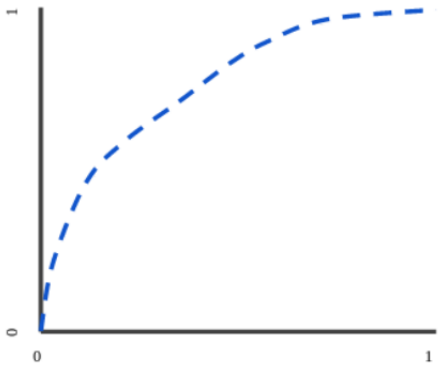


Figure 3.3: An example of a ROC curve. The curve plots the TPR (y-axis) vs FPR (x-axis) at different categorization thresholds.

[9] https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

3.7 Summary

This chapter has given a general overview of the arbitrary-length text categorization task as well as the short text categorization task, which is the main topic of the thesis. Although the provided overview has been rather general, the more specific details about short text categorization are presented in the remainder of the thesis. Moreover, the state-of-the-art approaches that are related to text categorization have been introduced in this chapter. These approaches utilize standard machine learning and neural networks methods. In addition, a standard machine learning based text categorization pipeline which includes text preprocessing, feature extraction (e.g., BoW and TF-IDF), training, testing steps, etc. has been presented. Finally, the evaluation metrics which aim to assess the performance of the text categorization models have also been presented.

4 Knowledge-Based Short Text Categorization

This chapter is one of the core chapters of this thesis. The chapter investigates two research questions defined in Section 1.2 (Research Question 1 and 2):

- How can a knowledge base be utilized to categorize short texts without requiring any labeled training data?
- How to learn the semantic representation of short texts and pre-defined categories with the help of a knowledge base?

In this regard, this chapter proposes a novel probabilistic model for Knowledge-Based Short Text Categorization (KBSTC), which is based on the following publication:

Türker, R., Zhang, L., Koutraki, M., & Sack, H. (2019). Knowledge-based short text categorization using entity and category embedding. The Semantic Web-16th International Conference, ESWC 2019, Portoro, Slovenia.

The proposed model does not require any labeled training data to categorize a short text, instead, it utilizes a knowledge base as an external source. KBSTC leverages entities and categories from the large knowledge base, which are further embedded into a common vector space, for which a new entity and category embedding model is proposed. Given a short text, its category (e.g., Business, Sports, etc.) can then be derived based on the entities mentioned in the text by exploiting semantic similarity between entities and categories. To validate the effectiveness of the proposed method, the experiments on two real-world datasets, i.e., AG News and Google Snippets have been conducted. The experimental results show that the proposed approach significantly outperforms the categorization approaches which do not require any labeled data, while it comes close to the results of the supervised approaches.

4.1 Introduction

Short text categorization is gaining more and more attention due to the availability of a huge number of text data, which includes search snippets, short messages as well as text data generated in social forums (Burel et al., 2017; Türker et al., 2018a; Türker et al., 2018b). Furthermore, people often voice their opinions on online platforms such as Twitter, Amazon, etc. in the form of a short text (Zeng et al., 2018). Hence, a large body of short text data is generated everyday. Although traditional text categorization methods perform well on long text such as news articles, yet, by considering the short text, most of them suffer from issues such as data sparsity and insufficient text length, which is no longer than 200 characters (Song et al., 2014). Furthermore, short texts are often malformed, i.e., the meaning of the texts is not explicitly expressed, and thus extracting meaningful features from short texts is a challenging task (Sakor et al., 2019). Moreover, simple text categorization approaches based on the bag-of-words (BOW) cannot properly represent short texts as the semantic similarity between single words is not taken into account (Wang et al., 2016b). Such methods only consider the word occurrences within the documents, they discard the sequential and semantic information of the words. Also, approaches that utilize word embeddings, which consider the semantic relation of the words, perform better when dealing with longer text. Because in the case of longer text the ambiguities can be resolved based on the provided context information within the given text. On the other hand, in the case of short text, where the available context is rather limited and each word obtains significant importance, such approaches often lead to inaccurate results.

Another characteristic of existing approaches is that they are designed in the paradigm of supervised learning. Therefore, they all require a significant amount of labeled training data. Manually labeling of such data can be a rather time-consuming and costly task. Especially, if the text to be labeled is of a specific scientific or technical domain, crowd-sourcing based labeling approaches do not work successfully and only expensive domain experts are able to fulfill the manual labeling task. Alternatively, semi-supervised text categorization approaches (Nigam et al., 2000; Xuan et al., 2017) have been proposed to reduce the labeling effort. These approaches mainly aim to perform text categorization by leveraging limited labeled data as well as unlabeled data. The key idea is improving the categorization performance by combining unlabeled data with a small amount of labeled data. Yet, due to the diversity of the documents in many applications, generating small training set for semi-supervised approaches still remains an expensive process (Li et al., 2016a). Moreover, (Singh et al., 2008) show that leveraging labeled data may not always provide gain on the performance. In fact, the performance of the semi-supervised learning approach might fall down due to the characteristics of the unlabeled data.

To overcome the requirement for labeled data, a number of *dataless text categorization* methods have been proposed (Song & Roth, 2014; Chang et al., 2008). These methods do not require any labeled data as a prerequisite. Instead, they rely on the semantic similarity

between a given document and a set of predefined categories to determine which category the given document belongs to. More specifically, documents and categories are represented in a common semantic space based on the words contained in the documents and category labels, which allows calculating a meaningful semantic similarity between documents and categories. The categorization process depends on this semantic similarity. On the other hand, (Druck et al., 2008; Chen et al., 2015a) proposed dataless categorization models which require manually designed seed words, i.e., representative keywords for each label. The models have provided promising categorization accuracy. However, the performance of such models is mainly affected by the selection of seed words. Furthermore, the most prominent and successful dataless categorization approaches are designed for long documents.

Motivated by the already mentioned challenges, this chapter proposes a novel probabilistic model for Knowledge-Based Short Text Categorization (KBSTC), which does not require any labeled training data. The proposed approach differs from the traditional text categorization approaches since it is designed for the task of *short* text categorization. It is able to capture the semantic relations between the entities represented in a short text and the predefined categories. In order to capture such semantic relations, this chapter also proposes a new network embedding technique which is capable of embedding both entities and categories from a large knowledge base into a common vector space. Finally, the category of the given text can be derived based on the semantic similarity between entities present in the given text and the set of predefined categories. The similarity is computed based on the vector representation of entities and categories.
Overall, the main contributions of this chapter are as follows:

- a new paradigm for short text categorization, based on a knowledge base;
- a probabilistic model for short text categorization;
- a new method of entity and category embedding;
- an evaluation using standard datasets for short text categorization.

4.2 Related Approaches

The aim of this work is to assign a category (e.g., *Business*, *Sports*, etc.) to a given short text by utilizing entity and category embeddings without requiring any labeled data. Thus, our work is mainly related to three prior studies: Short Text Categorization, Dataless Text Categorization as well as Entity and Category Embedding.

The state-of-the-art short text categorization approaches are presented in Section 3.2. Therefore, in this section, dataless text categorization approaches and embedding models that are relevant to the proposed entity and category embedding model are discussed.

4.2.1 Dataless Text Categorization

In order to address the problem of missing labeled data, (Chang et al., 2008) have introduced a dataless text categorization method by representing documents and category labels in a common semantic space. As a source, the online encyclopedia, Wikipedia was utilized supported with Explicit Semantic Analysis (ESA) (Gabrilovich et al., 2007) to quantify semantic relatedness between the labels to be assigned and the documents. As a result, it was shown that ESA is able to achieve better categorization results than the traditional BOW representations. The method demonstrated remarkable categorization performance without using any labeled data. In fact, the model obtained comparable results with the supervised methods which have utilized 100 labeled samples for training. Furthermore, (Song & Roth, 2014) proposed a dataless hierarchical text categorization by dividing the dataless categorization task into two steps. In the semantic similarity step, both labels and documents were represented in a common semantic space, which allows calculating semantic relatedness between documents and labels. In the bootstrapping step, the approach made use of a machine learning based categorization procedure with the aim of iteratively improving categorization accuracy. Similar to (Chang et al., 2008), the performance of the proposed approach is evaluated on the same datasets. Since the approach relies on the similarity between the label names and the texts, the representative label names are essential for the approach. Therefore, the authors expanded the original label names by adding more relevant descriptions. The experimental results come close to the result of the supervised approaches which trained on thousands of labeled data samples. The baselines are based on the traditional machine learning models, i.e., Naive Bayes (NB), Support Vector Machine (SVM), etc. with the features calculated based on the term frequency and the inverse document frequency to perform categorization.

In addition, Latent Dirichlet Allocation (LDA) was utilized for dataless text categorization (Li et al., 2016a; Hingmire et al., 2013). (Chen et al., 2015b) proposed an LDA based dataless categorization method referred to as Descriptive LDA (DescLDA). DescLDA is an extension of the standard LDA model, in which a *describing device (DD)* is applied. The DD is a model which helps to infer Dirichlet priors from categories/labels of the given dataset.

The aforementioned studies are designed for the categorization of long and well structured documents such as news articles. In contrast to these approaches, our proposed approach differs in two main aspects. First, all the mentioned studies were designed for the categorization of documents of arbitrary length. However, the main purpose of KBSTC is to categorize short text documents without the necessity of labeled training data. Second, none of the mentioned approaches did make use of the entities present in a short text document. To represent a document, all the mentioned approaches consider the words contained in the document and they ignore the entities. However, entities carry much more information than words.

4.2.2 Entity and Category Embeddings

Section 2.1.6 and Section 2.1.7 discuss the technical details of embedding models that are presented in this section. Hence, in this section, the technical details are omitted.

To be able to capture the semantic relation between the entities present in short texts and the predefined categories, the embedding of entities and categories into a common vector space is crucial for KBSTC. There exist several embedding models, which can be employed to generate entity and category embeddings. For instance, RDF2Vec (Ristoski & Paulheim, 2016), DeepWalk, etc. RDF2Vec aims to learn the distributed representation of the vertices in an RDF graph. Further, it treats each type of vertices and edges equally. However, the connection between the vertices in a graph may vary. In other words, some entities might strongly related to each other than the other entities. Yet, such types of relations are ignored in RDF2Vec, instead, each edge between the vertices (e.g., entities) is treated equally. On the other hand, similar to RDF2Vec, DeepWalk (Perozzi et al., 2014) adopts a language modeling approach, i.e., Skip-gram to learn the representation of vertices in a large network. Such approaches are also designed for homogeneous networks in which there exists only one type of vertice and edge.

(Li et al., 2016d) have proposed the state of the art joint entity and category embedding models, namely, the Category Embedding (CE) model, and the Hierarchical Category Embedding (HCE) model. The first model CE learns simultaneously representations of entities and their associated categories based on the Skip-gram model (Mikolov et al., 2013a). In order to enhance the embedding quality, the authors extended the CE model and integrated the category hierarchy structure into the embedding space, which yielded the HCE model. The authors applied both models to the dateless categorization task and reported that HCE can represent semantic relatedness between entities and categories better than the CE model. Moreover, (Tang et al., 2015a) have proposed a model called PTE which uses labeled data and word co-occurrence information to build a heterogeneous text network, where multiple types of vertices exist and then apply the proposed algorithm to learn the embedding of text.

As stated before, capturing the meaningful semantic relatedness between the entities and categories from large knowledge bases is essential for KBSTC. Except for PTE, all the presented embedding models are designed for homogeneous networks where each vertex and edge are treated equally. However, it is assumed that the entity-entity, entity-category, category-category relation should be treated differently while computing the entity and category vectors. Therefore, inspired by PTE, a heterogeneous network entity, and category embedding model is proposed in this chapter. The model first constructs a weighted network of entities and categories, and then jointly learns their embeddings from the network. The proposed embedding model is presented in Section 4.6.

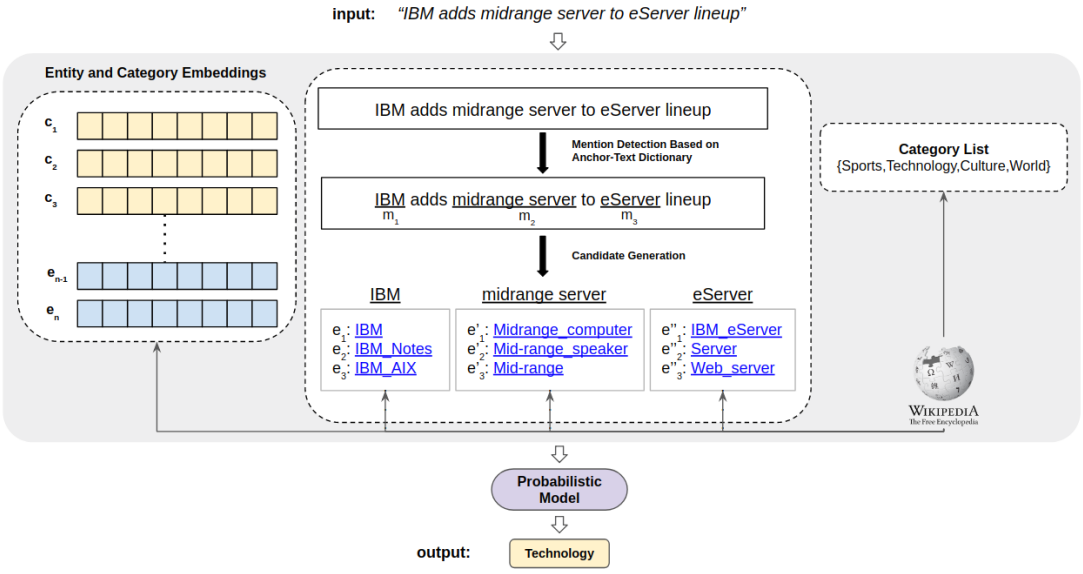


Figure 4.1: The workflow of the proposed KBSTC approach (best viewed in color)

4.3 Preliminaries and Overview

This section introduces the preliminaries and provides a formal definition of the Knowledge-Based Short Text Categorization (KBSTC) task, followed by the description of the proposed probabilistic approach for KBSTC.

Preliminaries. Given a knowledge base KB containing a set of entities $E = \{e_1, e_2, .., e_n\}$ and a set of hierarchically related categories $C = \{c_1, c_2, .., c_m\}$, we model KB as a graph $G_{KB} = (V, R)$ with $V = E \cup C$ as the set of vertices and $R = R_{EE} \cup R_{EC} \cup R_{CC}$ as the set of edges of the form (v_i, v_j) reflecting various relationships between the vertices v_i and v_j, where each edge in R_{EE} with $v_i, v_j \in E$ represents an entity-entity relation, each edge in R_{EC} with $v_i \in E$ and $v_j \in C$ represents an entity-category association, and each edge in R_{CC} with $v_i, v_j \in C$ reflects the category hierarchy.

In this work, as the knowledge base, we utilize Wikipedia, which is a collaboratively edited knowledge source, contains more than 5 million articles, where each article is associated with one or more categories. Moreover, Wikipedia contains a rich category hierarchy. In Wikipedia each article and each category page are considered as an entity in E and a category in C, respectively. In addition, each relationship (v_i, v_j) between the pair of vertices v_i and v_j are extracted from Wikipedia and the following rule applies:

- $(v_i, v_j) \in R_{EE}$ if and only if $v_i, v_j \in E$ and there is a link from the article v_i to the article v_j in Wikipedia,
- $(v_i, v_j) \in R_{EC}$ if and only if $v_i \in E, v_j \in C$ and the article v_i has the associated category v_j in Wikipedia, and
- $(v_i, v_j) \in R_{CC}$ if and only if $v_i, v_j \in C$ and v_i is subcategory of v_j in Wikipedia.

Definition (KBSTC task). Given an input short text t that contains a set of entities $E_t \subseteq E$ as well as a set of predefined categories $C' \subseteq C$ (from the underlying knowledge base KB), the output of the KBSTC task is the most relevant category $c_i \in C'$ for the given short text t, i.e., we compute the category function $f_{cat}(t) = c_i$, where $c_i \in C'$.

