ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Sentiment Analysis using Deep Convolutional Neural Networks with Distant Supervision

Master Thesis

Jan Deriu

April 22, 2016

Advisors: Prof. Dr. T. Hofmann, Dr. M. Jaggi, Dr. A. Lucchi

Department of Computer Science, ETH Zürich

Abstract

This thesis addresses the problem of predicting message-level sentiments of English micro-blog messages from Twitter. Convolutional neural networks (CNN) have shown great promise in the task of sentiment classification. Here we expand the CNN proposed by [31, 32] and perform an in-depth analysis to deepen the understanding of these systems.
In a first step we compare the performance of different architectures with focus on deeper architectures. We show how different choices of hyper-parameters impact the results and present the benefits and challenges of using deeper architectures.
Second, we introduce the procedure for achieving a high quality initialization of the CNN and show the massive impact it has on its predictive capabilities. This procedure consists of creating custom word embeddings followed by pre-training the network using 90M tweets where the polarity of a tweet is inferred by noisy labels.
Third, we present an in-depth analysis of the supervised phase. We show the impact of the pre-training on the supervised phase. We show how the choice of the optimization algorithm and the step size impact the variance of the score during the supervised phase.
The analysis is concluded by an inspection of the errors made by the system. This gives us insights into the inner workings of the CNN and hints as to what is needed for further improvement.
The CNN presented in this thesis is part of an ensemble which ranked 1st in the SemEval-2016 competition, Task 4a [22]. The task consists of classifying a given tweet as either positive, negative or neutral. This further demonstrates the potential these systems have in the field of sentiment analysis.

Contents

Chapter 1

Introduction

Automatic sentiment analysis aims to give a machine the ability to understand text in its polarity. It is a fundamental problem in natural language processing (NLP). This is an extremely challenging task due to the complexity of human language, which makes use of rhetorical devices such as sarcasm and irony. This thesis focuses on creating a system for automatically predicting the sentiments of Twitter messages called ”tweets”. Contextualization, informal language, misspellings, bad grammatical structure, the use of emoticons and abbreviations as well as Twitters’ limitation to 140 characters are additional complicating factors. We focus on classifying a tweet as *positive*, *negative* or *neutral* making it a *single label multiple class* problem. This increases the complexity compared to the *binary classification* task where the tweets are classified as either *positive* or *negative*. The *neutral* class is the main source of errors as it is difficult, even for humans, to differentiate neutral from non-neutral tweets. The many intricacies of human interactions and human culture pose further challenges.

Consider the following example,

> *President Barack Obama will welcome Nigeria’s new president to the White House Monday in an effort to bolster the fight against Boko Haram*

This tweet contains many interactions which make it difficult to classify. Knowledge about the entities is required. In order to infer the tweets’ sentiment, even for a human, it is important to know who or what Obama, Nigeria and Boko Haram are. Furthermore, there are the various interactions among the entities: Obama welcomes the Nigerian president to fight Boko Haram. Thus, there is a notion of collaboration between Obama and the Nigerian president and a notion of adversity between Boko Haram and the two presidents. Finally there is the aspect of perspective, for example

a coalition to fight Boko Haram may be seen as both positive or negative. From one perspective the collaboration of Obama and Nigeria's president might be positive since they fight against a terror organization. However the tweet might also be seen as neutral as it is simply sharing a news story. It could also be classified as negative since Obama and Nigeria's president are planning military actions that will lead to more casualties.
This small example demonstrates very well the many nuances of human language and human interactions. To understand this tweet a human has to be informed about the current events, know the notion of countries and their presidents, the notion of collaboration and adversity as well as to have the ability to imagine different perspectives. To then further infer the sentiment the tweet conveys it is important to understand the subtlety of human language. The phrases 'welcome' and 'effort to fight' are indicators of a positive sentiment. This combined with the knowledge that Boko Haram is a ruthless terror organization one may infer that the tweet is positive.

