technische universität
dortmund

Master thesis

A Machine Learning Approach for Aspect-based Sentiment Analysis on Social Media

Weihan Pang

March 2018

Gutachter:
Prof. Dr. Katharina Morik
M.Sc. Lukas Pfahler

Technische Universität Dortmund
Fakultät für Informatik
Lehrstuhl für Künstliche Intelligenz (LS-8)
http://www-ai.cs.tu-dortmund.de

Contents

Chapter 1

Introduction

In the connected modern world, customer feedback is a valuable source for insights on the quality of products or services. This feedback allows other customers to benefit from the experiences of others and enables businesses to react on requests, complaints or recommendations. However, the more people use a product or service, the more feedback is generated, which results in the major challenge of analyzing huge amounts of feedback in an efficient, but still meaningful way.

Recent years has seen rapid growth of research on sentiment analysis. Sentiment analysis has both business importance and academic interest [56]. The typical sentiment analysis focus on predicting the positive or negative polarity of the given texts. This task works for the text that has only one aspect and polarity. A more general and complicated task is to predict the aspects mentioned in a sentence and the sentiments associated with each one of them. This generalized task is called *Aspect-based Sentiment Analysis (ABSA)* [56], i.e., mining opinions from text about specific entities and their aspects, which can provide valuable insights to both consumers and businesses.

In the Germeval Shared Task 2017 competition[1], a shared task on aspect-based sentiment in social media customer feedback is proposed, which is to automatically analyze customer reviews about "Deutsche Bahn" - the german public train operator with about two billion passengers each year. The Germeval 2017 Task consists of four subtasks: *relevance classification* (subtask A); *document-level polarity* (subtask B); *aspect-level polarity* (subtask C); *opinion target extraction* (subtask D). This thesis aims to find appropriate machine learning approaches for these four subtasks.

[1]https://sites.google.com/view/germeval2017-absa/home

1.1 Motivation

The *relevance classification, document-level polarity and aspect-level polarity* subtasks are essentially text classification problems. Text classification is a classic topic for natural language processing, in which one needs to assign predefined categories to documents.

The traditional machine learning methods are applied for text classification, such as Naive Bayes (NB) [32], Maximum Entropy [2] and Support Vector Machines (SVM) [21] using features like unigrams, bigrams, i.e., each document is represented as word vectors. The last subtask, *opinion target extraction* can be treated as a sequence labeling problem, which is often addressed by Conditional Random Fileds (CRF) [25] or Hidden Markov Model [3]. These traditional machine learning algorithms cannot learn complicated invariant features, they are incorporated with large amounts of task-specific knowledge in the form of handcrafted feature engineering and data pre-processing. Therefore, feature extraction and selection play an important role for their classification performance.

In addition, online reviews are short texts that contain only a few sentences or even a few words. Sentiment analysis of short texts face a challenge because of the limited contextual information that they normally contain [10]. Solutions for the challenges incurred in these problems come from neural networks [23]. The neural networks can extract relevant features from words and sentences of any size [10]. Therefore one has to worry less about the feature engineering part than the traditional machine learning approaches.

Neural Networks are good at learning invariant features and contexts to predict sentiment, but not always optimal for classification or sequence labeling. While Support Vector Machines are good at producing decision surfaces from well-behaved feature vectors, but cannot learn complicated invariances [17]. Target to this problem, Convolutional Neural Networks (CNN) and Support Vector Machines (SVM) are combined for the first three classification subtasks, CNN is utilized as feature extractor to learn feature vector representations corresponding to each sentence. The learned vector representations are then fed to a SVM classifier as features for topic or sentiment classification [4]. Besides, the opinion target extraction subtask is treated as a sequence labeling task, the sequence labeling classifier is trained using Conditional Random Fields (CRF). The output of a Bidirectional Long Short Term Memory (BiLSTM) model is used as additional features. Because BiLSTM is good at modeling context information of each word [19], while Conditional Random Fields is good at sequence labeling for whole sentence.

1.2 Structure of this Thesis

This thesis consists of 7 chapters. First, the background of sentiment analysis is introduced, including text classification and sequential labeling problems of natural language processing field. Then the formal definition of sentiment analysis on different levels are introduced, and the Germeval subtasks are described in detail.

In Chapter 3, the architecture of machine learning approaches will be presented, including word representations, neural networks and classifiers. Then the four subtasks are constructed into a whole system. The structure of the system is described in Chapter 4 and which approaches are applied to which model of the system will be discussed.

Chapter 5 introduce the implementation part and the experiment results are discussed in Chapter 6. Finally, Chapter 7 concludes this thesis and gives an outlook to future research.

Chapter 2

Background

2.1 Sentiment Analysis and Opinion Mining

Sentiment analysis, also called opinion mining, is essentially a text classification problem. Sentiment analysis has been a widely researched area with large amounts of literatures on document, sentence, phrase and aspect level analyses in different domains. Sentiment classification is the most extensive topic of sentiment analysis. It aims to classify an opinion document as expressing a positive or negative opinion or sentiment [29]. In general, sentiment analysis has been investigated mainly at three levels: *Document-level Sentiment Analysis, Sentence-level Sentiment Analysis and Aspect-level Sentiment Analysis.* This thesis focuses on *Document-level and Aspect-level Sentiment Analysis* for subtask B and C.

Document-level Sentiment Analysis The task at this level is to classify whether a whole opinion document expresses a positive, negative or neutral sentiment. Neutral usually means no opinion. It is commonly known as document-level sentiment classification because it considers the whole document as a basic information unit, i.e., each document expresses opinions on a single entity (e.g., a single product or a single aspect of the product). Thus, it is not applicable to documents which evaluate or compare multiple entities.

Sentence-level Sentiment Analysis The task at this level goes to the sentences and determines whether each sentence expresses a positive, negative, or neutral opinion. If each sentence of a document contains a single opinion, sentence level sentiment classification goes further than document level sentiment classification as it moves closer to opinion targets and sentiments on the targets. However, there is no fundamental difference between document and sentence level classifications because sentences are just short documents [29]. So it is not considered in this thesis, since the datasets are social media comments that contain only a few sentences or even a few words.

Aspect-level Sentiment Analysis However, in most applications, the user needs to know additional details, i.e., what entities or aspects of entities are liked and disliked. As the document level, the sentence level analysis still does not do that. Although a sentence may have an overall positive or negative tone, some of its components may express opposite opinions. For example, although the sentence "Although the service is not that great, i still love this restaurant" clearly has a positive tone, we cannot say that this sentence is entirely positive. In fact, the sentence is positive about the restaurant, but negative about its service [29].

If we go to the aspect-level sentiment analysis, the problem is solved. It is highly dependent on both content and its aspects. It refers to determining the opinions or sentiments expressed on different aspects. An aspect is represented as an entity and aspect pair (E#A), e.g., the service and the food of the restaurant are respectively represented as *Restaurant#Service* and *Restaurant#Food*.

2.2 Natural Language Processing

Natural Language Processing, or NLP for short, is an area of research and application that explores how computers can be used to analyze, understand and manipulate natural language text or speech [7]. By utilizing NLP, developers can organize and structure knowledge to perform tasks such as translation, named entity recognition, relationship extraction, sentiment analysis, speech recognition, and topic segmentation.

2.2.1 Text Classification

Text classification is an important task in many areas of natural language processing, including sentiment analysis, question answering, or dialog management. In classification tasks, one needs to produce a classification function which can give the correlation between a certain feature and a class.

Let x denote a text and y denote the corresponding class. A training example is a pair (x_i, y_i) consisting of a text and its associated class label. We assume that the training examples are drawn independently and identically from the joint distribution $P(x_i, y_i)$, and we will refer to a set of N such examples as the training data. A classifier is a function h that maps from text to classes. The goal of the learning process is to find an h that correctly predicts the class $y = h(x)$ of new text x [9].

Traditional text classification mainly classifies documents of different topics, e.g., politics, sciences and sports. In such classification, topic-related words are the key features [29]. In this thesis, we focus on applying machine learning techniques to the sentiment classification problem. A challenging aspect of this problem that seems to distinguish sentiment classification from traditional topic-based classification is that topics are often identifiable by keywords alone, sentiment can be expressed in a more subtle manner [35]. In sentiment classification, sentiment or opinion words that indicate positive or negative opinions are more important, e.g., great, excellent, amazing, bad, worst, etc. [29]. For example, the sentence: *"Although the service of this restaurant is not that great, i still love its food."* is classifiable as restaurant relevant. In addition to the topic, the sentiment of the document can also be used to classify this text. A positive sentiment can be assigned to the entity food, whereas a negative one could be assigned to service. Furthermore, sentiment analysis can deal with more implicit expressions. For example, the sentence *"How could anyone sit through this movie?"* contains no single word that is obviously negative. Thus, sentiment seems to require more understanding than the usual topic-based classification [35].

In general, for the sentiment classification, sentiment is incredibly subjective, and depends upon a number of variables. Therefore, sentiment analysis goes far beyond traditional topic-based text classification, and the traditional text classification methods should be augmented in order to advance towards the problem of textual emotion recognition [48].

2.2.2 Sequence labeling

In many NLP problems, we would like to model pairs of sequences. Sequence labeling or tagging such as Part-of-speech tagging is perhaps the earliest, and most famous, example of this type of problem. The goal of sequence labeling is to build a model whose input is a sentence, for example:

The train is coming.

and output is a label sequence, for example:

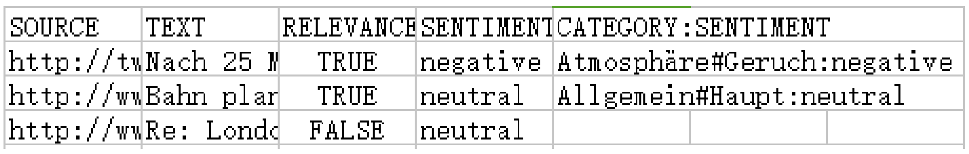
$$DNVAdj$$

(here we use D for a determiner, N for noun, and V for verb, and Adj for adjective).

Let $(x_i, y_i)_{i=1}^{N}$ be a set of N training data. Each training data is a sequence pair (x_i, y_i), where $x_i = x_{i_1}, x_{i_2}, ..., x_{i_l}$ and $y_i = y_{i_1}, y_{i_2}, ..., y_{i_l}$. The label sequence is the same length as the input sentence, therefore a single label is specified for each word in the sentence. The labeling classifier model h is constructed from the given training data. For an input sequence x, this model can correctly predict its corresponding labeling sequence $y = h(x)$ [9].

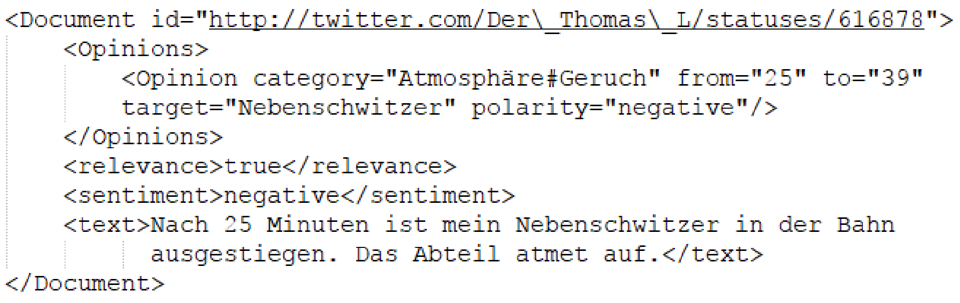
2.3 Task Description

The Germeval Task 2017 specifically focuses on the natural language processing problem of german language. The datasets and four subtasks are desribed in this section.



SOURCE	TEXT	RELEVANCE	SENTIMENT	CATEGORY:SENTIMENT
http://tw	Nach 25 M	TRUE	negative	Atmosphäre#Geruch:negative
http://ww	Bahn plar	TRUE	neutral	Allgemein#Haupt:neutral
http://ww	Re: Londo	FALSE	neutral	

Figure 2.1: Data in TSV format.



```
<Document id="http://twitter.com/Der\_Thomas\_L/statuses/616878">
    <Opinions>
        <Opinion category="Atmosphäre#Geruch" from="25" to="39"
        target="Nebenschwitzer" polarity="negative"/>
    </Opinions>
    <relevance>true</relevance>
    <sentiment>negative</sentiment>
    <text>Nach 25 Minuten ist mein Nebenschwitzer in der Bahn
          ausgestiegen. Das Abteil atmet auf.</text>
</Document>
```

Figure 2.2: Data in XML format.

Datasets Germeval Task 2017 provides both training and test data[1]. All the data consists of 22,000 messages from various social media and web sources.

The whole data is available in both TSV (Fig. 2.1) and XML (Fig. 2.2) format. Subtask (D), however, can only be done by using the XML format, as the spans of the opinion target expression are not available in TSV.

The aspects in the data are chosen from predefined inventories of categories. The Table A.1 in Appendix gives an overview on them. Each category is described as an Entity#Aspect pair, which has several sub-aspects (e.g., Atmosphere#Temperature, Atmosphere#Cleanliness, Atmosphäre#Geruch). While we provide these sub-aspects in the data, we will evaluate only on the categories.

Subtask A: Relevance Classification. Relevance classification is formulated as a binary text classification problem, true and false. It is to determine whether a social media post contains feedback about the "Deutsche Bahn" or if the post is off-topic/contains no evaluation.

For the post in Fig. 2.2, the task is to identify that the post is relevant.

Subtask B: Document-level Polarity. The task is the document-level sentiment classification because it considers the whole comment as a basic information unit. It aims to identify, whether the customer evaluates the "Deutsche Bahn" or travel as positive, negative or neutral.

For the post in Fig. 2.2, the task is to identify the polarity as negative.

[1]https://sites.google.com/view/ germeval2017-absa/data

Subtask C: Aspect-level Polarity. The objective of subtask (C) is to identify all aspects which are positively, negatively and neutrally evaluated within the review. A set of aspect terms are given for a specific entity, we need to determine the sentiment assigned to each unique aspect. In order to increase comparability, the aspects are previously divided into categories. Consequently, the aim of the subtasks is to identify all contained categories and their associated polarity. For the post in Fig. 2.2, the task is to identify the aspects (and their polarity): Atmosphäre#Geruch:negative.

Subtask D: Opinion Target Extraction. Subtask (D) identify the linguistic expression in the posts which are used to express the aspect-based sentiment (subtask C). The opinion target expression is defined by its starting and ending offsets. For the post in Fig. 2.2, the extracted opinion target is *Nebenschwitzer*.

Chapter 3

Machine Learning Approaches

Machine learning is a subfield of Artificial Intelligence dealing with algorithms that allow computers to learn. This usually means that an algorithm is given a set of data and subsequently infers information about the properties of the data, that information allows it to make predictions about other data that it might come across in the future. The ability to make predictions about unseen data is possible because almost all non-random data contains patterns that allow machines to generalize [45]. In order to generalize, the computer trains a model with what it determines are the important aspects of the data [46]. This chapter surveys the machine learning approaches applied to sentiment analysis-based applications. The main emphasis of this chapter is to discuss the research involved in applying machine learning methods for text classification and sequence labeling.

3.1 Word Representation

Neural networks have achieved good results in the fields of computer vision and speech. In the field of natural language processing, many researchers use neural network models to learn word vectors and then compound word vectors into sentence or paragraph vectors and apply them to classification tasks.

Instead of image pixels, the inputs of neural networks are sentences or documents represented as a matrix. Each row of the matrix corresponds to one token, typically a word, but it could be a character [59]. That is, each row is a vector that represents a word. There are two ways commonly used to represent word, one-hot representation and distributed representation [58].

3.1.1 One-hot Representation

Before neural networks was applied to natural language processing, text data was generally expressed by using one-hot representation. In one-hot representation, word is represented as a boolean vector whose length is equal to the size of vocabulary. For every word, the position corresponding to the word in the representation vector is set to one and the remainders are set to zeros, which make the feature convenient to be stored by a sparse model. "Bahn" and "Zug" are taken as an example:

"Zug" is represented as: [0001000000...]
"Bahn" is represented as: [0000000100...]

This representation method can solve some natural language processing problems by using statistical-based machine learning algorithms such as Naive Bayes, Support Vector Machines and Hidden Markov Models. Though it is used widely as it is simple and easy to implement, but the one-hot vectors in the representation space are independent even the original words are very similar [58], so we can not get the similarity among word vectors.

3.1.2 Distributed Representation

The idea of distributed representation was originally proposed by Hinton in 1986, it is different from one-hot representation [14]. The distributed representation of words, also known as word embeddings, is a real-valued, dense, and low-dimensional vectors. Word embeddings represent words as low dimensional vectors that are trained by a language model like word2vec [33] and make the related or similar words closer in the vector space. Thus it can overcome the disadvantage of one-hot representation that feature vectors cannot reflect the dependency relationship between words [58].

Let a word embedding $W : words \rightarrow \Re^n$ be a paramaterized function mapping words to high-dimensional vectors (perhaps 200 to 500 dimensions). For example, we might find:

$$W("Bahn") = (0.2, 0.7, -0.5, ...)$$

$$W("Zug") = (0.15, 0.6, -0.33, ...)$$

As mentioned before, the one-hot vectors are independent in the representation space and the similarity among words cannot be calculated. In contrast, word embeddings are easy to work with because they enable efficient computation of word similarities through low-dimensional matrix operations [28]. We can use cosine similarity as a metric to calculate similarity of words. Words that are semantically similar often occur near each other in text, and so embeddings that are good at predicting neighboring words are also good at representing similarity [22].

Word2vec[1] is a tool for computing continuous distributed representations of words, which was created by a team of researchers led by Tomas Mikolov at Google [33]. The word2vec tool takes a text corpus as input and produces the word vectors as output. They have published pre-trained vectors that are trained on part of Google News dataset (about

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

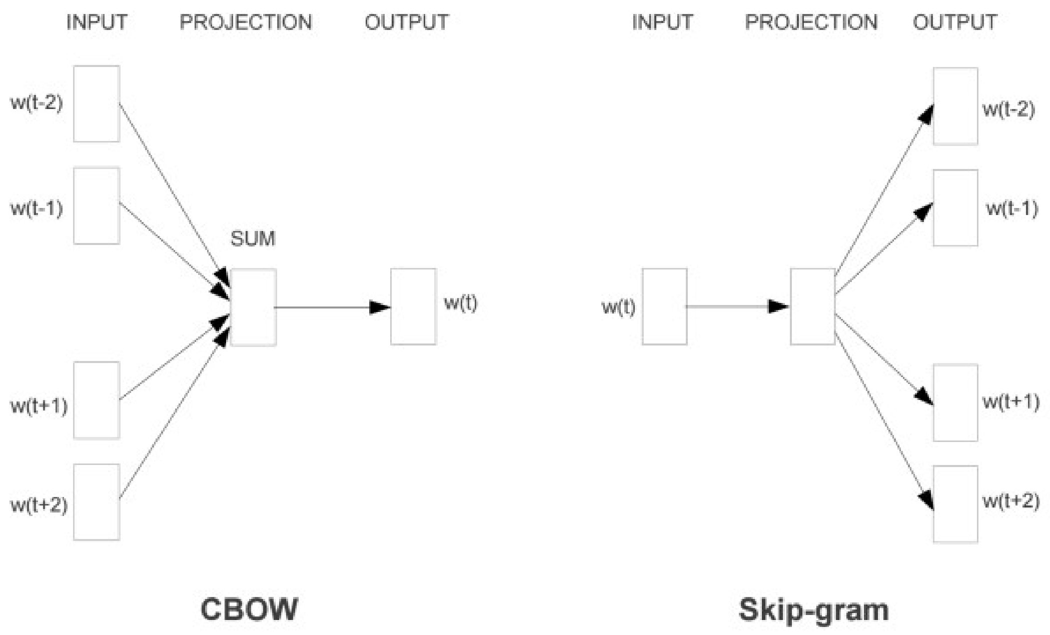


Figure 3.1: CBOW and Skip-gram model

100 billion words). The model contains 300-dimensional vectors for 3 million words and phrases.