KBSTC Overview. The general workflow of KBSTC is shown in Figure 5.1. In the first step, each entity mention present in a given short text t is detected. In order to detect entity mentions, first all n-grams from the input text are gathered and then the extracted n-grams matching surface forms of entities (based on the Anchor-Text dictionary) are selected as entity mentions. Next, for each mention, a set of candidate entities are generated based on a prefabricated Anchor-Text Dictionary, which contains all mentions and their corresponding Wikipedia entities. To construct the Anchor-Text Dictionary, all the anchor texts of hyperlinks in Wikipedia pointing to any Wikipedia articles are extracted, whereby the anchor texts serve as mentions and the links refer to the corresponding entities.
Given the short text t as "`IBM adds midrange server to eServer lineup`", the detected mentions are "`IBM`", "`midrange server`" and "`eServer`".
For each mention a set of entities are generated;
{"`IBM`", "`IBM_Notes`", "`IBM_AIX`",..},
{"`Midrange_computer`", "`Mid-range_speaker`", "`Mid-range`",..} and
{"`IBM_eServer`", "`Server`", "`Web_server`",..}.
Likewise the predefined categories,
`C'={Sports, Technology, Culture, World}`, are mapped to Wikipedia categories. Finally, applying the proposed probabilistic model (cf. Section 4.4) by utilizing the entity and category embeddings that have been precomputed from Wikipedia (cf. Section 4.6), the output of the KBSTC task is the semantically most relevant category for the entities present in t. Thereby, in the given example the category `Technology` should be determined.
Note that in this work we have utilized Wikipedia as a KB. However, KBSTC is applicable to any arbitrary domain as long as there exists a KB providing domain-specific entities and categories.

4.4 Probabilistic Approach

The KBSTC aims to assign the most relevant category for a given short text, and thus its task is formalized as estimating the probability of $P(c|t)$ of each predefined category c and an input short text t. The result of this probability estimation can be considered as a score for each category. Therefore, the most relevant category c for a given text t should maximize the probability $P(c|t)$. Based on Bayes' theorem, the probability $P(c|t)$ can be rewritten as follows:

$$P(c|t) = \frac{P(c,t)}{P(t)} \propto P(c,t)\,, \tag{4.1}$$

where the denominator $P(t)$ can be ignored as it has no impact on the ranking of the categories.

To facilitate the following discussion, we first introduce the concepts of *mention* and *context*. For an input text t, a *mention* is a term in t that can refer to an entity e and the *context* of e is the set of all other mentions in t except the one for e. For each candidate entity e contained in t, the input text t can be decomposed into the mention and context of e, denoted by m_e and C_e, respectively. For example, given the entity e as `IBM`, the input text `"IBM adds midrange server to eServer lineup."` can be decomposed into a mention m_e as `"IBM"` and a context C_e as `{"midrange server", "eServer"}`, where `"midrange server"` and `"eServer"` can refer to the context entities `Midrange_computer` and `IBM_eServer`, respectively.

Based on the above introduced concepts, the joint probability $P(c,t)$ is given as follows:

$$\begin{aligned} P(c,t) &= \sum_{e \in E_t} P(e,c,t) && (4.2)\\ &= \sum_{e \in E_t} P(e,c,m_e,C_e) \\ &= \sum_{e \in E_t} P(e)P(c|e)P(m_e|e,c)P(C_e|e,c) && (4.3)\\ &= \sum_{e \in E_t} P(e)P(c|e)P(m_e|e)P(C_e|e)\,, && (4.4) \end{aligned}$$

where E_t represents the set of all possible entities contained in the input text t. We assume that in Equation (4.3) m_e and C_e are conditionally independent given e, in Equation (4.4) m_e and C_e are conditionally independent of c given e. The intuition behind these assumptions is that a mention m_e and a context C_e only rely on the entity e which refers to and co-occurs with, such that once the entity e is fixed, m_e and C_e can be considered as conditionally independent. The main problem is then to estimate each probability in Equation (4.4), which will be discussed in the next section.

4.5 Model Parameter Estimation

The proposed probabilistic model has four main components, i.e., $P(e)$, $P(c|e)$, $P(m_e|e)$ and $P(C_e|e)$. This section provides the estimation of each component in detail.

Entity Popularity. The probability $P(e)$ captures the popularity of the entity e. Here, we simply apply a uniform distribution to calculate $P(e)$ as follows:

$$P(e) = \frac{1}{N} \tag{4.5}$$

where N is the total number of entities in the KB.

Entity-Category Relatedness. The probability $P(c|e)$ models the relatedness between an entity e and a category c. With the pre-built entity and category embeddings (cf. Section 4.6), there are two cases to consider for estimating $P(c|e)$. Firstly, when the entity e is directly associated with the category, denoted by c_{a_e}, in KB, i.e., e appears in some Wikipedia articles that have associated category c_{a_e}, the probability $P(c_{a_e}|e)$ can be approximated based on similarity as

$$P(c_{a_e}|e) = \frac{sim(c_{a_e}, e)}{\sum\limits_{c'_{a_e} \in C_{a_e}} sim(c'_{a_e}, e)}, \tag{4.6}$$

where C_{a_e} is the set of categories that are directly associated with e, and $sim(c_{a_e}, e)$ denotes the cosine similarity between the vectors of the category c_{a_e} and the entity e in the embedding space. Secondly, in case where the entity e is not directly associated with the category c, the hierarchical structure of categories in KB is considered. More specifically, the categories in C_{a_e} are incorporated into the estimation of the probability $P(c|e)$ as follows:

$$\begin{aligned} P(c|e) &= \sum_{c_{a_e} \in C_{a_e}} P(c_{a_e}, c|e) \\ &= \sum_{c_{a_e} \in C_{a_e}} P(c_{a_e}|e) P(c|c_{a_e}, e) \\ &= \sum_{c_{a_e} \in C_{a_e}} P(c_{a_e}|e) P(c|c_{a_e}), \end{aligned} \tag{4.7}$$

where we consider that e is related to c only through its directly associated category c_{a_e}, such that once c_{a_e} is given, e and c are conditionally independent.

In Equation (4.7), the probability $P(c_{a_e}|e)$ then can be simply calculated based on Equation (4.6) and the probability $P(c|c_{a_e})$ that captures the hierarchical category structure, is estimated as follows:

$$P(c|c_{a_e}) = \begin{cases} \frac{1}{|A_{c_{a_e}}|} & \text{if } c \text{ is an ancestor of } c_{a_e}, \\ 0 & \text{otherwise}, \end{cases} \tag{4.8}$$

where $A_{c_{a_e}}$ is the set of ancestor categories of c_{a_e}, which can be obtained by using the category hierarchy in KB.

Mention-Entity Association. The probability $P(m_e|e)$ of observing a mention m_e given the entity e is calculated based on the *Anchor-Text Dictionary* as follows:

$$P(m_e|e) = \frac{count(m_e, e)}{\sum\limits_{m'_e \in M_e} count(m'_e, e)}, \tag{4.9}$$

where $count(m_e, e)$ denotes the number of links using m_e as anchor text pointing to e as the destination, and M_e is the set of all mentions that can refer to e.

Entity-Context Relatedness. The probability $P(C_e|e)$ models the relatedness between the entity e and its context C_e that consists of all the other mentions in the input text t except m_e. Each mention in C_e refers to a context entity e_c from the given KB. The probability $P(C_e|e)$ can be calculated as follows:

$$\begin{aligned} P(C_e|e) &= \sum_{e_c \in E_{C_e}} P(e_c, C_e|e) \\ &= \sum_{e_c \in E_{C_e}} P(e_c|e) P(C_e|e_c, e) \\ &= \sum_{e_c \in E_{C_e}} P(e_c|e) P(C_e|e_c) \qquad (4.10) \\ &= \sum_{e_c \in E_{C_e}} P(e_c|e) P(m_{e_c}|e_c), \qquad (4.11) \end{aligned}$$

where E_{C_e} denotes the set of entities that can be referred to by the mentions in C_e. In Equation (4.10), the context C_e is conditionally independent of e given the context entity e_c, and in Equation (4.11) e_c is assumed to be only related to its corresponding mention $m_{e_c} \in C_e$ such that the other mentions in C_e can be ignored.

Similar to $P(c_{a_e}|e)$ (cf. Equation (4.6)), the probability $P(e_c|e)$ in Equation (4.11) can also be estimated based on the pre-built entity and category embeddings. Let $sim(e_c, e)$ be the cosine similarity between the entity vectors of e_c and e. Then the probability $P(e_c|e)$ can be calculated as follows:

$$P(e_c|e) = \frac{sim(e_c, e)}{\sum\limits_{e' \in E} sim(e', e)}, \tag{4.12}$$

where E is the set of all entities in KB. In addition, the probability $P(m_{e_c}|e_c)$ in Equation (4.11) can be calculated based on Equation (4.9).

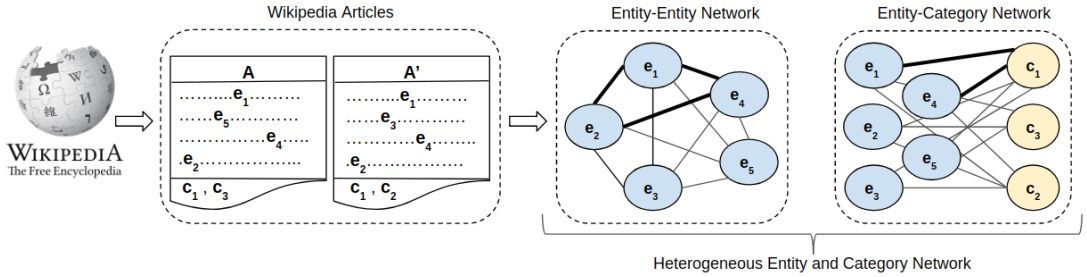


Figure 4.2: The Entity Category Network Construction (best viewed in color)

4.6 Joint Entity And Category Embedding

This section provides a description of the proposed embedding method that embeds entities and categories into a common vector space by integrating knowledge from a large knowledge base. The generated embedding model is originated from PTE (Tang et al., 2015a) (cf. Section 2.1.7) which is a heterogeneous network embedding technology. Firstly, the entity-category network construction in Section 4.6.1 is presented, and subsequently, the joint entity and category embedding model is presented in Section 4.6.2.

4.6.1 Network Construction

The semantic representation of a text is crucial for text categorization algorithms. Traditional text categorization approaches use words to generate a feature set, and adopt machine learning algorithms. However, it is assumed that in short text words tend to be ambiguous, however entities carry much more information (Wang et al., 2016a). Therefore, in this study entities and their semantic relation with categories are used as a main feature to categorize the given short text.

To calculate the meaningful semantic relatedness, the proper semantic representation of entities and categories in a common vector space is essential for the proposed approach. For this reason, the entity and category network is firstly constructed, which will be later utilized to generate the entity and category embeddings.

Figure 4.2 depicts the process of the entity-entity and entity-category network construction, where the *heterogeneous network* consists of both entity vertices and category vertices, and accordingly two types of edges, i.e., edges between two entity vertices and edges between an entity vertex and a category vertex. The weights of the edges between different vertices are crucial due to their significant impact on the embedding model (cf. Section 4.6.2). By leveraging the hyperlink structure in Wikipedia, we propose a method to calculate the edge weights for both entity-entity and entity-category networks:

- **Weights for Entity-Entity Edges.** In order to explain the weight calculation, firstly the concept of *linked entity* has to be defined. The hyperlinks that are present in an arbitrary Wikipedia article and refer to another Wikipedia article are called linked entities. The weight of an edge between an entity-entity pair is the number of Wikipedia articles where both entities appear as a linked entity.

- **Weights for Entity-Category Edges.** The weight of an edge between an entity-category pair is the number of Wikipedia articles where the entity appears as a linked entity and simultaneously the corresponding article containing the linked entity belongs to the category in Wikipedia.

As shown in Figure 4.2, the linked entities and the associated categories for each Wikipedia article are used to generate the entity-entity and the entity-category edges. The edges of (e_1, e_2), (e_1, e_4), (e_2, e_4), (e_1, c_1) and (e_4, c_1) are thicker due to their higher co-occurrence frequency.

4.6.2 Embedding Model

As introduced before, the overall heterogeneous network consists of two homogeneous networks, i.e., the entity-entity and entity-category networks. Similar to PTE (Tang et al., 2015a), to embed each of these networks, our proposed embedding model aims to capture the second-order proximity (cf. Section 2.1.7). More specifically, the second-order proximity is calculated between two vertices in a network by considering their common (shared) vertices. Therefore, according to the second order proximity vertices that share many same neighbors should be placed closely in the vector space.

To model the second-order proximity of a homogeneous network, for each edge (v_i, v_j), the conditional probability $p(v_j|v_i)$ is defined as follows (Tang et al., 2015b):

$$p(v_j|v_i) = \frac{exp(-\vec{u}_j^T \cdot \vec{u}_i)}{\sum\limits_{v_k \in V} exp(-\vec{u}_k^T \cdot \vec{u}_i)}\,, \tag{4.13}$$

where V is the set of vertices connected with v_i in the network, $\vec{u}_i$, $\vec{u}_j$ and $\vec{u}_k$ are the vectors of vertices v_i, v_j and v_k, respectively. The empirical probability of $p(v_j|v_i)$ can be defined as:

$$\hat{p}(v_j|v_i) = \frac{w_{ij}}{d_i}, \tag{4.14}$$

where d_i is the out-degree of v_i and w_{ij} is the weight of the edge (v_i, v_j).

In order to preserve the second-order proximity, the conditional distribution $p(v_j|v_i)$ is made close to $\hat{p}(v_j|v_i)$ based on the KL-divergence over the entire set of vertices in the

network, such that the model minimizes the following objective function:

$$O_{homo} = -\sum_{(v_i,v_j)\in E} w_{ij} \log\left(p(v_j|v_i)\right), \tag{4.15}$$

The embedding of the individual entity-entity and entity-category networks can be learned by utilizing the second-order proximity between vertices. However, our goal is to simultaneously learn the embedding of the constructed heterogeneous network by minimizing the following objective function:

$$O_{heter} = O_{ee} + O_{ec}, \tag{4.16}$$

where O_{ee} and O_{ec} are the objective functions defined in Equation (4.15) for the homogeneous entity-entity and entity-category networks, respectively. To optimize the objective function in Equation (4.16), we adopt a similar approach as described in (Tang et al., 2015a), where all the edges are firstly collected from these two homogeneous networks as two sets, one for entity-entity edges and the other for entity-category edges, and then in each training iteration, edges are sampled from both sets to update the model. Readers can refer to (Tang et al., 2015a; Tang et al., 2015b), for the detailed optimization process.

4.7 Experiments

This section provides a detailed description of the datasets and the baselines for evaluating the proposed approach, followed by the experimental results as well as a comparison to the existing state-of-the-art approaches in the related areas.

4.7.1 Datasets

To validate the effectiveness of the proposed method experiments are conducted on two different benchmarks: AG News and Google Snippets.

- **AG News (AG)[1]:** This dataset is adopted from (Zhang & LeCun, 2015), which contains both titles and short descriptions (usually one sentence) of news articles. News data is one of the largest parts of the available text data on the Web (Zhang & LeCun, 2015). Therefore, it is an important task to evaluate a text categorization algorithm with this type of data. The data distribution of the training and test datasets is shown in Table 4.1. In our experiments, the dataset has two versions, where one contains only titles and the other contains both titles and descriptions. The total number of entities and the

[1] http://goo.gl/JyCnZq

Category	#Train	#Test
Business	30,000	1,900
Sports		
World		
Sci/Tech		
Total	120,000	7,600

Table 4.1: The data distribution of the AG News dataset

Category	#Train	#Test
Business	1200	300
Computers	1200	300
Cult-arts-entertainment	1880	330
Education-Science	2360	300
Engineering	220	150
Health	880	300
Politics-Society	1200	300
Sports	1120	300
Total	10,060	2,280

Table 4.2: The data distribution of the Google Snippets dataset

average number of entities and words per text in the test datasets are shown in Table 4.3.

