Stefan Rigobert Falk, BSc

Supervised Aspect Category Detection in Sentiment Analysis for Opinionated Text

MASTER'S THESIS
to achieve the university degree of
Diplom-Ingenieur
Master's degree programme: Computer Science

submitted to
Graz University of Technology

Supervisor
Dipl.-Ing. Dr.techn. Roman Kern

Institute of
Interactive Systems and Data Science

Graz, September 2017

Acknowledgment

Mein besonderer Dank gilt meinem Betreuer Dipl.-Ing. Dr.techn Roman Kern, der mich mit konstruktiver Kritik und Anregungen im Zuge dieser Arbeit unterstützt und mir damit sehr geholfen hat.

Ebenso gilt mein Dank meinen Eltern Doris und Gerhard Falk sowie meiner Schwester Nina, auf deren Unterstützung ich im Zuge des Studiums stets bauen durfte und die meine Sorgen immer auch ein wenig mitgetragen haben.

Meinem guten Freund David Ganster möchte ich vor allem für die Zeit während des Studiums, seiner Geduld während unseren hitzigen Debatten aber auch für sein Beispiel danken.

Weiteres möchte ich mich auch bei Philipp Kober, Florian Kubin und vielen anderen Freunden, Bekannten und Studienkollegen für ihre Gesellschaft und die gemeinsame Arbeit während und innerhalb des Umkreises der Studienzeit bedanken, jedoch auch für die Zeiten außerhalb dessen unmittelbaren Umfangs.

Abschließend gilt mein Dank auch meiner Freundin Eveline Baumschlager, die mich in den letzten Jahren des Studiums unterstützt hat und für mich zum Antrieb meines Handelns wurde.

Stefan Falk,

Graz im Juli 2017

Statutory Declaration

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.

Graz, _____ _____

Date Signature

Eidesstattliche Erklärung[1]

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am _____ _____

Datum Unterschrift

[1] Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008; Genehmigung des Senates am 1.12.2008

Abstract

The growth of user-generated data in the past decades has led to an increase in research being conducted in the field of natural language processing (NLP). Neural networks have shown promising results in several different language related tasks such as sentiment detection (Socher et al. 2013) or opinion mining (Pang and Lee 2008) both which have become a hot topic with the emergence of social networks and platforms that allow users to write reviews and express opinions towards entities. Detecting the sentiment of short texts (Severyn and Moschitti 2015) can be particularly challenging as missing context information can be encoded in just very few phrases. Building upon the information retrieved from sentiment detection, by combining it with information received from *aspect category detection* systems, allows the determination of positive or negative opinions towards entities or particular aspects of such. Aspect category detection is the task of obtaining the targeted aspect of an opinionated expression. It is the attempt to find out *what* is being talked about or referred to.

The objective of this thesis is to develop system for aspect category detection in terms of NLP information retrieval using neural networks. Particularly, the requirements for the system lean on the definition of Task 5 Slot 1 (Pontiki et al. 2016) for constrained systems of the Semantic Evaluation challenge of 2016.

Contents

List of Figures

1 Introduction

In this introductory chapter, a general overview and a description of the main topic of this thesis is provided. The purpose of it is to help understand what makes aspect category detection an interesting field of research and list relevant research questions on which this thesis focuses. Also, an outline is provided by the last section of this chapter listing the upcoming chapters of this thesis and gives an overview of their individual content.

1.1 Motivation

With the continuous growth of user-generated content in the last decades, *aspect analysis* or *aspect category detection* has become a more relevant niche of natural language processing (NLP), which is of particular importance for aspect-based sentiment detection. Aspect category detection can be described as the task of finding the entity which is being talked about or towards an opinion is being expressed, whereas sentiment detection is the task of finding the polarity of such an opinion, e.g. positive or negative, in a text. In the sentence "The food was expensive but very good." the word "food" could be extracted as the entity which is being talked about and represented using the label `FOOD`. It is also possible to differentiate more closely between the price and the taste of the food by using labels such as `FOOD#PRICE` and `FOOD#QUALITY`. This however depends on the respective task.

Challenges like the SemEval-2016 Aspect Based Sentiment Detection task[1] (Pontiki et al. 2016) are representatives of an increasing interest in this field.

[1] SemEval-2016 Aspect Based Sentiment Detection challenge: http://alt.qcri.org/semeval2016/task5/ (2017-06-19)

Mining, analyzing and understanding written natural language can help many sectors in different industries to better understand the market and their customers. Over social networks such as Twitter, even real time events can be detected, which may not only help a particular industry but can potentially help emergency services, to gather information about tragic events such as natural disasters (Sakaki, Okazaki, and Matsuo 2010).

This shows how the understanding and recognition of aspects, which can be referred to withing a sentence, paragraph or document, can be of major importance for many different information retrieval processes related to language.

From this, the focus of this thesis emerges, which is the development of a system for aspect detection. In particular, the thesis aims to accomplish reasonable results for Task 5, Slot 1 of the SemEval-2016 challenge.

1.2 Research Questions

As described in the previous chapter, the goal of this thesis is to provide a framework for the classification task of aspect categories in written text using a supervised approach. The key research questions, which form the main objectives of this thesis, can be stated as follows:

- What are useful features for aspect category detection?
- Can word embeddings for a small dataset be found by using similar techniques as Word2Vec (Mikolov et al. 2013) to classify aspect categories?
  - How can word embeddings be trained w.r.t. their aspect category labels?
  - How to generate samples for training such word embeddings?

As an overview, the following section provides the outline of all upcoming chapters by summarizing their contents individually.

1.3 Outline

The following Chapter 2 represents a literature review and provides an overview of different approaches which other participants implemented during the course of Task 5 Slot 1 of the SemEval-2016 challenge, namely aspect detection.

Chapter 3 presents the essential methods which are made use of in later chapters including an introduction to neural networks and their mathematical representation and a look into word embeddings and Google's Word2Vec algorithm (Mikolov et al. 2013).

The entire Chapter 4 is dedicated to feature engineering. In this chapter, different kinds of word embeddings are presented which are used as word features and are later combined to form a sentence representation. This chapter also describes a Word2Vec inspired method to generate word embeddings for certain words in a grammatical dependency graph.

Chapter 5 explains the construction of the feedforward neural network multi-label classifier, the generation of training samples and how the classification process is conducted, in detail.

After presenting the details of the classifier, Chapter 6 presents the results of the system developed in the course of this thesis. The performance of each feature separately and combined are presented and further compared to the results of former participants of the SemEval-2016 challenge.

The final Chapter 7 presents the conclusion of this thesis and potential future work.

2 Literature Review

The literature review conducted during the course of this thesis focuses on constrained systems which target Task 5, Aspect Based Sentiment Detection (ABSA), Slot 1 for constrained systems of the SemEval-2016 challenge. In order to give an overview of related work and provide insight into different approaches to solve aspect detection tasks, the following papers have been studied. The ordering goes from higher to lower ranked unconstrained systems of the named task.

2.1 Aspect Detection during SemEval-2016

In his paper "BUTknot at SemEval-2016 Task 5: Supervised Machine Learning with Term Substitution Approach in Aspect Category Detection," Machacek 2016 describes manually compiled *Term Groups*, which are lists of words containing highly descriptive words for each available aspect category. For each category, the frequencies of words are computed, and those which occur more often are manually checked and grouped by assigning them to the corresponding group. However, the question remains whether this isn't equivalent to using an external similarity dictionary which is not permitted by the rules for constrained systems as stated on the challenge's website[1]. For classification, Machacek 2016 used a binary classifier provided by the Vowpal Wabbit[2] learning system (Langford, Li, and Strehl 2007).

Brun, Perez, and Roux 2016 explain in their article "XRCE at SemEval-2016 Task 5: Feedbacked Ensemble Modeling on Syntactico-Semantic Knowledge

[1] SemEval-2016 ABSA: http://alt.qcri.org/semeval2016/task5/ (2017-06-19)
[2] Vowpal Wabbit on GitHub: https://github.com/JohnLangford/vowpal_wabbit (2017-06-19)

for Aspect Based Sentiment Analysis" a two step classification process utilizing the output of a Conditional Random Field (CRF) which has been specialized at word level on the available training data, e.g. the labeled opinion target phrases, to classify terms into one or more aspect categories. In a second step, at sentence level, the classification models associate aspect categories of sentences with probabilities. The aspect categories are then assigned using a threshold over the assigned probabilities.

Hercig et al. 2016 describe in "UWB at SemEval-2016 Task 5: Aspect Based Sentiment Analysis" a rich feature set for an English corpus that is being used for a maximum entropy classifier with an optimized threshold for all available aspect classes. The feature set consists of different kinds of Bag of Words variations, such as *Bag of Words around Verb* which is described as two bags of five words around verbs. Another similar feature described is the *Bag of 5 Word at the Beginning of Sentence* which considers only words at the end of a sentence. Also a *Bag of Bigram* feature, the occurrence of a bigram in a context window, is used. They make further use of classic *Bag of Words* and an additional variation of it that uses a POS filter to remove certain kind of words. Furthermore, TF-IDF features are computed from the training data. Different feature sets are described depending on the corpus domain. For the SemEval-2016 ABSA restaurant domain, additional features such as document vectors (Le and Mikolov 2014) and character n-grams are made use of as well. Altogether, Hercig et al. 2016 use a broad variety of features.