There are two main learning algorithms in word2vec : Continuous Bag-of-Words model (CBOW) and continuous Skip-gram model. They are described as the following.

Skip-gram model Skip-gram model predict the context words $[w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2}]$ based on the current word w_t. More precisely, it uses each current word as an input to a log-linear classifier with continuous projection layer, and predicts words within a certain range before and after the current word [58]. Next, the architecture of skip-gram model is introduced in detail.

Fig. 3.2 shows a simple neural network with a single hidden layer. First, a vocabulary of words from our training documents is builded. In this example, we have a vocabulary of 10,000 unique words, an input word "Bahn" is represented as an one-hot vector. This vector have 10,000 components (one for every word in the vocabulary) and a "1" is placed in the position corresponding to the word "Bahn", and "0"s in all of the other positions. The output of the network is a single vector (also 10,000 dimensional) containing, for every word in the vocabulary, that a randomly selected nearby word is that vocabulary word[2].

But neither the well-trained neural network nor the output are used. Instead, the goal is the learned weights of the hidden layer.

The output neurons use softmax, but there is no activation function on the hidden layer neurons. As shown in figure 3.2, if the word vectors are learned by 300 features

[2]http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

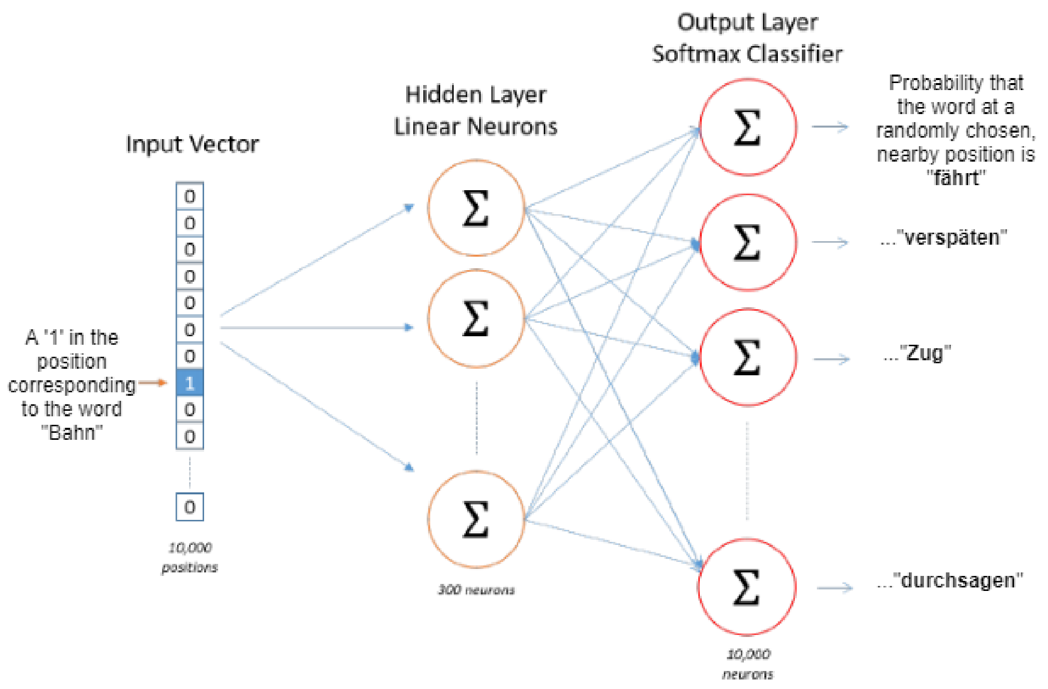

Input Vector
0
0
0
0
0
0
0
1
0
0
0
10,000 positions
A '1' in the position corresponding to the word "Bahn"
Hidden Layer Linear Neurons
300 neurons
Output Layer Softmax Classifier
10,000 neurons
Probability that the word at a randomly chosen, nearby position is "fährt"
..."verspäten"
..."Zug"
..."durchsagen"

Figure 3.2: Architecture of Skim-gram model

(300 features is what Google used in their published model trained on the Google news dataset) [3]. Then the hidden layer is represented by a weight matrix with 10,000 rows (one for every word in the vocabulary) and 300 columns (one for every hidden neuron).

If we look at the rows of this weight matrix (Fig. 3.4), these are actually word vectors that we need. So the end goal of all of this is just to learn this hidden layer weight matrix.

Therefore, the skip-gram model actually learns two separate embeddings for each word w: the *word embedding* v and the *context embedding* c. These embeddings are then encoded in two matrices, the *word matrix* W and the *context matrix* C. Fig. 3.4 shows an example, each row i of the *word matrix* W is the 1×5 vector embedding v_i for word i in the vocabulary, each column i of the *context matrix* C is a 5×1 vector embedding c_i for word i in the vocabulary [22]. If we multiply the *word matrix* W by a *context matrix* C, it will effectively just select the matrix row corresponding to the "1" of the vector embedding v_i.

This means that the hidden layer of this model is just operating as a lookup table. The output of the hidden layer is just the "word vector" for the input word.

CBOW The CBOW (continuous bag of words) model is roughly the mirror image of the skip-gram model. It is also based on a predictive model, but predicting the current word w_t from the context window of words around it [22].

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

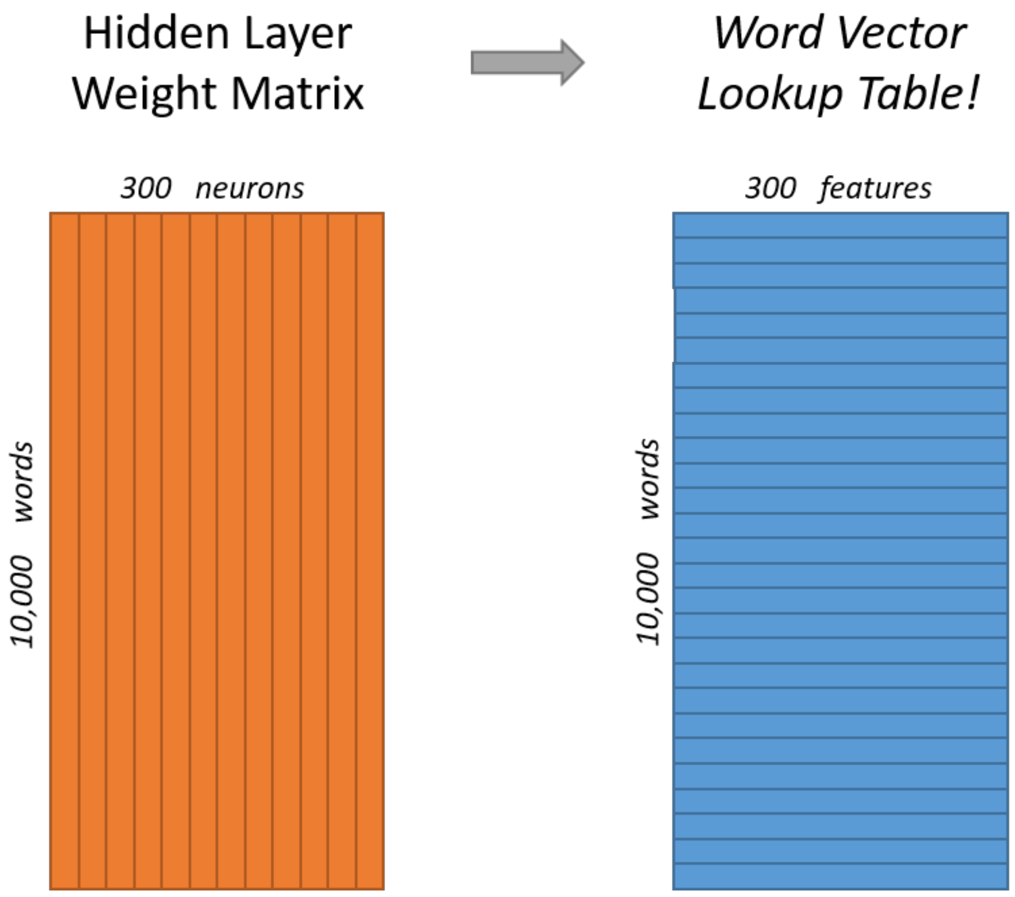


Figure 3.3: Weight matrix of hidden layer.

$$[0 \quad 0 \quad 0 \quad 1 \quad 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = [10 \quad 12 \quad 19]$$

Figure 3.4: An example for the computation of input and hidden layer.

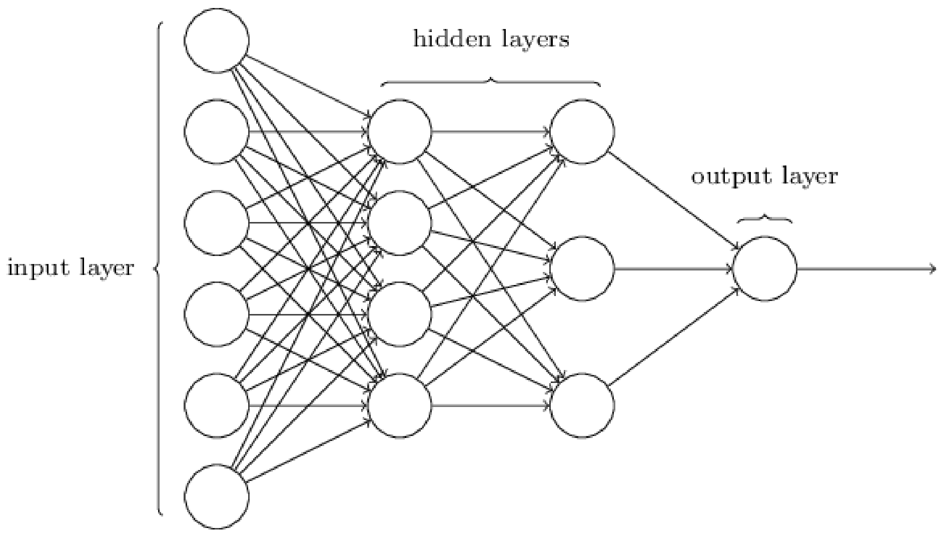


Figure 3.5: Neural network model with two hidden layers.

While CBOW and skip-gram are similar algorithms and produce similar embeddings, they do have slightly different behavior, and often one of them will turn out to be the better choice for any particular task [22]. Word embeddings trained by CBOW contains more syntax information which can obtain a better result in syntax test while trained by Skip-gram contains more semantic information which can perform better in semantic test. Obviously, the semantic information plays a more important role in sentiment, as several sentiment words which are not grammatical can also express the sentiment of document [58]. Thus skip-gram model is used to train the word embeddings for the tasks of this thesis.

The pre-trained word embeddings can then used as input for neural networks, and performs better than words which are randomly initialized [23].

3.2 Neural Networks

Neural Networks (NNs) are widely used in a variety of NLP tasks such as Machine Translation [26], Question Answering [11] and Text Summarization [42]. Neural Networks are computing systems vaguely inspired by the biological neural networks that constitute brains [53]. A standard neural network consists of a input layer, output layer and one or more hidden layers. Fig. 3.5 shows a neural netwrok model with two hidden layers.

Each layer consists of many simple, connected processors called neurons, each producing a sequence of real-valued activations. Input neurons get activated through sensors perceiving the environment, other neurons get activated through weighted connections from previously active neurons. Some neurons may influence the environment by triggering actions. Learning or credit assignment is about finding weights that make the neural network exhibit desired behavior. Depending on the problem and how the neurons are connected, such behavior may require long causal chains of computational stages , where each stage transforms (often in a non-linear way) the aggregate activation of the network [44].

The training and development datasets provided by Germeval Shared Task 2017 are short texts that contain only a few sentences or even a few words. Sentiment analysis of short texts face a challenge because of the limited contextual information that they normally contain [10]. Solutions for the challenges incurred in these problems come from neural networks [23]. *Convolutional Neural Network (CNN)* and *Recurrent Neural Network (RNN)* are two efficient neural network models in machine learning methods, the former is able to extract local feature from words and sentences of any size and the latter is designed for learning general sequences. Another advantage is that one has to worry less about the feature engineering part than the other machine learning approaches.

In recent years, applying neural network to text classification (e.g., Sentiment Analysis, Spam Detection or Topic Categorization.) was proven to be competitive to traditional models.

Yoon Kim [23] has partially improved the convolutional neural network for text sentiment analysis tasks, and achieved good classification performance. Zhang et al. [59] apply convolutional neural networks only on characters without the knowledge of words. Working on characters also has the advantage that abnormal character combinations such as misspellings and emotions may be naturally learnt. The recurrent neural network is applied to opinion target extraction(subtask D), recurrent network could easily be adapted to perform sequence labeling instead of text classification. Furthermore, for opinion target extraction, syntactic relationships and long-distance dependencies may play a significant role and that such phenomena may be better modeled with a recurrent network [50].

The neural network architecture without manual feature-engineering can be trained to do different tasks on varies language datasets.

3.2.1 Convolutional Neural Networks

Yann LeCun et. al. have developed first application of Convolutional Neural Networks in [27]. A convolutional neural network is a special neural network used to process meshed topological data. For example, topological data can be time-series data of a one-dimensional lattice structure over a time interval, as well as it can be a picture of a two-dimensional grid structure of pixels. It is named as convolution neural network because the network model contains convolution operations, in addition, it also includes the pooling operation.

Convolution neural network is a kind of artificial neural network, which is a simple neural network constructed by using convolution operation instead of matrix multiplication operation of at least one layer in neural network [34]. Convolutional neural network has been able to enhance the machine learning system is mainly because the use of three

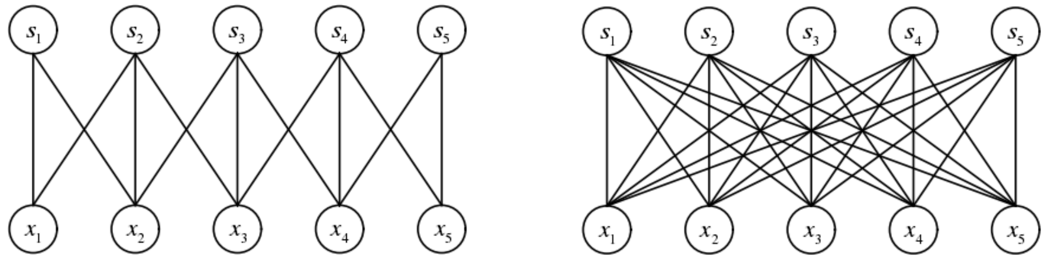


Figure 3.6: The input node and output node connection strategy of convolutional neural network (left) and traditional neural network (right).

important concepts: local receptive fields, shared weights (or weight replication), and spatial or temporal pooling.

In addition, convolutional neural networks can also handle different sized inputs.

The two diagrams in Figure 3.6 show the sparse connection of convolutional neural network and the matrix multiplication connection of traditional neural network respectively, where x_1, x_2, x_3, x_4, x_5 are the input nodes and s_1, s_2, s_3, s_4, s_5 are the output nodes. Traditional neural networks connect input nodes and output nodes through matrix multiplication operations, which means that there is a connection between each input node and each output node. Convolutional neural networks, on the other hand, employ a strategy of sparse connections (also called sparse weights) that implement sparse connections between input and output nodes by making the convolution kernel smaller than the input. Sparse connections mean that convolutional neural networks only need store a small number of parameters, which in turn can reduce the memory requirements of the model and increase its statistical efficiency.

The output of the convolutional layer is then input into the pooling layer, and the pooled operation adjusts the output by the pooled function. The pooled function replaces these nodes with their statistical significance in the output node of the previous layer, for example, the max pooling operation finds the maximum on the rectangular neighborhood of output. Other pooling operations include finding average or the weighted average over rectangular neighborhoods. No matter what kind of pooling operation, their role is that when the input of network changes slightly, the output can remain unchanged. This means that most values in the output of the pooled layer do not change when the input to the network is slightly shifted. Figure 3.7 shows an example of a pooling operation.

As the changes from the left to right in Figure 3.7 can be seen, when the input to the pooling layer (i.e., the output of convolutional layer) shift a unit, left and right examples get the same pooling layer output. If the model is only concerned with the representation of the feature but does not care about the exact location of the feature, then the invariance of the pooling operation is very important to the model. Besides, pooling also allows the model to accept input data of different lengths. Using a pooling operation in a convolutional neural

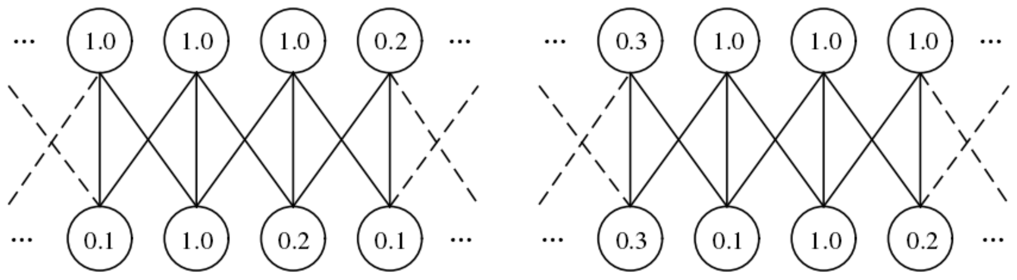


Figure 3.7: Left: An example of the maximum pooling operation, the window size of the pooling operation is 3; Right: A new example that right-shift the output of the convolutional layer of left example.

network can be viewed as adding a priori knowledge to the model - the function learned by the model remains unchanged for small changes. In addition, pooling operations can dramatically increase the statistical efficiency of network models.

Convolution neural network plays a very important role in the evolution of deep learning. It is one of the successful examples of adding human brain structure into machine learning. It is also one of the earliest depth models successfully applied in various fields.