- **Google Snippets (Snippets)[2]:** This is a well-known dataset for short text categorization, which was introduced in (Phan et al., 2008) as a short text categorization benchmark. The dataset contains short snippets from Google search results and its distribution is shown in Table 4.2. Several short text categorization approaches (Chen et al., 2011; Dai et al., 2013; Wang et al., 2016b; Bouaziz et al., 2016) have used this dataset to asses the performance of the proposed models. As shown in Table 4.3, the test dataset has an average of 8.9 entities and an average of 17.97 words in each snippet.

Dataset	Avg. #Ent	Avg. #Word
AG News (Title)	3.21	7.14
AG News (Title+Description)	11.83	38.65
Google Snippets	8.90	17.97

Table 4.3: The statistical analysis of the test datasets

As the focus of this work is the KBSTC task, where the goal is to derive the most relevant category from the knowledge base for a given short text. Hence, to adapt these datasets the labels/categories have been aligned with the categories in the underlying knowledge base.

[2] `http://jwebpro.sourceforge.net/data-web-snippets.tar.gz`

Model	AG (title)	AG (title+description)	Snippets
Dataless ESA	53.5	64.1	48.5
Dataless Word2Vec	49.5	52.7	52.4
NB+TF-IDF	86.6	90.2	64.4
SVM+TF-IDF	**87.6**	**91.9**	69.1
LR+TF-IDF	87.1	91.7	63.6
KBSTC+Our Embedding	67.9	80.5	**72.0**

Table 4.4: The categorization accuracy of KBSTC against baselines (%)

More specifically, each label/category in these datasets are manually mapped to its corresponding Wikipedia category, e.g., the category *Sports* from the AG dataset is mapped to the Wikipedia category *Sports*[3]. Furthermore, as KBSTC does not depend on any training/labeled data, the training datasets of AG and Snippets are only used for the training of the supervised baseline methods. Lastly, to measure the performance of KBSTC, the categorization accuracy (the ratio of correctly categorized data over all the test data (cf. Section 3.6)) have been used.

4.7.2 Baselines

To demonstrate the performance of the KBSTC approach, the following dataless and supervised categorization methods have been selected as baselines:

- **Dataless ESA and Dataless Word2Vec:** As described in Section 4.2, the dataless approaches do not require any labeled data or training phase, therefore, they can be considered as the most similar approaches to KBSTC. Two variants of the state-of-the-art dataless approach (Song & Roth, 2014) are considered as baselines, which are based on ESA (Gabrilovich et al., 2007) and Word2Vec (Mikolov et al., 2013b), respectively.

- **NB, SVM, LR:** Additional baselines include the traditional supervised models (cf. Section 3.5), i.e., Naive Bayes (NB), Support Vector Machine (SVM) and Logistic Regression (LR), with the features calculated based on the term frequency and the inverse document frequency (TF-IDF).

4.7.3 Evaluation of KBSTC

Table 4.4 shows that the accuracy of the proposed probabilistic KBSTC approach (cf. Section 4.4) based on our entity and category embedding model (cf. Section 4.6) in comparison

[3] `https://en.wikipedia.org/wiki/Category:Sports`

to the baselines on the AG and Snippets datasets.

It is observed that the KBSTC approach considerably outperforms the dataless categorization approaches. While Dataless ESA and Dataless Word2Vec have been assessed with two different news datasets, namely, 20Newsgroups Data, RCV1 Data, which consist of long news articles, and achieved promising results in (Song & Roth, 2014), they cannot perform well with the short text due to the data sparsity problem. In other words, dataless approaches construct the document vectors by utilizing the words present in documents. However, short text documents are often malformed, i.e., the meaning of the sentence is not explicitly expressed, therefore, using only words for the categorization task may lead to inaccurate results. Furthermore, for Dataless approaches the label names are crucial to compute the most accurate vector representation of the labels, therefore, in (Song & Roth, 2014), the labels have been enriched by their description before obtaining their vector representation. However, the datasets (AG News and Google Snippets) that have been used in these experiments do not contain any label description. As shown in Table 4.1 and Table 4.2 the labels are only words. Lack of label descriptions could be another reason why the dataless approaches have performed poorly on these datasets however, KBSTC is capable of performing short text categorization without requiring any additional description of labels.

Remarkably, KBSTC outperforms all the baselines on the Snippets dataset, however, all supervised approaches outperform KBSTC on the AG dataset. Especially, SVM as explained in detail in Section 3.5 is one of the most extensively employed standard baselines for text categorization. Even, today, several sophisticated deep neural network approaches are struggling to outperform SVM. The reason of the differences among the obtained accuracy can be attributed to the different characteristics of the two datasets. AG is a larger dataset with more training samples (cf. Table 4.1) in comparison to Snippets (cf. Table 4.2). Further, the samples of the Snippets dataset are not equally distributed among the categories, hence, it is an imbalanced dataset. Moreover, the AG dataset provides only 4 different categories in comparison to 8 categories of the Snippets dataset. Those differences might be the reason for the significant decrease in accuracy for the supervised approaches on the Snippets dataset in comparison to the AG dataset. This could be an indicator that the size of the training data and the number of classes make a real impact on the categorization accuracy for the supervised approaches. Since KBSTC does not require or use any labeled data, the number of the available training samples has no impact on its accuracy.

Regarding the results of KBSTC, the AG (title+description) dataset yields better accuracy than the Snippets dataset, which in turn, results in better accuracy than the AG (title) dataset. The reason might be found in the nature of the datasets. As shown in Table 4.3, the average number of entities per text in AG (title+description) is greater than Snippets, followed by AG (title). AG (title+description) provides richer context, i.e., larger number of entities, therefore, KBSTC obtains the best accuracy with this dataset. Often a richer context with more entities can make the categorization more accurate.

Overall, the results in Table 4.4 have demonstrated that for short text categorization,

Model	AG (title)	AG (title+description)	Snippets
KBSTC+HCE	67.0	79.6	**72.3**
KBSTC+DeepWalk	57.1	74.2	64.3
KBSTC+RDF2Vec	62.7	77.5	68.2
KBSTC+Our Embedding	**67.9**	**80.5**	72.0

Table 4.5: The categorization accuracy of KBSTC with different embedding models (%)

KBSTC achieves a high accuracy without requiring any labeled data, a time-consuming training phase, or a cumbersome parameter tuning step.

4.7.4 Evaluation of Entity and Category Embedding

To assess the quality of the proposed entity and category embedding model (cf. Section 4.6), we have compared it with the following embedding models:

- **HCE** (Li et al., 2016d) is a joint embedding of entities and hierarchical categories. The model learns simultaneously the representation of entities and their associated categories from large knowledge bases. Although the authors proposed two flavors of the entity and category embedding, namely, CE and HCE (cf. Section 2.1.7). The obtained experimental result demonstrates that HCE can capture better semantic relations between the entities and categories. Therefore, in these experiments, HCE has been considered for the comparison. Further, among the baseline embedding models, HCE is the most similar embedding model to the proposed embedding approach.

- **DeepWalk** (Perozzi et al., 2014) uses a language modeling approach (Mikolov et al., 2013a) to learn the distributed representation of vertices in large social networks such as Flickr and YouTube. The algorithm can only be employed by the networks with binary edges.

- **RDF2Vec** (Ristoski & Paulheim, 2016) adopts a language modeling approach (Mikolov et al., 2013a) to learn the representation of vertices in a directed RDF graph.

While the Wikipedia entity and category embeddings generated by HCE can be directly used, DeepWalk has been applied on the network constructed using Wikipedia and RDF2Vec has been applied on the RDF graph of DBpedia to obtain the needed embeddings. Then, these embeddings are integrated into KBSTC to compute the entity-category and entity-context relatedness (see Equation (4.6) and Equation (4.12)).

The results of KBSTC with different embedding models are shown in Table 4.5. The proposed entity and category embedding model outperforms all other embedding models for the KBSTC task on the AG dataset, while HCE performs slightly better than our model on the Snippets dataset.

HCE is a more specific embedding model that has been designed to learn the representation of entities and their associated categories from Wikipedia. Moreover, it also considers the hierarchy between the categories. Therefore, it is expected that the semantic relatedness between entities and categories in the HCE model would be captured better than DeepWalk and RDF2Vec, as it has been designed especially for this task. However, HCE is not flexible to be adapted to other networks. In other words, HCE can be trained only with the network which contains entities and hierarchically related categories. Further, including more elements such as words into the embedding space is not straightforward. In contrast to HCE, our model can deal with more general networks. For example, with words and word-category relations as an additional type of vertices and edges in the heterogeneous network described in Sec. 4.6.1, it is straightforward to adapt our embedding model by involving a new object function O_{wc} into Eq. (4.16), which is considered as our future work.

On the other hand, although DeepWalk and RDF2Vec aim to learn the representation of vertices in general networks and RDF graphs, respectively, they have been either designed for homogeneous networks or treated each type of vertices and edges in an RDF graph equally. The experimental results also indicate that our embedding model enables to capture a better semantic representation of vertices by taking into account different types of networks, i.e., the entity-entity and entity-category networks.

4.7.5 Evaluation of Entity Linking

Dataset	#Doc	Avg-Ent	Avg-Word
DBpedia Spotlight	58	5.69	32
N^3 RSS-500	500	1.18	34

Table 4.6: The statistical analysis of the entity linking benchmarks

As discussed in Section 4.4, the first step of KBSTC is to detect entity mentions in a given short text and then for each mention to generate a candidate list of entities based on the anchor text dictionary, which is employed to determine the most relevant category for the input text based on the proposed probabilistic approach. An alternative way could be to first use an existing entity linking (EL) system to obtain the referent entity for each mention and then based on that derive the category of the input short text. The reason we did not adopt the latter solution is that most of the existing EL systems rely on the rich context of the input text for the collective inference to boost the overall EL performance. However, due to the lack of

such context in a short text, existing EL systems might not perform well in our case, i.e., the correct entities in the input short text cannot be found, which plays a vital role in the KBSTC approach.

Instead of directly using an existing EL system, our probabilistic approach actually involves an internal step of EL for the input short text t, where the main difference is that we consider a list of candidate entities for each mention. The output is a set of possible entities E_t present in t with the confidence score of each entity $e \in E_t$ as $P(e)P(m_e|e)P(C_e|e)$ (cf. Equation (4.4)), where $P(e)$ captures the popularity of e, $P(m_e|e)$ and $P(C_e|e)$ reflect the likelihood of observing the mention m_e and the context C_e given e. By incorporating the confidence score of each $e \in E_t$ and its relatedness to each predefined category c, represented by $P(c|e)$, we can compute the final joint probability $P(c, t)$ to determine the most relevant category for t (see Eq. (4.4))

Methods	Spotlight	N^3 RSS-500
$P(e\|m)$	0.69	0.64
AIDA	0.25	0.45
AGDISTS	0.27	**0.66**
Babelfly	0.52	0.44
DBpedia Spotlight	**0.71**	0.20

Table 4.7: The comparison of Anchor Text Dictionary with EL Systems Micro F1 Results

To evaluate the effectiveness of the EL step in our approach, the experiments have been conducted on two datasets from the general entity linking benchmark GERBIL (Usbeck et al., 2015):

- **DBpedia Spotlight:** The dataset is released in (Mendes et al., 2011). The dataset contains relatively short text samples, named and nominal entities present in the samples are annotated.

- N^3 **RSS-500:** This dataset is one of the N^3 datasets (Röder et al., 2014). Similar to the DBpedia Spotlight dataset, N^3 RSS-500 consists of short texts which are annotated by the domain experts.

We have chosen these two datasets for the EL evaluation, because among the other EL benchmarks, these are the ones, which contain only short text, similar to our test datasets (cf. Table 4.6). To make our EL method be comparable with existing EL systems, in the experiments we also generate one single entity for each mention, which maximizes the confidence score,

computed by $P(e)P(m_e|e)P(C_e|e)$. The results of Micro F1 for various EL systems and our method are shown in Table 4.7. It is observed that our EL method achieves promising results for both datasets, which are very close to the best results yielded by the state-of-the-art EL systems. More importantly, because of the insufficient context of short text required by the collective inference for EL, it is difficult to provide the correct referent entity for each mention in many cases, such that our EL method used in KBSTC takes into account a list of candidate entities with their confidence scores for each mention.

4.7.6 Using Wikipedia as a Training Set

To further demonstrate the effectiveness of the proposed KBSTC approach, an additional experiment has been conducted. The results in Table 4.4 indicate that supervised methods can perform well in case of the existence of a sufficient amount of training data. However, in a real-world scenario, the labeled data might not be available and this is the case most of the time. Further, labeling a dataset is an expensive and time-consuming task. An alternative solution to the expensive manual process of compiling a labeled training dataset would be to automatically extract the training data from existing publicly available sources such as Wikipedia.

Method	AG (title+description)	Google Snippets
SVM+TF-IDF	59.9	53.9
KBSTC	**80.5**	**72.0**

Table 4.8: The categorization accuracy of KBSTC against a traditional categorization model, which is trained on the Wikipedia dataset and tested on AG and Snippets (%)

To do so, Wikipedia which contains more than 5 million articles has been leveraged. Each article in Wikipedia is associated with one or more categories. To generate the training data, for each category from the two datasets (AG and Snippets), training samples have to be assembled. For this purpose, Wikipedia articles associated with the corresponding categories (or their subcategories) are first collected. In other words, if the dataset category is *Technology* then all the Wikipedia articles associated with the category *Technology* are collected. Thereafter, 10,000 Wikipedia articles are then randomly selected as training data per category, which constitute the training datasets for AG and Snippets. Since SVM achieved the best results among the supervised approaches (cf. Table 4.4), two SVMs are trained with the generated training data for AG and Snippets, respectively. In the experiments, we have used the original test datasets from AG and Snippets for evaluating the trained SVMs.

The results are shown in Table 4.8, which indicate that the KBSTC approach achieved higher accuracy in comparison to the SVMs. More interesting, the same approach (SVM+TF-

IDF) trained with the AG and Snippets datasets achieved the accuracy scores of 91.9% and 69.1% (see Table 4.4), while it only achieved the accuracy scores of 59.9% and 53.9% when trained with the collected Wikipedia articles. This provides us some insights that it might not be suitable to directly use Wikipedia as the training datasets for supervised approaches and also serves as the motivation of the KBTSC approach proposed in this work.

From these results, it can be concluded that not only the size of the training set is important for the categorization performance but also the nature and characteristics of training data play an important role in the performance of the categorization model. In other words, train and test datasets should have similar characteristics in order to obtain reasonable accuracy from the supervised categorization approaches. This fact indicates that categorization accuracy highly depends on the nature of the chosen training dataset. However, the performance of KBSTC does not depend on the availability of any labeled data, as KBSTC does not have any training phase.

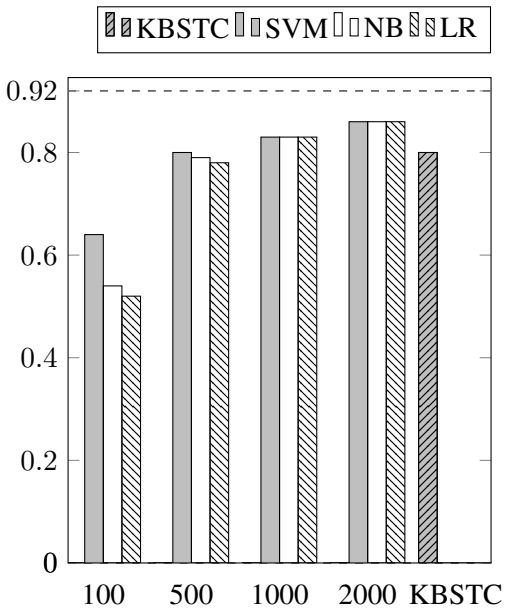


Figure 4.3: The performance of the supervised approaches for different training set sizes sampled from the AG News dataset. The x axis corresponds to the number of training samples, the y axis corresponds to the achieved accuracy score. The dashed line represents the best accuracy score 92.4%, achieved by n-gram-TF-IDF+MLR by using all the AG News training set.