The constrained system developed by Xenos et al. 2016, described in "AUEB-ABSA at SemEval-2016 Task 5: Ensembles of Classifiers and Embeddings for Aspect Based Sentiment Analysis," uses generated lexicons (stemmed and unstemmed) derived from available training data. Lexicons provide scores for uni- and bigrams by computing their F1-Score, Precision and Recall following the approach of Karampatsis, Pavlopoulos, and Malakasiotis 2014 in "AUEB: Two Stage Sentiment Analysis of Social Network Messages." For each score the average, median, maximum and minimum values are used to form features for words. For classification a Support Vector Machine (SVM) is trained for the available categories.

Toh and Su 2016 describe in "NLANGP at SemEval-2016 Task 5: Improving Aspect Based Sentiment Analysis using Neural Network Features" a set of

binary features which consists of the word itself (bigrams are also used for Slot 1) and name lists. The name lists are derived using the available training data. One list contains words which are very often labeled as opinion targets (or aspects), whereas the other considers those words which occur often just as part of an opinion target.

2.2 Summary

This chapter presented several different approaches towards aspect detection which have been applied during the course of the SemEval-2016 Aspect Based Sentiment Detection challenge. The former participants tried various different features ranging from binary features such as bag of words (Hercig et al. 2016), to real-valued word representations using each word's classification strength as measured by a certain metric such as the F1-score (Xenos et al. 2016).

The following chapter, an overview of the theoretical foundations for different methods which are being made use of in the course of the thesis is provided. Using this foundation, the chapter after the next one describes the engineered features which were developed in order to tackle Task 5 Slot 1 of the SemEval-2016 challenge.

3 Background

In this chapter, the most relevant methods which are made use of for the development of the system are introduced and explained. The purpose of this chapter is to provide a basic understanding of neural networks and how their simplest implementations can be utilized for regression of classification tasks as well as give a short introduction into word embeddings and Google's famous Word2Vec algorithm (Mikolov et al. 2013). The materials used along this chapter, and thus recommended for reading, are referenced accordingly during the course of this chapter.

3.1 Artificial Neural Networks

In this section, a short introduction to artificial neural networks is provided to provide a basic understanding of the machine learning techniques used by this thesis. The textbook "Pattern Recognition and Machine Learning" by Bishop 2006 has been used as a reference and this section tries to follow its notation and terminology.

Artificial neural networks are a type of computational model that use a fixed number of adaptive basis functions which can be used in several different machine learning domains such as classification or regression. Nowadays, many different network architectures have proven to be useful in a variety of tasks. A very classic architecture, due to its relative simple structure, is the *feedforward* or *multilayer perceptron* neural network, which will be used in this section to explain the basic characteristics and functionality of neural networks.

3.1.1 Linear Models for Regression

To begin with, we look at a simple multivariable linear model.

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + \ldots + w_M x_M$$

Where $\mathbf{x} = (x_1, \ldots, x_M)^T$ is a input predictor vector and $\mathbf{w} = (w_0, \ldots, w_M)^T$ are the coefficients. Here, w_0 is a parameter which allows fixed offsets in the data and is usually referred to as *bias*. The major limitation of this model is it is only a linear function of its parameters $\mathbf{x}$ and $\mathbf{w}$. It is however possible, to obtain a more useful model by taking linear combinations of a fixed set of nonlinear functions of the input variables. These functions are called *basis functions*, which allow the model to be nonlinear with respect to the input variables, yet stay linear with respect to their parameters. For that reason, such models maintain simpler analytical properties.

$$\begin{aligned} y(\mathbf{x}, \mathbf{w}) &= w_0 + w_1 \phi_1(\mathbf{x}) + \ldots + w_M \phi_M(\mathbf{x}) \\ &= w_0 + \sum_{j=1}^{M} w_j \phi_j(\mathbf{x}) \end{aligned}$$

It is common to introduce a dummy basis function $\phi_0(\mathbf{x}) = 1$ to further simplify this expression to

$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M} w_j \phi_j(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}) \tag{3.1}$$

having $\mathbf{w} = (w_0, \ldots, w_M)^T$ and $\boldsymbol{\phi} = (\phi_0, \ldots, \phi_M)^T$.

This model can be used for a supervised learning task by making it subject to a target function and minimize a *sum-of-squares* error function. It can be shown that the sum-of-squares errors function is related to a procedure called *maximum likelihood method*.

3.1.2 Maximum-Likelihood

Assuming a target variable t that is given by a function $y(\mathbf{x}, \mathbf{w})$ with additive zero mean Gaussian noise given by the random variable ϵ with precision β

(inverse variance) such that

$$t = y(\mathbf{x}, \mathbf{w}) + \epsilon,$$

it is possible to write this as a likelihood function

$$p(t|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(t|y(\mathbf{x}, \mathbf{w}), \beta^{-1}) \tag{3.2}$$

where $\mathcal{N}(\cdot)$ is the Gaussian normal distribution.

For a data set $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, where all $\mathbf{x}_i$ are drawn independently from the distribution (3.2), with corresponding target values given by the vector $\mathbf{t} = \{t_1, \ldots, t_N\}^T$, the likelihood function for the targets $\mathbf{t}$ is given as a product of likelihood functions

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|y(\mathbf{x}, \mathbf{w}), \beta^{-1}), \tag{3.3}$$

by recalling that the joint probability of two independent events is given by the product of the marginal probabilities for each event.

By taking the logarithm of (3.3), the log-likelihood function takes the form

$$\begin{aligned}
\ln p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) &= \ln \prod_{n=1}^{N} \mathcal{N}(t_n|y(\mathbf{x}, \mathbf{w}), \beta^{-1}) \\
&= \sum_{n=1}^{N} \ln \mathcal{N}(t_n|y(\mathbf{x}, \mathbf{w}), \beta^{-1}) \\
&= \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi) - \beta \frac{1}{2} \sum_{n=1}^{N} \{t_n - \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n)\}^2.
\end{aligned} \tag{3.4}$$

The gradient of the log-likelihood with respect to the weight parameters $\mathbf{w}$ is

$$\nabla \ln p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \beta \sum_{n=1}^{N} \{t_n - \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n)\} \boldsymbol{\phi}(\mathbf{x})^T.$$

Setting the gradient function to zero

$$0 = \sum_{n=1}^{N} t_n \boldsymbol{\phi}(\mathbf{x}_n)^T - \mathbf{w}^T \Big(\sum_{n=1}^{N} \boldsymbol{\phi}(\mathbf{x}_n) \boldsymbol{\phi}(\mathbf{x}_n)^T \Big),$$

and further solving it for $\mathbf{w}$, it gives the *normal equation* for the least squares problem

$$\mathbf{w}_{ML} = (\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{t}.$$

3.1.3 Feedforward Neural Networks

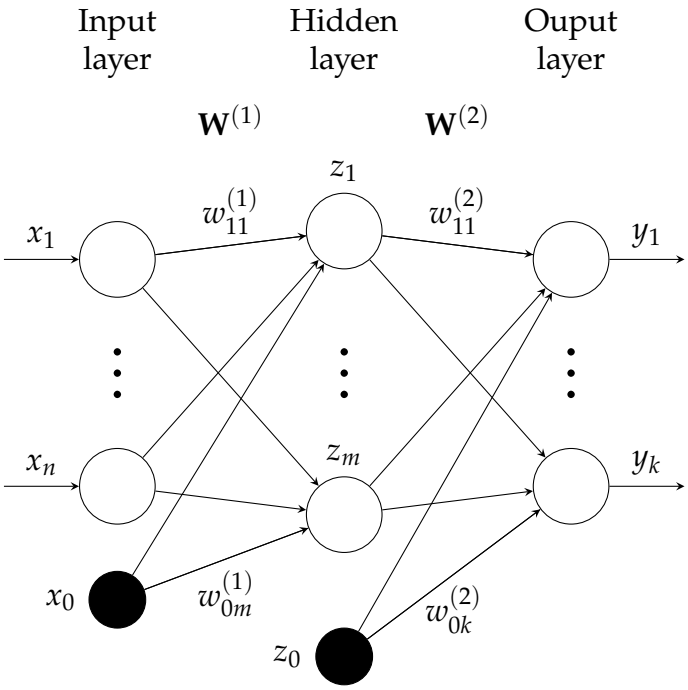


Figure 3.1: Depiction of a two-layer feedforward neural network.

Equation (3.1) can be generalized for classification or regression tasks by transforming the linear function using a nonlinear function $f(\cdot)$ such that

$$y(\mathbf{x}, \mathbf{w}) = f\Big(\sum_{j=1}^{M} w_j \phi_j(\mathbf{x})\Big) = f\Big(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x})\Big) \tag{3.5}$$

where $f(\cdot)$ is called *activation function* or *squashing function* and its inverse is often referred to as *link function*. The class of models which take the form of (3.5) are called *generalized linear models*).

For neural networks this equation gets extended such that the basis functions $\phi_j(\mathbf{x})$ depend on adjustable parameters. Thus, each basis function of a neural network is itself a nonlinear function that takes a linear combination of inputs where the coefficients of the linear combination are adaptive parameters.

By defining $j = 1, \ldots, M$ linear combinations of inputs $x_1, \ldots, x_D$, the resulting quantities

$$a_j^{(1)} = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)},$$

referred to as *activations* of layer (1), are passed to a nonlinear activation function $h(\cdot)$ such that

$$z_j = h(a_j^{(1)}).$$

Each z_j corresponds to the output of the basis functions in (3.5). However, in the context of neural networks, these quantities are being interpreted as the output of the *hidden units*. The adjustable parameters $\mathbf{W}^{(1)} = \{w_{ji}^{(1)}\}$, in the context of neural networks often referred to as *weights*, can be interpreted as connections of the inputs of a neural network to its hidden units which is indicated by the superscript (1). In particular a weight $w_{ji}^{(1)}$ can be understood as the connection of the i-th input to the j-th hidden unit of the inputs.