Many different approaches have been researched to tackle this difficult task. The focus of this thesis is to study the use of deeper convolutional neural networks (CNN) as well as distant supervised training for text polarity classification, and experimentally compare a variety of architectures. CNNs have shown great promise in the task of sentiment analysis [31] as they exceed the results of systems based on hand-crafted features [22]. We show that CNNs are capable of automatically learning the patterns of small phrases which in earlier systems had to be manually specified. For example it learns that 'not good' is negative, 'not bad' is positive and 'very good' is more positive than 'good'.
Our approach extends the 1-layer CNN proposed by [31] by introducing a deeper architecture, i.e. a CNN with more convolutional+pooling layers. The use of deeper architectures has shown to increase the predictive performance in the field of vision [37]. We systematically measure the accuracy for deeper architectures by increasing the number of convolutional+pooling layers.

We underline the importance of a good initialization for the neural network. We initialize the CNN with a two step procedure: (i) we create high quality word embeddings based on word2vec [17] and (ii) train the CNN on a large amount of weakly labelled data, i.e. data where the label is inferred from the emoticons in the tweet. The word embeddings are used to represent the input to the CNN. The distant supervised phase is crucial for achieving a good predictive performance. The reason being that the CNN has a large amount of parameters to be trained, in fact there are more than 74M parameters. Thus, a large amount of data is required for training. However, the amount of available supervised data is by far not enough, so we have to rely on weakly labelled data. Furthermore, we also show the extent to which different amounts of weakly labelled data used in the distant supervised phase influence the final prediction accuracy. We then study the interplay of pre-trained word-embeddings with the distant supervised and supervised training phases. We then show the effects the distant supervised phase has on the supervised phase. We conclude by an in-depth analysis of the errors made by the classifier to give more insights into its inner workings.

We evaluate the performance of our approach on the datasets of the SemEval-2016 competition, Task 4a [22], on a test set of 20K Twitter messages. SemEval provides a platform where different systems can be compared to each other in a controlled environment using the same test and training data.

The thesis is structured as follows: In Chapter 2 we give an overview of the various methods used to tackle the task of automatic sentiment analysis. Chapter 3 contains an overview of the data used throughout this thesis. In Chapter 4 we look at a system based on the widely used hand-crafted feature engineering approach. In Chapter 5 we present the CNN which was part of the ensemble that won the SemEval 2016 competition, Task 4a. In Chapter 6 we perform an in-depth analysis of the CNN. We conclude the discussion in Chapter 7.

Chapter 2

Related Work

A large variety of approaches exists in the field of sentiment analysis. In this chapter we look at several examples.

Unsupervised and Rule-Based Methods The earliest methods are based on heuristics and pre-selected words and work primarily on unsupervised data. These methods do not perform as well as data driven approaches which we will introduce later. For example [8] predicts the sentiment of adjectives by (i) creating a graph where the vertices are adjectives and the edges express whether two adjectives express the same or opposite sentiment and (ii) applying a clustering algorithm on the graph to classify the adjectives as either positive or negative. The information if two adjectives have the same or opposite sentiment is extracted by looking at the type of conjunction the adjectives appear in: 'and'-conjunctions hint at identical sentiment and 'but'-conjunctions hint at opposite sentiment.
Another technique, to predict the sentiment of adjectives, is to compute the semantic associaton via pointwise mutual information (PMI) of words to pre-selected positive and negative seed-words [39]. Let x and y be words or phrases then the PMI is defined as $\text{PMI}(x,y) = \log(\frac{p(x,y)}{p(x)p(y)})$. The PMI is used to measure the association of two words where $p(x,y)$ can be approximated by counting how often the words x and y co-occur in a corpus, and $p(x)$ is approximated by counting the occurrences of x.
The PMI technique is also used to predict the sentiment of movie reviews [38]. They first extract phrases consisting of two words (e.g. 'unpredictable plot') from the sentences using a part-of-speech (POS) tagger and then select those phrases whose tags match certain patterns. They compute the PMI of these phrases to the words 'excellent' and 'poor' to infer the sentiment. The predicted sentiment of these short phrases is used to infer the overall sentiment of the sentence from which the phrases were extracted.

Supervised Methods Supervised machine learning techniques learn patterns from a labelled dataset [25]. A large set of features is automatically generated from the sentences where these features have to be specified manually. Then different types of classifiers (such as SVM, Naive Bayes etc.) are trained on these features. Most work in this area is focused on feature engineering. For example [24] experimented with the use of n-grams, which are contiguous sequences of n words, and substrings. In Chapter 4 of this thesis a model based on the feature engineering approach will be discussed in detail.