4.7.7 Partitioning the Training Data

As stated before, the performance of the supervised approaches highly depends on the size and the characteristics of the training data as well as the parameters of the model. In the previous experiments (cf. Section 4.7.6), it has been proved that the nature of the training data plays a vital role in the performance of the categorization method. To additionally show the impact of the size of the labeled data on the text categorization task, a further experiment has been conducted.

Here, four smaller training datasets from the AG News training dataset have been randomly sampled. The new datasets are of size 100, 200, 1000, and 2000 texts. For each sampled datasets TF-IDF has been calculated as a feature and three different supervised categorizers (i.e. Support Vector Machine (SVM), Logistic Regression (LR), and Naive Bayes (NB)) have been trained on these features. The complete AG news test dataset has been considered for testing. The experimental results of the accuracy scores are depicted in Figure 4.3. Although with the entire dataset supervised approaches seem to perform well (cf. Table 4.4), with fewer training samples (e.g., 100, 500 samples) SVM, NB and LR perform poorly. In fact, it should be noted that manual labeling of 100 samples is a labor-intensive and time-consuming task. However, it is clear that by increasing the size of the training data the accuracy of the supervised approaches improves. The results suggest that the size of the training dataset has a huge impact on the accuracy of the supervised methods. However, KBSTC performs solid and stable and achieves adequate accuracy without requiring any labeled data.

4.8 Summary and Conclusion

This chapter has investigated two research questions (cf. Section 1.2 (Research Question 1 and 2)):

- How can a knowledge base be utilized to categorize short texts without requiring any labeled training data?
- How to learn the semantic representation of short texts and pre-defined categories with the help of a knowledge base?

In this regard, a novel probabilistic model for Knowledge-Based Short Text Categorization (KBSTC) has been proposed. It is a new paradigm for short text categorization based on a large knowledge base. KBSTC does not require any labeled training data, instead, it considers entities present in the input text and their semantic similarity to the predefined categories to categorize short text. In order to calculate the semantic similarity between the entities and categories, a new embedding model has been proposed. The proposed model tries to capture the relatedness between the entities and hierarchically related categories from knowledge bases. To do so, it first creates entity-entity and entity-category networks from a given knowledge

base. Then, jointly learns the low dimensional distribution of the vertices of the networks, i.e., entities and categories. In conclusion, the experimental results have proven that KBSTC is capable of categorizing short text in an unsupervised way with high accuracy. Further, it outperforms all the baselines, which do not require any labeled data for short text categorization. Since the performance of the supervised approaches depends on the characteristics of the training set (e.g., size, balanced or unbalanced), KBSTC provides better performance than the supervised approaches on some datasets (e.g., Snippets). Moreover, the samples with more entities have enabled KBSTC to provide better accuracy. Besides, the experimental results have also demonstrated that the proposed joint entity and category embedding model, which embeds entities and categories from a large knowledge base to a common vector space captures better semantic relation between entities and categories than the baseline embedding models. As for future work, our aim is to include words along with entities for the KBSTC task, which requires also the extension of the proposed embedding model towards the additional inclusion of word embeddings into the common entity and category vector space. Further, the performance of KBSTC will also be evaluated on social media text such as tweets.

5 Weakly Supervised Short Text Categorization

This chapter is one of the core chapters of this thesis and it investigates the following research question which is defined in Section 1.2 (Research Question 3):

- How to combine a deep neural network with a knowledge base to perform short text categorization without requiring any hand-labeled data?

In this regard, this chapter proposes a weakly supervised short text categorization approach which is based on the following publication:

Türker, R., Zhang, L., Alam, M., & Sack, H. (2020). Weakly supervised short text categorization using world knowledge. The 19th International Semantic Web Conference, ISWC 2020.

The proposed model consists of two main modules:

1. a data labeling module, which leverages an external Knowledge Base (KB) to compute probabilistic labels for a given unlabeled training data set,
2. a categorization model based on a Wide & Deep learning approach.

The effectiveness of the proposed method is validated via evaluation on multiple datasets, i.e., AG news, Google Snippets, Twitter, DBpedia. To assess the performance of the proposed method extensive experiments have been conducted. The experimental results show that the proposed approach outperforms unsupervised state-of-the-art categorization approaches and achieves comparable performance to supervised approaches.

5.1 Introduction

Due to the rapid growth of the Web content, online short text data such as search snippets, news feeds, short messages, etc. is drastically multiplying (Chen et al., 2011). Furthermore, the reason of the significant amount of availability of short text data can be also attributed to the online platforms where people express their opinion in short text forms. Hence, short text categorization has become a crucial task for a wide range of applications including sentiment analysis and news feed categorization (Hu et al., 2019). While conventional text classification methods such as Support Vector Machines (SVMs) have demonstrated their success in classifying long and well structured text, as e.g., news articles. However, in case of short text they seem to have a substandard performance (Zeng et al., 2018). Moreover, unlike paragraphs or documents, categorization of short text is a much more challenging task due to its main characteristics such as:

- limited context, i.e., only a few words within the text,
- sparsity and
- ambiguity as it does not have enough contextual information.

Hence, the traditional categorization methods based on Bag of Words (BoW) (Wang et al., 2016b) or approaches that utilize word embeddings perform poorly if directly applied to short text. Furthermore, approaches that utilize word embeddings for categorization perform better when dealing with longer text, in which case, even if a word is ambiguous, such ambiguity will be handled based on the given context. In the case of short text, where the available context is rather limited and each word obtains significant importance, such approaches often lead to inaccurate results, especially, on new and rare words. Thus, to overcome these challenges, it is indispensable to use external sources such as Knowledge Bases (KBs) to enrich and obtain more advanced text representations (Wang et al., 2017).

Recently, several deep learning approaches have been proposed for short text categorization, which demonstrated remarkable performance in this task (Chen et al., 2017; Ma et al., 2018). The two main advantages of these models for the categorization task are that

- a minimum effort is required for feature engineering,
- their categorization performance is better in comparison to traditional text categorization approaches (Meng et al., 2018).

However, the requirement of large amounts of labeled data remains the main bottleneck for neural network based approaches (Meng et al., 2018). Furthermore, the models that are trained with a limited amount of labeled data tend to face an overfitting problem (Chen et al., 2015a). Acquiring labeled data for the categorization task is costly and time-consuming. Especially, if the data to be labeled is of a specific domain then only a limited number of domain experts are able to label them correctly. In other words, the domain experts need to read such a significant

amount of data carefully and label them according to their domain knowledge (Meng et al., 2018). This is a very expensive and labor intensive task.

To overcome the requirement for labeled data bottleneck, several *dataless* (Li et al., 2016a; Hingmire et al., 2013), *semi supervised* (Nigam et al., 2000; Xuan et al., 2017), and *weakly supervised* (Meng et al., 2018; Meng et al., 2019) categorization algorithms have been proposed. The *dataless categorization* algorithms do not require any labeled data to perform text categorization. Instead, they project each predefined label and document into a common vector space by exploiting the words present in the labels and the documents. As a second step, based on the vector similarity a label is assigned to each document. However, the most prominent dataless categorization methods are designed for long text, e.g., news article categorization (Li et al., 2016a). Moreover, other dataless approaches leverage label names/descriptions as seed words for supervision (Chen et al., 2015a). Therefore, the performance of such systems highly depends on manually designed seed words or label descriptions.

In addition, for addressing the labeled data scarcity problem, *semi supervised* text categorization algorithms have been proposed. These approaches leverage both labeled and unlabeled data to perform categorization. Nevertheless, they still require some set of labeled data. Yet, generating small training sets for semi supervised methods still remains an expensive process due to the diversity of the documents in many applications (Li et al., 2016b). Furthermore, there has been a considerable amount of studies in *weakly supervised* text categorization approaches. Most of these methods require user-given weak supervision sources such as some labeled documents, class related keywords, etc. for the categorization task. Providing manually designed such sources is still an expensive task. Besides, existing weakly supervised text categorization solutions mostly rely on hard-coded heuristics, such as looking for specific keywords or syntactical patterns in text, which still requires domain expertise and is especially prone to noise. Moreover, the most well-known weakly supervised methods are designed for long text categorization.

Motivated by the aforementioned challenges, this chapter proposes a novel model for **We**akly **S**upervised **S**hort **Te**xt **C**ategorization using World Knowledge (WESSTEC). The proposed approach does not require any labeled data for short text categorization. It exploits a knowledge base and embedding models such as Doc2Vec (Le & Mikolov, 2014), LINE (Tang et al., 2015b), Word2Vec (Mikolov et al., 2013b) etc. as weak supervision sources without requiring any manual effort. Instead, given a list of labels and unlabeled short text documents, the proposed method first associates each text with its relevant concepts in the KB to enhance the semantic representation of short texts and then generates labels for each document by utilizing the aforementioned embedding models. In the second step, words and concepts from the labeled documents are exploited for training a Wide & Deep learning based categorization model (Cheng et al., 2016). Finally, the trained model is used to categorize new short text documents.

Overall, the main contributions of this chapter are as follows:

- a new paradigm for short text categorization, based on a knowledge based weak supervision;
- a method to combine weak supervision sources to generate labeled data, which can be used for any arbitrary categorization model;
- adaptation of a Wide & Deep model for weakly supervised short text categorization;
- utilizing multiple features, i.e, both words and entities present in a given short text and their combination for the Wide & Deep model;
- an experimental evaluation using four different standard datasets for short text categorization.

5.2 Related Approaches

This study aims to categorize short text documents under a weak supervision setting without requiring any manually labeled data. The related studies on *Short Text Categorization* is presented in Section 3.2 and thus this section introduces prior related studies on *Weakly Supervised Text Categorization*.

There has been a considerable amount of studies related to weakly supervised text categorization to address the problem of missing labeled data (Meng et al., 2018; Meng et al., 2019; Rabinovich et al., 2018).

(Meng et al., 2018) propose a weakly supervised neural network based text categorization approach. The proposed model consists of two main modules:

- a pseudo-document generator, which utilizes three types of user provided seed information in order to generate pseudo labeled documents. The types of seed information are:
  - label surface names,
  - class-related keywords,
  - labeled documents.
- a self-training module, which first trains a neural model with the pseudo labeled documents, then applies a bootstrapping strategy to refine the model. To bootstrap the pre-trained model, its highly confident predicted documents are utilized.

The performance of the model has been assessed on three different datasets as well as with different seed information. The model outperforms all the baselines with different weak supervision sources.

The same authors extend the aforementioned model in a similar paradigm of weak supervision to be able to classify the text documents into a given category/label hierarchy (Meng

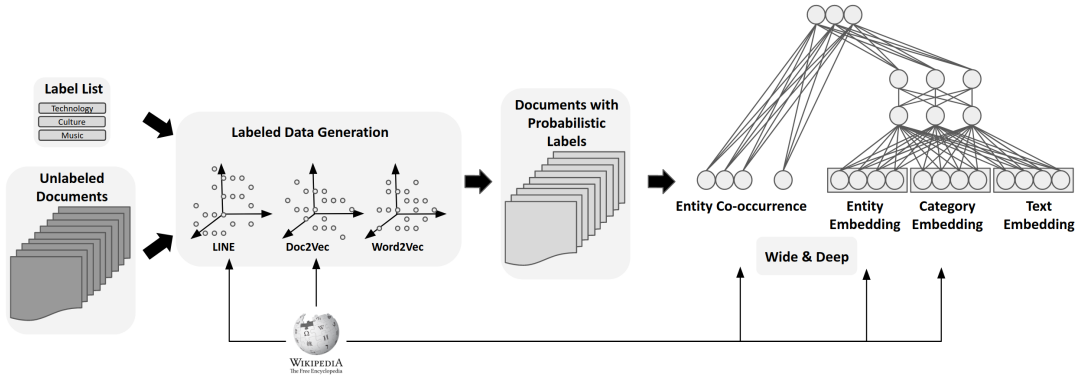


Figure 5.1: The workflow of WESSTEC

et al., 2019).

On the other hand, (Hingmire & Chakraborti, 2014) proposed Latent Dirichlet Allocation (LDA) based weakly supervised text categorization approach. LDA is a generative model which does not require any labeled data. Further, it assumes that each document in a given corpus is a mixture of latent topics. The proposed method finds a likely set of topics for each given document and the human annotator assigns meaningful category labels to these latent topics. Then the labeled latent topics are utilized to categorize documents.

The presented methods require user-given weak supervision sources such as class-related keywords, a small amount of labeled data, etc. Hence, the requirement for domain expertise is still inevitable. On the contrary, the proposed approach in this chapter, i.e., WESSTEC does not require such manually designed weak supervision sources for the categorization task. Instead, it utilizes unsupervised embedding models such as Word2Vec, Doc2Vec, and LINE as weak supervision sources.

5.3 Weakly Supervised Short Text Categorization

This section provides a formal definition of the short text categorization task, followed by the description of the proposed approach.

Problem Formulation. Given an input short text t and n predefined labels $L = \{l_1, l_2, .., l_n\}$, the output is the most relevant label $l_i \in L$ for the given short text t, i.e., the label function $f_{lab}(t) = l_i$, where $l_i \in L$ is computed.

Method Overview. The general workflow of WESSTEC is shown in Figure 5.1. As

stated before, WESSTEC contains two main modules, namely, *Labeled Data Generation* and the categorization model, which is based on Wide & Deep (Cheng et al., 2016).

Given a list of labels and a set of unlabeled short text documents, the Labeled Data Generation module is responsible for generating *probabilistic training labels* for each document. To do so, it utilizes three different embedding models, i.e., LINE (Tang et al., 2015b), Doc2Vec (Le & Mikolov, 2014) and Word2Vec (Mikolov et al., 2013b) to estimate the probability of each predefined label for a given document. This module aims to generate documents with probabilistic labels as training data. It should be noted that the generated training data, i.e., labeled short text documents, can be leveraged by any supervised model for the training phase.

The second main module of the workflow is a Wide & Deep learning based categorization model (Cheng et al., 2016), which utilizes the documents with the probabilistic labels for training. Several different feature sets, such as words and entities, are extracted from the documents to train the Wide & Deep model. Furthermore, categories that are associated with the entities present in documents have also been utilized as a feature set. As shown in Figure 5.1, Wikipedia has been leveraged as a KB in this work.

Section 5.3.1 and Section 5.3.2 provide a detailed description of each module and the feature sets that have been utilized by each module.

5.3.1 Labeled Data Generation

The aim of this module is to generate labeled documents from a given label list and unlabeled set of documents (cf. Figure 5.1). In other words, given a short text t and n labels $L = \{l_1, l_2, .., l_n\}$, the goal of this module is to produce a probabilistic label for t as $\mathbf{y_t} = [p_1, p_2, ..., p_n]$, where $p_i \in [0, 1]$, and p_i is the corresponding probability of l_i for t. To this end, this module utilizes three different embedding models, namely, LINE, Doc2Vec and pre-trained Word2Vec to capture the semantic correlations between the predefined labels and the words as well as entities present in a short text. First, each document and label is projected into common vector spaces, then the probabilistic labels of given texts are calculated based on the semantic similarity between documents and the set of predefined labels.

The network embedding models that have been employed in this work are presented as follows:

- **LINE** is a network embedding model, which is designed to learn embedding of arbitrary types of large-scale networks (weighted, directed, undirected, etc.). The model has been trained by utilizing Wikipedia hyperlink structure to obtain a vector representation of each entity from Wikipedia. In other words, from Wikipedia hyperlink structure, an *entity-network* has been constructed to be utilized by this model. More technical details about the construction of the *entity-network* can be found in Section 4.6.1.

To obtain a document vector with the help of LINE, we simply take the average of entity vectors present in that document. To extract entities from a document an *anchor text dictionary* (cf. Section 4.3) is used. The anchor text dictionary is constructed by leveraging the anchor texts of hyperlinks of Wikipedia, which are pointing to any article in Wikipedia. The anchor texts are considered as entity mentions and the links refer to the corresponding entities.