This process can be repeated to stack multiple *layers* of neurons in a feedforward neural network. Here the word *layer* can be ambiguous. Sometimes it is used to refer to a sets of activation units, for example the network in Figure 3.1 could be called a three-layer neural network for each layer of activation units, namely input, hidden and output layer. However, in this study we follow Bishops terminology by counting the number of adjustable weight matrices in the network. Thus we call the network shown in Figure 3.1 a two-layer neural network due to its adjustable weight matrices $\mathbf{W}^1$ and $\mathbf{W}^2$. The activations $k = 1, \ldots, K$ for the next layer are again linear combinations of its inputs which are in this case simply the outputs z_j of

the previous layers such that

$$a_k^{(2)} = \sum_{j=1}^{K} w_{kj}^{(2)} z_j + w_{k0}^{(2)}$$

where $\mathbf{W}^{(2)} = \{w_{kj}^{(2)}\}$ are again the connecting weights to the next layer. For a two layer network as shown in Figure 3.1, the activations $a_k^{(2)}$ can be interpreted as the outputs $y_k = a_k^{(2)}$ of the network for standard regression problems. However, it is possible to further transform the activations in order to solve classification problems. One example for multiple binary classification problems is to transform the output using a sigmoidal function such that

$$y_k = \sigma(a_k^{(2)})$$

where

$$\sigma(a) = \frac{1}{1 + \exp(-a)}.$$

For multiclass problems the softmax function can be used which is given by

$$y_k = \frac{\exp(a_k^{(2)})}{\sum_{i=1}^{K} \exp(a_i^{(2)})}$$

holding

$$\sum_{i=1}^{K} y_i = 1.$$

Each y_i, interpreted as a probability for a particular class label represented by the i-th dimension, could, for example, be thresholded such that the real-valued output gets transformed over into a binary-valued output vector. A y_k above a given threshold might be interpreted as *true*, which means the k-th label has been assigned to the input shown to the network. This is of course only one way to make use of the output of such a layer under many.

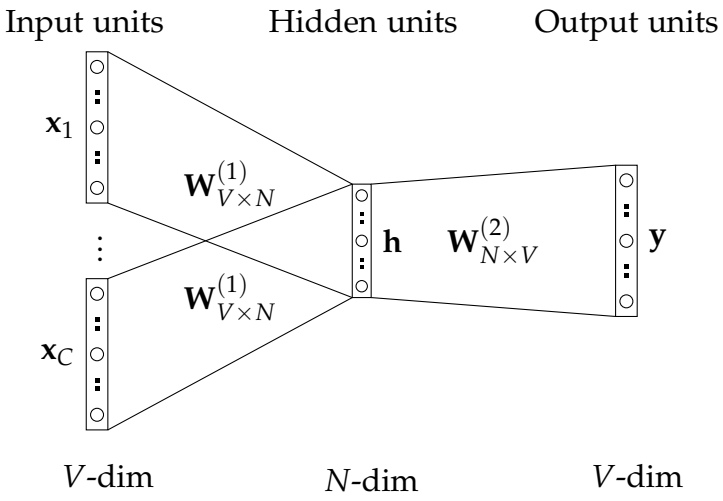


Figure 3.2: The Word2Vec continuous bag-of-words (CBOW) model as introduced by Mikolov et al. 2013.

3.2 Word Embeddings

A word embedding is commonly understood as a vector of real numbers that is a numerical representation of a word which it is mapped to. Technically, a word embedding is a mathematical embedding from a space with one dimension per word to a continuous vector space. This continuous vector space usually has a much lower dimensionality than the original vector space. Also, depending on the underlaying training method, word embeddings can, among other things, capture contextual and semantic information. This can allow word embeddings to be useful even for analogy tasks.

3.2.1 Word2Vec Continuous Bag-of-Words

There are several different methods which can be used for the creation of word embeddings, one of which is Google's Word2Vec algorithm (Mikolov et al. 2013). A detailed explanation is provided by Rong 2014, which also goes into mathematical derivations and details of the algorithm as opposed

to the original paper. As this method is being made use of in the course of this study, a general introduction to this algorithm is provided.

Algorithm

The simplest version of this algorithm is the continuous bag-of-word (CBOW) model. Figure 3.2, which looks similar to the depictions used by Mikolov et al. 2013 and Rong 2014, illustrates the architecture of the network for learning word embeddings using such a model. Conceptually, this network is designed to predict a word at the output, given a set of context words at the input.

Sticking to the terminology used in Section 3.1.3, Figure 3.2 shows a two-layer feed forward neural network. The weight matrix $\mathbf{W}^{(1)}_{V\times N}$, or the first layer of the network, connects the input to its hidden units. The input of this network is a V-dimensional many-hot vector $\mathbf{x} = \{x_i\}$ where $x_i \in \{0,1\}, \forall i = 1,\ldots,V$ which represents the set of context words. The hidden units $\mathbf{h}$ take the form

$$\mathbf{h} = \frac{1}{C}\mathbf{W}^{T\,(1)}_{V\times N}(\mathbf{x}_1 + \ldots + \mathbf{x}_C) \tag{3.6}$$

where each $\mathbf{x}_i$ is a one-hot encoded vector such that $\mathbf{x} = \sum_{i=1}^{C} \mathbf{x}_i$. Thus, the hidden units $\mathbf{h}$ are linear, as they do not transform their inputs with a non-linear activation function. As stated by Rong 2014, this can be interpreted as "copying" all input words to the hidden layer by taking the sum of all rows of the matrix $\mathbf{W}^{T\,(1)}_{V\times N}$ and take the average by dividing this sum by the number of context words C which is given by Equation (3.6).

The hidden units are connected to the output units by the second layer, another weight matrix $\mathbf{W}^{(2)}_{N\times V}$. Thus, the output of the hidden units, which is $\mathbf{h}$, gets propagated over $\mathbf{W}^{(2)}_{N\times V}$ such that

$$u_j = \mathbf{v}^T_{w_j}\mathbf{h},$$

where $\mathbf{v}_{w_j}$ is the j-th column of the matrix $\mathbf{W}^{(2)}_{N\times V}$, which represents a score u_j for each word in the vocabulary. Using a softmax activation function for the output units $\mathbf{y} = \{y_j\}$, the result is the posterior distribution of words

$$y_j = \frac{\exp(u_j)}{\sum_{k=1}^{V} \exp(u_k)}$$

where each y_j is the output of the j-th unit in the output layer representing the probability for word w_j given the input words $w_{I,1}, w_{I,2}, \ldots, w_{I,C}$ or simply

$$y_j = p(w_j | w_{I,1}, \ldots, w_{I,C}). \tag{3.7}$$

Loss Function

The loss function E is the negative maximum of Equation (3.7)

$$\begin{aligned}
E &= -\max p(w_O | w_1, \ldots, w_C) \\
&= -\max y_{j^*} \\
&= -\max \log y_{j^*} \\
&= -\Big(u_{j^*} - \log p(w_O | w_{I,1}, \ldots, w_{I,C})\Big) \\
&= -u_{j^*} + \log p(w_O | w_{I,1}, \ldots, w_{I,C}) \\
&= -u_{j^*} + \log \sum_{j'=1}^{V} \exp(u_{j'}) \\
&= -\mathbf{v}'^{T}_{w_o} \cdot \mathbf{h} + \log \sum_{j'=1}^{V} \exp(\mathbf{v}'^{T}_{w_{j'}} \cdot \mathbf{h})
\end{aligned}$$

which will be minimized during the training process through a backpropagation algorithm. This ought to describe the very fundamentals of the Word2Vec algorithm. Further derivations e.g. the updating rule for the weights are not going to be examined in this work but can be found in Rong 2014 and other sources.

3.3 Summary

This chapter explained the key methods used by the system implemented in the course of this thesis. Artificial neural networks and their mathematical representation have been explained in Section 3.1 by providing an overview of their origin from simple linear models for regression and its extension to non-linear models by using non-linear activation functions called neurons. In addition, a simple feedforwad neural network architecture has been described including how such a model can be used for regression or classification tasks.

Further, word embeddings were explained and how they encode features. Additionally, Google's Word2Vec algorithm, for generating a particular type of embeddings, was explained as well using the excellent work conducted by Mikolov et al. 2013 and the complementary explanation work written by Rong 2014.

4 Features

This chapter describes features which have been derived from a labeled data set like one given for the SemEval-2016 challenge for Task 1 Slot 1.

In the Section 4.1 two different kinds of probability vector word embeddings are engineered and explained. Also shown is a weighting method for words, using the Kullback-Leibler distance measure over the word distributions.

Further, Section 4.2 explains the construction of class-context based word embeddings which are the result of using a slightly modified version of Google's Word2Vec algorithm (Mikolov et al. 2013). This modified algorithm makes use of generated training samples drawn from an annotated corpus using the grammatical dependency graph received from the Stanford CoreNLP Toolkit (Manning et al. 2014).

4.1 Probability Vector Word Embeddings

Each word w_i receives a representation as a probability vector $\mathbf{p}_i$ inside a lookup table $\mathbf{P}$ which is the matrix $\{p_{ij}\} = \{\mathbf{p}_i\}$.