Sentiment Lexicons Many of the machine learining models incorporate information gathered from lexicons into their features [20, 9]. These lexicons are mappings from a word to a score indicating the sentiment expressed by the word. Most lexicons are only word-based but they can also be extended to include bigrams as we will see later. Lexicons can be crafted manually [20, 41, 14] or they can be generated automatically in an unsupervised fashion using noisy labels, i.e. labels automatically inferred from either hashtags or emoticons. A classifier is trained on unigram and bigram features. The result is a mapping that assigns a sentiment score to each unigram or bigram expressing the affinity to each sentiment class [6, 19, 9, 30]. In Chapter 4 we will present these lexicons in more detail as part of the already mentioned system.

Structured Models The models mentioned above focus on one single level of granularity. They detect the sentiment either on the document, sentence, phrase or word level. The goal of structured methods is to capture the more intricate aspects of an opinion by differentiating between several levels of granularity. For example the author might criticise one aspect of the product but at the same time appreciate another aspect of the same product. Thus, the overall sentiment of the review might be positive (document level) but single aspects of the review might be negative (sentence or phrase level). These interactions of sentences can be modelled using a graphical representation.
For example in [16] the document is modelled as a undirected graph where the label of each sentence is dependent on the label of its neighbouring sentences and the label of the document. The label of the document depends on the labels of the sentences . The output of the classifier is thus a joint label $y = (y^d, y^s) = (y^d, y_1^s, ..., y_n^s)$ where y^d represents the label of the overall document and y_i^s is the label of the i^{th} sentence. The features for sentence s_i are represented by the feature mapping $f(y^d, y_{i-1}^s, y_{i+1}^s, s_i)$. It contains the labels of the document and the neighbouring sentences as well as unigram, bigram and trigrams which occur in sentence s_i. The labels are inferred using standard sequence classification techniques.

Another example of a graphical model is given by [42] where they represent the sentiment as a directed graph. The graph represents relations between opinion expressions (e.g. 'good', 'sharp', 'bad'..) and their modifiers (e.g. 'Focus accuracy') as well as relations among opinion expressions. This approach works on the sentence and phrase level. The candidate vertices are found using standard sequential labelling methods with simple features on the character level (e.g. character, marker if it's a letter, a digit or punctuation, marker if the word is in a sentiment lexicon, marker if it's the start or the end of a word). The features are extracted on the level of edges $f(e_{ij})$ where e_{ij} is the edge between the vertices x_i and x_j and f is the feature map. The inference solves following maximization problem: $y = argmax_y \sum_{e_{ij}} \alpha^T f(e_{ij})$ where α are the parameters to be learned.

Deep Learning Neural networks have shown great promise in several NLP applications. Examples are semantic analysis [33], machine translation [5] and sentiment analysis [35]. In particular, shallow CNNs have recently improved the state-of-the-art in text polarity classification by a significant increase in accuracy [12, 11, 31, 32, 10, 29]. The main idea of these methods is to use supervised training on a network with convolutional filters, acting as a sliding window over the sequence of given words, followed by max-pooling. Typically, common word embedding techniques [18, 26] are used to represent each input word. Such neural network architectures share a lot of similarity with CNN methods used in computer vision, which have achieved excellent results in several tasks, e.g. object detection [37] and image segmentation [15]. CNNs for image analysis typically consist out of many more layers than those in currently successful NLP applications [37].

Deeper networks increase the representation power of the system, but also significantly increase the number of parameters and hence require larger amounts of (labelled) training data. This is a major impediment that partly explains why the use of deep architectures in NLP applications has been very limited. An exception being the 6-layer CNN was proposed in [45] for character-level text classification.

The use of distant-supervised or unsupervised learning has been an active research direction in machine learning and in particular NLP applications. Unsupervised training has been shown empirically to be beneficial for supervised machine learning tasks [3]. Distant supervised pre-training can be seen as an improved approach compared to unsupervised pre-training of neutral networks. Here the main idea is to infer weak labels from data without manual labeling. This approach has for example been used for text polarity classification where significantly larger training sets were generated from texts containing emoticons [6, 31]. These methods have shown that training a CNN on such larger datasets, followed by additional classical supervised training on a smaller set of manually annotated labels, yields improved performance. Nevertheless, there is no systematic method known for enabling transfer learning in this way, and so most applications of distant pre-training currently follow empirical rules.