Word-level Convolutional Neural Networks

In 2014, Yonn Kim proposed a convolution neural network for sentence classification [23]. The model is shown in Figure 3.8. It is a one-dimensional convolutional neural network model with two different size filters. This model will be used as a distributed sentence feature extractor for various classification tasks.

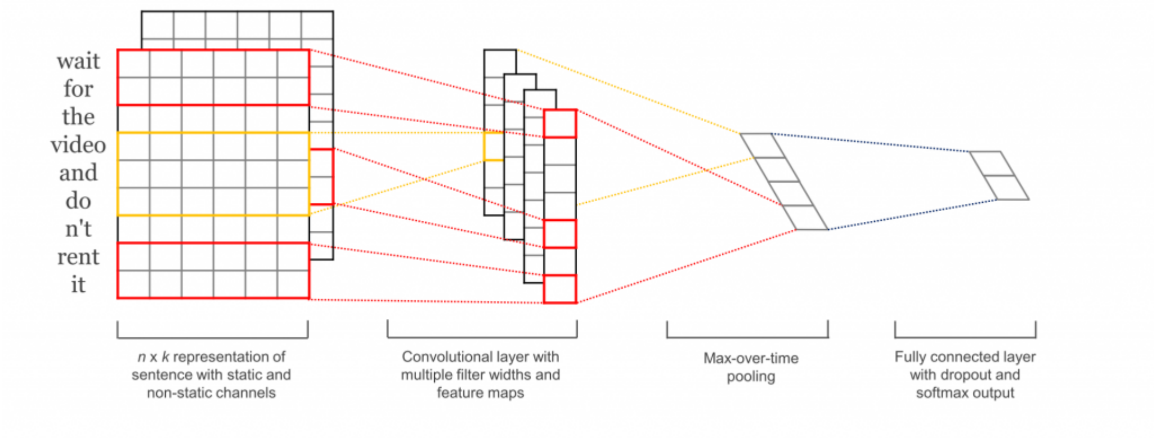

wait
for
the
video
and
do
n't
rent
it
n x k representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

Figure 3.8: Convolutional neural networks for sentence classification

It is composed of a single convolutional layer followed by a non-linearity, max pooling and a softmax classification layer.

(1) Input layer: The word vectors corresponding to the input of convolutional neural

network can be either randomly initialized or pre-trained from word2vec. Yoon Kim [23] proposed three model variants:

(a) CNN-rand: The word vectors of the input samples for convolutional neural network are randomly initialized, which are then fine-tuned by backpropagation during the model training.

(b) CNN-static: The input word vectors are pre-trained vectors from word2vec, but the word vectors keep unchanged during training and only the other parameters of the model are learned.

(c) CNN-non-static: The input word vectors are pre-trained vectors, and they are fine-tuned during training CNN model.

(2) Convolutional layer: Learning the local features between adjacent words in the input sample through multiple convolution filters of different sizes.

(3) Pooling layer: This layer gets the most important features by taking the maximum value of the output of the convolution layer, and ensures that the same length of output can be obtained from different lengths of input.

(4) Classification layer: This layer is fully connected softmax layer, the output is the probability distribution of the input samples on each label.

More formally, for the input layer, letting the d-dimensional vector of the i-th word in the input sample be w, then a formulation of a sample vector s of length n (padded with row vectors of 0s to n if the sentence length is less than n) is formulated as:

$$\mathbf{s} = [w_1, w_2, ..., w_{n-1}, w_n] \tag{3.1}$$

The vector $\mathbf{s}$ is obtained by concatenating end-to-end vectors of all the input samples. Let $\mathbf{s}_{i:i+j}$ be a part of the vector $\mathbf{s}$ consisting of the i_{th} to the $(i+j)_{th}$ word vectors in the input sample, concatenated together end to end.

To learn to capture and compose features of individual words in a given sentence from low-level word embeddings into higher level semantic concepts, the neural network applies a series of transformations to the input sentence matrix S using convolution, nonlinearity and pooling operations, which are described next. [47]

For the convolutional layer, the convolutional layer is connected to the input layer by two convolution filters of different window sizes. Let one of the convolution filters of window size h be $v \in \mathbf{R}^{hk}$. This convolution filter is applied to a window of h words on the input sample and a new feature c_i is produced . The formula for generating a new feature is as follows:

$$c_i = f(v \cdot w_{i:i+h-1} + b) \tag{3.2}$$

Where f is a non-linear function such as the hyperbolic tangent, $b \in \mathbf{R}$ is a bias term, and both b and v are parameters in the convolutional neural network. This filter is applied to each possible window of words in the sentence $\{w_{1:h}, w_{2:h+1}, ..., w_{n-h+1:n}\}$ to produce a feature map:

$$\mathbf{c} = [c_1, c_2, ..., c_{n-h+1}] \tag{3.3}$$

with $\mathbf{c} \in \mathbf{R}^{n-h+1}$.

After the convolution operation, the max-pooling operation is then applied over the feature map and the maximum value $\hat{c} = max\{c\}$ is taken as the feature corresponding to this particular filter.[5]

The idea of the pooling layer is to capture the most important features of the corresponding convolution filter. In addition, max-pooling allows samples of different lengths to be used as input to the convolutional neural networks. Because convolutional neural networks have multiple window-sized convolution filters, the output of the pooling layer is a feature vector whose values correspond to the features of the convolution filter.

The last layer is the fully connected layer. The features form the max-pooling layer are passed to a fully connected softmax layer whose output is the probability distribution over labels [23].

Character-level Convolutional Networks

So far, the models presented above is based on words. But there is also research in applying CNNs directly on characters. An example of character-level text classification not requiring any tokenization is given by Zhang et al. (2015) [59]. In their work, the authors perform text classification using character-level CNNs on very large datasets and obtain comparable results to traditional models based on words [20].

When trained on largescale datasets, deep convolutional neural network do not require the knowledge about the words and syntactic or semantic structure of a language. This simplification of engineering could be crucial for a single system that can work for different languages, since characters always constitute a necessary construct regardless of whether segmentation into words is possible. Working only on characters also has the advantage that abnormal character combinations such as misspellings and emoticons may be naturally learnt [59].

The first step is to build an alphabet. The alphabet used in this model consists of 74 characters, including 30 german letters, 10 digits, 33 other characters and the new line character. Then an all-zero vector is padded to the one-hot encoding, in order to handle the characters that are not in the character label (Fig. 3.9). The non-space characters are:

$$abcdefghijklmnopqrstuvwxyz\text{äöü}\beta 0123456789$$

$$-, ; .!? :' \text{''} / \backslash | \_ @\#\$\%\hat{}\&* \tilde{}\text{`} + - =<> ()[]\{\}$$

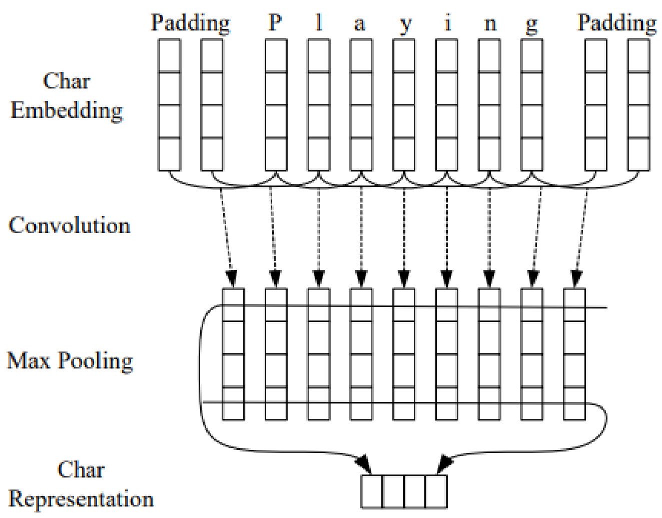


Figure 3.9: The convolution neural network for extracting character-level representations of words.

Then a sentence containing n words (word sequences) is represented as:

$$x = (x_1, x_2, ..., x_n)$$

Where x_i represents the *id* of the words in the alphabet, and then we can get one-hot vector of each word, the dimension m is the alphabet size.

Then, the sequence of characters is transformed to a sequence of such m sized vectors with fixed length l_0. Any character exceeding length l_0 is ignored, and any characters that are not in the alphabet including blank characters are quantized as all-zero vectors. The character quantization order is backward so that the latest reading on characters is always placed near the begin of the output, making it easy for fully connected layers to associate weights with the latest reading [59].

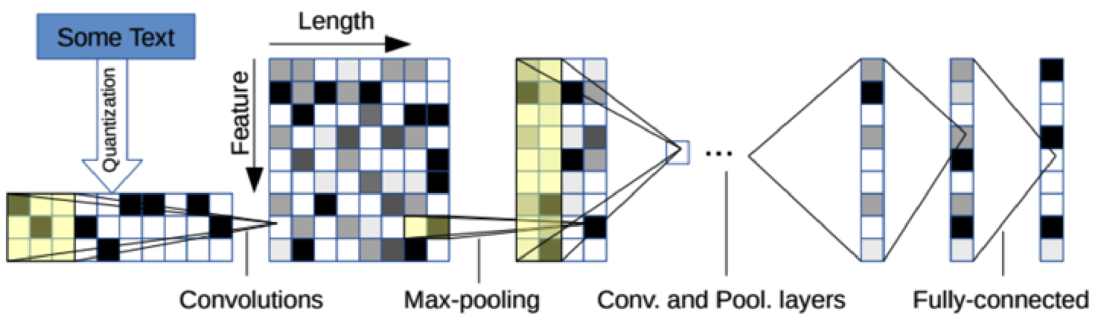
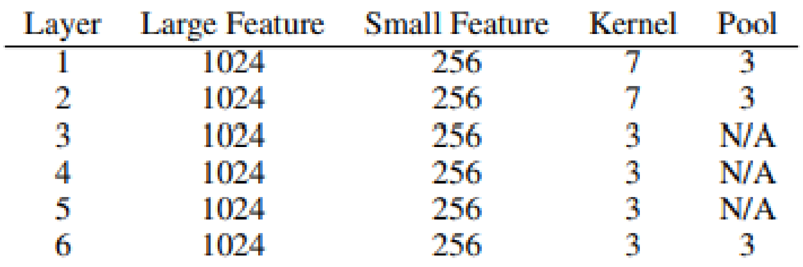


Figure 3.10: Zhang: Character-level Convolutional Networks

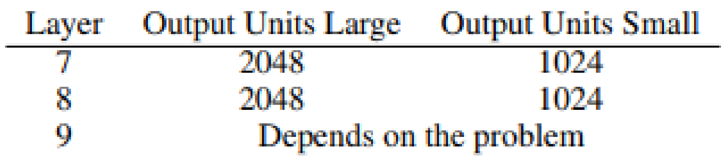
Zhang has desighed two CNNs – one large and one small and the difference between them is the feature map size. They are both 9 layers deep with 6 convolutional layers and

3 fully-connected layers. Figure 3.11 and 3.12 gives an illustration. The kernel size of the first two convolutional layers are set as 7, and the rest of the four layers kernel size are 3.



Layer	Large Feature	Small Feature	Kernel	Pool
1	1024	256	7	3
2	1024	256	7	3
3	1024	256	3	N/A
4	1024	256	3	N/A
5	1024	256	3	N/A
6	1024	256	3	3

Figure 3.11: Convolutional layers followed by pooling layers.



Layer	Output Units Large	Output Units Small
7	2048	1024
8	2048	1024
9	Depends on the problem	

Figure 3.12: Fully-connected layers of the character-level convolutional network. The number of output units for the last layer is determined by the problem. For example, for a 10-class classification problem it will be 10.

The input have number of features equal to 74 due to the character quantization method, and the input feature length is 1014. The pre-trained method such as word2vec are not used for the input word vectors in the model. It seems that 1014 characters could already capture most of the texts of interest.There are also two dropout layers between the three fully-connected layer to regularise the loss.

3.2.2 Recurrent Neural Networks

Unfortunately, the feed-forward neural network can not handle the constant input data. Recurrent Neural Network (RNN) [16]can overcome this shortcoming, which is a neural network has recurrent connections, so it can store information inside the network and also accept sequences of different lengths as input.

In order to better understand the loop in a recurrent neural network, this section first introduces the structure of a flow graph developed in a recurrent neural network, followed by the description of the recurrent neural network model.

Flow graph is a way to formalize a series of computational structures that are mappings of inputs and parameters to outputs and loss functions. For a recurrent computation structure, it can be expanded into a flow graph composed of repeating structures. For example, consider the classic form of a dynamic system [12]:

$$s_t = f_\theta(s_{t-1}) \tag{3.4}$$

Where s_t is the state of the system at time step t. The flow graph of the dynamic system is shown in Figure 3.13 .

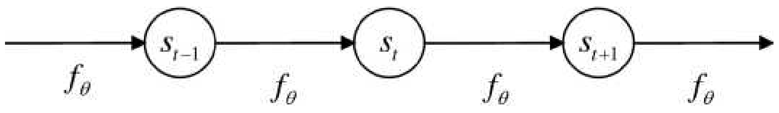


Figure 3.13: Expanded flow graph structure of a traditional dynamic system

In the flow graph, the transfer function from the previous state to the latter state is unified as f_θ, that is, the transfer function is applied to all time steps t.

Another example is a dynamic system driven by an external signal x:

$$s_t = f_\theta(s_{t-1}, x_t) \tag{3.5}$$

The structure of the system is shown in Figure 3.14. It can be seen from the figure that the state node contains the information of almost all past sequences. Formula 3.5 implicitly defines the formula 3.6:

$$s_t = g_t(x_t, x_{t-1}, x_{t-2}, ..., x_2, x_1) \tag{3.6}$$

Where function g_t maps all the past sequence to the current state. The formula 3.6 is in fact a part of the recurrent neural network definition and s_t can be considered as a summary of the input sequence before the current time step t. If a recurrent neural network is used for statistical language modeling, it is typical to give a series of previous words to predict the next word. The recurring flow graph shown in Figure 3.14 helps to understand and define the expanded recurrent neural network structure.

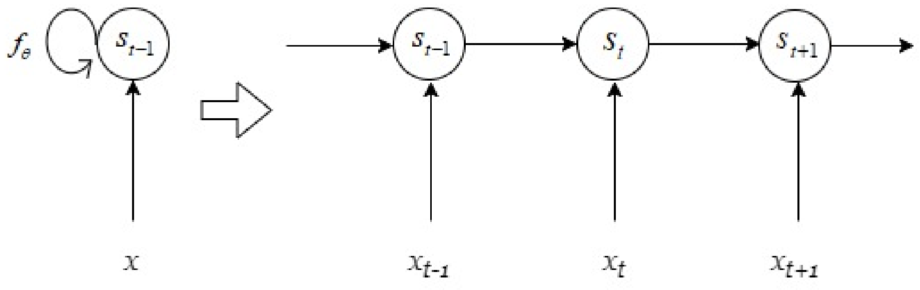


Figure 3.14: The neuronal structure containing the loop structure and its unrolled flow graph structure

Recurrent Neural Network With the idea of the flow graph described above, a wide variety of recurrent structure can be designed. The recurrent structure before and after unfolding are shown in Figure 3.14. Assuming that the hidden layer uses hyperbolic

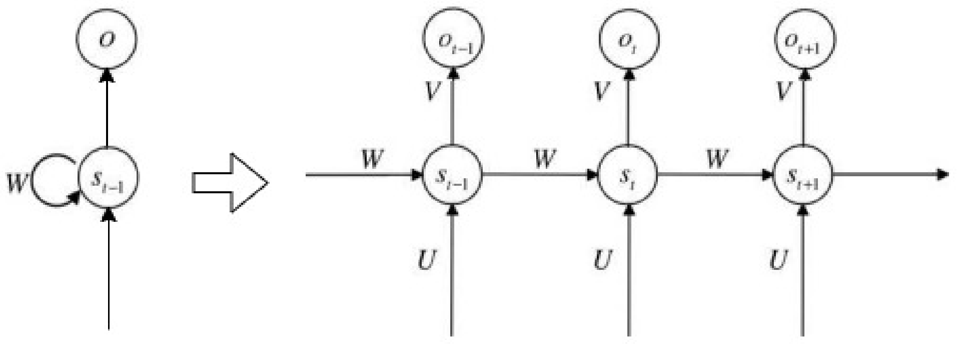


Figure 3.15: An unfolded recurrent neural network.

tangent nonlinear functions and the output layer (for classification problems) uses the softmax function, the forward propagation formulas corresponding to the recurrent neural network model in Figure 3.15 are as shown in Formulas 3.7 and 3.8:

$$s_t = tahn(b + W \cdot s_{t-1} + U \cdot x_t) \tag{3.7}$$

$$o_t = softmax(c + V \cdot s_t) \tag{3.8}$$

Where matrix U is the parameter of the input layer to the hidden layer, matrix V is the parameter of the hidden layer to the output layer, matrix W is the parameter of the hidden layer to the hidden layer, b and c are the bias of the tanh function and the softmax function respectively, x_t is input at time t, s_t is the hidden layer state at time t, and o_t is the output at time t.

Figure 3.15 shows a recurrent neural network structure that maps an input sequence to an output sequence of the same length. For a given input output sequence pair (x, y), the total loss $L(x, y)$ is the sum of the losses L_t over all time steps:

$$L(x, y) = \sum_t L_t = \sum_t -logp_{y_t} \tag{3.9}$$

Where y_t is the category corresponding to the time step t in the output sequence.

Long Thort-Item Memory networks

For the traditional recurrent neural network, the transfer weight matrix has a great influence on the learning process of the recurrent neural network. If the value of the weight matrix is very small (formally, the main feature value of the weight matrix is less than

1.0), it will lead to the gradient vanishing. Gradient vanishing refers to the phenomenon that the learning process becomes very slow or even stop due to the gradient signal too small. On the contrary, if the weight matrix is very large (i.e., the main feature value of the weight matrix is greater than 1.0), the gradient signal will be too large to make the learning process divergent. This phenomenon is also called gradient explosion.

For recurrent neural network, it is difficult to learn long-distance dependence information. In theory, recurrent neural networks can solve the loss of long-distance dependence information through parameter selection. However, in practice, it can not successfully learn long-distance dependence information. Long Short-Term Memory (LSTM) was proposed by Hochreiter and Schmidhuber in 1997 [15, 30] and has recently been improved and applied by Graves [13]. It is a specific recurrent neural network (RNN) architecture that was designed to model temporal sequences and their long-range dependencies more accurately than conventional RNN [43]. In recent years, LSTM has achieved great success in many fields. The LSTM model can help recurrent neural network to overcome its drawbacks and achieve good results on many tasks.