- **Doc2Vec** creates the distributed representation of documents by utilizing the context words present in the corresponding documents. This model has been trained on Wikipedia articles and contains a vector representation of each entity of Wikipedia. Note that we consider each Wikipedia article page as an entity. To form a document vector for a given text, the average of entity vectors present in that text is considered.

- **Word2Vec** learns the low dimensional distributed representation of words. We use the pre-trained Word2Vec model[1] for our approach. To create document vectors with Word2Vec, the average of the word vectors in that document is considered.

Moreover, each given label is also mapped to its corresponding vector in the respective vector space, e.g., the label *Music* is mapped to the word vector of *Music* from Word2Vec and it is also mapped to the entity vector of *Music* from Doc2Vec and LINE.

After embedding each text and label into common vector spaces, each embedding model assigns the most similar label to each text based on the vector similarity between the text and the labels. As there are three embedding models, for each given text three labels are generated. These labels can overlap or conflict. Then, the goal of the remaining process of this module is to convert the outputs of the embedding measures into *probabilistic training labels*. In order to achieve that a heuristic approach has been employed.

Based on outputs of each embedding measure for all texts, the heuristic approach estimates the *confidence* of each embedding model by considering the output label agreement and disagreement rates. The *confidence* of an embedding model EM_i is defined as follows:

$$C_{EM_i} = \frac{Agg_{EM_i} + noneAgg}{TotalAgg + noneAgg}, \tag{5.1}$$

where Agg_{EM_i} is the number of documents, on which the model EM_i agreed for a label with at least one of the other embeddings, $noneAgg$ is the number of documents, on which none of the embedding models agreed for the assigned label and $TotalAgg$ is the number of documents which at least two embedding models agreed on their labels (i.e., $TotalAgg = \#TotalDocuments - noneAgg$).

The confidence values are exploited to convert the generated labels into probabilistic

[1] https://code.google.com/archive/p/word2vec/

training labels $\mathbf{y_t}$. For each text, the preferred label from each embedding measure will be weighted using its confidence and then all three weighted labels are combined together, which could result in three probabilistic labels when three measures disagree with each other or two probabilistic labels when two measures agree or one label when all agree on it. Finally, these values are normalized to produce the probabilistic training labels $\mathbf{y_t}$.

Given a short text t and n labels $L = \{l_1, l_2, .., l_n\}$, let $\mathbf{y_t} = [p_1, p_2, ..., p_n]$ denote t's probabilistic training labels, where $p_i \in [0, 1]$. To calculate the probability p_i of the label l_i for t, we define the following formula:

$$p_i(t) = \frac{\sum_{j=1}^{e} C_{EM_j} I^i_{EM_j}(t)}{\sum_{k=1}^{n} \sum_{j=1}^{e} C_{EM_j} I^k_{EM_j}(t)}\,, \tag{5.2}$$

where e is the total number of embedding models that are utilized in *labeled data generation* module, C_{EM_j} is the confidence of embedding model EM_j, n is the total number of predefined labels and $I^i_{EM_j}$ is defined as

$$I^i_{EM_j}(t) = \begin{cases} 1 & \text{if } EM_j \text{ assigns } l_i \text{ to } t\ , \\ 0 & \text{otherwise}\,. \end{cases} \tag{5.3}$$

5.3.2 Wide and Deep Model for Short Text Categorization

The Wide & Deep model originally proposed for recommendation systems. One of the main contributions of this chapter is the adaption of Wide & Deep learning paradigm to the weakly supervised short text categorization task. The second main module of the workflow (Figure 5.1) illustrates the adaption of the Wide & Deep learning based categorization model to short text categorization. To the best of our knowledge, this is the first attempt of utilizing the Wide & Deep model for short text categorization.

The model consists of two main components i.e., Wide Component and Deep Component. Moreover, the model has the ability of memorizing feature interactions and generalizing feature combinations by jointly training the wide and deep components as shown in Figure 5.1 (right).

In the following, first the Wide model and Deep model separately are introduced and then joint Wide & Deep model are presented:

- **Wide Model:** The wide part has the ability of memorizing feature interactions. In other words, it is able to learn the frequent co-occurrence of features. Hence, we design this model to be able to capture the correlation between the co-occurrence of features and the target labels. In our approach, ***Entity co-occurrence*** information of each document is used as a feature for the wide part (cf. Figure 5.1). Given a short text

t let $\mathbf{x_t} = [x_1, x_2, x_3, ..., x_m]$ denote the m entities present in t. To construct the d dimensional *Entity co-occurrence* feature vector we apply cross-product transformation (Cheng et al., 2016) as:

$$\phi_k(\mathbf{x_t}) = \prod_{i=1}^{m} x_i^{c_{ki}} \quad c_{ki} \in \{0, 1\}, \tag{5.4}$$

where c_{ki} is a boolean variable that is 1 if the *i*-th feature is part of the *k*-th transformation ϕ_k, and 0 otherwise. The wide part is a model of the form as:

$$P(Y = l_i|t) = softmax(\mathbf{w}_i^T \phi(\mathbf{x_t}) + b_i), \tag{5.5}$$

where t is a given short text, $\phi(\mathbf{x_t}) = [\phi_1(\mathbf{x_t}), \phi_2(\mathbf{x_t}), ..., \phi_d(\mathbf{x_t})]$ is the cross product transformations of $\mathbf{x_t}$, $\mathbf{w}_i = [w_1, w_2, ..., w_d]$ and b_i are the model parameters corresponding to the i-th label l_i. The *softmax* function is defined as:

$$softmax(z_i) = \frac{e^{z_i}}{\sum_{z_j \in \mathbf{z}} e^{z_j}}, \tag{5.6}$$

for $i = 1, 2, .., n$ and $\mathbf{z} = (z_1, z_2, ..., z_n) \in \mathbb{R}^n$.

We give the following example to illustrate how an *Entity co-occurrence* feature vector can be formed. Given a short text *"Motorola and HP in Linux tie-up"*, the extracted entities are $E' = \{Motorola, HP, Linux\}$ and the possible entity pairs are $E'_p = \{(Motorola, HP), (Motorola, Linux), (HP, Linux)\}$. The dimension of the vector is the number of all the possible entity pairs of the dataset and each dimension corresponds to an entity pair. For each entity pair $e_{p_i} \in E'_p$, the value of the corresponding dimension of the vector would be 1 and the rest would be 0.

- **Deep Model:** The deep part is a neural network, which is capable of generalization of feature combinations through low-dimensional dense embeddings. In our approach, three different embedding vectors, i.e., ***Entity Embedding***, ***Category Embedding*** and ***Text Embedding*** are utilized as an input to the deep part (cf. Figure 5.1).

  To construct each feature vector, different embedding models are utilized, i.e., for *Entity Embedding* LINE, for *Category Embedding* the joint entity and category embedding model (cf. Section 4.6) and for *Text Embedding* Word2Vec. The joint entity and category embedding model has been proposed by (Türker et al., 2019) to capture the semantic relations between entities and categories from a KB. This model first constructs a weighted network of entities and categories, and then jointly learns their embeddings from the network.

  In order to form an *Entity Embedding* vector for a given text, entities present in the document are extracted with the help of a prefabricated Anchor-Text dictionary (cf.

Section 4.3) and then the average of the vector representations of these entities is taken. For the *Category Embedding* feature vector, all the categories that are directly associated with the entities appearing in the text are collected from Wikipedia, then the average of the category vector representations is taken. Finally, for a given text a *Text Embedding* feature vector is constructed by taking the average of the word vector representations in that document.

The deep part is a feed forward neural network, which takes low-dimensional embedding vectors as an input i.e., $[\mathbf{e}_t^e, \mathbf{e}_t^c, \mathbf{e}_t^t]$, where $\mathbf{e}_t^e$ is the entity embedding, $\mathbf{e}_t^c$ is the category embedding and $\mathbf{e}_t^t$ is the text embedding.
The deep part is the model of the form as:

$$P(Y = l_i|t) = softmax(\mathbf{w}_i^T a^{(lf)} + b), \tag{5.7}$$

where $\mathbf{w}_i$ are the weights that are applied on the final activation $a^{(lf)}$ for the i-th label l_i, l is the layer number and f is the activation function which is ReLU.

We have built 3-layer feed forward neural network for the deep part and each hidden layer of this model performs the following computation (Cheng et al., 2016):

$$a^{(l+1)} = f(W^{(l)} a^{(l)} + b^{(l)}), \tag{5.8}$$

$a^{(l)}$ is activations, $b^{(l)}$ is bias and $W^{(l)}$ is model weights at l-th layer.

- **Wide & Deep Model:** The wide and the deep components are combined for joint training by back propagating the gradients from the output of both wide and deep parts simultaneously. The combined model is illustrated in Fig. 5.1 (right). For a given short text t the prediction of Wide & Deep model is:

$$P(Y = l_i|t) = softmax({\mathbf{w}_{wide_i}}^T \phi(\mathbf{x}_t) + {\mathbf{w}_{deep_i}}^T a^{(lf)} + b_i). \tag{5.9}$$

In order to deal with the probabilistic training labels, we configure our model to train with a noise-aware loss function, i.e., cross-entropy between the probability of each training label and the output of the softmax function, which is defined as:

$$H(p, q) = -\sum_n p_i(t) * log(P(Y = l_i|t)) \tag{5.10}$$

Note that the reason of exploiting different feature sets in Labeled Data Generation and Wide & Deep modules is mainly two-fold: (1) Combining different features into Labeled Data Generation module requires much more feature engineering efforts. In other words, the Wide & Deep model can automatically learn the weights of the feature sets, however, it is not the same case with the proposed heuristic model designed

for labeled data generation. (2) There are some features (e.g., entity co-occurrence) that cannot be straightforwardly integrated into heuristic algorithms to help calculate semantic similarity between input text and labels and do the labeling. However, such "non-heuristic" features can be transferred into the final categorization model trained on labeled data generated by the heuristic algorithms using other features. Overall, we expect the trained model to provide performance gains over the heuristics that it is trained on both by applying to "non-heuristic" features (e.g., entity co-occurrence), and by learning to generalize beyond heuristics, i.e., putting weights on more subtle features that each individual heuristic algorithm cannot cover.

5.4 Experiments

This section provides a description of the datasets and the baselines, followed by the experimental results and a comparison to the state-of-the-art text categorization approaches.

5.4.1 Datasets

In Section 4.7.1 two real-world datasets, namely, AG News, Google Snippets have been leveraged to assess the performance of KBSTC which has been proposed in Chapter 4. In these experiments in order to extend the evaluation process four different datasets, i.e., AG News, Google Snippets, DBpedia as well as Twitter dataset have been used to demonstrate the performance of the proposed approach. The details of the AG News and Google Snippets are given in Section 4.7.1. Therefore, in this section, DBpedia and Twitter datasets are introduced in more detail.

- **AG News** (Zhang & LeCun, 2015) contains titles and short descriptions of news articles.

- **Snippets** (Phan et al., 2008) contains short snippets from Google search results.

- **DBpedia Ontology** is also adopted from (Zhang & LeCun, 2015). The well-known text categorization benchmark has been built by selecting 14 non-overlapping classes from DBpedia 2014. The dataset distribution is shown in Table 5.1. The dataset contains short abstracts of DBpedia entities. In comparison to the other datasets, the DBpedia dataset has the highest number of categories, further, it has the largest training samples per category.

- **Twitter** [2] is a topic categorization dataset, contains tweets belong to 6 different cat-

[2] https://github.com/madhasri/Twitter-Trending-Topic-Classification/tree/master/data

egories. The dataset distribution is shown in Table 5.2. The Twitter dataset is pre-processed, in other words, the dataset does not contain hash symbols, emoticons, user mentions, etc.

Table 5.3 shows the distribution of the datasets and the average number of entities and words per text in each dataset. More details about AG news and Google Snippets datasets can be found in Section 4.7.1.

Furthermore, as WESSTEC does not require any labeled training data, the training datasets of AG News, Snippets, DBpedia and Twitter have been used without their labels. In other words, the training set of each dataset without their labels have been utilized as an input to *Labeled Data Generation* module of the WESSTEC framework (cf. Figure 5.1) to generate the training labels.

Category	#Train	#Test
Company	40,000	5,000
Educational Institution		
Athlete		
Office Holder		
Mean of Transportation		
Building		
Natural Place		
Village		
Animal		
Building		
Plant		
Album		
Film		
Written Work		

Table 5.1: The data distribution of the DBpedia dataset

Category	#Train	#Test
Business	1673	601
Entertainment	1700	500
Food-Lifestyle	1901	900
Politics	1730	996
Sports	1375	400
Technology	1500	300

Table 5.2: The data distribution of the Twitter dataset

Dataset	#Category	#Train	#Test	Avg. #Ent	Avg. #Word
AG News	4	120,000	7,600	11.83	38.65
Google Snippets	8	10,060	2,280	8.90	17.97
DBpedia	14	560,000	70,000	15.30	46.49
Twitter	6	9,879	3,697	4.31	12.36

Table 5.3: The statistics for the short text datasets

5.4.2 Baseline Approaches

To demonstrate the performance of the proposed approach, the following models have been selected as baselines:

- **Dataless ESA and Dataless Word2Vec (cf. Section 4.2.1):** Two variants of the state-of-the-art dataless approach (Song & Roth, 2014) are considered as baselines which are based on different methods to compute word similarity, i.e., ESA (Gabrilovich et al., 2007) and Word2Vec (Mikolov et al., 2013b).
- **KBSTC (cf. Chapter 4):** Knowledge-based short text categorization, which does not require any labeled data for short text categorization. Instead, it relies on the semantic similarity between the given short text and predefined labels to categorize a given short text. Therefore, KBSTC can be considered as the most relevant prior work to WESSTEC.
- **SVM+tf-idf (cf. Section 3.5):** In this model, the term frequency-inverse document frequency (tf-idf) is calculated as features for a subsequent Support Vector Machine (SVM).
- **CNN (Zhang & Wallace, 2017)+Word2Vec, CNN+Ent and CNN+Category:** A Convolutional Neural Network (CNN) is applied on text, entity and category matrices separately. These matrices are constructed by using Word2Vec, LINE, joint entity and category embedding model(cf. Section 4.6), respectively.
- **LSTM (cf. Section 2.1.5):** The standard LSTM model is composed of a single LSTM layer followed by a dense output layer.
- **charCNN (Zhang & LeCun, 2015):** This model learns character embeddings using "one-hot" encoding. Subsequently, CNN is applied for the categorization process.
- **BERT (Devlin et al., 2019) (cf. Section 2.1.6):** The state-of-the-art language representation model[3] have been commonly leveraged to derive sentence embeddings. To produce BERT embeddings, first, each sentence has been passed through pre-trained BERT, then the outputs of the model have been averaged, which is the most common way of obtaining sentence embeddings from BERT (Reimers & Gurevych, 2019). In the

[3] https://github.com/google-research/bert

experiments, the BERT embeddings have been generated as features for the subsequent 3-layer feed forward neural network.

5.4.3 Feature Sets

This section describes the feature sets that have been extracted from the *Documents with Probabilistic Labels* (cf. Figure 5.1) and utilized to train the Wide & Deep model. To construct feature sets, words and entities present in texts as well as parent categories of entities from Wikipedia have been leveraged.

As shown in Figure 5.1, the wide part exploits the **Entity Co-occurrence (Ent Co)** information as a feature and the deep part utilizes three different feature sets, namely, **Text Embedding (Text)**, **Entity Embedding (Entity)** and **Category Embedding (Category)** vectors as well as their combinations, such as Text+Entity (cf. Table 5.4) refers to the concatenation of text embeddings and entity embeddings. The detailed construction of the feature sets is explained in Section 5.3.2.

5.4.4 Evaluation of WESSTEC

Table 5.4 depicts the categorization accuracy of the Wide & Deep model of WESSTEC, in comparison to individual Wide-only and Deep-only models with different features on AG News, Snippets, DBpedia, Twitter datasets.