Definiton 1 (Probability Vector) *A probability vector is a non-negative vector with the property that the sum over all elements add up to unity. Formally, let $p_1, p_2, \ldots, p_n$ be a set of probabilities such taht $\mathbf{p} = \{p_0, p_1, \ldots, p_n\}^T$, then $\mathbf{p}$ is called probability vector, if*

$$\sum_{i=0}^{n} p_i = 1$$

For a labeled data set, it is possible to calculate a frequency matrix $\mathbf{F} = (f_{ij})$, where each element represents the frequency of the i-th word in the j-th class. By further normalizing each row of $\mathbf{F}$ with $\mathbf{n} = \{n_i\}$ where $n_i = (\sum_{i=0}^{|C|} f_{ij})^{-1}$ such that

$$\begin{aligned}(\mathbf{F}^T\mathbf{n})^T &= \{(f_{ji}n_i)^T\} \\ &= \{p_{ij}\} \\ &= \mathbf{P}\end{aligned}$$

each element p_{ij} of matrix $\mathbf{P}$ represents the conditional probability for a class c_j given a word w_i.

$$\mathbb{P}[C = c_j \mid W = w_i] = p_{C|W}(c_j \mid w_i)$$

Thus, matrix $\mathbf{P}$ consists of $\mathbf{p}_i$ probability vectors where the i-th vector in the matrix corresponds to the i-th word in the vocabulary.

4.1.1 Target Context Feature

Since the provided labels do correspond to single opinion target expressions, e.g. the phrase *"spicy tuna roll"* with `FOOD#QUALITY` in *"The spicy tuna roll was unusually good!"*, one way of creating a frequency matrix $\mathbf{F}_t$ is to count how many times a word occurred inside a target expression for each class but also count how many times a word did occur in a sentence which was not labeled by any class. For this purpose, an additional class `NONE` was introduced to account for such a case. The final embeddings $\mathbf{P}_t$ are then calculated with $\mathbf{P}_t = (\mathbf{F}_t^T\mathbf{n}_t)^T$.

4.1.2 Sentence Context Feature

Similar to the Target Expression probability vector feature, it is possible to count how many times a word occurred only within a sentence that contained a certain aspect and ignore whether the word was part of the target expression or not. The final matrix $\mathbf{P}_s = (\mathbf{F}_s^T\mathbf{n}_s)^T$ provides according word embeddings. Note that there is also a `NONE` class for sentences which do not contain a target expression.

4.1.3 Probability Vectors Weight Feature

word	$D_{KL}\left(p_{C\|w_i \in W} \| p_C\right)$
wine	1.1842
glass	1.1430
wines	1.0729
list	1.0424
stocked	1.0257
priced	1.0188
..	..

word	$D_{KL}\left(p_{C\|w_i \in W} \| p_C\right)$
..	..
been	0.0410
one	0.0392
my	0.0388
have	0.0340
to	0.0332
i	0.0237

Table 4.1: The highest and lowest ranked words in the corpus according to the Kullback-Leibler divergence measure by comparing the distribution over class labels of words to the actual distribution of class labels.

Intuitively, the information of how important a word is in terms of expressiveness with respect to the entire corpus, and its respective distribution over class labels, can be exploited. Given a distribution of class labels over sentences, each word in the corpus has a distribution of its own over available labels. From this, it is possible to introduce a measure which assumes that the closer a word's distribution complies with the distribution of class labels, the less significant a word represents one or more particular classes. A common measure for the difference between probability distributions is the Kullback-Leibler divergence.

Definiton 2 (Kullback-Leibler divergence) *Let p, q be two discrete probability distributions on X. The non-symmetric difference between two probability distributions, relative entropy or Kullback-Leibler divergence of p with respect to q is defined as*

$$D_{KL}(p||q) = \sum_{x \in X} p(x) \log \frac{p(x)}{q(x)}$$

By comparing the distribution of words over class labels to the distribution of class labels by making use of the Kullback-Leibler divergence, the resulting

weight for each word can be interpreted as a measure of the significance of a word within the corpus by calculating a weight vector

$$\mathbf{s} = \{s_i\} = \left\{ D_{KL}(p_{C|w_i \in W} || p_C) \right\}$$

where each s_i is a weight for the corresponding word w_i. Table 4.1 shows the highest and lowest ranked words according to the Kullback-Leibler divergence measure applied on the sentence context probability vectors as described in Section 4.1.2.

Note that by using a threshold, the set of words could further be split into two sets of words where one set corresponds to highly descriptive, the other to low descriptive words. Lower descriptive words may further be used as a list of stopwords. A word is highly descriptive, if its distribution over class labels is very different from the distribution of class labels. This would correspond to a larger probability vector weight feature. The explanation for lower descriptive words follows analogously. However, for the following classification task, no such list of stopwords is generated or used.

4.1.4 Visual Evaluation

It is helpful to visualize a sentence by stacking word embeddings for each word to a sentence embedding matrix. Using probability vectors, where each element represents a particular class, each row of such a matrix represents a corresponding word in a sentence, whereas each column represents a target class.

In Figure 4.1 two such matrices are compared using the sentence context probability vectors for one and using the sentence context probability vectors each multiplied with the Kullback-Leibler divergence weight for the other. The difference is clearly visible, as less descriptive words were down- and more descriptive words were upscaled.

By calculating the sum of all rows of such sentence embeddings, it is possible to further compare the different vectors visually. Figure 4.2 shows the relative change for a sentence embedding that was calculated in one case with the pure sentence context probability vectors and in another case

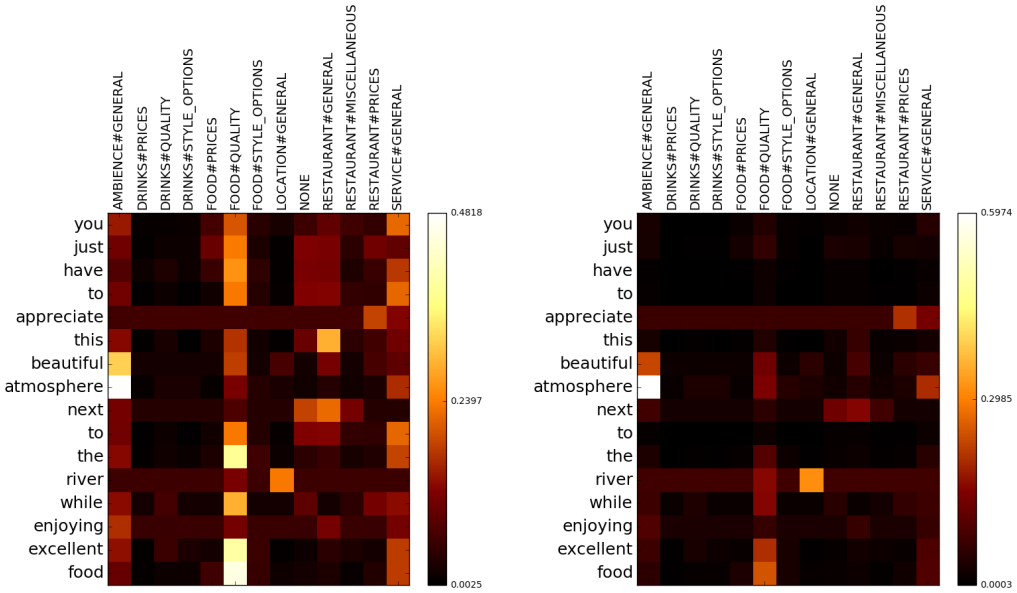


Figure 4.1: Visualization of a sentence using probability sentence context probability vectors without using the Kullback-Leibler divergence weight (lhs) and having it applied (rhs).

with said vectors but also having the Kullback-Leibler divergence weight applied. This example highlights, how a less common class label, here `FOOD#PRICES`, becomes relatively more emphasized, as may be expected in the given example sentence.

The resulting word embeddings can further be visualized to highlight word similarities by applying a dimensionality reduction method such as Principal Component Analysis (PCA). By selecting a certain set of words and reducing their corresponding embedding vectors to two or three dimensions, it is possible to gain intuitive insight into the underlying feature space.