The long-short-term memory model proposes a new structure called memory cell. Its structure is shown in Figure 3.16. Memory cell consists of three main components: input gate, forget gate, output gate. The gate is to regulate the interaction between the memory cell and its environment. On the one hand, the input gate can allow the input signal to change the state of the memory cell or prevent the input signal. On the other hand, the output gate can allow the memory cell to act on another neuron or prevent it from affecting other neurons. Finally, forget gate can adjust the self-looping connection of memory cells, which can allow memory cells to remember or forget their previous states as needed.

The formula below will be used to formally describe how the network layer in the memory cell is updated at each time step t, in these formulas:

$x_t --$ The memory cell input at time step t;

$h_t --$ The hidden layer value of memory cell at time step t;

$\sigma --$ Sigmoid function, output a value between 0 to 1;

First, the start value i_t of the input gate at time step t and the candidate value of memory cell state c_t are calculated:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{3.10}$$

$$c_t = tahn(W_c x_t + U_c h_{t-1} + b_c) \tag{3.11}$$

Then, the start value f_t of the forget gate in the memory cell at time step t is calculated:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{3.12}$$

The next step is to calculate the new state value C_t of the memory cell at the time step t:

$$C_t = i_t \times C_t + f_t \times C_{t-1} \tag{3.13}$$

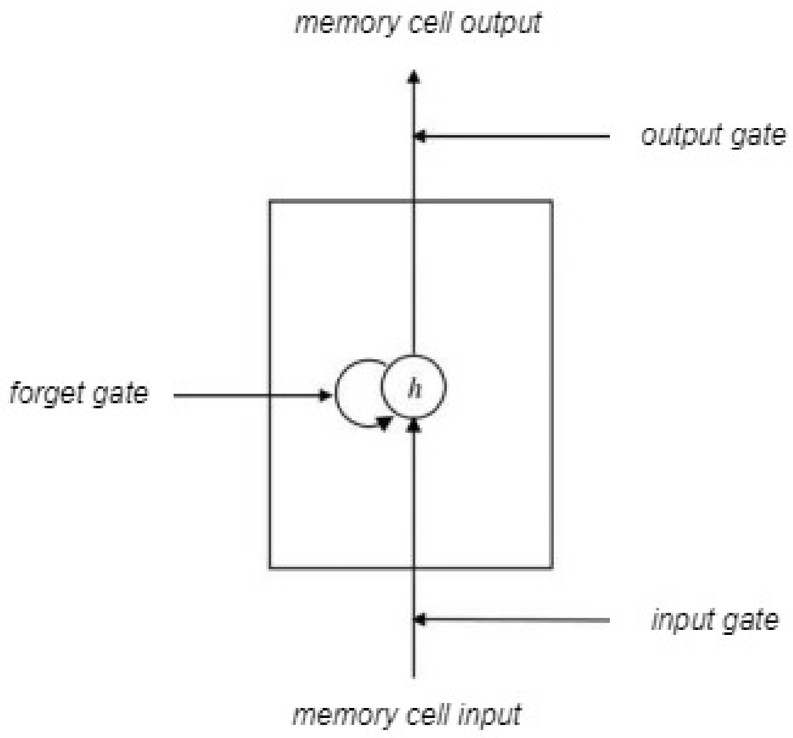


Figure 3.16: Memory cell structure

Finally, the value of output gate and memory cell output value are calculated:

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + V_0 C_t + b_o) \tag{3.14}$$

$$h_t = o_t \times tahn(C_t) \tag{3.15}$$

Bidirectional LSTM

For many sequence labeling tasks it is beneficial to have access to both past (left) and future (right) contexts. However, the hidden state h_t of LSTM takes information only from past, knowing nothing about the future. An elegant solution is Bidirectional LSTM (BiLSTM) [31]. A Bidirectional LSTM model can take into account any arbitrary amount of context on both sides of a word and eliminates the problem of limited context [6].

The basic idea of BiLSTM is that each forwards and backwards sequence is respectively two LSTMs, and both are connected to the output layer. This structure capture both past and future contextual information for each time step of the input sequence. Figure 3.17 shows an unfolded BiLSTM model.

The word embedding $(x_1, x_2, ..., x_n)$ of each word of a sentence is taken as the input of BiLSTM at each time step. Then the hidden state sequence $(\overrightarrow{h_1}, \overrightarrow{h_2}, ..., \overrightarrow{h_n})$ of the output of the forwards LSTM and the hidden state sequence $(\overleftarrow{h_1}, \overleftarrow{h_2}, ..., \overleftarrow{h_n})$ of the backwards LSTM are concatenated at each time step $h_t = [\overrightarrow{h_t}, \overleftarrow{h_t}] \in \mathbf{R}^m$,then the complete hidden state sequence is formed:

$$(h_1, h_2, ..., h_n) \in \mathbf{R}^{n \cdot m}$$

Then the m-dimensional hidden state vector is mapped to k-dimensional, and k is the number of labels. Thus, the extracted sentence features are denoted as matrix $O = (o_1, o_2, ..., o_n) \in \mathbf{R}^{n \cdot k}$. Each dimension o_{ij} of $o_i \in \mathbf{R}^k$ can be regarded as the scoring value that the word x_i is classified as j_{th} tag.

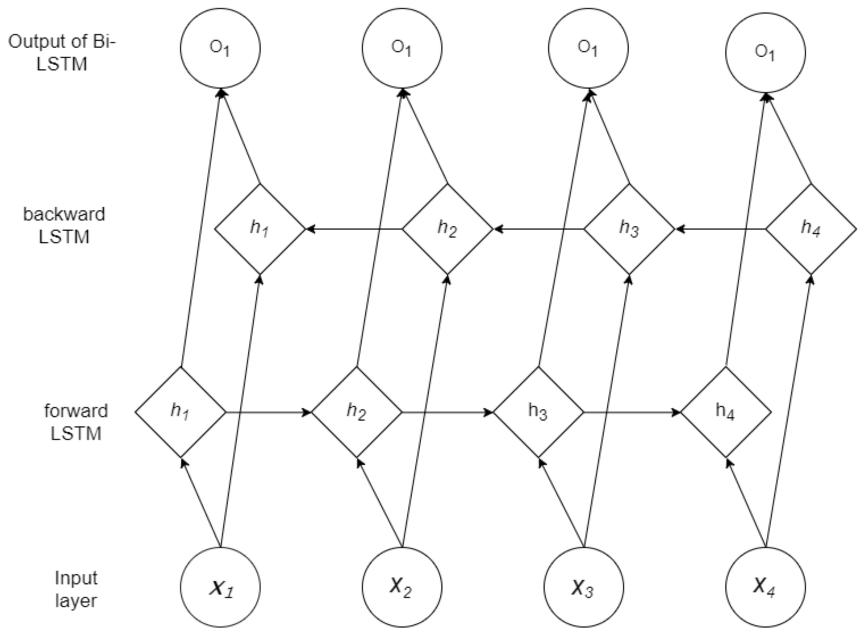


Figure 3.17: Bi-directional LSTM

3.3 Baseline Models

3.3.1 Support Vector Machines

Support Vector Machines (SVM) [8] is a powerful tool for classification problems. It is a non-statistical approach and make no assumptions about the distribution of the data [38].

It is used to maximize the separation in feature space. When the dataset is linearly separable, a linear classifier, i.e., a linear separable support vector machine is learned by maximizing the hard interval. When the dataset is approximately linearly separable, a linear classifier is obtained by maximizing the soft interval, i.e., Linear Support Vector Machine. When data sets are inseparable, nonlinear support vector machines is learned by using kernel techniques and maximizing soft intervals. The kernel technique refers to the inner product of the input feature vectors mapped into the feature space by the kernel function. The principle is equivalent to learning the linear support vector machine implicitly in the high dimensional feature space. We begin with the basic principles involved in the simplest linear separable support vector machine.

The basic idea of SVM learning is to find a seperate hyperplane that can correctly divide the data set and maximize the geometric interval. For a linearly separable dataset, there are lots of linearly separable hyperplanes, but the hyperplane with maximized geometric intervals is unique. Maximizing separation is to classify the dataset with a sufficiently large confidence in finding a hyperplane with maximized margin. Not only the positive and negative instances, but also the points that are most difficult to classify can be separated. The hyperplane thus has good classification predictability for unknown samples, maximizing the model generalization.

The maximum interval method for learning a linear support vector machine is described as follows:

Input: The linearly separable dataset $T = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$, where the corresponding input to x_i is an n-dimensional feature vector, only the binary classification is considered, therefore $y_i \in \{1, -1\}, i = 1, 2, ..., N$.

Output: Maximum margin of separated hyperplane and decision function:

$$min_{w,b} \frac{1}{2} ||w||^2 \tag{3.16}$$

$$s.t. \quad y_i(w \cdot x_i + b) \geq 0, i = 1, 2, ..., N \tag{3.17}$$

Find the optimal solution w^*, b^* ,and the optimal margin seperating hyperplane: $w^* \cdot x + b^* = 0$.

The classification decision function is: $f(x) = sign(w^* \cdot x + b^*)$.

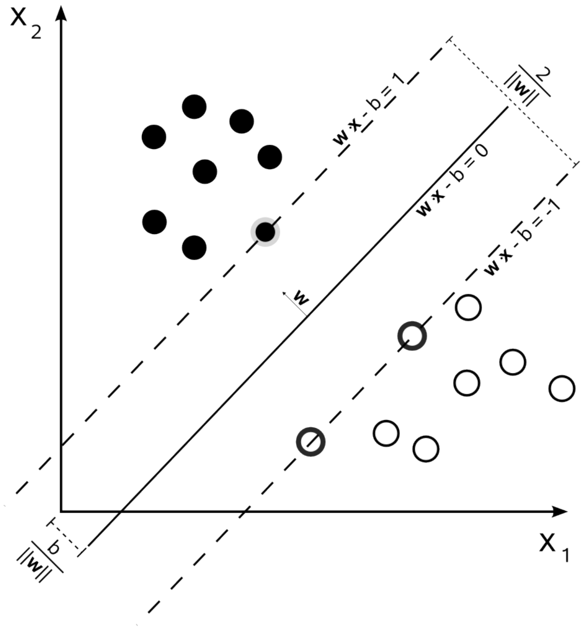


Figure 3.18: Linear separable support vector machines

Under the condition of linear separability, the support vector contains the nearest sample points to the separation hyperplane, and the support vector sample points are the points that make the constraint inequality equal. As shown in Fig. 3.18, the points on the dotted line represent the hyperplane support vector. That is, the support vector is used to determine the separation hyperplane. If we move the support vectors, we will change the solution, but if we move the other instance points outside the bounds, the solution will not change even without these sample points. Since the support vector plays a decisive role in determining the separation hyperplane, this classification model is called support vector machine.

When there is a sample that does not satisfy the constraint that the function interval is greater than or equal to 1, the sample points need to be introduced with slack variables, which are called the soft interval maximization compared with the original hard interval maximization. Due to the existence of singular points in most real-world data, the linear SVM with soft interval maximization is more universal. The mathematical expression after changing the constraint is:

$$min_{w,b,\epsilon}\frac{1}{2}||w||^2 + C\sum_{i=1}^{N}\epsilon_i \tag{3.18}$$

$$s.t. \quad y_i(w \cdot x_i + b) \geq 1 - \epsilon, i = 1, 2, ..., N \tag{3.19}$$

$$\epsilon_i \geq 0, i = 1, 2, ..., N \tag{3.20}$$

where ϵ_i is the classification loss for the sample i, it will be 0 if the classification is correct, or a linear value if the classification is biased, $\sum_{i=1}^{N}\epsilon_i$ is the total error. The goal of our optimization is to minimize this value. A smaller value indicates a higher classification accuracy. Another optimization is to minimize C, the penalty term. In principle, C can select any numbers greater than 0 as needed. The bigger C is, the higher the attention to the total error in the whole optimization process is, the higher the requirement for reducing the error is.

This problem will be a hard-margin SVM problem when C tends to infinity, i.e., a sample of classification errors is not allowed. When C tends to 0, we no longer care about the classification is correct, the larger interval the better, then we will not be able to find a meaningful solution and the algorithm will not converge. Here is a set of Guassian kernel / soft-margin SVM experimental results under different C :

Nonlinear support vector machine (SVM) uses kernel function to map the indivisible nonlinear data set in low-dimensional space into the high-dimensional feature space. Through such nonlinear transformation, the original nonlinear problem is transformed into a linear problem. Under the new feature space, the nonlinear classification problem can be solved by linear support vector machine, the following gives the definition of kernel function:

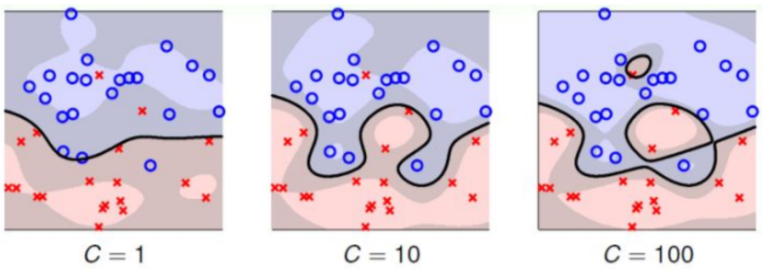


Figure 3.19: Experimental results of Guassian kernel / soft-margin SVM when C is 1,10,100.

Let X be the input space, H be the feature space, if there exists a mapping ϕ from X to H, so that for any two input vectors x, z in X, the function K satisfies the condition $K(x, z) = \phi(x) \cdot \phi(z)$, then K is the kernel function, ϕ is the corresponding mapping function, and $\phi(x) \cdot \phi(z)$ is the inner product of the corresponding high dimensional feature space.

Commonly used kernel functions include:

Polynomial kernel function:

$$K(x, z) = (x \cdot z + 1)^p \tag{3.21}$$

Radial basis function:

$$K(x, z) = exp(-\gamma ||x - y||^2) \tag{3.22}$$

Gaussian kernel function:

$$K(x, z) = exp\{-\frac{||x - z||^2}{2\mu^2}\} \tag{3.23}$$

In practice, the kernel function is often relied on domain knowledge, and the validity of kernel function needs to be tested by cross-validation set. The optimization algorithm includes sequence minimum optimization(SMO) algorithm.

3.3.2 Conditional Random Fields

Kernel-based approaches, such as SVM, which maximize the margin of confidence of the classifier, are adopted for many typical classification tasks, which assign a label to a single object. Their popularity stems both from the ability to use high-dimensional feature spaces, and from their strong theoretical guarantees. However, many real-world tasks involve sequential, spatial, or structured data, where multiple labels must be assigned. Existing kernel-based methods ignore structure in the problem, assigning labels independently to each object, losing much useful information [51]. Conversely, probabilistic graphical models, such as Markov networks, Conditional Random Fileds (CRF), can represent correlations between labels.

Conditional Random Fields is a statistical-based model. In 2001, it was first proposed by John Lafferty et al. [25]. The conditional probability of the global optimal output label could be calculated under the given global conditions.

The CRF model is a kind of undirected graph model. $G = (V, E)$ is defined as an undirected graph, a point $v \in V$ corresponds to a random variable $y \in Y$. If X is a global condition, then (X, Y) is a conditional random field when the entire output label Y satisfies the Markov property.

Among them, the chained structure is a common situation, which is to obtain the conditional probability of the entire output labeled sequence in the global sense under the condition of the given input sequence.

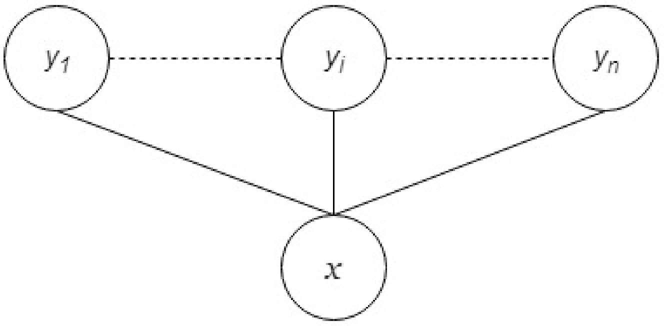


Figure 3.20: The undirected graph of conditional random fileds

As shown in Figure 3.20 , let $x =< x_1, x_2, ..., x_i, ..., x_n >$ be the input sequence and $y =< y_1, y_2, ..., y_i, ..., y_n >$ be the output labeled sequence. Under the condition of the input sequence x, the probability of outputting the labeled sequence y is defined as:

$$P_\theta(y|x) = \frac{1}{Z(x)} exp(\sum_i \sum_k \mu_k s_k(y_i, x) + \sum_i \sum_k \lambda_k t_k(y_{i-1}, y_i, x)) \tag{3.24}$$

In Equation 3.24, the parameters μ_k and λ_k are respectively the weights of two functions. $Z(x)$ is the normalization factor, which gives a probabilistic results. $s_k(y_i, x)$ represents the state feature function of the input sequence at position i, and $t_k(y_{i-1}, y_i, x)$ represents the transition feature function between $i - 1$ and i position of input sequence. For the conciseness of the following expressions, the two feature functions are expressed in a unified form: $f_k(y_{i-1}, y_i, x)$. This is just a form. In practice, the state function $s_k(y_i, x)$ only considers a single output y_i at position i.

From Formula 3.24, we can see that the key to construct a CRF model is to solve four problems: input and output sequence representation, feature function definition and parameter estimation. Among them, the definition of the feature functions is particularly critical, because it will directly affect the effectiveness of the whole CRF model.

Each feature function corresponds to an event whose value is '0' or '1', and when its value is '1', the event corresponding to this eigenfunction increases by one time.

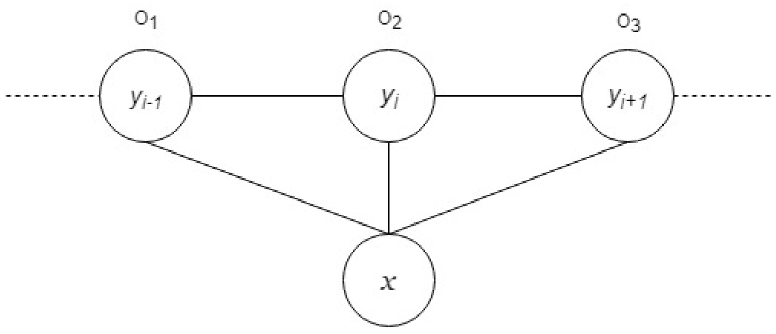


Figure 3.21: Part of input sequence and output sequence

As shown in Figure 3.21, let the feature function $f(y_{i-1}, y_i, x)$ correspond to the following events: $x_i = I_2, x_{i-1} = I_1, x_{i+1} = I_3$, the output at position i is labeled as: $y_i = O_2$. It can be expressed as a function of form:

$$f(y_{i-1}, y_i, x) = \begin{pmatrix} b(x,i) & if \quad y_i = O_2 \\ 0 & otherwise \end{pmatrix} \tag{3.25}$$

$b(x, i)$ is the observed value of the input sequence, when an input situation occurs, it takes a value of '1', otherwise '0'.

$$b(x, i) = \begin{pmatrix} 1 \quad if \quad x_i = I_2, x_{i-1} = I_1, x_{i+1} = I_3 \\ 0 \quad otherwise \end{pmatrix} \tag{3.26}$$

In summary, the feature function $f(y_{i-1}, y_i, x)$ value of '1' means:

In the conditional sequence x, when $x_{i-1} = I_1, x_{i+1} = I_3, x_i = I_2$, the output at the i position is labeled as $y_i = O_2$, this event is increased by 1 count.