It has been observed that the jointly trained Wide & Deep model outperforms the individual Wide-only and Deep-only models on each dataset. As stated before, the Wide model capable of memorizing feature interactions and on the other hand the Deep model capable of generalization of feature combinations. Hence, the reason here can be attributed to the benefit of utilizing the Wide & Deep model to achieve both memorization and generalization of features for short text categorization.

In addition, we have randomly sampled some instances from the wrongly categorized samples with the Deep part. It has been observed that some of these instances, have been correctly categorized after combining the Wide part with the Deep part and jointly training the model.

The wide model performs best on the AG News dataset. This dataset has the least number of categories and the length of the samples are not as limited as Twitter dataset, therefore, it is easier for the Wide model to handle this type of a dataset in comparison to the other datasets. Further, in contrast to the Snippets and Twitter datasets, AG News contains an equal number of samples per class, i.e., it is a balanced dataset.

The reason for the general low accuracy of the Wide model (in comparison to the Deep model and Wide & Deep model) is that a very sparse set of features, i.e., entity occurrence

Model	Feature	AG News	Snippets	DBpedia	Twitter
Wide	Entity Co-occurance (Ent Co)	0.561	0.447	0.499	0.278
Deep	Text	0.802	0.795	0.786	0.555
	Entity	0.790	0.764	0.775	0.521
	Category	0.773	0.698	0.754	0.444
	Text+Entity	0.793	0.785	0.779	0.524
	Text+Category	0.801	0.794	0.786	0.554
	Entity+Category	0.792	0.771	0.771	0.534
	Text+Entity+Category	0.792	0.786	0.785	0.529
Wide & Deep	Ent Co+Text	0.807	0.792	0.786	0.556
	Ent Co+Entity	0.791	0.774	0.768	0.520
	Ent Co+Category	0.792	0.693	0.774	0.446
	Ent Co+Text+Entity	0.787	0.802	0.776	0.53
	Ent Co+Text+Category	**0.814**	0.803	**0.792**	**0.581**
	Ent Co+Entity+Category	0.791	0.770	0.766	0.544
	Ent Co+Text+Entity+Category	0.790	**0.805**	0.778	0.572

Table 5.4: The categorization accuracy of different models with different features

information, have been used to train the model. Although Deep Neural Networks (DNNs) are computationally more expensive, it is a well-known fact that they can be much more powerful than the linear models. Therefore, in these experiments, the Deep model always outperforms the Wide model on each dataset. Similar to the Wide model, with the Deep model the best categorization accuracy has been obtained on the AG news.

On the other hand, despite the specific properties of Tweets (e.g., out-of-vocabulary words) WESSTEC can still obtain reasonable accuracy on the Twitter dataset. To illustrate the difficulty of categorizing tweets, we give the following tweet from the Twitter dataset as an example: *"BSE NSE Stock Tip HINDUSZI"*, which is labeled as *"Business"*. The categorization of such tweets is rather difficult for many standard categorization models, which rely on only words. However, WESSTEC enriches text representations by leveraging entities present in texts and their associated categories with the help of a KB. For the given example the detected entities are `Bombay_Stock_Exchange`, `National_Stock_Exchange_of_India` and `Stock`, which capture very useful information for categorization of the tweet. Further, even for out-of-vocabulary words such as *"BSE"*, WESSTEC can still detect entities, which are crucial for the categorization task.

This study has also investigated the impact of each feature combination on the catego-

rization performance. The Deep model performs the best when utilizing only words. Whereas, the Wide & Deep model enjoys the combination of the feature sets. However, it has been observed that using entity features in both wide and deep parts could result in a bias of the whole model towards entity information, which might not reflect the entire semantics of text, especially when the text is longer such that there could be some more words that cannot be detected as entities (e.g., in AG News and DBpedia). This suggests that our Wide & Deep model (Ent-Co+Text+Category) using Entity Co-occurrence (Ent-Co) as a feature in the wide part as well as Text Embedding (Text) and Category Embedding (Category) as features in the deep part could be the most promising combination. The results in Table 5.4 also shows that (Ent-Co+Text+Category) clearly yields best results on AG News and DBpedia datasets and performs only slightly worse than (Ent-Co+Text+Entity+Category) on Snippets dataset (with the difference of 0.002 for accuracy).

Overall, the experiments show that, firstly, it is possible to perform short text categorization with a high accuracy in the complete absence of labeled data with our proposed approach and secondly, the Wide & Deep model can be successfully applied for the short text categorization problem.

Since WESSTEC achieves almost the best performance with the combination of Ent-Co+Text+Category features, we use the results of this model for the comparison between WESSTEC and other approaches in the rest of the experiments.

5.4.5 Comparison of WESSTEC with the Unsupervised Approaches

Table 5.5 presents the categorization accuracy of WESSTEC in comparison to the text categorization approaches that do not require any labeled data.

It is observed that the proposed approach based on the Wide & Deep model considerably outperforms the dataless approaches as well as KBSTC. Although the dataless approaches achieved promising results in the case of longer news articles in (Song & Roth, 2014), they cannot perform well on short text due to the data sparsity problem.

Model	AG News	Snippets	DBpedia	Twitter
Dataless ESA (Song & Roth, 2014)	0.641	0.485	0.551	0.317
Dataless Word2Vec (Song & Roth, 2014)	0.527	0.524	0.679	0.5
KBSTC (cf. Chapter 4)	0.805	0.720	0.460	0.359
WESSTEC	**0.814**	**0.803**	**0.792**	**0.581**

Table 5.5: The categorization accuracy against the unsupervised baselines

KBSTC is the most recent related work to WESSTEC. It is a probabilistic model and does not require any labeled training data to perform short text categorization. Instead, the cat-

egory of the given text is derived based on the semantic similarity between the entities present in the text and the set of predefined categories. KBSTC utilizes only entities and ignores the words. Whereas WESSTEC first generates documents with probabilistic labels from a given unlabeled document set, then it utilizes those documents to train a Wide & Deep model to categorize new documents. In other words, WESSTEC exploits words present in text as well as entities and their directly associated categories from a KB for categorization. In addition, the proposed model leverages both textual information (in Doc2Vec model) and structural information (in LINE model) from KBs to better capture the semantic representation of entities, however, KBSTC uses only structural information of entities. Further, while KBSTC labels the input text only based on the heuristics of semantic similarity, WESSTEC adapts an additional categorization model using Wide & Deep learning. Hence, the proposed model is much more sophisticated and utilizes more features than the KBSTC model. Therefore, as expected the categorization performance has been improved with the proposed approach.

5.4.6 Comparison of WESSTEC with the Supervised Approaches

In order to show the effectiveness of the Wide & Deep Module (cf. Section 5.3.2), its performance has been compared with the supervised baselines. The generated training sets of respective datasets (cf. Section 5.3.1) have been utilized to train Wide & Deep as well as the baseline models. The respective original test datasets have been used for evaluating the trained models. Table 5.6 reports the categorization performance.

Model	AG	Snippets	DBpedia	Twitter
SVM+tf-idf	0.808	0.696	0.784	0.513
CNN+W2V	0.796	0.787	0.784	0.542
CNN+Ent	0.794	0.703	78.24	0.456
CNN+Category	0.779	0.656	0.762	0.449
LSTM	0.786	0.693	0.796	0.473
charCNN	0.773	0.497	0.760	0.472
BERT	0.806	0.801	**0.804**	0.560
Wide & Deep	**0.814**	**0.803**	0.792	**0.581**

Table 5.6: The categorization accuracy against the supervised baselines. The baselines have been trained with the generated training sets (cf. Section 5.3.1) of respective datasets.

The results show that the proposed Wide & Deep model can yield better accuracy in comparison to the baselines. This is due to the fact that in contrast to other approaches,

the Wide & Deep model is capable of both memorization and generalization of features and thus it performs the best among all the approaches. Moreover, especially on the Snippets dataset, Wide & Deep model significantly outperforms all the baselines. The reason here can be attributed to the different characteristics of this dataset. The Snippets dataset has less average number of entities, words per text and the size of the training set is much smaller in comparison to other datasets (cf. Table 5.3). In contrast to baselines, the proposed Wide & Deep model utilizes different resources from a KB to enrich the semantic representations of texts. Thus, it is capable of categorizing of such a dataset with a high accuracy.

Another advantage of the Wide & Deep model over the baselines is different feature combinations (e.g., entity co-occurrence, text embedding, entity embedding, etc.) can be easily exploited by the model for the categorization task.

Furthermore, a statistical significance test, namely, the 5x2cv paired *t*-test (Dietterich, 1998) has been also performed to compare the results of Wide & Deep and BERT. This test has been proposed to overcome the drawbacks of other significance tests (e.g., resampled paired *t*-test) and it is based on five iterations of two-fold cross validation. According to 5x2cv paired *t*-test, the experimental results are significantly different at 95% level of significance with 5 degrees of freedom.

Overall, the obtained results in Table 5.6 suggest that in comparison to the baselines the Wide & Deep model is better suited for the short text categorization task by utilizing the generated labeled data for training.

5.4.7 Evaluation of the Generated Labeled Data

To evaluate the performance of each embedding model, i.e., Word2Vec, Doc2Vec and LINE in the context of labeling the training data, we have conducted a set of experiments. First, each of the unlabeled documents and predefined labels has been projected into common vector spaces. Then each embedding model has assigned the most similar label to the documents based on the vector similarity. Additionally, by considering a simple majority vote of all the embedding models each document has also been labeled. The accuracy of labeled datasets has been calculated by comparing them with the original hand-labeled data. Table 5.7 presents the accuracy of the labeled training data based on the individual embedding models and the majority vote. The results suggest that considering all the embedding models for the labeling task can help in assigning more accurate labels. Therefore, to estimate the probabilistic labels for each training sample, all the embedding models have been used in the *Labeled Data Generation* module (cf. Section 5.3.1).

Further experiments have been conducted to asses the performance of the Wide & Deep model when it is trained on the training samples that are labeled based on majority vote. Table 5.8 presents the categorization accuracy. The results show that using probabilistic labels in WESSTEC leads to higher-quality supervision for training the end categorization model.

Model	AG News	Snippets	DBpedia	Twitter
Vector Similarity LINE	0.776	0.657	0.708	0.536
Vector Similarity Doc2Vec	0.651	0.644	0.672	0.479
Vector Similarity Word2Vec	0.612	0.692	0.702	0.527
Vector Similarity (Majority)	**0.778**	**0.709**	**0.757**	**0.555**

Table 5.7: The accuracy of generated training data based on the embedding models

Model	AG News	Snippets	DBpedia	Twitter
Wide & Deep (Majority)	0.812	0.799	0.772	0.559
WESSTEC	**0.814**	**0.803**	**0.792**	**0.581**

Table 5.8: The categorization accuracy of WESSTEC against the Wide & Deep model trained on majority vote based training set

5.5 Summary and Conclusion

This chapter has investigated the following research question (cf. Section 1.2 (Research Question 3)):

- How to combine a deep neural network with a knowledge base to perform short text categorization without requiring any hand-labeled data?

In this regard, an approach called WESSTEC, which does not require any labeled data for short text categorization, has been proposed. It is a new paradigm for weakly supervised short text categorization using world knowledge. The model contains two main modules, namely, labeled data generation and categorization model which is based on Wide & Deep learning. The first module, i.e., the labeled data generation module responsible for generating labeled training data from unlabeled documents by utilizing three different embedding models, i.e., Word2Vec, LINE, Doc2Vec. Several features, i.e., words, entities, and their associated categories from the underlying knowledge base are extracted from the labeled documents to train the Wide & Deep categorization model. Finally, the new documents are categorized with the help of this model. In conclusion, the experimental results have proven that WESSTEC is capable of categorizing short text documents with high accuracy without requiring any hand-labeled data. Furthermore, the Wide & Deep model which has been proposed for a recommendation system can be successfully applied for the short text categorization problem. Moreover, the jointly trained Wide & Deep model outperforms the individual Wide-only and Deep-only models on each dataset. Further, it also significantly outperforms the categorization approaches which do not require any labeled data. The experimental results have also proven

that the Wide & Deep model is better suited for the short text categorization task by utilizing the generated labeled data in comparison to the baselines, e.g., BERT, CNN, etc.
As for future work, we aim to

1. improve the labeled data generation process by exploiting advanced weak supervision approaches such as Snorkel (Ratner et al., 2017);
2. adopt WESSTEC with different KBs;
3. evaluate the performance of WESSTEC on more text categorization benchmarks.

6 Conclusion

This chapter provides a summary of the research questions, which are introduced in Chapter 1.2 and the main contributions of the thesis. Moreover, the thesis is concluded with possible future directions.

6.1 Summary

The main focus of the thesis has been performing short text categorization effectively without requiring any labeled data. In this regard, two main challenges, namely, the *"labeled data requirement"* and the *"non-standard characteristics of short texts"* are faced. These challenges have been discussed in Chapter 1.1. To address these challenges the main research question of the thesis have been introduced as follows:

How to perform short text categorization effectively without requiring any hand-labeled data?

This broad research question has broken down into three specific research questions according to the challenges and tasks that they are concerned about. Each question and the findings are given as follows:

Research Question 1. *How can a knowledge base be utilized to categorize short texts without requiring any labeled training data?*

In Chapter 4, a novel short text categorization approach so-called KBSTC has been introduced. The model does not require any labeled data to perform short text categorization, instead, it exploits a knowledge base, i.e., Wikipedia as an external source. KBSTC is a probabilistic model, which does not have any training phase, further, it relies on the semantic similarity between the entities present in a text and predefined categories to assign the most relevant category for the given short text. The main contributions of this chapter include a new paradigm for short text categorization, which is based on a knowledge base without requiring any labeled data.

Research Question 2. *How to learn the semantic representation of short texts and predefined categories with the help of a knowledge base?*

In Section 4.6.1 we have aimed to address this research question by proposing a new entity and category embedding model. The embedding model enables to quantify the meaningful semantic relatedness between the short texts, which contain a set of entities and predefined categories. More specifically, the model first constructs entity-entity and entity-category networks by exploiting a hyperlink structure of a large knowledge base, i.e., Wikipedia. Then the model embeds these networks into a common vector space. After generating vector representation of each entity and category, the low dimensional representation of short texts is formed by leveraging the entities present in texts.

Research Question 3. *How to combine a deep neural network with a knowledge base to perform short text categorization without requiring any hand-labeled data?*

In Chapter 5, we have proposed a novel approach, which is called WESSTEC to address this research question. The model has been designed to perform weakly supervised short text categorization without requiring any hand-labeled data. It consists of two main modules: (1) a labeled data generation module, (2) a categorization model, which is based on a deep neural network. The contributions of this approach include a principle way of combining different weak supervising sources (e.g., embedding models) without requiring any manual effort to label unlabeled short text data and adaption of Wide & Deep model for short text categorization. As the name indicates, the first module is responsible for generating labeled short text data. To train the deep neural network model effectively, the representation of the generated labeled data by the first module is enriched by exploiting a knowledge base. As stated before, due to the main characteristics of short texts (e.g., sparsity), it is indispensable to enhance the text representations by leveraging external sources.

6.2 Outlook

Chapter 4 and Chapter 5 have already presented relevant future works for the proposed methods. However, this section aims to provide more general, possible future directions. In the following we define the future research questions:

Future Work 1. *How to generalize the proposed models to perform categorization of arbitrary-length documents?*

The main focus of this thesis has been to perform short text categorization without requiring any labeled data, instead, utilizing a knowledge base as an external source. In this regard, in

Chapter 4 and Chapter 5 two models have been presented for short text categorization. The experiments show that the proposed models can categorize short texts with high accuracy. As a subsequent step, we plan to generalize the proposed approaches for the categorization of arbitrary-length documents, including, long documents (e.g., news articles), etc. The main advantage of arbitrary-length documents is that they contain more contextual information which could help to ease the categorization process. However, there are still several challenges that need to be addressed in this regard. Long text often contain entities and words that do not contribute to the categorization task. In fact, such noise could harm the categorization performance. Hence, as for future work, we plan to investigate how to perform arbitrary-length text categorization by leveraging the most representative features of documents without requiring any hand-labeled data.