Figure 4.3 shows two PCA plots for a certain set of words. Here ”food”, ”meal”, ”fish”, ”restaurant”, ”place”, ”location”, ”wine”, ”beer” and ”glass”, were chosen purposely to form three groups ”food”, ”restaurant” and ”drinks” in the context of the restaurant domain. Taking a closer look at both plots in Figure 4.3, it appears that similar words do have a tendency to

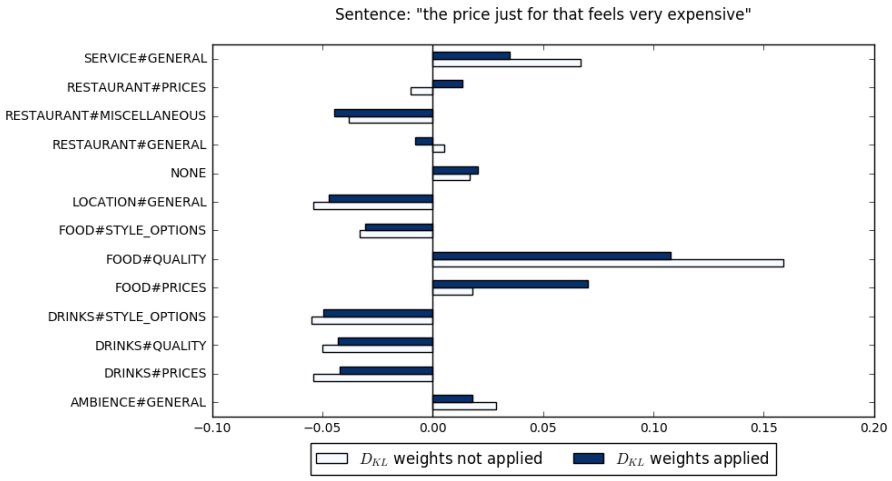
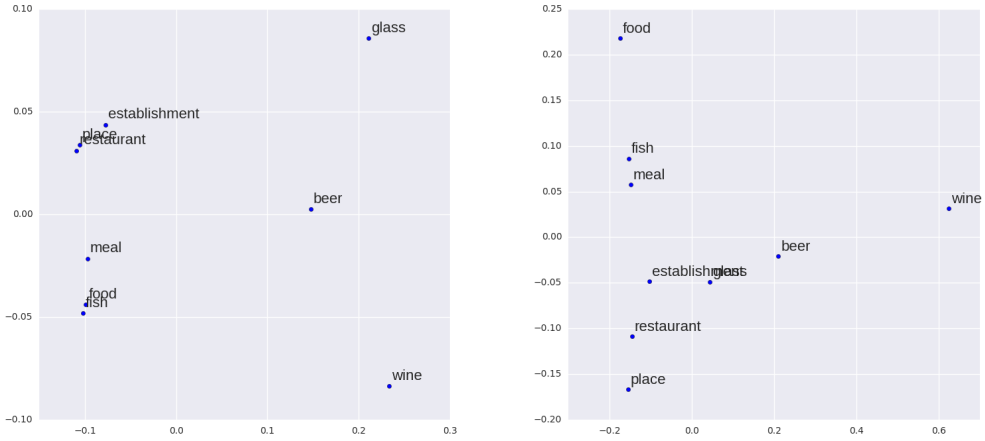


Figure 4.2: Example of two sentence vectors for one example sentence. One calculated using sentence context probability vectors, the other using sentence context probability vectors having the Kullback-Leibler divergence weight applied to emphasize less common classes like FOOD#PRICES in this example.

be closer together (where ”similar” should be interpreted carefully). The word ”place” must not necessarily have a high similarity with the word ”restaurant”, but in the context of the *domain* one could argue that this makes sense. Having the context in mind, the word ”location” could also be interpreted as a synonym for the word ”restaurant” and thus also for the word ”place”. However, in Figure 4.4, where corresponding cosine similarities are shown for each word to all other words, it can be seen that ”location” does have a negative cosine similarity with the words ”restaurant” and ”place”. Thus, the Subfigures 4.3a and 4.3b might lead to the false conclusion that the word ”location” is relatively similar to the words ”restaurant” and ”place”.

4.2 Class-Context Based Word Embeddings

One potential drawback of probability vectors, as described in the previous section is, that they do not consider only *one* context word, that is, the conditional probabilities for a class c_j given a particular word w_i, or $\mathbb{P}[C =$



(a) *Sentence* context feature vectors. (b) *Target* context feature vectors.

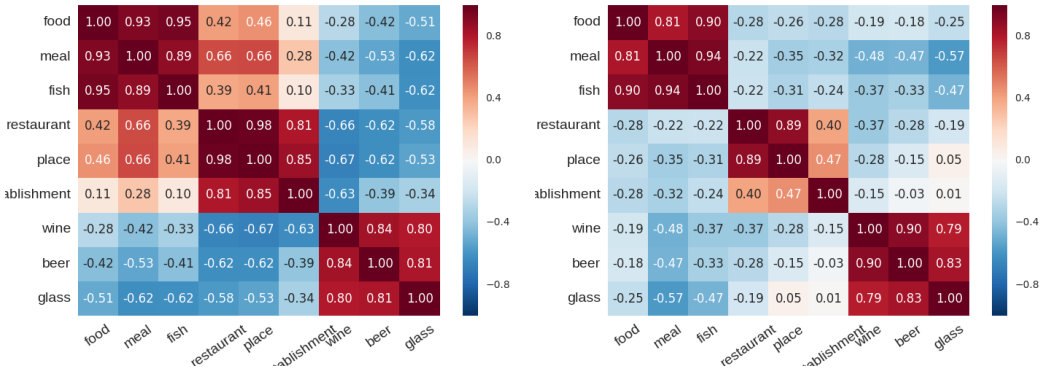
Figure 4.3: PCA plots using different feature vectors for a particular set of words.

$c_j \mid W = w_i]$. However, it can be useful to engineer a word embedding feature which tries to capture a wider context window for a number of words $w_k, \ldots, w_l$, $\mathbb{P}[C = c_j \mid W = w_k, \ldots, W = w_l]$, in order to find new word representations.

Google's Word2Vec algorithm, a neural network based method described by Mikolov et al. 2013, aims to model the distribution of a word given all words in a fixed-sized context window, or $\mathbb{P}[W = w_i \mid W = w_j, \ldots, W = w_k]$.

In order to create class-context based word embeddings, the target for the network has been changed from a given word to a given class, thus modeling $\mathbb{P}[C = c_j \mid W = w_k, \ldots, W = w_l]$. Similar to the Word2Vec algorithm, the resulting weights, which represent the word vectors for the weight matrix of the first layer of the network, should capture contextual information during the training process and bring similar words closer together within the feature space.

Two major components are required: training samples derived from an annotated data set and a training method which is a slightly modified version of Google's Word2Vec algorithm. The following section explains



(a) *Sentence* context feature vectors. (b) *Target* context feature vectors.

Figure 4.4: Cosine similarity matrices for sentence (lhs) and target (rhs) context probability vector features.

how training samples are generated and how they are used for training the class-context based word embeddings.

4.2.1 Generating Training Samples for Word Embeddings

Before word embeddings, which have been trained on the context of words given a particular class, can be found, a method is required to derive training samples from a given data set.

To derive training samples for a sentence, the entire target phrase and all directly connected words are drawn from the sentence to form *one* training sample. A concrete example for a sentence, as shown in Figure 4.5, would be to select the target phrases "food" with the label `FOOD#QUALITY` and "portions" with the label `FOOD#STYLE_OPTIONS` and in order to create two samples

```
FOOD#QUALITY       - The, food, lousy
FOOD#STYLE_OPTIONS - the, portions, tiny
```

for training the word embeddings. The required grammatical dependency graph has been extracted using the Stanford CoreNLP Natural Language Processing Toolkit (Manning et al. 2014).

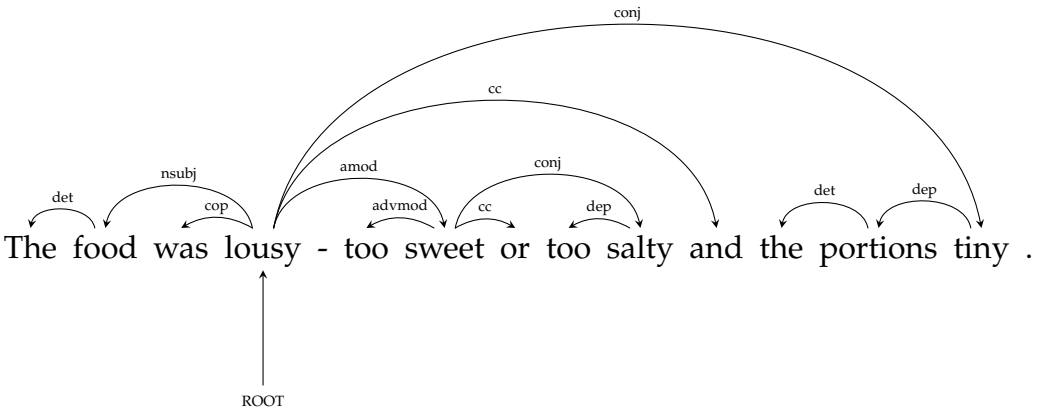


Figure 4.5: The grammatical dependencies for a given sentence using Stanford CoreNLP Natural Language Processing Toolkit.

The following listing shows an actual example of generated training samples for the network generated using the provided data for Task 5 of the SemEval-2016 challenge.

```
..
["RESTAURANT#GENERAL", ["great", "place", "this"]]
["SERVICE#GENERAL", ["the", "waitress", "patient"]]
["FOOD#QUALITY", ["food", "phenomenal", "the"]]
["SERVICE#GENERAL", ["prompt", "friendly", "great", "service"]]
["FOOD#QUALITY", ["and", "great", "pizza", "service"]]
["SERVICE#GENERAL", ["pizza", "fantastic", "service"]]
["SERVICE#GENERAL", ["a", "shorter", "wait", "small", "was"]]
["LOCATION#GENERAL", ["a", "of", "block", "magnificent", "end"]]
..
```

Other methods

For the generation of training samples two other strategies were tested as well. Since the results did not improve under those two methods they are not explained in detail. However, withing the scope of this thesis it was not

possible to determine why these approaches did not improve the results compared to the method explained above. For completeness, the other two approaches are summarized in the following.

The first approach was to look at a sentence and its labels. Now, for each label, the respective probabilities of all the words within the sentence were looked up. Each word then gets drawn according to its probability p_ij (see also Section 4.1), which is the probability of the word for a particular class label. The idea behind this approach was to generate more general samples and increase the chance to collect words which are not reachable by just one hop in the grammatical dependency graph. However, using this approach did not improve the results.