The training of the conditional random field is to estimate the parameter λ of each feature function, usually using the maximum likelihood estimation.

Maximum Likelihood Estimation [54] is a commonly used estimation method in frequency theory. It is assumed that training data consists of a set of data points $D = < x^{(k)}, y^{(k)} >, k = 1, ..., N$, each of them is independent and is generated according to a joint experimental distribution. Accordingly, the likelihood function of the training data D of the conditional random field model is:

$$L(\lambda) = \prod_{x,y} logp(y^{(k)}|x^{(k)}, \lambda) \tag{3.27}$$

The parameter λ that maximizes the likelihood function is defined as:

$$\lambda_{ML} = argmaxL(\lambda)$$

It can also make the distribution of the model closer to the empirical distribution, but directly applied maximum likelihood estimation may lead to over-learning problems. Gaussian smoothing factor is introduced into the conditional random field to solve the problem [55], then Formula 3.27 is reformed as:

$$L(\lambda) = \prod_{x,y} logp(y^{(k)}|x^{(k)}, \lambda) - \sum_{j} \frac{(\lambda_j)^2}{2\sigma^2}$$

This function is a concave function, which ensures that the function can converge to the global maximum.

3.4 Train a Classifier on top of Neural Network Features

3.4.1 Combine Convolutional Neural Networks and Support Vector Machines

In recent years, classifiers based on word embeddings and CNN have achieved good performance in sentiment analysis tasks. CNN is good at learning invariant features, however, CNN only uses fully connected softmax layers as the classification layer [23]. The fully connected layer can not classify the non-linearly distributed data effectively [4], while Support Vector Machines are good at producing decision surfaces from well-behaved feature vectors, but cannot learn complicated invariances [14].

In this thesis, the model (Fig. 3.22) that combining convolutional neural network and support vector machines is applied for text classification. In this model, skip-gram model is employed to construct the word embeddings. Each word x_i in a sentence is mapped from one-hot vector to a low-dimensional dense word embedding $x_i \in \mathbf{R}^d$ by using a pre-trained or randomly initialized embedding matrix, d is the embedding dimension. The CNN model is then applied to learn feature vector representations of the input samples. The output of CNN, distributed feature representations of the input samples are fed into SVM. Such a combined model is expected to combine the advantages of a convolutional neural network with a support vector machines [4, 14].

The concrete implementation algorithm of this model is shown in Algorithm 3.1.

3.4.2 Combine Bidirectional-LSTM and Conditional Random Fields

As mentioned in the introduction of Bidirectional-LSTM, if softmax is further used for the output O, it is equivalent to a k-class classification at each time step independently. Therefore, the previously labeled information is not considered.

While Conditional Random Fields (CRF) is a traditional sequence labeling model that consider the correlations between labels in neighborhoods and jointly decode the best chain of labels for a given input sentence, instead of decoding each label independently [31]. Therefore, it considers more linear weighted combinations of local features of the whole sentence. In particular, CRF calculates the joint probability and optimizes the whole sequence, instead of stitching the optimal value for each time step. However, CRF can not consider long-term contextual information like LSTM.

```
Input: Train data $D_{train}$, Test Data $D_{test}$
Output: Class label of test data
 1: Import word vector set $W$ that is trained by CBOW or Skip-gram model
 2: Initialize the parameters of CNN model
 3: for each sentence $s \in D_{train}$ do
 4:     Get the word vectors $s = [w_1, w_2, ..., w_{n-1}, w_n]$ of all words in $s$ from $W$
 5:     Compute the feature value $c_i = f(v \cdot w_{i:i+h-1} + b)$ through convolutional operation;
 6:     All feature values are formed to a feature vector $\mathbf{c} = [c_1, c_2, ..., c_{n-h+1}]$;
 7:     Get the important feature voctors through pooling operation. $\hat{c} = max(\mathbf{c})$;
 8: end for
 9: Export the well-trained parameters $P_cnn$ of CNN and the feature vector $S_train$ of
    train data
10: Initialize hyperplane parameters $P_{svm}$ of SVM
11: for each vector $s_{train} \in S_{train}$ do
12:     Minimize classification error;
13:     Adjust hyperplane parameters iteratively;
14: end for
15: Export well-trained hyperplane parameters
16: for each sentence $s \in D_{test}$ do
17:     Get the feature vectors $S_{test}$ of sentences through trained CNN;
18: end for
19: for each vector $s_{test} \in S_{test}$ do
20:     Use well-trained SVM to classify test sample;
21:     Output class label;
22: end for
```

Algorithmus 3.1: Text classification algorithm based on convolutional neural network and support vector machines.

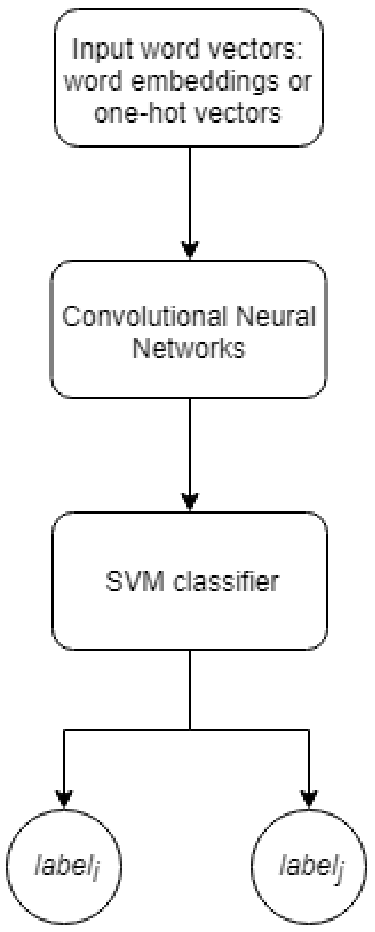

Input word vectors: word embeddings or one-hot vectors
Convolutional Neural Networks
SVM classifier
$label_i$
$label_j$

Figure 3.22: A CNN-SVM model

Therefore, BiLSTM combined with CRF has become a pretty good model for sequence labeling, the structure is shown in Figure 3.23.

The first layer of the model is the word embedding layer. The distributed representation of each word is fed into Bidirectional LSTM (BiLSTM) that automatically extracts context information of each word. The output dimension of BiLSTM layer is label size, which is equivalent to the transition probability from each word to a tag. Then the output of BiLSTM network at each time stamp is fed into the CRF layer that implements sentence-level sequence tagging. A CRF layer is represented by lines which connect consecutive output layers. A CRF layer has a state transition matrix A as parameters, then $A_{i,j}$ represents the transition score to model the transition from tag_i to tag_j for a pair of consecutive time steps [60]. With such a layer, we can efficiently use past and future tags to predict the current tag.

Thus, this BiLSTM-CRF model can use both past and future input features thanks to a bidirectional LSTM component and sentence level tag information instead of individual positions thanks to a CRF layer [18].

```
Input: Train data $D_{train}$, Test Data $D_{test}$
Output: Class label of test data
 1: Import word vector set $W$ that is trained by CBOW or Skip-gram model
 2: Initialize the parameters of Bi-LSTM model
 3: for each sentence $s \in D_{train}$ do
 4:    Get the word vectors $s = [w_1, w_2, ..., w_{n-1}, w_n]$ of all words in $s$ from $W$
 5:    for Forward pass of Bidirectional LSTM model do
 6:       forward pass for forward state LSTM;
 7:       forward pass for backward state LSTM;
 8:    end for
 9:    for Backward pass of Bidirectional LSTM model do
10:       backward pass for forward state LSTM;
11:       backward pass for backward state LSTM;
12:    end for
13:    The representation sequence $[h_1, h_2, ..., h_{n-1}, h_n]$ is generated by the memory cell ;
14:    The representation vector $\mathbf{h}$ is generated by a pooling operation ;
15:    The class labels of the input samples are obtained by the output layer;
16:    Adjust the parameters $O_i$ of the model and the word vector of the input sample by
       backpropagation iteration;
17: end for
18: Export the well-trained parameters $O_{train} = (o_1, o_2, ..., o_n) \in \mathbf{R}^{n \cdot k}$ of Bi-LSTM model
    parameter
19: for each vector $s_{train} \in S_{train}$ do
20:    Maximize the conditional likelihood;
21:    Adjust parameter $A$ iteratively;
22: end for
23: Export well-trained parameters
24: for each sentence $s \in D_{test}$ do
25:    Find the word vectors $s = [w_1, w_2, ..., w_{n-1}, w_n]$ of all words in $s$ from $W$;
26:    Get the feature vectors $O_{test}$of sentences;
27: end for
28: for each vector $s_{test} \in S_{test}$ do
29:    Tagging the input words by using well-trained CRF;
30:    Output label of input words;
31: end for
```

Algorithmus 3.2: Sequential labeling algorithm based on bidirectional LSTM and conditional random filed.

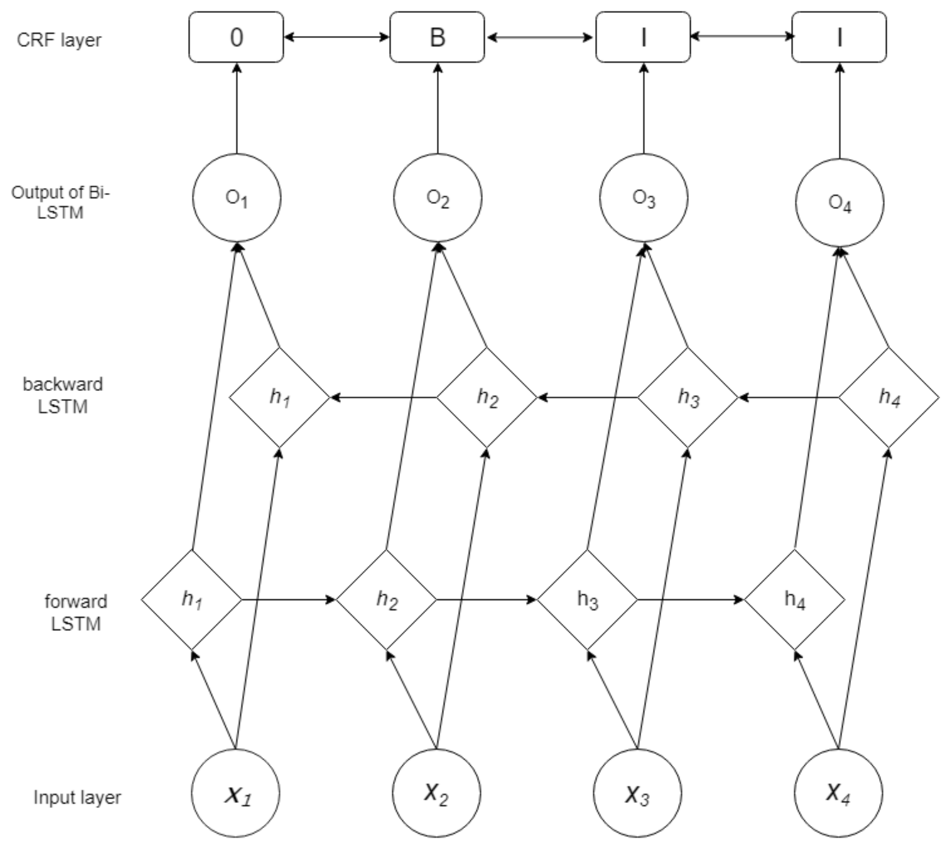


Figure 3.23: A Bi-LSTM-CRF model

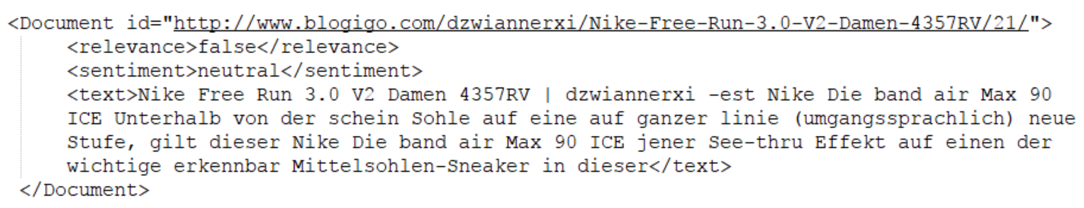
Chapter 4

System Structure

The machine learning approaches introduced in Chapter 3 will be applied to four subtasks of Germeval Shared Task 2017. In this chapter, a system is constructed to integrate these four subtasks together. The Fig. 4.2 shows the overall architecture of the system. The system is decomposed into three parts: a relevance model, a document-level sentiment model and an aspect-level sentiment model (Fig. 4.3) that is consist of aspect model and aspect-level polarity model.

The relevance model takes in a review and outputs the "Deutsche Bahn" relevant reviews. The document-level sentiment model also takes in all reviews as input, the "Deutsche Bahn" irrelevant reviews are then classified as neutral. However, the aspect-level sentiment model only takes in the relevant reviews from relevant model, since a irrelevant review is neutral at document level and a neutral review contains no opinions. An example of our dataset (Fig. 4.1) shows that the irrelevant review does not have an opinion tag.

The aspect model of aspect-level sentiment model takes in a sentence vector (or a set of word vectors) and outputs a probabilistic distribution over the aspects (E#A pairs). The sentiment model takes in a sentence vector and outputs the corresponding sentiment of the sentence. For the aspect-level sentiment analysis, the sentiment is connected to target aspects by augmenting the word vectors with aspect-specific rescaling. These models and the method to link aspects and corresponding sentiments are described in the following sections [56].



```
<Document id="http://www.blogigo.com/dzwiannerxi/Nike-Free-Run-3.0-V2-Damen-4357RV/21/">
    <relevance>false</relevance>
    <sentiment>neutral</sentiment>
    <text>Nike Free Run 3.0 V2 Damen 4357RV | dzwiannerxi -est Nike Die band air Max 90
    ICE Unterhalb von der schein Sohle auf eine auf ganzer linie (umgangssprachlich) neue
    Stufe, gilt dieser Nike Die band air Max 90 ICE jener See-thru Effekt auf einen der
    wichtige erkennbar Mittelsohlen-Sneaker in dieser</text>
 </Document>
```

Figure 4.1: An example of an irrelevant review.

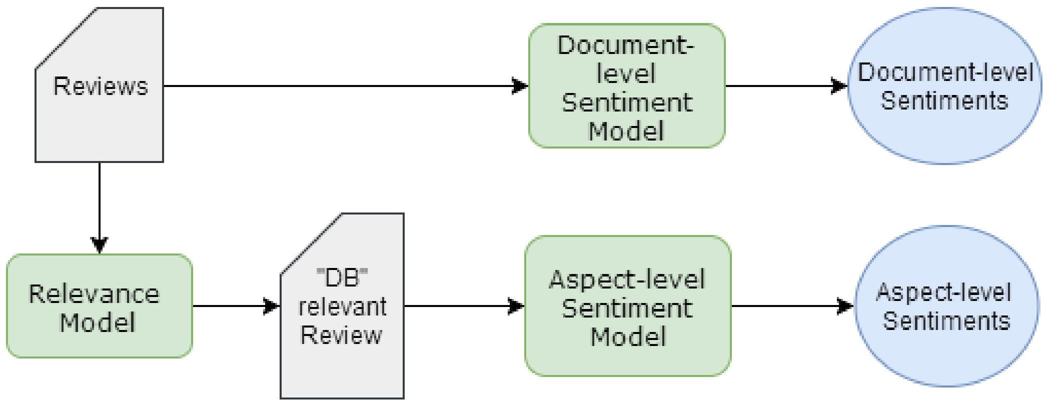

Reviews
Document-level Sentiment Model
Document-level Sentiments
Relevance Model
"DB" relevant Review
Aspect-level Sentiment Model
Aspect-level Sentiments

Figure 4.2: System Architecture

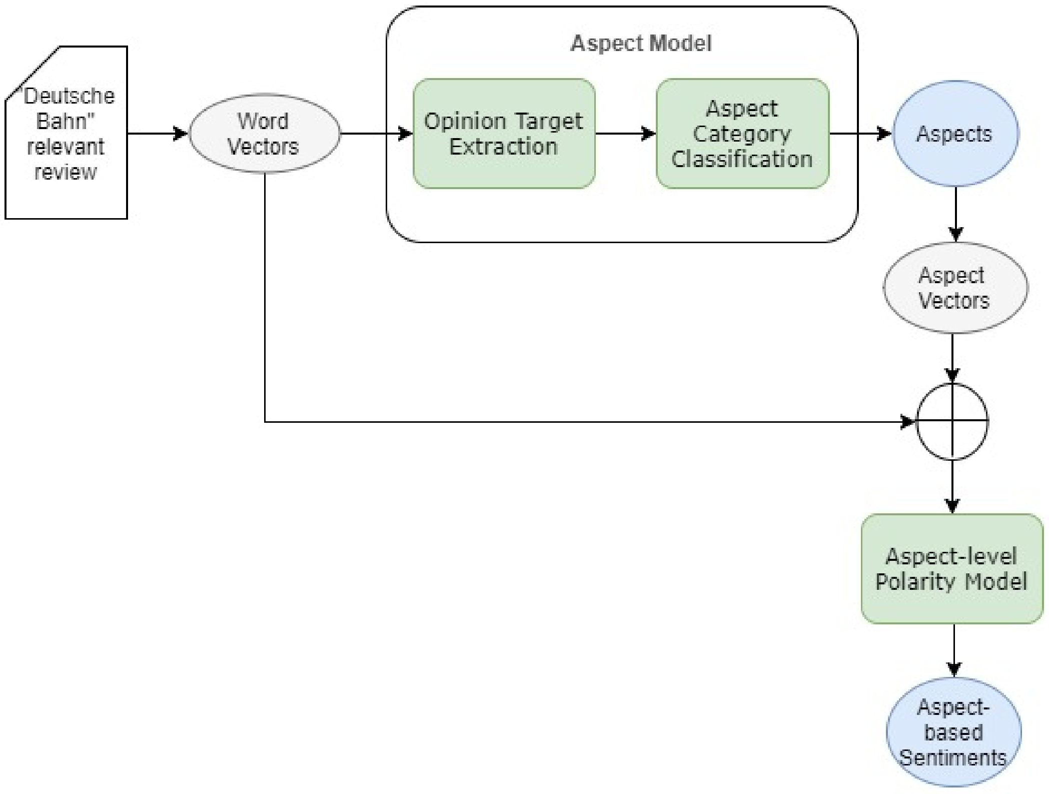

"Deutsche Bahn" relevant review
Word Vectors
Aspect Model
Opinion Target Extraction
Aspect Category Classification
Aspects
Aspect Vectors
Aspect-level Polarity Model
Aspect-based Sentiments

Figure 4.3: Aspect-level Sentiment Model

The system and the results of models are illustrated through the example of Fig. 2.2.
Reviews : Nach 25 Minuten ist mein Nebenschwitzer in der Bahn ausgestiegen. Das Abteil atmet auf.
"Deutsche Bahn" Relevant Reviews : Nach 25 Minuten ist mein Nebenschwitzer in der Bahn ausgestiegen. Das Abteil atmet auf.
Aspect Model :
*Opinion target extraction :*Nebenschwitzer
Aspect cateforization : Atmosphäre#Geruch
Aspect : "Atmosphäre#Geruch"
Document-level Sentiments : Negative
Aspect-level Sentiments : Negative

4.1 Relevance Model

The WE-CNN-SVM model described in Chapter 3.4.1 is adopted for the relevance classification. Since the word embeddings have shown previously to be beneficial to text classification tasks, requiring only minimal feature engineering effort [29]. The first layers embeds words into low-dimensional vectors[1]. If the words do not appear in the pre-trained word embedding, they are initialized randomly. The next layer performs convolutions over the embedded word vectors using multiple filter sizes. For example, sliding over 3, 4 or 5 words at a time. Next, the results of the convolutional layer are max-pooled into a long feature vector. The vectors in the fully connected layer of CNN are regarded as the distributed feature representations, and then these distributed representations are regarded as feature vectors in a SVM classifier [4].