Future Work 2. *How to perform categorization of text documents into hierarchically related categories?*

In the course of this thesis, the proposed models (cf. Chapter 4 and Chapter 5) are designed for the categorization of the text documents into flat-structured categories. However, depending on the domain, the predefined categories could be hierarchically related. Therefore, we aim to extend the proposed models to enable the hierarchical categorization of text documents. To do so, as a first step we plan to integrate categories' hierarchy information into an embedding space. This will enable differentiating between the samples that come from the hierarchically related domains, e.g., basketball (which is a subcategory of sports) and sports. Further, for the categorization process, a categorization model, which is capable of categorizing samples into hierarchically related categories will be leveraged.

Future Work 3. *How to combine several different weak supervision sources to label unlabeled documents?*

In Chapter 5, a novel weakly supervised short text categorization model has been proposed. The model has two main components, namely, labeled data generation module and categorization module, which is based on a supervised model. The labeled data generation module utilizes three different embedding models, namely, word2vec, Doc2Vec, LINE (cf. Section 2.1.6 and Section 2.1.7) to heuristically label unlabeled short text documents without requiring any manual effort. As for future work, we aim to include more embedding models in order to enhance the labeling process. The main challenge of this idea is the effective combination of several different embedding models. There exist an approach called Snorkel (Ratner et al., 2017), which is designed to generate labeled data by leveraging several different labeling functions. However, this model requires manually designed labeling functions, i.e., weak supervision sources. In contrast to this model, we aim

to avoid any manual effort while combining the output of several different embedding models.

In this thesis, we have investigated the main research question of how to perform short text categorization effectively without requiring any labeled data using knowledge bases as an external source. In this context, two novel short text categorization models namely, Knowledge-Based Short Text Categorization (KBSTC) and Weakly Supervised Short Text Categorization using World Knowledge (WESSTEC) have been introduced and evaluated in this thesis. The extensive experiments have been conducted to assess the performance of the proposed short text categorization models. The experimental results of two models have demonstrated that it is possible to perform short text categorization effectively without requiring any hand-labeled data with high accuracy.

Bibliography

Aggarwal, C. C. (2018a). *Machine learning for text*. Springer.

Aggarwal, C. C. (2018b). *Neural Networks and Deep Learning - A Textbook*. Springer.

Aggarwal, C. C. & Zhai, C. (2012). A survey of text classification algorithms. In Aggarwal, C. C. & Zhai, C. (Eds.), *Mining Text Data*, pages 163–222. Springer.

Ahmad, W. U., Chang, K., & Wang, H. (2019). Context attentive document ranking and query suggestion. In Piwowarski, B., Chevalier, M., Gaussier, É., Maarek, Y., Nie, J., & Scholer, F. (Eds.), *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*, pages 385–394. ACM.

Alam, M. (2015). *Interactive Knowledge Discovery over Web of Data. (Découverte interactive de connaissances dans le web des données)*. PhD thesis, University of Lorraine, Nancy, France.

Almeida, F. & Xexéo, G. (2019). Word embeddings: A survey. *CoRR*, abs/1901.09069.

Arsov, N. & Mirceva, G. (2019). Network embedding: An overview. *CoRR*, abs/1911.11726.

Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., & Ives, Z. G. (2007). Dbpedia: A nucleus for a web of open data. In *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007.*, pages 722–735.

Bellomarini, L., Fakhoury, D., Gottlob, G., & Sallinger, E. (2019). Knowledge graphs and enterprise AI: the promise of an enabling technology. In *35th IEEE International Conference on Data Engineering, ICDE 2019, Macao, China, April 8-11, 2019*.

Berners-Lee, T., Fielding, R., Masinter, L., et al. (1998). Uniform resource identifiers (uri): Generic syntax.

Bollacker, K. D., Evans, C., Paritosh, P., Sturge, T., & Taylor, J. (2008). Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, pages 1247–1250.

Borst, P., Akkermans, H., & Top, J. L. (1997). Engineering ontologies. *Int. J. Hum. Comput. Stud.*, 46(2):365–406.

Bouaziz, A., ’e lia da Costa Pereira, C., Pallez, C. D. ., & ’e d é ric Precioso, F. (2016).

Introducing semantics in short text classification. In Gelbukh, A. F. (Ed.), *Computational Linguistics and Intelligent Text Processing - 17th International Conference, CICLing 2016, Konya, Turkey, April 3-9, 2016, Revised Selected Papers, Part II*, volume 9624 of *Lecture Notes in Computer Science*, pages 433–445. Springer.

Burel, G., Saif, H., & Alani, H. (2017). Semantic wide and deep learning for detecting crisis-information categories on social media. In *The Semantic Web - ISWC 2017 - 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part I*, pages 138–155.

Carlson, A., Betteridge, J., Wang, R. C., Jr., E. R. H., & Mitchell, T. M. (2010). Coupled semi-supervised learning for information extraction. In *Proceedings of the Third International Conference on Web Search and Web Data Mining, WSDM 2010, New York, NY, USA, February 4-6, 2010*, pages 101–110.

Chakravarthy, V., Joshi, S., Ramakrishnan, G., Godbole, S., & Balakrishnan, S. (2008). Learning decision lists with known rules for text mining. In *Third International Joint Conference on Natural Language Processing, IJCNLP 2008, Hyderabad, India, January 7-12, 2008*, pages 835–840. The Association for Computer Linguistics.

Chang, M., Ratinov, L., Roth, D., & Srikumar, V. (2008). Importance of semantic representation: Dataless classification. In Fox, D. & Gomes, C. P. (Eds.), *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, pages 830–835. AAAI Press.

Chen, Jin, X., & Shen, D. (2011). Short text classification improved by learning multi-granularity topics. In Walsh, T. (Ed.), *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 1776–1781. IJCAI / AAAI.

Chen, J., Hu, Y., Liu, J., Xiao, Y., & Jiang, H. (2019). Deep short text classification with knowledge powered attention. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 6252–6259.

Chen, P., Sun, Z., Bing, L., & Yang, W. (2017). Recurrent attention network on memory for aspect sentiment analysis. In Palmer, M., Hwa, R., & Riedel, S. (Eds.), *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 452–461. Association for Computational Linguistics.

Chen, X., Xia, Y., Jin, P., & Carroll, J. A. (2015a). Dataless text classification with descriptive LDA. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, pages 2224–2231. AAAI Press.

Chen, X., Xia, Y., Jin, P., & Carroll, J. A. (2015b). Dataless text classification with descriptive lda. In *AAAI*, pages 2224–2231. AAAI Press.

Cheng, H., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., Anderson, G., Corrado, G., Chai, W., Ispir, M., Anil, R., Haque, Z., Hong, L., Jain, V., Liu, X., & Shah, H. (2016). Wide & deep learning for recommender systems. In Karatzoglou, A., Hidasi, B., Tikk, D., Shalom, O. S., Roitman, H., Shapira, B., & Rokach, L. (Eds.), *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, DLRS@RecSys 2016, Boston, MA, USA, September 15, 2016*, pages 7–10. ACM.

Cilibrasi, R. L. & Vitanyi, P. M. (2007). The google similarity distance. *IEEE Transactions on knowledge and data engineering*, 19(3):370–383.

Cohen, W. W. & Singer, Y. (1999). Context-sensitive learning methods for text categorization. *ACM Transactions on Information Systems (TOIS)*, 17(2):141–173.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. P. (2011). Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537.

Conneau, A., Schwenk, H., Barrault, L., & LeCun, Y. (2016). Very deep convolutional networks for natural language processing. *CoRR*, abs/1606.01781.

Conneau, A., Schwenk, H., Barrault, L., & LeCun, Y. (2017). Very deep convolutional networks for text classification. In Lapata, M., Blunsom, P., & Koller, A. (Eds.), *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*, pages 1107–1116. Association for Computational Linguistics.

Cortes, C. & Vapnik, V. (1995). Support-vector networks. *Mach. Learn.*, 20(3):273–297.

Cui, P., Wang, X., Pei, J., & Zhu, W. (2019). A survey on network embedding. *IEEE Trans. Knowl. Data Eng.*, 31(5):833–852.

Dai, Z., Sun, A., & Liu, X. (2013). Crest: Cluster-based representation enrichment for short text classification. In Pei, J., Tseng, V. S., Cao, L., Motoda, H., & Xu, G. (Eds.), *Advances in Knowledge Discovery and Data Mining, 17th Pacific-Asia Conference, PAKDD 2013, Gold Coast, Australia, April 14-17, 2013, Proceedings, Part II*, volume 7819 of *Lecture Notes in Computer Science*, pages 256–267. Springer.

de Vries, G. K. D. (2013). A fast approximation of the weisfeiler-lehman graph kernel for RDF data. In Blockeel, H., Kersting, K., Nijssen, S., & ’y, F. Z. (Eds.), *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part I*, volume 8188 of *Lecture Notes in Computer Science*, pages 606–621. Springer.

de Vries, G. K. D. & de Rooij, S. (2015). Substructure counting graph kernels for machine learning from RDF data. *J. Web Semant.*, 35:71–84.

Deibe, M. A. (2018). *Query processing over graph-structured data on the web*. PhD thesis, Karlsruhe Institute of Technology, Germany.

Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2019). BERT: pre-training of deep bidirectional transformers for language understanding. In Burstein, J., Doran, C., & Solorio, T. (Eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10:1895–1923.

Druck, G., Mann, G. S., & McCallum, A. (2008). Learning from labeled features using generalized expectation criteria. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2008, Singapore, July 20-24, 2008*, pages 595–602. ACM.

Duque, A. B., Santos, L. L. J., Macêdo, D., & Zanchettin, C. (2019). Squeezed very deep convolutional neural networks for text classification. In Tetko, I. V., Kurková, V., Karpov, P., & Theis, F. J. (Eds.), *Artificial Neural Networks and Machine Learning - ICANN 2019: Theoretical Neural Computation - 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17-19, 2019, Proceedings, Part I*, volume 11727 of *Lecture Notes in Computer Science*, pages 193–207. Springer.

Dürst, M. J. & Suignard, M. (2005). Internationalized resource identifiers (iris). *RFC*, 3987:1–46.

Ehrlinger, L. & Wöß, W. (2016). Towards a definition of knowledge graphs. In Martin, M., Cuquet, M., & Folmer, E. (Eds.), *Joint Proceedings of the Posters and Demos Track of the 12th International Conference on Semantic Systems - SEMANTiCS2016 and the 1st International Workshop on Semantic Change & Evolving Semantics (SuCCESS'16) co-located with the 12th International Conference on Semantic Systems (SEMANTiCS 2016), Leipzig, Germany, September 12-15, 2016*.

Fabian, M., Gjergji, K., Gerhard, W., et al. (2007). Yago: A core of semantic knowledge unifying wordnet and wikipedia. In *16th International World Wide Web Conference, WWW*.

Färber, M., Bartscherer, F., Menne, C., & Rettinger, A. (2018). Linked data quality of dbpedia, freebase, opencyc, wikidata, and YAGO. *Semantic Web*, 9(1):77–129.

Färber, M., Ell, B., Menne, C., & Rettinger, A. (2015). A comparative survey of dbpedia, freebase, opencyc, wikidata, and yago. *Semantic Web Journal*.

Fellbaum, C. (Ed.) (1998). *WordNet: an electronic lexical database*. MIT Press.

Gabrilovich, E., Markovitch, S., et al. (2007). Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJcAI*, volume 7, pages 1606–1611.

Gao, H., Wang, Z., & Ji, S. (2018). Large-scale learnable graph convolutional networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discov-*

ery & Data Mining, KDD 2018, London, UK, August 19-23, 2018, pages 1416–1424. ACM.

Genkin, A., Lewis, D. D., & Madigan, D. (2007). Large-scale bayesian logistic regression for text categorization. *Technometrics*, 49(3):291–304.

Gomaa, W. H., Fahmy, A. A., et al. (2013). A survey of text similarity approaches. *International Journal of Computer Applications*, 68(13):13–18.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

Gruber, T. R. et al. (1993). A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–221.

Gutmann, M. & Hyvärinen, A. (2012). Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *J. Mach. Learn. Res.*, 13:307–361.

Harispe, S., Ranwez, S., Janaqi, S., & Montmain, J. (2013). Semantic measures for the comparison of units of language, concepts or entities from text and knowledge base analysis. *CoRR*, abs/1310.1285.

Hingmire, S. & Chakraborti, S. (2014). Topic labeled text classification: a weakly supervised approach. In Geva, S., Trotman, A., Bruza, P., Clarke, C. L. A., & Järvelin, K. (Eds.), *The 37th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '14, Gold Coast , QLD, Australia - July 06 - 11, 2014*, pages 385–394. ACM.

Hingmire, S., Chougule, S., Palshikar, G. K., & Chakraborti, S. (2013). Document classification by topic labeling. In *SIGIR*, pages 877–880. ACM.

Hitzler, P., Krötzsch, M., & Rudolph, S. (2010). *Foundations of Semantic Web Technologies*. Chapman and Hall/CRC Press.

Hogan, A., Blomqvist, E., Cochez, M., d'Amato, C., de Melo, G., Gutierrez, C., Gayo, J. E. L., Kirrane, S., Neumaier, S., Polleres, A., et al. (2020). Knowledge graphs. *arXiv preprint arXiv:2003.02320*.

Hu, L., Yang, T., Shi, C., Ji, H., & Li, X. (2019). Heterogeneous graph attention networks for semi-supervised short text classification. In Inui, K., Jiang, J., Ng, V., & Wan, X. (Eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 4820–4829. Association for Computational Linguistics.

Hu, Z., Huang, P., Deng, Y., Gao, Y., & Xing, E. P. (2015). Entity hierarchy embedding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1292–1300. The Association for Computer

Linguistics.

Ifrim, G., Bakir, G. H., & Weikum, G. (2008). Fast logistic regression for text categorization with variable-length n-grams. In Li, Y., Liu, B., & Sarawagi, S. (Eds.), *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008*, pages 354–362. ACM.

Jacobs, I. & Walsh, N. (2004). Architecture of the world wide web, volume one. w3c recommendation. *World Wide Web Consortium (W3C)*.

Joachims, T. (1997). A probabilistic analysis of the rocchio algorithm with TFIDF for text categorization. In Fisher, D. H. (Ed.), *Proceedings of the Fourteenth International Conference on Machine Learning (ICML 1997), Nashville, Tennessee, USA, July 8-12, 1997*, pages 143–151. Morgan Kaufmann.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In Nedellec, C. & Rouveirol, C. (Eds.), *Machine Learning: ECML-98, 10th European Conference on Machine Learning, Chemnitz, Germany, April 21-23, 1998, Proceedings*, volume 1398 of *Lecture Notes in Computer Science*, pages 137–142. Springer.

Jurafsky, D. & Martin, J. H. (2009). *Speech and Language Processing (2Nd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014). A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 655–665. The Association for Computer Linguistics.

Kim, Y. (2014). Convolutional neural networks for sentence classification. In Moschitti, A., Pang, B., & Daelemans, W. (Eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751. ACL.

Koller, D. & Sahami, M. (2007). Hierarchically classifying documents with very few words. In *ICML Conference*.

Komarek, P. & Moore, A. (2003). Fast logistic regression for data mining, text classification and link detection. *Proceedings of NIPS2003*.

Kowsari, K., Meimandi, K. J., Heidarysafa, M., Mendu, S., Barnes, L. E., & Brown, D. E. (2019). Text classification algorithms: A survey. *Information*, 10(4):150.

Landauer, T. K. & Dumais, S. T. (1997). A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2):211.

Le, Q. V. & Mikolov, T. (2014). Distributed representations of sentences and documents. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014,*

Beijing, China, 21-26 June 2014, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 1188–1196. JMLR.org.