The second approach was to use the grammatical dependency graph and make hops with a decreasing probability. For example, the first hop was taken with a chance of 50%, but the next one only with a chance of only 25%. Different settings have been tried without showing an improvement.

4.2.2 Training and Evaluation

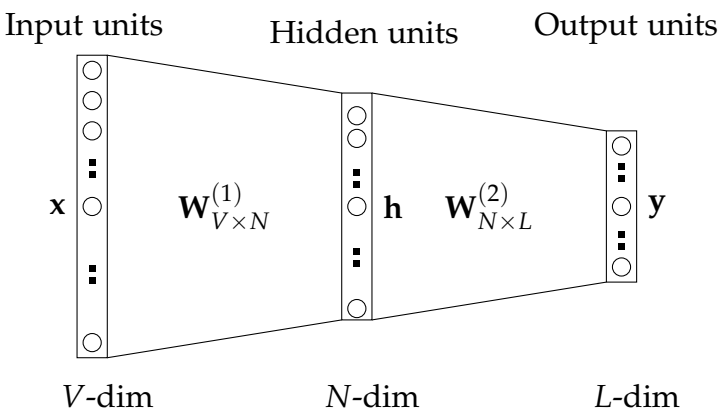


Figure 4.6: Schematic illustration of the feedforward neural network used for learning the class-context based word embeddings.

In order to obtain word embeddings, a feedforward neural network with a hidden layer for the input words and an output layer for target labels are

used. As in Mikolov et al. 2013 and Rong 2014, the activation function of the hidden layer is the average of all words in the current context of present words.

The weights for the hidden layer, the matrix $\mathbf{W}^{(1)}$, is a $V \times N$ sized matrix where V is the number of words in the vocabulary and N the size of the resulting embeddings, which is similar to the vanilla Word2Vec algorithm. The actual modification is the hidden layer, which connects to the softmax output layer over the $N \times L$ sized weight matrix $\mathbf{W}^{(2)}$, where L is the number of class labels, instead of V, again the size of the vocabulary, as in vanilla Word2Vec. In contrast to Word2Vec, which tries to predict a word given a set of context words, this algorithm tries to predict a class label given a set of context words. Figure 4.6 illustrates the overall architecture of the network.

The input vector $\mathbf{x}$ is a bag-of-words representation of the context words whereas the output vector $\mathbf{y}$ is the one-hot encoded target vector representing the target class label to predict.

Training

For the restaurant domain, the word embeddings are trained for 200 epochs with $N = 27$ and using a small batch size of 10 samples each. The network weights are initialized randomly using a uniform distribution. Additionally, the training samples are shuffled randomly after each epoch. The network uses the *Adam optimizer* (Kingma and Ba 2014) for approximating a binary cross-entropy loss function.

The model was implemented using Keras Chollet et al. 2015 with Tensorflow (Abadi et al. 2015) as backend in Python 3.5.

Evaluation

Figure 4.7 (lhs) shows another PCA plot of the same words already used in Section 4.1 and a corresponding cosine similarity correlation matrix (rhs).

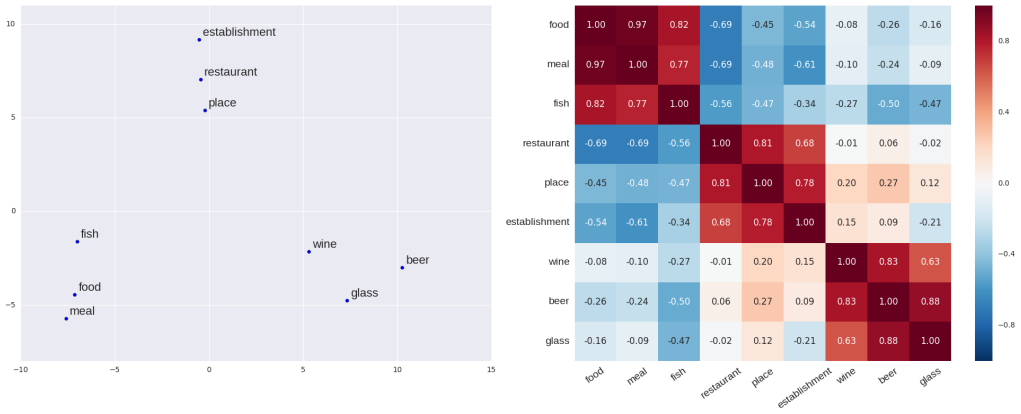


Figure 4.7: (lhs) PCA plot for a given set of words using class-context based word embeddings and (rhs) the corresponding cosine similarities of those words.

Figure 4.8 shows the 15 most frequent words for each class and their corresponding embeddings, taken from $\mathbf{W}^{(1)}$, concatenated horizontally. Clearly visible patterns can be seen which appear to separate categories.

Noting the very uniform vectors in Figure 4.8, further investigation revealed that these are words which are, although highly frequent, not present in the training data for the class-context based word embeddings.

4.3 Summary

This chapter explained how features for aspect detection can be extracted from a labeled data set and illustrated it using the data provided for the SemEval-2016 challenge of Task 1 Slot 1.

The extraction of two different types of probability vectors was shown in Section 4.1. The first type, which included the context of the whole sentence, and the second type, which included only the context of a given target phrase. In addition, a weight, or a ranking measure, using the Kullback-Leibler divergence function, for words in a corpus was explained. This was used to apply weights to the probability vectors, which created another set of word embeddings.

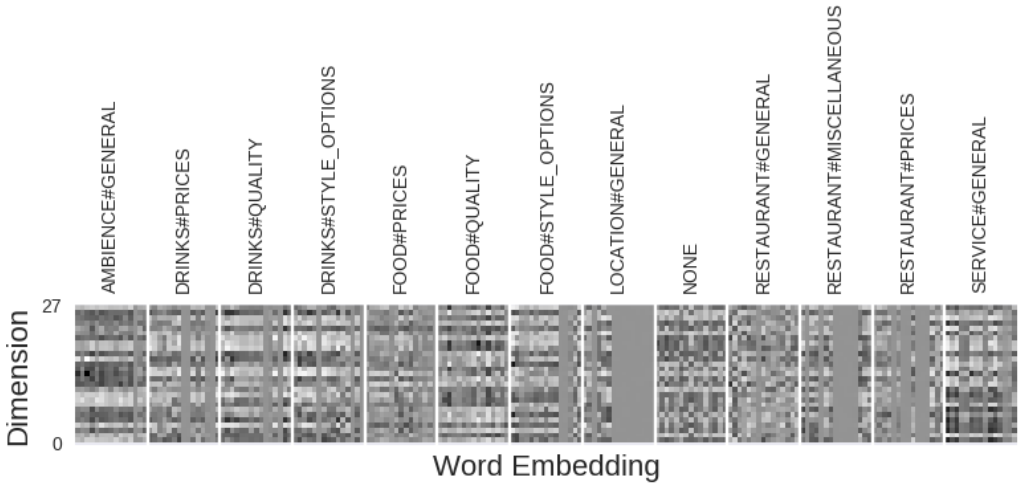


Figure 4.8: The final class-context based word embeddings of the 15 most frequent words for each category by concatenating their vectors horizontally. Each group of 15 word embeddings is separated by a white vertical line to visually separate the categories from each other.

In Section 4.2, class-context based word embeddings, trained on training samples extracted from the labeled data set, have been derived using a Word2Vec-based method.

The following chapter tests the introduced features and evaluates the system build on these features against the results of previous participants of Task 1 Slot 1 of the SemEval-2016 for constrained systems.

5 Classifier Implementation and Training

The classifier is implemented using Python 3.5 and the Keras (Chollet et al. 2015) neural network API framework using the Tensorflow (Abadi et al. 2015) backend. In the following, the architecture of the classifier and its implementation are presented.

5.1 Multilabel Feedforward Neural Network Classifier

5.1.1 Implementation

For classification, a two-layer feedforward neural network is implemented using rectified linear units for the first and tanh units for the second (output) layer. The input layer's size depends on the features being used whereas the size of the output layer corresponds to the number of available classes in all cases. To prevent overfitting the regularization technique *dropout* (Srivastava et al. 2014) is applied on the second layer with a dropout rate of 10%.

5.1.2 Generating Training Samples

Each training sample consists of an input vector $\mathbf{x}_i$, which represents an entire sentence, and its corresponding target vector $\mathbf{t}_i$, which is a binary vector representing all labels of the sentence.

In particular, the input of the network depends on the selected features. In general all selected features are concatenated to form a single feature vector $\mathbf{v}_{ik}$. In the beginning, all representations are fetched to form a feature vector for each word in a sentence. The final input of the network, or a single training input sample, is the sum of all feature vectors $\mathbf{x}_i = \mathbf{v}_{i1} + \mathbf{v}_{i2} + \ldots + \mathbf{v}_{ik}$.

The target vectors are binary encoded classes where each dimension corresponds to one class. Since the network uses tanh as the output layer, each dimension t_k of the target vector $\mathbf{t}_i$ takes values in $\{-1, 1\}$ where 1 corresponds to `true` and -1 to `false`.

5.1.3 Classification

Running the network in classification mode requires the sample input vector to be propagated through the network and the output to be converted to a binary vector. The conversion is done by simply comparing each output value of the output vector against 0. If the output of a node is greater than 0, the sentence is assigned the corresponding label of the output dimension. Analogously if the value is smaller than or equal to 0, the label is not assigned.