4.2 Document-level Sentiment Model

Since document-level sentiment polarity is a multi-class classification problem, the sentiment is predicted via 3 binary classifiers. The one-vs-one strategy is used to train a binary classifier. Each classifier is trained using a Support Vector Machine. The classifier is enhanced by adding pre-trained word embeddings and neural network features learned from convolutional neural network proposed by Yoon Kim [52].

4.3 Aspect-level Sentiment Model

This model is to determine whether an opinion on an aspect is positive, negative or neutral. As introduced in Chapter 2.1, an aspect is an entity and aspect pair. Thus aspect-based sentiment analysis covers both entities and aspects. As defined by Bing [29], the objective

[1]http://www.wildml.com/2015/12/implementing-a-cnn-for-text-classification-in-tensorflow/

at aspect-level is to discover every quintuple $(e_i, a_{ij}, s_{ijkl}, h_k, t_l)$ in a given document d, where s_{ijkl} indicates the sentiment of the opinion holder h_k about the aspect a_{ij} of entity e_i at t_l time of opinion. To achieve this goal, three core tasks have to be performed : *opinion target extraction; aspect categorization; aspect-level polarity.* The aspect model combines the *opinion target extraction* (subtask D) with *aspect categorization* [29].

Since the document-level sentiment model described in Section 4.2 is aspect-agnostic. It works fairly well with sentences of uni-sentiment. However, when judging sentences with multiple conflicting sentiments, the output is hard to predict [56]. To solve this problem, the aspect vectors will be concatenated with the input word vectors.

Besides, when applying CNN to word vectors, the convolutional layers can be viewed as weighted sum of the word vectors with respect to the shared weight matrix. Then, the largest value is selected by max-pooling layer. Thus, the magnitude of word vectors has a strong influence on the behavior of the CNN. If a word vector is scale up (or down) uniformly on all dimensions, its impact at the max-pooling layer will be enhanced (or reduced). The two observations above result to rescale word vectors by connecting with the corresponding aspect vectors before feeding them into the CNN [56].

4.3.1 Aspect Model

This model is to extract all opinion target expressions of the entities, and then categorize these opinion target expressions into clusters. Each opinion target expression cluster of entity represents a unique aspect [29].

Slot 1: Opinion Target Extraction

The Opinion Target Expression (OTE) is defined by its starting and ending offsets. While the dataset is available in both TSV and XML formats, the OTE task can only be done using the XML format. The opinion target extraction task is tackled as a sequence labeling problem. Doing so allows us to extract an arbitrary number of multi-word expressions in a given text [20].

The conventional *BIO* tagging scheme is used to represent sentence as a sequence of tokens labeled :

Words:	Nach	25	Minuten	ist	mein	Nebenschwitzer	in	der	Bahn	ausgestiegen.
Labels:	O	O	O	O	B	I	O	O	O	O

Table 4.1: An example sentence with labels in BIO format. The target is *mein Nebenschwitzer*,the label *B* indicates the beginning of a target, *I* indicates that the word is inside a target, and *O* indicates a word belongs to no target.

The sequence labeling classifier is trained using Conditional Random Fields (CRF). The word embeddings and output of a BiLSTM model are used as additional features. Thus, the BiLSTM-CRF model mentioned in Chapter 3.4.2 is adopted for this task, which can process sequence-typed input data.

Slot 2: Aspect Categorization

An opinion target expression is an actual word or phrase that appears in the text indicating an aspect category, while an aspect category represents a unique aspect. Aspect categories are typically coarser than the aspect terms of Slot 1 (*opinion target extraction*) [36]. So after opinion target extraction, we also need to categorize the extracted opinion targets, because people often write the same entity or targets in different ways [29]. The Germeval Task has given 19 predefined aspect categories (Table A.1). We need to recognize that each extracted opinion targets refers to at least one of the 19 categories. The process of grouping opinion target expressions into aspect categories (aspects) is called aspect categorization.

Aspect category classification is based on a set of one-vs-all binary classifiers, one classifier for each category found in the training dataset [52]. Each classifier is trained using a Support Vector Machine. The classifier is then enhanced by adding neural network features learned from a Convolutional Neural Network model.

4.3.2 Aspect-level Polarity

The aspect-level polarity of a comment faces more challenges because it is not only determined by the content but is also highly related to the concerned aspect [57]. As well as a comment may contain more than one aspects of the different categories or even a same category. This task is to determine the orientation of sentiment expressed on each aspect in a sentence.

The aspect obtained from aspect model 4.3.1 is splitted into its constituent tokens, e.g. Zugfahrt#Strechennetz –>Zug fahrt, Strechennetz [41]. Then the aspect tokens are concatenated with the words of input sentence and each character of these words are quantized by using one-hot encoding, and then each character is transformed to its corresponding character embedding using a character embedding matrix. The one-hot vectors are fed into Character-level Convolutional Neural Networks to extract features. The sentiment is then predicted via Support Vector Machine classifier.

Chapter 5

Implementation

The implementation of proposed models are constructed based on using Python programming language working on Eclipse platform. Python has a large number of scientific libraries for data processing and machine learning approaches. These libraries are presented in the following sections and their relevance to this thesis are described.

5.1 Library

5.1.1 Scientific Python

This section presents the libraries that have been developed specifically for scientific work and are used by this thesis. First, Scikit-Learn is introduced. Scikit-learn is a free "open source" library for Python and includes a variety of classification, regression and clustering algorithms.

One of the advantages of implementing with Python is that the simplicity of Python is matched with the speed of compiled programming languages. Therefore, highly complex algorithms can be applied in a very short period of time. Another feature that contributes to the popularity of Scikit-learn is that a uniform interface is available for all algorithm classes. All classification algorithms have the functions: *fit* and *predict*. The *fit* function trains the algorithm and *predict* executes the predictions after training.

The fact that these methods are implemented by all algorithms, it is sufficient only to change the class instance and then use another method to solve the present problem. For the requirements of this thesis are the properties mentioned above exactly helpful. Furthermore, Scikit-learn integrates the popular libraries Numpy and SciPy, which allowing good interoperability with other libraries. The following introduce NumPy and scikit.

NumPy (short of Numerical Python) is a basic Python Scientific Computing package that provides fast complicated functions for mathematical and numerical routines. In addition, NumPy enriches the programming language Python with powerful data structures for efficient arithmetic with large arrays and matrices. Furthermore, the module offers a huge number of high-quality mathematical functions to work with these matrices and arrays. The core functionality of NumPy is based on the data structure "ndarray" (ndimensional array), a contiguous storage area of fixed size. Different with the list data structure of python, "ndarrays" are homogeneous typed: all elements of an array must be of the same data type.

The SciPy (Scientific Python) package extends the functionality of NumPy. It provides many user-friendly and efficient numerical routines such as routines for numerical integration and optimization. SciPy also works with NumPy data formats, making it an ideal match.

5.1.2 BeautifulSoup

The Opinion Target Extraction task can only be done using the XML format. Therefore, the python library Beautiful Soup[1] in version 4, shortly called BS4, is used to extract information from XML documents. It is a tree-based parsing, in which the entire document is loaded into memory. It works on the DOM (Document Object Model), i.e. on the syntax tree of the document. Beautiful Soup basically offers very simple and intuitive methods to search and edit the DOM. For the application of the Opinion Target Extraction, only the search and filter function is used, because the information is only extracted and the documents do not need to be modified. Once a document has been read in with Beautiful Soup, it will be converted to a corresponding BS4 object. All data in this object is converted to Unicode and saved as UTF-8.

5.1.3 Tensorflow

TensorFlow[2] is an open source software library for numerical calculations using data flow graphs. Nodes represent math operations in the figure, and edges in the figure represent multidimensional data arrays, i.e., tensors, that relate to each other. Its flexible architecture allows you to expand computing on a variety of platforms, such as one or more CPUs (or GPUs), servers, mobile devices, and more in desktop computers. TensorFlow was originally developed by researchers and engineers from the Google Brain Group (part of the Google Institute for Machine Intelligence) for machine learning and deep neural networks, but its versatility makes it widely used in other computations field.

Tensors As the name indicates, tensorflow is a framework to define and run computations involving tensors. A tensor is a generalization of vectors and matrices to potentially higher dimensions.For example, 1D-tensor is a vector, 2D-tensor is a matrix and 2D-

[1]https://www.crummy.com/software/BeautifulSoup/bs4/doc/

[2]https://www.tensorflow.org/

tensor is a cube. Internally, TensorFlow represents tensors as n-dimensional arrays of base datatypes.

Graph Scopes Any layer or model that are defined inside a TensorFlow graph scope will have all of its variables and operations created as part of the specified graph.

Tensorboard The operations involved in TensorBoard are usually complicated and incomprehensible computations that occur in training large, deep neural networks. To make it easier to understand, debug and optimize TensorFlow programs, a visualization tool called TensorBoard is provided.

First, Tensorflow has a concept of a summaries, which allow us to keep track of and visualize various quantities during training and evaluation. For example, we can keep track of how loss and accuracy evolve over time, and even more complex quantities, such as histograms of layer activations. Summaries are serialized objects, and they are written to disk using a SummaryWriter.

Then we can use TensorBoard to show the TensorFlow image, draw a quantitative index of the image generated and additional data. TensorBoard runs by reading event file of TensorFlow. The TensorFlow event file includes the main data that will be involved in running TensorFlow. The Fig. 5.1 shows an example for a convolutional neural network with filter size 3, 4 and 5.

The general life cycle of the summary data in the TensorBoard is as follows: first, create a TensorFlow graph of the data we want to summarize, and then choose which node we want to summarize.

5.1.4 Keras

Keras[3] is a high-level neural network API, Keras is written in pure Python and based on Tensorflow, Theano and CNTK backend. Keras was born to support rapid experimentation and quickly convert your ideas to results. keras is highly modular, minimalist, and scalable, so it support simple and fast prototyping and model CNN and RNN, or a combination of the two models, switch CPU and GPU seamless.

Keras's design principle is user-friendly. Keras is an API designed for humans. The user experience is always the primary and central part of the consideration. Keras provides a consistent and concise API that drastically reduces the workload for users in general use, while Keras provides clear and practical bug feedback. A model can be understood as a sequence of layers or as a graph of data. Fully configurable modules can be freely combined at the least cost. Specifically, the network layer, loss functions, optimizers, initialization strategies, activation functions, and regularization methods are all separate modules that

[3]https://keras.io/

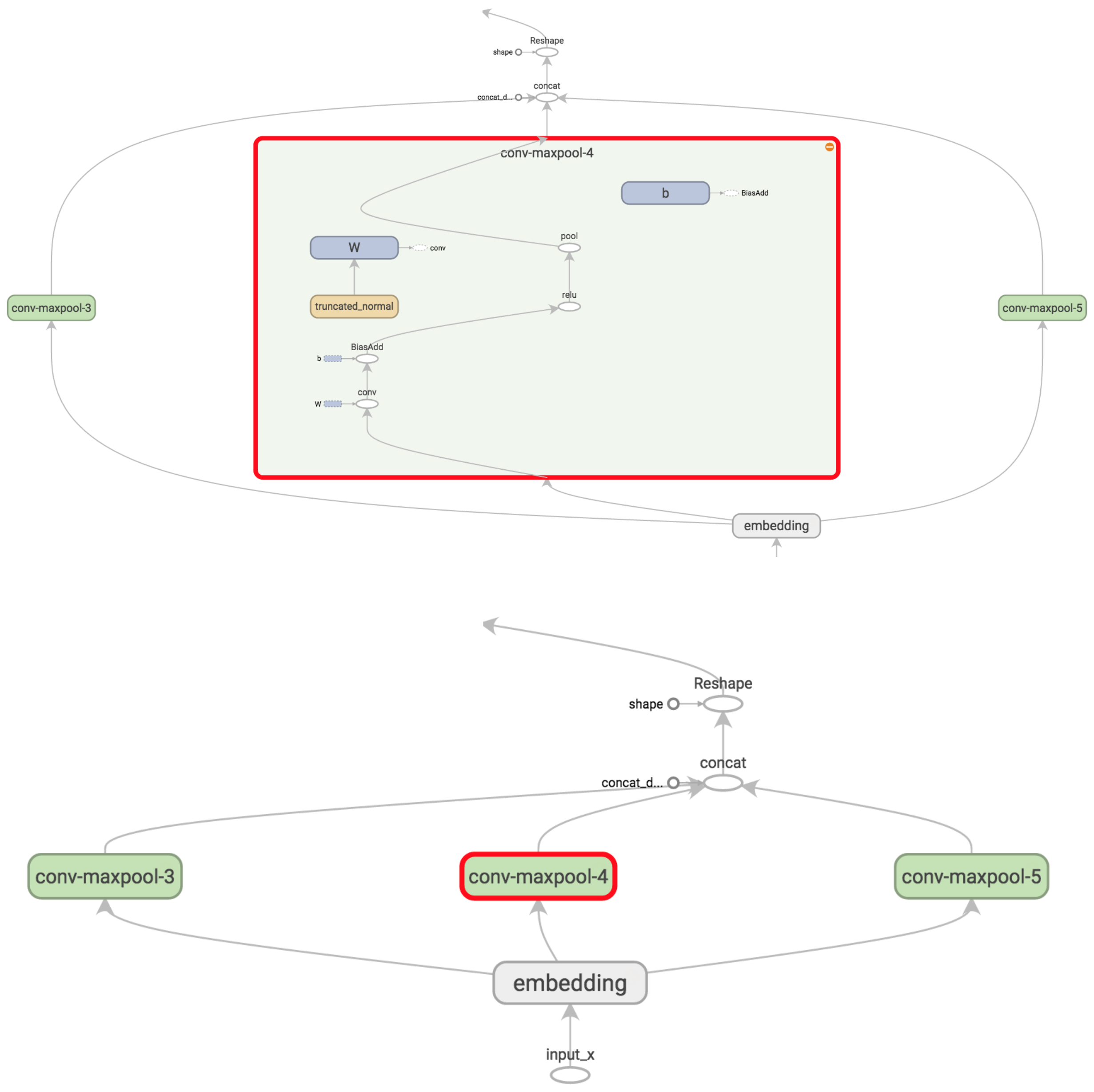


Figure 5.1: Visualizing the operations of CNN in TensorBoard

we can use to build our own model. Keras is scalability. Adding new modules is super easy, just follow the existing module and write new classes or functions. The convenience of creating new modules makes Keras more suitable for advanced research. Furthermore, Keras does not have a separate model profile type, the model is described by python code, making it more compact and easier to debug, and offers expanded convenience.

Chapter 6

Experiments

In this chapter, the evaluation metrics and methods for network training and regularization are intruduced. Then the experiments for evaluating the proposed models for four subtasks of Germeval 2017 are discussed. First,the mode of input word vectors are evaluated. Second, the proposed neural networks and whether a combination of neural network as feature extractor and support vector machine or conditional random fields as classifier is useful for the improvement of performance, are evaluated. Third, the experimental results of baseline system of Germeval 2017 and the performance of different methods on a specific subtask, are compared.

6.1 Evaluation Metrics

In order to simplify and speed up the process of model optimization, the metric *Accuracy* is used as the evaluation metric during the training phase. The accuracy equation declares in the next equation:

$$Accuracy = \frac{\text{true positive} + \text{true negative}}{\text{true positive} + \text{false positive} + \text{true negative} + \text{false negative}} \tag{6.1}$$

The official measure of the Germeval 2017 task is the *micro average F1-measure.* Micro-averaging, which counts the occurrence of a global confusion matrix for each instance of a data set regardless of category, and calculates the corresponding metric. Besides, the *F1_measure* is designed to give a more truthful result even when working with imbalanced classes dataset, because each instance is weighted the same as the others. Therefore, *micro average F1-measure* is used as evaluation metric during the test phase. The *Precision*, *Recall* and *F1-measure* equations are:

$$Precision = \frac{\text{true positive}}{\text{true positive} + \text{false positive}}. \tag{6.2}$$

$$Recall = \frac{\text{true positive}}{\text{true positive} + \text{false negative}}. \tag{6.3}$$

$$F1 - measure = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{6.4}$$

where *true positive* is defined when a text is correctly classified as positive, *false positive* is a negative text which is classified as positive, the *true negative* is a text correctly classified as negative, and *false negative* is a positive text but is classified as negative.

6.2 Network Training and Regularization

The neural networks contain multiple non-linear hidden layers and this makes them very expressive models that can learn very complicated relationships between their inputs and outputs. With limited training data, however, many of these complicated relationships will be the result of sampling noises, so they will exist in the training dataset but not in real test dataset even if it is drawn from the same distribution [49].

Thus the neural networks always tend to easily overfit especially on small and medium sized datasets. To mitigate the overfitting issue, we can augment the cost function with L2-norm regularization terms for the parameters of the networks. In addition, dropout is applied to improve regularization of the deep neural networks [23].

Dropout is a technique for addressing this problem, because the key idea is to randomly drop units (along with their connections) from the neural network during training. This prevents units from co-adapting too much. During training, dropout samples from an exponential number of different "thinned networks". At test time, it is easy to approximate the effect of averaging the predictions of all these thinned networks by simply using a single unthinned network that has smaller weights [49].