Lei, T., Barzilay, R., & Jaakkola, T. S. (2015). Molding cnns for text: non-linear, non-consecutive convolutions. In Màrquez, L., Callison-Burch, C., Su, J., Pighin, D., & Marton, Y. (Eds.), *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1565–1575. The Association for Computational Linguistics.

Lenat, D. B. (1995). CYC: A large-scale investment in knowledge infrastructure. *Commun. ACM*, 38(11):32–38.

Li, C., Xing, J., Sun, A., & Ma, Z. (2016a). Effective document labeling with very few seed words: A topic model approach. In Mukhopadhyay, S., Zhai, C., Bertino, E., Crestani, F., Mostafa, J., Tang, J., Si, L., Zhou, X., Chang, Y., Li, Y., & Sondhi, P. (Eds.), *Proceedings of the 25th ACM International Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016*, pages 85–94. ACM.

Li, C., Xing, J., Sun, A., & Ma, Z. (2016b). Effective document labeling with very few seed words: A topic model approach. In Mukhopadhyay, S., Zhai, C., Bertino, E., Crestani, F., Mostafa, J., Tang, J., Si, L., Zhou, X., Chang, Y., Li, Y., & Sondhi, P. (Eds.), *Proceedings of the 25th ACM International Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016*, pages 85–94. ACM.

Li, Y. H. & Jain, A. K. (1998). Classification of text documents. *The Computer Journal*, 41(8):537–546.

Li, Y., Zheng, R., Tian, T., Hu, Z., Iyer, R., & Sycara, K. P. (2016c). Joint embedding of hierarchical categories and entities for concept categorization and dataless classification. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 2678–2688.

Li, Y., Zheng, R., Tian, T., Hu, Z., Iyer, R., & Sycara, K. P. (2016d). Joint embedding of hierarchical categories and entities for concept categorization and dataless classification. In Calzolari, N., Matsumoto, Y., & Prasad, R. (Eds.), *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 2678–2688. ACL.

Liu, P., Qiu, X., & Huang, X. (2016). Recurrent neural network for text classification with multi-task learning. In Kambhampati, S. (Ed.), *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 2873–2879. IJCAI/AAAI Press.

Liu, Y. & Lapata, M. (2019). Text summarization with pretrained encoders. In Inui, K., Jiang, J., Ng, V., & Wan, X. (Eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural*

Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019, pages 3728–3738. Association for Computational Linguistics.

Lu, Z., Zhu, Y., Pan, S. J., Xiang, E. W., Wang, Y., & Yang, Q. (2014). Source free transfer learning for text classification. In Brodley, C. E. & Stone, P. (Eds.), *AAAI*. AAAI Press.

Lukovnikov, D., Fischer, A., & Lehmann, J. (2019). Pretrained transformers for simple question answering over knowledge graphs. In Ghidini, C., Hartig, O., Maleshkova, M., Svátek, V., Cruz, I. F., Hogan, A., Song, J., Lefrançois, M., & Gandon, F. (Eds.), *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part I*, volume 11778 of *Lecture Notes in Computer Science*, pages 470–486. Springer.

Ma, Y., Peng, H., & Cambria, E. (2018). Targeted aspect-based sentiment analysis via embedding commonsense knowledge into an attentive LSTM. In McIlraith, S. A. & Weinberger, K. Q. (Eds.), *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5876–5883. AAAI Press.

Mahdisoltani, F., Biega, J., & Suchanek, F. M. (2015). YAGO3: A knowledge base from multilingual wikipedias. In *CIDR 2015, Seventh Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 4-7, 2015, Online Proceedings*. www.cidrdb.org.

Max Schmachtenberg, Christian Bizer, A. J. & Cyganiak, R. (2014).

Mendes, P. N., Jakob, M., 'e s Garc ´ i a - Silva, A., & Bizer, C. (2011). Dbpedia spotlight: shedding light on the web of documents. In Ghidini, C., Ngomo, A. . C. N., Lindstaedt, S. N., & Pellegrini, T. (Eds.), *Proceedings the 7th International Conference on Semantic Systems, I-SEMANTICS 2011, Graz, Austria, September 7-9, 2011*, ACM International Conference Proceeding Series, pages 1–8. ACM.

Meng, Y., Shen, J., Zhang, C., & Han, J. (2018). Weakly-supervised neural text classification. In *CIKM*.

Meng, Y., Shen, J., Zhang, C., & Han, J. (2019). Weakly-supervised hierarchical text classification. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 6826–6833. AAAI Press.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013a). Efficient estimation of word representations in vector space. In Bengio, Y. & LeCun, Y. (Eds.), *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In Burges, C. J. C., Bottou, L., Ghahramani, Z., & Weinberger, K. Q. (Eds.), *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 3111–3119.

Nies, T. D., Beecks, C., Neve, W. D., Seidl, T., Mannens, E., & de Walle, R. V. (2014). Towards named-entity-based similarity measures: Challenges and opportunities. In Alonso, O., Kamps, J., & Karlgren, J. (Eds.), *Proceedings of the 7th International Workshop on Exploiting Semantic Annotations in Information Retrieval, ESAIR '14, Shanghai, China, November 7, 2014*, pages 9–11. ACM.

Nigam, K., McCallum, A., Thrun, S., & Mitchell, T. M. (2000). Text classification from labeled and unlabeled documents using EM. *Mach. Learn.*, 39(2/3):103–134.

Paulheim, H. (2017). Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic Web*.

Peng, H., Li, J., He, Y., Liu, Y., Bao, M., Wang, L., Song, Y., & Yang, Q. (2018). Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In Champin, P., Gandon, F. L., Lalmas, M., & Ipeirotis, P. G. (Eds.), *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, pages 1063–1072. ACM.

Perozzi, B., Al-Rfou, R., & Skiena, S. (2014). Deepwalk: online learning of social representations. In Macskassy, S. A., Perlich, C., Leskovec, J., Wang, W., & Ghani, R. (Eds.), *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pages 701–710. ACM.

Phan, X. H., Nguyen, M. L., & Horiguchi, S. (2008). Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In Huai, J., Chen, R., Hon, H., Liu, Y., Ma, W., Tomkins, A., & Zhang, X. (Eds.), *Proceedings of the 17th International Conference on World Wide Web, WWW 2008, Beijing, China, April 21-25, 2008*, pages 91–100. ACM.

Rabinovich, E., Sznajder, B., Spector, A., Shnayderman, I., Aharonov, R., Konopnicki, D., & Slonim, N. (2018). Learning concept abstractness using weak supervision. In Riloff, E., Chiang, D., Hockenmaier, J., & Tsujii, J. (Eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 4854–4859. Association for Computational Linguistics.

Ratner, A., Bach, S. H., Ehrenberg, H. R., Fries, J. A., Wu, S., & Ré, C. (2017). Snorkel: Rapid training data creation with weak supervision. *Proc. VLDB Endow.*, pages 269–282.

Reimers, N. & Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. In Inui, K., Jiang, J., Ng, V., & Wan, X. (Eds.), *Proceedings of the 2019*

Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019, pages 3980–3990. Association for Computational Linguistics.

Resnik, P. (1995). Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI 95, Montréal Québec, Canada, August 20-25 1995, 2 Volumes*, pages 448–453. Morgan Kaufmann.

Ristoski, P. & Paulheim, H. (2016). Rdf2vec: RDF graph embeddings for data mining. In Groth, P. T., Simperl, E., Gray, A. J. G., Sabou, M., Krötzsch, M., Lécué, F., Flöck, F., & Gil, Y. (Eds.), *The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016, Proceedings, Part I*, volume 9981 of *Lecture Notes in Computer Science*, pages 498–514.

Röder, M., Usbeck, R., Hellmann, S., Gerber, D., & Both, A. (2014). N^3 - A collection of datasets for named entity recognition and disambiguation in the NLP interchange format. In Calzolari, N., Choukri, K., Declerck, T., Loftsson, H., Maegaard, B., Mariani, J., Moreno, A., Odijk, J., & Piperidis, S. (Eds.), *Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC 2014, Reykjavik, Iceland, May 26-31, 2014*, pages 3529–3533. European Language Resources Association (ELRA).

Rosenblatt, F. (1961). Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, Cornell Aeronautical Lab Inc Buffalo NY.

Ross, S. (1976). A first course in probability. *HwaTia, Taiwan*, pages 136–137.

Sahami, M., Dumais, S., Heckerman, D., & Horvitz, E. (1998). A bayesian approach to filtering junk e-mail. In *Learning for Text Categorization: Papers from the 1998 workshop*, volume 62, pages 98–105. Madison, Wisconsin.

Sakor, A., Mulang, I. O., Singh, K., Shekarpour, S., Vidal, M., Lehmann, J., & Auer, S. (2019). Old is gold: Linguistic driven approach for entity and relation linking of short text. In Burstein, J., Doran, C., & Solorio, T. (Eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 2336–2346. Association for Computational Linguistics.

Sarkar, D. (2016). Text analytics with python.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47.

Singh, A., Nowak, R. D., & Zhu, X. (2008). Unlabeled data: Now it helps, now it doesn't. In *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancou-*

ver, British Columbia, Canada, December 8-11, 2008, pages 1513–1520.

Skansi, S. (2018). *Introduction to Deep Learning - From Logical Calculus to Artificial Intelligence*. Undergraduate Topics in Computer Science. Springer.

Song, G., Ye, Y., Du, X., Huang, X., & Bie, S. (2014). Short text classification: A survey. *Journal of multimedia*, 9(5):635–644.

Song, Y. & Roth, D. (2014). On dataless hierarchical text classification. In Brodley, C. E. & Stone, P. (Eds.), *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*, pages 1579–1585. AAAI Press.

Srinivasan, S. (2017). *Guide to Big Data Applications*, volume 26. Springer.

Studer, R., Benjamins, V. R., & Fensel, D. (1998). Knowledge engineering: Principles and methods. *Data Knowl. Eng.*, 25(1-2):161–197.

Su, J. & Zhang, H. (2006). A fast decision tree learning algorithm. In *Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, July 16-20, 2006, Boston, Massachusetts, USA*, pages 500–505. AAAI Press.

Suchanek, F. M., Kasneci, G., & Weikum, G. (2007). Yago: a core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, pages 697–706.

Tahery, S. & Farzi, S. (2020). Customized query auto-completion and suggestion - A review. *Inf. Syst.*, 87.

Tang, J., Qu, M., & Mei, Q. (2015a). PTE: predictive text embedding through large-scale heterogeneous text networks. In Cao, L., Zhang, C., Joachims, T., Webb, G. I., Margineantu, D. D., & Williams, G. (Eds.), *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, pages 1165–1174. ACM.

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., & Mei, Q. (2015b). LINE: large-scale information network embedding. In Gangemi, A., Leonardi, S., & Panconesi, A. (Eds.), *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015*, pages 1067–1077. ACM.

Teng, Z., Vo, D., & Zhang, Y. (2016). Context-sensitive lexicon features for neural sentiment analysis. In Su, J., Carreras, X., & Duh, K. (Eds.), *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1629–1638. The Association for Computational Linguistics.

Türker, R., Mehwish, Alam, & Sack, H. (2020). Weakly supervised short text categorization using world knowledge. In *ISWC 2020*.

Türker, R., Zhang, L., Koutraki, M., & Sack, H. (2018a). TECNE: knowledge based text

classification using network embeddings. In *Proceedings of the EKAW 2018 Posters and Demonstrations Session co-located with 21st International Conference on Knowledge Engineering and Knowledge Management (EKAW 2018), Nancy, France, November 12-16, 2018*, volume 2262, pages 53–56.

Türker, R., Zhang, L., Koutraki, M., & Sack, H. (2018b). "the less is more" for text classification,. In *Proceedings of the Posters and Demos Track of the 14th International Conference on Semantic Systems co-located with the 14th International Conference on Semantic Systems (SEMANTiCS 2018), Vienna, Austria, September 10-13, 2018*.

Türker, R., Zhang, L., Koutraki, M., & Sack, H. (2019). Knowledge-based short text categorization using entity and category embedding. In Hitzler, P., 'a ndez, M. F., Janowicz, K., Zaveri, A., Gray, A. J., 'o pez, V. L., Haller, A., & Hammar, K. (Eds.), *The Semantic Web - 16th International Conference, ESWC 2019, Portoro v z, Slovenia, June 2-6, 2019, Proceedings*, volume 11503 of *Lecture Notes in Computer Science*, pages 346–362. Springer.

Usbeck, R., Röder, M., Ngomo, A. N., Baron, C., Both, A., Brümmer, M., Ceccarelli, D., Cornolti, M., Cherix, D., Eickmann, B., Ferragina, P., Lemke, C., Moro, A., Navigli, R., Piccinno, F., Rizzo, G., Sack, H., Speck, R., Troncy, R., Waitelonis, J., & Wesemann, L. (2015). GERBIL: general entity annotator benchmarking framework. In Gangemi, A., Leonardi, S., & Panconesi, A. (Eds.), *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015*, pages 1133–1143. ACM.

Vrandecic, D. & Krötzsch, M. (2014). Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

Waitelonis, J., Exeler, C., & Sack, H. (2015). Linked data enabled generalized vector space model to improve document retrieval. *NLP & DBpedia@ ISWC*, 15:34–44.

Wang, C., Song, Y., Li, H., Zhang, M., & Han, J. (2016a). Text classification with heterogeneous information network kernels. In Schuurmans, D. & Wellman, M. P. (Eds.), *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 2130–2136. AAAI Press.

Wang, J., Wang, Z., Zhang, D., & Yan, J. (2017). Combining knowledge with deep convolutional neural networks for short text classification. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 2915–2921.

Wang, P., Xu, B., Xu, J., Tian, G., Liu, C.-L., & Hao, H. (2016b). Semantic expansion using word embedding clustering and convolutional neural network for improving short text classification. *Neurocomputing*.

Wang, R., Li, Z., Cao, J., Chen, T., & Wang, L. (2019). Convolutional recurrent neural networks for text classification. In *International Joint Conference on Neural Networks,*

IJCNN 2019 Budapest, Hungary, July 14-19, 2019, pages 1–6. IEEE.

Wang, X., Liu, Y., Sun, C., Wang, B., & Wang, X. (2015). Predicting polarities of tweets by composing word embeddings with long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1343–1353. The Association for Computer Linguistics.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P. S. (2019). A comprehensive survey on graph neural networks. *CoRR*, abs/1901.00596.

Xuan, J., Jiang, H., Ren, Z., Yan, J., & Luo, Z. (2017). Automatic bug triage using semi-supervised text classification. *CoRR*, abs/1704.04769.

Yao, L., Mao, C., & Luo, Y. (2019). Graph convolutional networks for text classification. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 7370–7377. AAAI Press.

Zeng, J., Li, J., Song, Y., Gao, C., Lyu, M. R., & King, I. (2018). Topic memory networks for short text classification. In *EMNLP*.

Zhang, R., Lee, H., & Radev, D. R. (2016). Dependency sensitive convolutional neural networks for modeling sentences and documents. In Knight, K., Nenkova, A., & Rambow, O. (Eds.), *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 1512–1521. The Association for Computational Linguistics.

Zhang, X. & LeCun, Y. (2015). Text understanding from scratch. *CoRR*, abs/1502.01710.

Zhang, X., Zhao, J. J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., & Garnett, R. (Eds.), *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 649–657.

Zhang, Y. & Wallace, B. C. (2017). A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. In Kondrak, G. & Watanabe, T. (Eds.), *Proceedings of the Eighth International Joint Conference on Natural Language Processing, IJCNLP 2017, Taipei, Taiwan, November 27 - December 1, 2017 - Volume 1: Long Papers*, pages 253–263. Asian Federation of Natural Language Processing.

Zhou, C., Sun, C., Liu, Z., & Lau, F. C. M. (2015). A C-LSTM neural network for text classification. *CoRR*, abs/1511.08630.

Zukarnain, N., Abbas, B. S., Wayan, S., Trisetyarso, A., & Kang, C. H. (2019). Spelling

checker algorithm methods for many languages. In *2019 International Conference on Information Management and Technology (ICIMTech)*, volume 1, pages 198–201. IEEE.