6 Results

In this chapter, the results for the features are presented. The results are generated by applying a system which works with the previously described features on the SemEval-2016 Aspect Based Sentiment Detection data set and comparing the scores to the *second* placed system, namely *XRCE* (Brun, Perez, and Roux 2016), in Slot 1 for unconstrained systems.

The reason for comparing the results to the second placed system instead of the first placed, namely *BUTknot* (Machacek 2016), is, because Machacek 2016 describes features which were generated by "*manually*" checking and sorting words into "*high precision*" list of terms. The definition of an *unconstrained* system, however, is a system which uses only the provided data and does not use additional resources such as lexicons or additional training data[1]. The following arguments should support the fact that the *BUTknot* system (Machacek 2016) is using additional resources.

The word "*savory*" does not occur in the entire corpus of Task 5 of the SemEval-2016 challenge thus for the sake of argument, it may be assumed that it occurred once. Having only *one* sample of this word won't provide very descriptive probability vectors such as they are described in Section 4.1. However, using pre-trained word vectors such as the GoogleNews word vectors[2], we can determine that the nearest four words to *savory* are *tangy*, *delicious*, *spicy* and *flavorful*, in that order. Using these embeddings, and considering the word count of *delicious*, one may decide that the rare word *savory* should have a more similar probability distribution over class labels than the word *delicious* thus improving the quality of *savory*'s word

[1] The rules for constrained and unconstrained systems for Task 5 of the SemEval-2016 challenge can be found at http://alt.qcri.org/semeval2016/task5/ (2017-06-19)

[2] Pre-trained word embeddings (GoogleNews corpus): https://github.com/mmihaltz/word2vec-GoogleNews-vectors (2017-06-19)

embedding. However, instead of using pre-trained embeddings, one could also *"manually"* sort words into groups or adjust their probability vectors in order to improve their overall word representation.

6.1 Single Feature

For comparison, each feature has been used solely for the classification task and has been trained and evaluated for 100 epochs in a first phase. The evaluation results are shown in Figure 6.1.

Starting with the weakest result, the class-context based word embeddings explained in Section 4.2 reach a maximum F1 score of 0.5893 which seems to be its upper performance limit.

The probability vectors, as explained in Section 4.1, do show a better performance when used individually than the class-context based word embeddings. As the graph shows, the sentence context based probability vectors (Section 4.1.2), as well as the target context based probability vectors (Section 4.1.1) show very similar results. With 0.6608, the sentence context based probability vectors appear to perform comparable with the target context based probability vectors, which show a maximum F1 score of 0.6529.

The best performance using only one feature comes from the sentence context based probability vectors using the Kullback-Leibler weight as described in Section 4.1.3. With a F1 score of 0.6608, the sentence context based probability vectors slightly outperform the target context based ones which show a score of 0.6529.

As can be seen from the final overall rankings in Table 6.3, all features, except the class-context based word embeddings outperform at least the baseline and place a system at rank 5, as can be seen in Table 6.3.

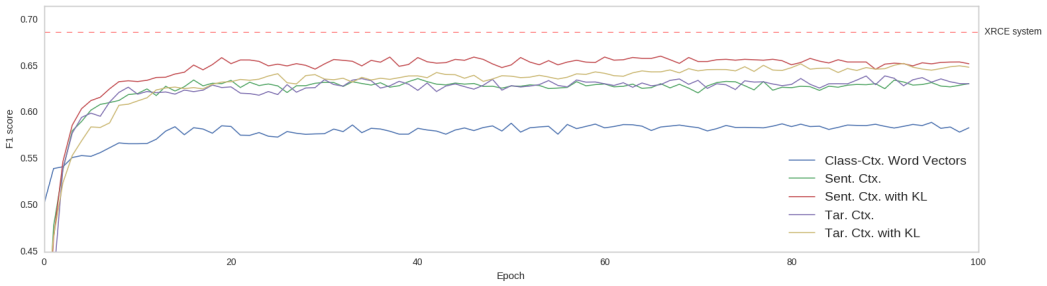


Figure 6.1: The development of all available features and variations. The highest F1 score is reached by Sentence Context vectors *with* applied Kullback-Leibler weights. The lowest score is given by the Class Context Word Vectors.

6.2 Features Combined

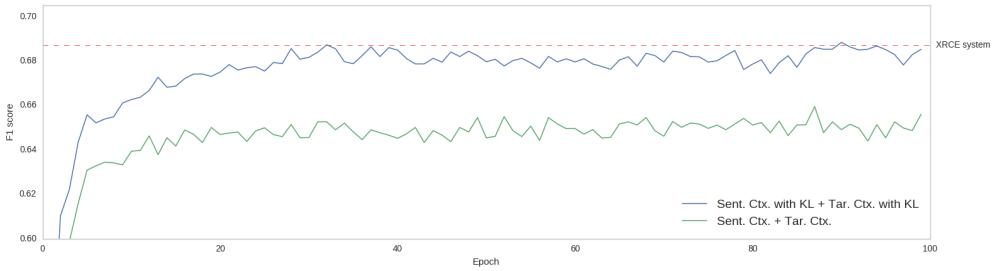
In the second phase, several combinations of the features are tested. The sentence and target context based probability vectors were combined and evaluated. Also, in order to see the impact of the Kullback-Leibler weighting, a test was performed with, and another without additional weighting. Figure 6.2a shows the F1 evolution over 100 epochs for each feature.

The graph shows a maximum F1 scoreof 0.6594 for the combination of the two probability vector types *without* using additional weighting. Using the weights, the performance goes up to 0.6883 which is a considerable improvement.

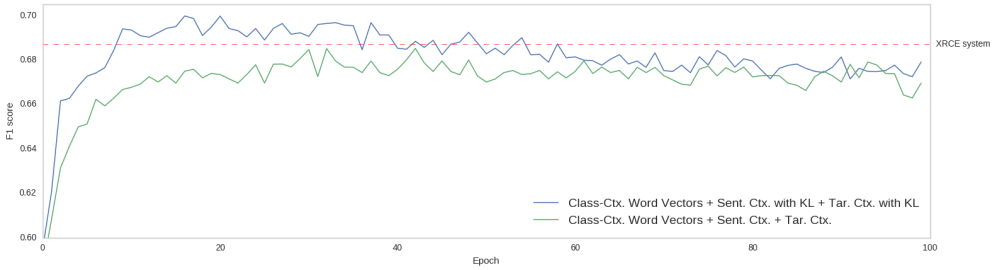
By comparing these results to the overall rankings in Table 6.3, the combination of weighted probability vectors is comparable to the 2nd best placed system *XRCE*.

The last remaining comparison was performed in the third phase using not only the probability vectors (Section 4.1) but also the class-context based word embeddings (Section 4.2). Once again, one run used the weighted versions of the probability vectors and another the unweighted versions. The results are presented in Figure 6.2b.

As can be clearly seen in Figure 6.2b, including the class-context based embeddings does increase the performance considerably. This feature combined with the two types of probability vectors delivers a F1 score of 0.6851



(a) The development of the F1 score by the combination between Sentence and Context probability vectors presenting one run with and another without Kullback-Leibler weights applied. The use of the weights does show a significant positive impact taking the F1 score close to the official result of the *XRCE* system.



(b) The development of the F1 score using all available features combined. Further, a run with and another without using Kullback-Leibler weights is shown. The development of the run using the weights shows that it could potentially outperform the official results of the *XRCE* system.

Figure 6.2: Results for combined features.

even without using the Kullback-Leibler weights. However, the best result is the combination of all features and additional weighting for the probability vectors which reaches a maximum F1 score of 0.6997. As Figure 6.2b also highlights, these features can even outperform the *XRCE* system.

6.3 Feature Scores Comparison

The previous section compared the different features and feature combinations by showing their F1 evolution over 100 epochs each. Table 6.1 provides an overview of the evaluation results and also shows the precision and recall for each feature set.

Feature	Prec.	Rec.	F1 score
Sentence Context	0.7072	0.5787	0.6365
Target Context	0.7272	0.57065	0.6395
Sentence Context with KL	0.7213	0.6096	0.6608
Target Context with KL	**0.7491**	0.5787	0.6529
Sentence & Target Context	0.7125	0.6137	0.6594
+ with KL	0.7247	0.6554	0.6883
Class-Ctx. Word Vectors	0.7054	0.5060	0.5893
+ Sentence & Target Context	0.7276	0.6473	0.6851
+ *with KL*	0.7402	**0.6635**	**0.6997**

Table 6.1: An overview for different feature combinations and the resulting F1 scores with corresponding precision and recall values.

As shown in Table 6.1, the combination of class-context based word vectors with sentence and target context based probability vectors, which were weighted using the Kullback-Leibler weights, has the highest F1 score, as well as the highest recall. The highest precision of this feature set is only slightly outperformed by weighted target context based probability vectors.

Table 6.2 presents the impact of each feature by successively removing features one at a time. The table does not show all possible feature combinations but shows the F1 score change by removing features in a comprehensible manner.

Feature	F1 score	Diff.	Comment
All Features	0.6997	-	-
- Class-Ctx Word Vectors	0.6883	−0.0114	-
- KL	0.6594	−0.0403	-
- Sentence Context	0.6395	−0.0602	Target Context only
- Target Context	0.6365	−0.0632	Sentence Context only

Table 6.2: The development of the F1 score by successively removing features.