6.2.1 Data Processing

Data Augmentation Text classification typically performs better with large training datasets [39]. Especially the character-level CNN, which extract information from raw signals, usually require large-scale datasets to work.

In order to reduce the generalization error of the deep learning models, the dataset will often be augmented to mitgate the overfitting problem. In the image processing, scaling, panning and rotating the image will not change its structure. For the speech recognition, adjusting the tone, speed of speech and noise will not change the result. However, in the text processing, the order of characters can not be changed because the order represents the semantics. Therefore, the best way to do data augmentation would have been using human rephrases of sentences, but this is unrealistic and expensive due the large volume of samples in our datasets. As a result, the most natural choice in data augmentation for us is to replace words or phrases with their synonyms [59]. The thesaurus can be obtained from SentiWS [37], a publicly available german language resource for sentiment analysis [50].

Imbalancement of datasets The second challenge is the imbalance of the dataset, about 83% reviews are topic (Deutsche Bahn) relevance and 68% are neutral, 6% are positive, 26% are negative. There are two methods to address the imbalance problem of datasets, we can either choose an appropriate performance metric or resample the datasets. The metrics such as Confusion Matrix, Precision, Recall and F1 Score are designed to give a more truthful result when working with imbalanced classes. There are two main methods of resampling that can use to even-up the classes: add copies of instances from the under-represented class called over-sampling; delete instances from the over-represented class, called under-sampling. The random sampling approach is very easy to implement and fast to run. But the random under-sampling always leads to loss of information. And random over-sampling adopts a simple replicated sample strategy to increase the minority samples, it can easily lead to overfitting problem, that is, the features learned by the model are too specific and not generalized.

SMOTE Synthetic Minority Oversampling Technique is an improvement based on random over-sampling algorithm. The minority class is over-sampled by taking each minority class sample and introducing synthetic examples along the line segments joining any/all of the k minority class nearest neighbors. Depending upon the amount of over-sampling required, neighbors from the k nearest neighbors are randomly chosen [5].

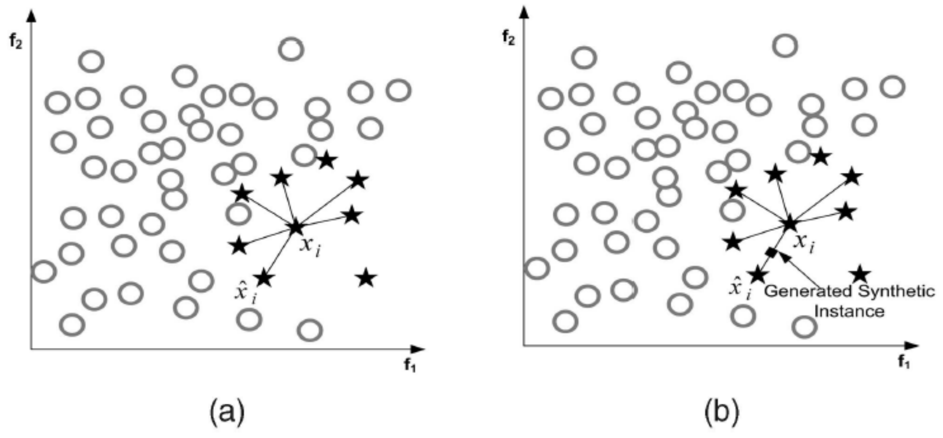


Figure 6.1: Synthetic Minority Oversampling Technique

Synthetic samples are generated in the following way: Take the difference between the feature vector (sample) under consideration and its nearest neighbor. Multiply this difference by a random number between 0 and 1, and add it to the feature vector under consideration. This causes the selection of a random point along the line segment between two specific features. This approach effectively forces the decision region of the minority class to become more general [5]. The new samples which are artificial synthesized based on the minority samples are then added to the datasets.

6.2.2 Optimization Algorithm

It is found that model performance is very sensitive to optimization algorithm. The adam optimization algorithm [24] is used to train the neural networks.

As shown in Figure 6.2, when we train the convolutional neural network proposed by Yoon Kim(Chapter 3.2.1) with the default parameters, gradient descent optimizer does not guarantee good convergence as adam optimizer.,

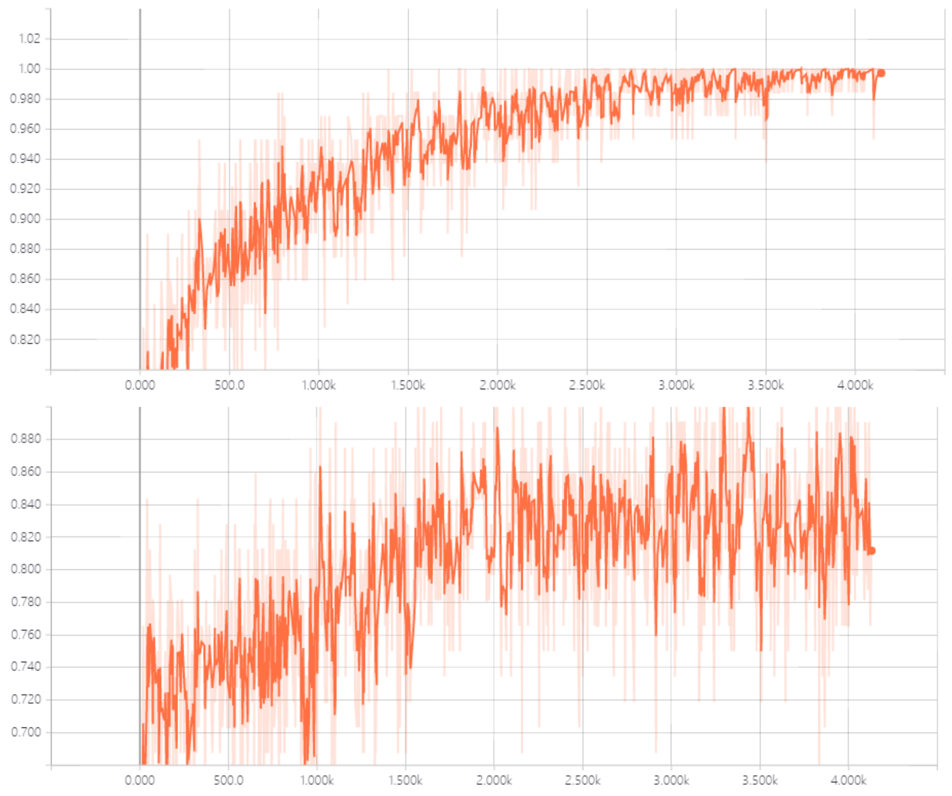


Figure 6.2: Training accuracy of CNN model that are respectively trained by Adam Optimizer (above) and Gradien Descent Optimizer (below).

Adam (Adaptive Moment Estimation) optimization algorithm is an extension of traditional stochastic gradient descent algorithm. It iteratively updates neural network weights based on training data. Stochastic gradient descent keeps a single learning rate (i.e., alpha) updates all weights and the learning rate does not change during training. However, choosing a proper learning rate can be difficult. A learning rate that is too small leads to painfully slow convergence, while a learning rate that is too large can hinder convergence and cause the loss function to fluctuate around the minimum or even to diverge [40].

Adam calculates independent adaptive learning rates for different parameters by calculating first-order and second-order moments of the gradient.

6.3 Task Evaluation

Baseline System: The Germeval 2017 provided a baseline system that contains two classifiers. A linear SVM classifier is used for *relevance classification*, *document-level polarity* and *aspect-level polarity* subtasks. The term frequency and a german sentiment lexicon are used as features for SVM classifier. The opinion target extraction subtask uses a CRF classifier with the token (without standardization/lemmatization/lowercasing) and the Part-of-Speech (POS) tag as features. Both features are unigram features on the current token, no preceding or following tokens are taken into account[1].

6.3.1 Relevance Classification

The CNN-SVM model presented in Chapter 4.1 is experimentally evaluated on the dataset that are provided by Germeval 2017. The Table 6.1 shows the statistical information for this dataset.

	True	False	Total
Train	17041	3895	20936
Test	2050	535	2585
Total	19091	4430	23521

Table 6.1: Class statistic of dataset for relevance classification task.

Training phase In order to verify the effectiveness of the CNN-SVM model proposed in Chapter 4.1, which is based on convolutional neural networks and support vector machines, the CNN-based sentence classification algorithm proposed by Kim [23] and SVM baseline system are used as comparison models.

We use a learning rate of 0.001, batch_size of 64, dimensionality of embedding of 128, dropout keep probability of 0.5 to train models. The parameters (filter size and number of filters per filter size) of CNN and parameters (C and $gamma$) of SVM are tuned during training phase.

Then optimization process of CNN-SVM model is divided into three steps. First of all, convolution neural network is modeled. Then, the parameters of convolutional neural network are adjusted. Finally, the parameters of support vector machines are adjusted. (1)The convolution neural network model is adjusted:

[1]https://sites.google.com/view/germeval2017-absa/baseline-system

The CNN-rand, CNN-static and CNN-non-static models introduced in Chapter 3.2.1 are compared. We experiment all convolutional neural network and support vector machine parameters with the default parameters, and choose the appropriate CNN mode. Table 6.2 shows the experimental results of adjusting the three modes.

Mode	F1-score
CNN-rand	0.810
CNN-static	0.821
CNN-non-static	0.857

Table 6.2: Experimental results of three modes of convolution neural network on the test dataset.

The experimental results in Table 6.2 show that the classification accuracy of CNN-non-static mode is the highest. From the definitions of these three modes, we can see that the CNN-non-static mode uses pre-trained word vectors compared to CNN-rand mode, so that has more prior knowledge. Compared with CNN-static mode, CNN-non-static mode fine-tunes the word vector during training, so that the word vector can learn more meaningful task-related vector expression. In the following experiments, the CNN-non-static mode will be used for experiments.
(2) The parameters *Filter* and *Hidden_unit* of CNN are adjusted, the specific meaning of these two parameters:

(a) *Filter*: the size of the three sliding windows in the convolutional layer of CNN.

(b) *Hidden_unit*: the number of each window size filter [4].

The product of *Filter* and *Hidden_unit* number is the dimension of the sentence feature vector output by the pooling layer. The parameter *Filter* determines the number of contiguous words that the convolution operation is applied. For the text, long-distance words are less relevant, so their size should not be too large. The convolutional neural network takes *filter* [3,4,5] and *hidden_unit* 100 as default. The candidate value of parameter *Filter* is selected as [1,2,3], [2,3,4], [3,4,5], [4,5,6], [5,6,7]. The parameter *Hidden_unit* determines the number of nodes of convolutional layer and the dimension of the output feature vectors. If its value is too small, the model will be lack of representation ability. If its value is too large, the model will be too complicated and leads to a longer training time. The candidate value of parameter *Hidden_unit* is selected as 50, 100, 150, 200. For the above two parameters, a grid search method (searching for all possible combinations of these two parameters) is used to search for the optimal parameters.

The Table 6.3 shows the experimental results of different parameter combinations of CNN on the training dataset.

	Hidden_unit			
Filter	50	100	150	200
[1,2,3]	0.781	0.786	0.836	0.857
[2,3,4]	0.789	0.810	0.797	0.809
[3,4,5]	0.853	0.857	0.863	0.845
[4,5,6]	0.833	0.871	**0.873**	0.843
[5,6,7]	0.795	0.810	0.786	0.825

Table 6.3: Experimental results: Micro F1_score of different parameter combinations of convolutional neural networks on the training set.

The experimental results in Table 6.3 show that when *Filter* = [4, 5, 6], *Hidden_unit* = 150, the highest accuracy of 0.873 is achieved on the training set. Therefore, in the subsequent model test phase. the parameters of CNN are set to *Filter* = [4,5,6] and *Hidden_unit* = 150.

(3) Then the parameters of Support Vector Machines (SVM) are adjusted. The SVM model has two important parameters: C and *gamma*.

(a) C: Penalty factor of support vector machine, which is the tolerance of error. It is mainly used to balance the complexity of the support vector and the misclassification rate in the optimization function. When C is larger, the loss function will be larger. Then we will have more support vectors, which means that support vectors and hyperplane models are more complex and easy to overfitting, i.e., the classification accuracy on the training dataset is high but very low on the test dataset. The generalization ability of the classifier is reduced. In contrast, small C always leads to underfitting.

(b) *gamma*: Kernel coefficient for RBF kernel function. This parameter implicitly determines the distribution of the data after it is mapped to the new feature space. The larger the gamma, the fewer the support vectors. Thus, the number of support vectors affects the speed of training and prediction.

For the optimization of SVM parameters, the method used in this thesis is to let C and *gamma* take values within a certain range. All the combinations of these two parameters are tested on training dataset by grid search. Finally, the C and *gamma* with the highest accuracy are selected as the best parameters of SVM. When more than one set of C and *gamma* in the experimental result achieve the highest classification accuracy, a set with the smallest C value is selected as the best parameters. When there are multiple *gamma* corresponding to the minimum C, the first searched C and *gamma* pair are the best parameters. The reason for this strategy is that high C always leads to SVM overfitting. Therefore, the smaller penalty C among all the parameter pairs C and *gamma* that can achieve the highest accuracy on the test dataset is considered as the optimal parameter.

The candidate parameters C are [0.01, 0.1, 1, 10, 100] and *gamma* are [0.01, 0.1, 1, 10]. The Table 6.4 shows the experimental results on the training dataset of different parameters of support vector machines.

	C				
gamma	0.01	0.1	1	10	100
0.01	0.879	**0.886**	0.765	0.778	0.810
0.1	0.862	0.883	0.774	0.768	0.757
1	0.863	0.863	0.858	0.827	0.757
10	0.852	0.846	0.858	0.827	0.634

Table 6.4: Experimental results on the test dataset of different parameters of support vector machine

The experimental results in Table 6.4 show that when $C = 0.1$ and $gamma = 0.01$, the highest accuracy of 0.886 is achieved.

Compared with the CNN model proposed by Kim, the classification performance of the CNN-SVM model in relevance classification tasks is improved by 1.3%. This shows that it is effective to replace the fully connected softmax layer of CNN by support vector machine classifier and the performance has improved.

Test phase In the test phase, the CNN-SVM model and CNN model are evaluated on the test dataset and the experimental results are compared with baseline system.

First, for the mode of convolutional neural networks in the CNN-SVM model, CNN-non-static is chosen as the model for the test phase. The CNN and SVM of the CNN-SVM model use the optimal parameter results in the training phase. Among them, the optimal parameters of CNN are: *Filter* = [4,5,6], *Hidden_unit* = 150. The optimal parameters of SVM are: C = 0.1, *gamma* = 0.01. The parameters of CNN proposed by Kim are the same as the CNN of CNN-SVM model. During the test phase, the F1_score is used as evaluation metric. The Table 6.5 shows the experimental results.

The experimental results on the test data show that compared with the CNN-based model, the F1-score of CNN-SVM model has improved by 2.1%. Therefore, the replacement of the last fully connected softmax layer of CNN with SVM contributes to the classification performance.

Model	Micro_F1
CNN	0.866
CNN-SVM	0.887
Baseline	0.852

Table 6.5: Experimental results of CNN-SVM model and CNN model on test dataset

6.3.2 Document-level Polarity

The document-level polarity task is also a text classification problem, but with three classes. The task uses the same CNN-SVM model as relevance classification task. Word embeddings and convolutional neural networks are utilized to learn feature vector representation, then the sentiment is predicted via SVM classifiers. The evaluation metrics and training process are the same as that of relevance classification 6.3.1.

The datasets are shown in Table 6.6.

	Positive	Negative	Neutral	Total
Train	1216	5228	14492	20936
Test	155	617	1811	2583
Total	1371	5845	16303	23519

Table 6.6: Class statistic of dataset for document-level polarity task.

Training phase In order to verify the effectiveness of the CNN-SVM model applied to document-level sentiment classification, the CNN and SVM baseline system are used as comparison models.

We use a learning rate of 0.001, batch_size of 64, dimensionality of embedding of 128, dropout keep probability of 0.5 to train models. The parameters (*Filter* and *Hidden_unit*) of CNN and parameters (C and *gamma*) of SVM are tuned during training phase.
(1)The convolution neural network mode is adjusted. The experimental results in Table 6.7 show that the classification accuracy of CNN-non-static mode is also the highest.

Mode	F1-score
CNN-rand	0.701
CNN-static	0.732
CNN-non-static	0.734

Table 6.7: Experimental results of three modes of convolution neural network on the training dataset.

(2)The two parameters *Filter* and *Hidden_unit* of convolutional neural network are adjusted. The Table 6.8 shows the experimental results of different parameter combinations of convolutional neural networks on the test dataset.

	Hidden_unit			
Filter	50	100	150	200
[1,2,3]	0.654	0.661	0.657	0.735
[2,3,4]	0.653	0.723	0.687	0.665
[3,4,5]	0.664	0.734	0.716	0.672
[4,5,6]	0.717	**0.736**	0.725	0.703
[5,6,7]	0.715	0.698	0.658	0.667

Table 6.8: Experimental results of different parameter combinations of convolutional neural networks on the training set

The experimental results in Table 6.8 show that when *Filter* = [4, 5, 6], *Hidden_unit* = 100, the highest accuracy of 0.736 is achieved on the test set. Therefore, in the subsequent model test phase, the parameters of convolutional neural network are set to *Filter* = [4,5,6] and *Hidden_unit* = 100.

(3)The parameters C and *gamma* of support vector machine are adjusted.

The Table 6.9 shows the experimental results on the training dataset of different parameters of support vector machines.

	C				
gamma	0.01	0.1	1	10	100
0.01	0.659	0.710	0.687	0.682	0.710
0.1	0.648	0.723	**0.757**	0.684	0.687
1	0.673	0.734	0.738	0.717	0.689
10	0.654	0.686	0.719	0.725	0.694

Table 6.9: Experimental results on the test dataset of different parameters of support vector machines.

The experimental results show that when $C = 1$ and *gamma* = 0.1, the highest accuracy of 0.757 is achieved.

Compared with the CNN-based model, the classification performance of the CNN-SVM model in classification tasks is improved by 2.1%.

Test phase The CNN-SVM model and CNN model are tested on the test dataset and the experimental results are compared with baseline system.

First, CNN with CNN-non-static mode is chosen as the model for the test phase. The optimal parameters of convolutional neural network are: *Filter* = [4,5,6], *Hidden_unit* = 100. The optimal parameters of SVM are: $C = 1$, *gamma* = 0.1.

The experimental results on the test data show that compared with the CNN-based model, the micro F1-score of CNN-SVM model has improved by 1.8%. Therefore, the replacement of the last full connective layer of convolutional neural networks with SVM also contributes to the multi-class classification performance.