6.4 Comparing Results

Considering the absolute best result of all feature sets, the proposed system could potentially reach the second place in the overall ranking of constrained systems in Slot 1 of the SemEval-2016 Aspect Based Sentiment Detection challenge[3] with an F1 score of 0.6997 as shown in Table 6.3.

The table shows the official results of the challenge and places the system, which has been developed in the course of this thesis, amongst them using the name "**My System**". The results for *My System* are shown for comparative purposes and are *not* part of the official results of the challenge.

Rank	System	F1 score	Rank	System	F1 score
1	BUTkn	0.7149	7	TGB	0.6391
-	**My System**	**0.6997**	8	DMIS	0.6175
2	XRCE	0.6870	9	IIT-T.	0.6122
3	UWB	0.6781	10	*Baseline*	0.5992
4	AUEB.	0.6735	11	INSIG.	0.5830
5	NLANGP	0.6556	12	CENNL.	0.4057
6	LeeHu.	0.6545			

Table 6.3: The official ranking of Task 5 Slot 1 of the SemEval-2016 challenge. The result for *My System*, developed in the course of this thesis and its potential ranking is shown for a sole comparative reason and is *not* part of the official results.

[3] Official results SemEval-2016 Task 5 (ABSA): http://alt.qcri.org/semeval2016/task5/data/uploads/submissionsresults.zip (2017-06-19)

6.5 Summary

In this chapter, the results which were collected by using the data set provided in the SemEval-2016 Aspect Based Sentiment Detection challenge as training and testing framework were presented. Different feature sets were evaluated and compared to each other and their results were arranged clearly in Table 6.1. As a point of reference, the performance results of the features were compared against previous results obtained by previous participants of this challenge, which were shown in Table 6.3.

7 Conclusion

In the course of natural language processing, aspect category detection is a key issue for the understanding content in language. The complexity of natural language, and all its nuances in terms of embedded *meaning* embedded, represents huge obstacles for its analysis and to the information retrieval technologies. This thesis presents a system developed for aspect category detection as defined in Task 5 of the SemEval-2016 challenge (Pontiki et al. 2016), which shows comparable results to other systems evaluated on the corresponding data set of the challenge.

The proposed system works on different kinds of word embeddings which can be derived from a labeled data set. After computing the respective word embeddings, a two-layer feedforward neural network is trained to perform a multilabel classification task.

The performance of the system was presented without the use of any threshold optimization as compared to other systems and is better than other comparable systems. As explained in Chapter 5, the output layer of the feedforward neural network corresponds to each label with a single tanh unit, and thus does not require a threshold over labels as is required when using a softmax output layer.

7.1 Future Work

Along the thesis, multiple new questions and ideas emerged which could not be addressed in the course of it, thus are candidates for further research work. In the following, a few thoughts on open questions of the proposed approach are presented.

7.1.1 Class-Context Based Word Embeddings

Generating Training Samples

For the generation of training samples, a rather naive approach has been chosen. At this time, only one hop is considered in the grammatical dependency graph, which might be insufficient, e.g. due to conjunctions, to capture *all* context words for the training of class-context based word embeddings. This naive approach could be replaced by a more sophisticated method and re-evaluated.

Also, the other two approaches for drawing samples described in Section 4.2.1 could be investigated in more detail.

Robustness and Encoding

The number of labeled sentences in the Task 5 data set of the SemEval-2016 challenge consists of only 2,000 labeled sentences. This small number of sentences makes it hard for Google's Word2Vec algorithm (Mikolov et al. 2013) to find good representations and makes it harder for more complex networks such as CNNs to derive features for classification. Usually, although in *un*constrained systems, pre-trained word vectors are used or trained on a particular corpus of choice (see for example Toh and Su 2016). Firstly, it would be interesting to see how class-context based word embeddings work on a larger corpus, and secondly, it would be interesting to take a closer look at the word embeddings themselves in order to make statements about the information which they are encoding.

7.1.2 Extending to Unconstrained System

The proposed system works under the rules of Task 5 of the SemEval-2016 challenge for *constrained* systems. As stated in Chapter 6, this definition is rather vague. However, a quite obvious path for the proposed system would be to try out its delivering power when combined with additional resources

such as pre-trained word embeddings or additional (external) training data besides the provided data in Task 5 of the SemEval-2016 challenge.

7.2 Summary

The system developed for aspect category detection during the course of this thesis compares favorably with current state-of-the-art systems for the same task. For future work, some suggestions have been presented with respect to the focus of this work on constrained systems and the low availability of training data for the presented method.

Bibliography

[Aba+15] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: http://tensorflow.org/ (visited on 06/19/2017).

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006. ISBN: 0387310738.

[BPR16] Caroline Brun, Julien Perez, and Claude Roux. "XRCE at SemEval-2016 Task 5: Feedbacked Ensemble Modeling on Syntactico-Semantic Knowledge for Aspect Based Sentiment Analysis." In: *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. San Diego, California: Association for Computational Linguistics, 2016, pp. 277–281. URL: http://www.aclweb.org/anthology/S16-1044.

[Cho+15] François Chollet et al. *Keras*. https://github.com/fchollet/keras. 2015. (Visited on 06/19/2017).

[Her+16] Tomáš Hercig et al. "UWB at SemEval-2016 Task 5: Aspect Based Sentiment Analysis." In: *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. San Diego, California: Association for Computational Linguistics, 2016, pp. 342–349. URL: http://www.aclweb.org/anthology/S16-1055.

[KB14] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization." In: *CoRR* abs/1412.6980 (2014). URL: http://arxiv.org/abs/1412.6980.

[KPM14] Rafael Michael Karampatsis, John Pavlopoulos, and Prodromos Malakasiotis. "AUEB: Two Stage Sentiment Analysis of Social Network Messages." In: *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Dublin, Ireland: Association for Computational Linguistics and Dublin City University, 2014, pp. 114–118. URL: http://www.aclweb.org/anthology/S14-2015.

[LLS07] John Langford, Lihong Li, and Alex Strehl. *Vowpal wabbit online learning project*. 2007. (Visited on 06/19/2017).

[LM14] Quoc V. Le and Tomas Mikolov. "Distributed Representations of Sentences and Documents." In: *CoRR* abs/1405.4053 (2014). URL: http://arxiv.org/abs/1405.4053.

[Mac16] Jakub Machacek. "BUTknot at SemEval-2016 Task 5: Supervised Machine Learning with Term Substitution Approach in Aspect Category Detection." In: *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. San Diego, California: Association for Computational Linguistics, 2016, pp. 301–305. URL: http://www.aclweb.org/anthology/S16-1048.

[Man+14] Christopher D. Manning et al. "The Stanford CoreNLP Natural Language Processing Toolkit." In: *Association for Computational Linguistics (ACL) System Demonstrations*. 2014, pp. 55–60. URL: http://www.aclweb.org/anthology/P/P14/P14-5010.

[Mik+13] Tomas Mikolov et al. "Efficient Estimation of Word Representations in Vector Space." In: *CoRR* abs/1301.3781 (2013). URL: http://arxiv.org/abs/1301.3781.

[PL08] Bo Pang and Lillian Lee. "Opinion Mining and Sentiment Analysis." In: *Found. Trends Inf. Retr.* 2.1-2 (Jan. 2008), pp. 1–135. ISSN: 1554-0669. DOI: 10.1561/1500000011. URL: http://dx.doi.org/10.1561/1500000011.

[Pon+16] Maria Pontiki et al. "SemEval-2016 Task 5: Aspect Based Sentiment Analysis." In: *Proceedings of the 10th International Workshop on Semantic Evaluation*. SemEval '16. San Diego, California: Association for Computational Linguistics, 2016.

[Ron14] Xin Rong. "word2vec Parameter Learning Explained." In: *CoRR* abs/1411.2738 (2014). URL: http://arxiv.org/abs/1411.2738.

[SM15] Aliaksei Severyn and Alessandro Moschitti. "Twitter Sentiment Analysis with Deep Convolutional Neural Networks." In: *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '15. Santiago, Chile: ACM, 2015, pp. 959–962. ISBN: 978-1-4503-3621-5. DOI: 10.1145/2766462.2767830. URL: http://doi.acm.org/10.1145/2766462.2767830.

[Soc+13] Richard Socher et al. "Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank." In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, WA: Association for Computational Linguistics, 2013, pp. 1631–1642.

[SOM10] Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. "Earthquake Shakes Twitter Users: Real-time Event Detection by Social Sensors." In: *Proceedings of the 19th International Conference on World Wide Web*. WWW '10. Raleigh, North Carolina, USA: ACM, 2010, pp. 851–860. ISBN: 978-1-60558-799-8. DOI: 10.1145/1772690.1772777. URL: http://doi.acm.org/10.1145/1772690.1772777.

[Sri+14] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." In: *J. Mach. Learn. Res.* 15.1 (Jan. 2014), pp. 1929–1958. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=2627435.2670313.

[TS16] 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)*. San Diego, California: Association for Computational Linguistics, 2016, pp. 282–288. URL: http://www.aclweb.org/anthology/S16-1045.

[Xen+16] Dionysios Xenos et al. "AUEB-ABSA at SemEval-2016 Task 5: Ensembles of Classifiers and Embeddings for Aspect Based Sentiment Analysis." In: *Proceedings of the 10th International Workshop*

on Semantic Evaluation (SemEval-2016). San Diego, California: Association for Computational Linguistics, 2016, pp. 312–317. URL: http://www.aclweb.org/anthology/S16-1050.