Model	Micro_F1
CNN	0.725
CNN-SVM	0.743
Baseline	0.667

Table 6.10: Experimental results of CNN-SVM model and CNN model on test dataset

6.3.3 Aspect-level Polarity

The aspect-level polarity task is also a text classification problem with three classes. After concatenating aspect vector and word vectors, the character-level convolutional neural network (CharCNN) is applied to learn features from the new concatenated vectors, and then classified by support vector machines. A part of aspect categories are listed in Table 6.11 according to their frequency. The most frequent aspect *Allgemein* is almost 10 times as the second frequent aspect *Zugfahrt*. The reviews that cannot be assigned to any other 18 categories are categorized as *Allgemein*, including the reviews without opinion target as the example in Figure 6.3.

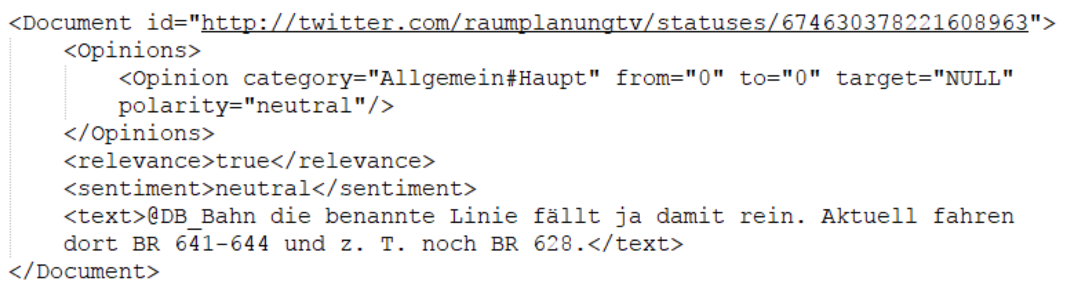
Aspect Categories	#
Allgemein	12564
Zugfahrt	1380
Sonstige Unregelmäßigkeiten	1110
Atmosphäre	930
Ticketkauf	470
Service und Kundenbetreuung	325
Sicherheit	315
Connectivity	254
...	...

Table 6.11: Class statistic of dataset for aspect-level polarity task.

Training phase In order to verify the effectiveness of the CharCNN-SVM model, the CharCNN-based model and SVM baseline system are used as the comparision models.

We use a learning rate of 0.01, batch_size of 128, dropout keep probability of 0.5 as default to train this model. The feature size and kernel size are the same as described in Chapter 3.2.1.

(1) The two character-level CNN modes with small (256) and large (1024) feature map size are evaluated.



```
<Document id="http://twitter.com/raumplanungtv/statuses/674630378221608963">
    <Opinions>
        <Opinion category="Allgemein#Haupt" from="0" to="0" target="NULL"
        polarity="neutral"/>
    </Opinions>
    <relevance>true</relevance>
    <sentiment>neutral</sentiment>
    <text>@DB_Bahn die benannte Linie fällt ja damit rein. Aktuell fahren
    dort BR 641-644 und z. T. noch BR 628.</text>
</Document>
```

Figure 6.3: A review without opinion target.

The Table 6.12 shows the experimental results of two sized character-level CNN on the test dataset.

Feature size	*Micro_F1*
256	0.487
1024	0.427

Table 6.12: Experimental results of character-level convolutional neural networks with small and large feature map size on the training set.

The experimental results in Table 6.12 show that the CharCNN with small feature size has achieved the highest accuracy of 0.487 on the training set.
(2)The parameters C and $gamma$ of support vector machines are adjusted.

The Table 6.13 shows the experimental results on the training dataset of different parameters of support vector machines.

	C				
gamma	0.01	0.1	1	10	100
0.01	0.357	0.441	0.378	0.372	0.272
0.1	**0.509**	0.493	0.364	0.348	0.287
1	0.319	0.354	0.482	0.315	0.286
10	0.334	0.340	0.315	0.323	0.297

Table 6.13: Experimental results on the training dataset of different parameters of support vector machine

The experimental results in Table 6.13 show that when $C = 0.01$ and $gamma = 0.1$, the highest accuracy of 0.509 is achieved.

Compared with the CharCNN-based model, the classification performance of the CharCNN-SVM model in classification tasks is improved by 2.2%.

Test phase In the test phase, the CharCNN model and CharCNN-SVM model are tested on the test dataset and the experimental results are compared.

The small CharCNN is tested on the test dataset. Then the optimal parameters of SVM $C = 0.01$, $gamma = 0.1$ are used. The Table 6.14 shows the experimental results.

The experimental results on the test data show that compared with the CharCNN-based model, the F1-score of CharCNN-SVM model has improved by 2.1%.

Model	*Micro F1*
CharCNN	0.326
CharCNN-SVM	0.347
Baseline	0.322

Table 6.14: Experimental results of CharCNN-SVM model and CharConvNets model on test dataset.

6.3.4 Opinion Target Extraction

Word embeddings and Bidirectional LSTM (BiLSTM) are utilized to learn feature vector representations, the tag of each word is predicted via Conditional Random Fields (CRF). Then the model is evaluated using the micro F1-score on the BIO tags.

Training phase During this training phase, we use batch size of 64, learning rate of 0.002, dropout rate of 0.5 as default. The BiLSTM model with 50 hidden units, 1 forward-LSTM layer and 1 backward-LSTM layer is used.

(1) First, the parameter *Hidden_unit* of Bidirectional LSTM is adjusted. The Table 6.15 shows the experimental results of different number of hidden unit of BiLSTM model on the training dataset.

Hidden_unit	50	100	150	200
Micro F1	0.346	**0.379**	0.286	0.270

Table 6.15: Experimental results of different hidden units of BiLSTM model.

The results indicate that when *Hidden_unit* = 100, the highest micro F1-score is achieved on the training set.

(2) Then the CRF layer is trained on top of BiLSTM. Same as above, the combined BiLSTM-CRF models with different hidden_units are evaluated. The Table 6.16 shows the experimental results of different number of hidden unit of BiLSTM-CRF model on the training dataset.

Hidden_unit	50	100	150	200
Micro F1	0.347	**0.423**	0.365	0.215

Table 6.16: Experimental results of different hidden units of BiLSTM-CRF model.

Test phase In the test phase, the BiLSTM model and BiLSTM-CRF model are tested on the test dataset and the experimental results are compared with CRF baseline.

Model	*Micro F1*
BiLSTM	0.282
BiLSTM-CRF	0.334
Baseline	0.278

Table 6.17: Experimental results of BiLSTM and BiLSTM-CRF model on test dataset.

The experimental results on the test data show that compared with the BiLSTM, the micro F1-score of combining BiLSTM and CRF has improved by 5.2%.

6.4 Summary

In this chapter, four subtasks of Germeval 2017 are evaluated seperately. The CNN-SVM model is adopted for subtask A and B, the subtask C adopts the Character-level CNN-SVM model. These models use the convolutional neural networks as a feature extractor that can automatically learn the distributed feature representation of the input samples. They also take advantages of the support vector machines that classify non-linear data effectively. As a result of this approach, we achieved micro F1-score of 88.7% for subtask A, 74.3% for subtask B and 34.7% for subtask C, which are better than CNN-based classification model. The subtask D is addressed by Bidirectional LSTM that can efficiently use both past and future input features. The output layer of BiLSTM is then replaced by a CRF layer. Experiment result shows that this combined model (33.4%) achieves better labeling performance than BiLSTM model (28.2%).

Chapter 7

Conclusion and Outlook

In this thesis, we aim to address topic and sentiment classification as well as sequence labeling problems by using machine learning approaches. The Support Vector Machines are adopted as classifier for topic and sentiment classification. We enhance the classifier by adding neural network features learned from Convolutional Neural Networks that take in pre-trained word embeddings or one-hot encoded character vectors as input vectors. Then, the most recent state-of-the-art sequence labeling model, BiLSTM-CRF is used for *opinion target extraction* subtask. This model treats Bidirectional LSTM as feature extractor which automatically learn both preceding and following contexts. A CRF layer is trained on top of BiLSTM that jointly decode labels for the whole sentence. These models are truly end-to-end, we do not need handcraft feature engineering or data preprocessing. We show that the proposed models can achieve a reasonable accuracy without handcrafted feature engineering. Experiments show that our CNN-SVM, CharCNN-SVM and BiLSTM-CRF models outperform the CNN, CharCNN and BiLSTM models, respectively.

It is found in the experiment, for the combined models, if the extracted features are not good enough, it is difficult to achieve an ideal result no matter how the parameters of classifier are adjusted. If the features are well extracted, then the influence of parameters of classifier on the model is not significant. It may only change the prediction results of a few samples after adjusting the parameters.

A major improvement for the future would be to create a german sentiment lexicon [57] in order to capture specific words. Both *relevance classification* and *document-level polarity* subtasks use the CNN-SVM model. Their experimental results show that the CNN-SVM model for relevance classification (88.7%) outperforms the CNN-SVM for document-level polarity (74.3%). As mentioned in Chapter 2.2.1, topic classification focuses on keywords, while sentiment analysis should deal with sentiment words and implicit expressions. Besides, the target words are always in the near of the sentiment words or expressions. Thus, a german sentiment lexicon would be beneficial to detect the target words.

Also, the *aspect-level polarity* and *opinion target extraction* subtasks can be benefited from taking into account attention mechanism. For *aspect-level polarity*, the attention mechanism can concentrate on different parts of a sentence when different aspects are taken as input [57]. The aspect embeddings are used to decide attention weights along with word embeddings of input samples. For *opinion target extraction*, the attention mechanism can be incorporated with the BiLSTM model [1]. It is helpful for sequence labeling task because it is capable of highlighting important part among the entire sequence for the labeling task.

Appendix A

Appendix

Category	**Description**
Allgemein	General statement about the DB, which can not be assigned to any other category.
Atmosphäre	The "Atmosphere" category provides feedback on subjects such as a pleasant and relaxed journey to the destination, noise levels, noise from the train or other passengers' harassment, the possibility of concentration or interaction with other travelers.
Connectivity	The "Connectivity" category provides feedback on issues such as the presence, speed, and stability of the wireless and mobile Internet connection, information and entertainment (ICE Portal), and the availability and quality of mobile phone reception.
Design	The "Design" category groups feedback on topics such as the visual design, interior design, carpet and general appearance of the train. For formulations such as "new train" the category should only be selected if the formulation is solitary, not if further specified as in "new train, nevertheless unpunctual ..."

Table A.1: Aspect Categories

Category	Description
Gastronomisches Angebot	The "gastronomic offer" category provides feedback on topics such as the gastronomy service and its availability, the choice and quality of food and drink on the menu and its prices, cleanliness and service in the Bordbistro, waiting time or experience during the stay in the Bordbistro as well as to the service at the place.
Informationen	The category "information" groups feedback on subjects, such as the content, timing and comprehensibility of the passenger information, by the staff or on display boards, both in the event of a fault, in the train or at the station. It is about information on delays, the reason of the delay, changes in the track, onboard bistros or information about connection possibilities.
DB App und Website	The "DB App and Website" category groups feedback on how to use the DB App and the DB homepage.
Service/Kundenbetreuung	The "Service / Customer Support" category groups feedback on issues such as friendliness, competence (accessibility) and availability of the train crew or the availability of the newspapers offered.
Komfort/Ausstattung	Includes general feedback on the topics of comfort such as comfort on the train, comfortable travel, stairs, general spaciousness and equipment especially at the seat, such as reserved and non-reserved seats, comfort seats, reservation indicator, leg roomor luggage rack
Gepäck	The category "luggage" groups feedback on the availability of sufficient space for luggage near the seat.

Table A.1: Aspect Categories

Category	Description
Auslastung und Platzangebot	The category "occupancy / vacancy" groups feedback on the availability of seats as well as sufficient / insufficient number of cars, incl. train / car availability, as well as feedback on availability and retrieval of the selected train / car according to the reservation.
Ticketkauf	The category "Ticket purchase" groups feedback on topics such as the cost of buying a ticket, the sales channels (online, travel agent, vending machine, train), the price information offered, the price-performance ratio of the journey or the process of ticket control ,
Toiletten	The category "toilets" groups feedback on subjects such as functionality, cleanliness, smell or hygiene of the toilets and the functional design of the toilet, as well as the availability of the consumables such as soap, paper towels or toilet paper.
Zugfahrt	The category "Punctuality and Connection" groups feedback on topics such as delays of all kinds or punctuality when reaching the destination, as well as the connection and whether connections are questionable or missed.
Reisen mit Kindern	The "Traveling with Children" category groups feedback on the area of childhood, such as design, placement, space and general topics when traveling with children.
Image	The category "Image" groups feedback on the public image of the DB.
QR-Code	The category "QR code" groups the QR code, which gives the customer the opportunity to complain and to express opinions.

Table A.1: Aspect Categories

Category	**Description**
Barrierefreiheit	The "Accessibility" category groups feedback on the barrier-free movement in trains and at the station.
Sicherheit	The "Security" category groups feedback on safety at the train and at the station.

Table A.1: Aspect Categories

List of Figures

Algorithms

Bibliography

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[2] Adam L Berger, Vincent J Della Pietra, and Stephen A Della Pietra. A maximum entropy approach to natural language processing. *Computational linguistics*, 22(1):39–71, 1996.

[3] Phil Blunsom. Hidden markov models. *Lecture notes, August*, 15:18–19, 2004.

[4] Yuhui Cao, Ruifeng Xu, and Tao Chen. Combining convolutional neural network and support vector machine for sentiment classification. In *Chinese national conference on social media processing*, pages 144–155. Springer, 2015.

[5] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[6] Jason Chiu and Eric Nichols. Sequential labeling with bidirectional lstm-cnns. In *Proc. International Conf. of Japanese Association for NLP*, pages 937–940, 2016.

[7] Gobinda G Chowdhury. Natural language processing. *Annual review of information science and technology*, 37(1):51–89, 2003.

[8] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[9] Thomas G Dietterich. Machine learning for sequential data: A review. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pages 15–30. Springer, 2002.

[10] Cicero dos Santos and Maira Gatti. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78, 2014.

[11] David Golub and Xiaodong He. Character-level question answering with attention. *arXiv preprint arXiv:1604.00727*, 2016.

[12] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[13] Alex Graves. Supervised sequence labelling with recurrent neural networks. 2012. *ISBN 9783642212703. URL http://books. google. com/books.*

[14] Geoffrey E Hinton. Learning distributed representations of concepts. In *Proceedings of the eighth annual conference of the cognitive science society*, volume 1, page 12. Amherst, MA, 1986.

[15] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[16] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

[17] Fu Jie Huang and Yann LeCun. Large-scale learning with svm and convolutional for generic object categorization. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 284–291. IEEE, 2006.

[18] Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*, 2015.

[19] Ozan Irsoy and Claire Cardie. Opinion mining with deep recurrent neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 720–728, 2014.

[20] Soufian Jebbara and Philipp Cimiano. Improving opinion-target extraction with character-level word embeddings. *arXiv preprint arXiv:1709.06317*, 2017.

[21] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer, 1998.

[22] Dan Jurafsky and James H Martin. *Speech and language processing*, volume 3. Pearson London:, 2014.

[23] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[24] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[25] John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

[26] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*, 2016.

[27] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[28] Omer Levy and Yoav Goldberg. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 302–308, 2014.

[29] Bing Liu. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167, 2012.

[30] Pengfei Liu, Shafiq Joty, and Helen Meng. Fine-grained opinion mining with recurrent neural networks and word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1433–1443, 2015.

[31] Xuezhe Ma and Eduard Hovy. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*, 2016.

[32] Andrew McCallum, Kamal Nigam, et al. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Citeseer, 1998.

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

[34] Ivars Namatēvs. Deep convolutional neural networks: Structure, feature extraction and training. *Information Technology and Management Science*, 20(1):40–47, 2017.

[35] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics, 2002.

[36] Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, AL-Smadi Mohammad, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphée De Clercq, et al. Semeval-2016 task 5: Aspect based sentiment analysis. In *Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016)*, pages 19–30, 2016.

[37] Robert Remus, Uwe Quasthoff, and Gerhard Heyer. Sentiws-a publicly available german-language resource for sentiment analysis. In *LREC*, 2010.

[38] Spencer David Rogers. Support vector machines for classification and imputation. 2012.

[39] Ryan Robert Rosario. *A Data Augmentation Approach to Short Text Classification*. PhD thesis, University of California, Los Angeles, 2017.

[40] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[41] Sebastian Ruder, Parsa Ghaffari, and John G Breslin. Insight-1 at semeval-2016 task 5: Deep learning for multilingual aspect-based sentiment analysis. *arXiv preprint arXiv:1609.02748*, 2016.

[42] Alexander M Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*, 2015.

[43] Haşim Sak, Andrew Senior, and Françoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Fifteenth annual conference of the international speech communication association*, 2014.

[44] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[45] Sarah Schrauwen. Machine learning approaches to sentiment analysis using the dutch netlog corpus. *Computational Linguistics and Psycholinguistics Research Center*, 2010.

[46] Toby Segaran. *Programming collective intelligence: building smart web 2.0 applications*. " O'Reilly Media, Inc.", 2007.

[47] Aliaksei Severyn and Alessandro Moschitti. Unitn: Training deep convolutional neural network for twitter sentiment classification. In *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*, pages 464–469, 2015.

[48] Phillip Smith, Mark Lee, John Barnden, and Peter Hancox. *Sentiment analysis: beyond polarity*. PhD thesis, Thesis Proposal, School of Computer Science, University of Birmingham, UK, 2011.

[49] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[50] Aleš Tamchyna and Kateřina Veselovská. Ufal at semeval-2016 task 5: recurrent neural networks for sentence classification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 367–371, 2016.

[51] Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin markov networks. In *Advances in neural information processing systems*, pages 25–32, 2004.

[52] Zhiqiang Toh and Jian Su. Nlangp at semeval-2016 task 5: Improving aspect based sentiment analysis using neural network features. In *Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016)*, pages 282–288, 2016.

[53] Marcel van Gerven and Sander Bohte. *Artificial neural networks as models of neural information processing.* Frontiers Media SA, 2018.

[54] Hanna Wallach. *Efficient training of conditional random fields.* PhD thesis, Master's thesis, University of Edinburgh, 2002.

[55] Hanna M Wallach. Conditional random fields: An introduction. *Technical Reports (CIS)*, page 22, 2004.

[56] Bo Wang and Min Liu. Deep learning for aspect-based sentiment analysis, 2015.

[57] Yequan Wang, Minlie Huang, Li Zhao, et al. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 606–615, 2016.

[58] Yanping Yin and Zhong Jin. Document sentiment classification based on the word embedding. In *4th International Conference on Mechatronics, Materials, Chemistry and Computer Engineering*, 2015.

[59] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015.

[60] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip HS Torr. Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1529–1537, 2015.

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet sowie Zitate kenntlich gemacht habe.

Dortmund, den March 5, 2018

Weihan Pang