Word2Vec CNNs for text analysis depend on high quality word embeddings to represent the input given to the neural network. Word2Vec [17] is based on a neural network which creates dense vector representations for words. These vectors have the property that, both syntactically and semantically, similar words are close to each other. Word2Vec implements two models: The continuous bag-of-words model (CBOW) and the Skip-gram model. CBOW predicts the current word given a context whereas the Skip-gram model predicts the context given a word. For both models the width of the context-window has to be specified: Typical widths are 5 to 10 words on each side of the center word. The user chooses the model and the width of the context-window and then the neural network is trained using gradient descent and backpropagation.

SemEval Workshop SemEval [22, 28] is a public competition which organises tasks where various systems are compared to each other in a controlled setting using the same training and test sets. The winner of SemEval 2015 Task 10 subtask B [7] created an ensemble of various methods which performed well in previous iterations of SemEval. The common theme of these methods is to train a classifier on a set of manually crafted features. The ensemble was created by means of a meta classifier which used as features the confidence scores of the various classifiers.
In the 2016 edition of SemEval the trend changed. The focus lies more on deep neural nets, out of the 10 best ranking teams in the 2016 edition 5 used a deep neural network [22]. The CNN proposed in this thesis was part of the ensemble which won this year's SemEval-2016 competition, Task 4 [22].

Chapter 3

Data

We used two types of data: supervised and unsupervised data.

The supervised data is provided by the organizers of the SemEval-2016[1] competition. The data is split into several datasets each containing tweets with their corresponding annotation [22]. An overview of the supervised data is given in Table 3.1. Combining Train 2016, Dev 2016, DevTest 2016, Train 2013 and Dev 2013 we have a total of 18044 labelled tweets for training. The *Test 2016* set contains about 20K tweets which is around 10$\times$ the size of the other test sets. Our discussion will focus mostly on the *Test 2016* set.

The unsupervised dataset is a set composed of 2.7B tweets. We filtered all the 218M tweets that contain emoticons. As we will see in chapters 5 and 6 these tweets are used for the creation of the word embeddings and for the distant supervised phase.

[1]http://alt.qcri.org/semeval2016/task4/index.php?id=data-and-tools

Table 3.1: Datasets and number of tweets (or sentences) provided by SemEval-2016. The data was divided into training (Train), development (Dev) and testing sets (Test).

Dataset	Total	Posit.	Negat.	Neutr.
Train 2016	5355	2749	762	1844
Dev 2016	1269	568	214	487
DevTest 2016	1779	883	276	620
Train 2013	8224	3058	1210	3956
Dev 2013	1417	494	286	637
Test 2016	20632	7059	10342	3231
Test 2015	2390	1038	365	987
Test 2014	1853	982	202	669
Test 2013	3813	1572	601	1640
Test SMS2013	2093	492	394	1207
Test Tw2014Sarcasm	86	33	40	13
Test LiveJournal2014	1142	427	304	411

Chapter 4

Feature Based Method

In this chapter we recreate the supoort vector machine (SVM) classifier proposed by [9, 19]. This classifier will be the baseline for the systems we introduce in Chapter 5. The approach of these classifiers is to combine different feature groups which are extracted from Twitter messages.

First, we introduce the model by presenting the feature groups as well as the lexicons. Second, we present the results obtained by this approach. Finally, we introduce the class weighting scheme for structural SVMs.

4.1 Model

4.1.1 Preprocessing

As a first step the tweets are preprocessed. We apply the following standard methods:

- Tokenize: We use ArkTweetNLP [23] which is optimized to handle word segmentation on Twitter.
- Lowercase: We lowercase all the words inside the tweet.
- Normalization: We normalize the URLs and the user-names to a replacement token.
- Negation: Based on [25] we mark words that appear after a negation word (e.g. shouldn't, not,..). The word is marked as negated by appending the '_NEG' token. A list of negation words is provided by Christopher Potts' tutorial[1].

[1]http://sentiment.christopherpotts.net/tokenizing.html

4.1.2 Lexicons

A lexicon is a mapping that assigns a *sentiment score* to a word. These lexicons can be created manually or automatically.

Manually Crafted Lexicons We used three lexicons: (i) NRC Emotion Lexicon [20] which employed Mechanical Turk[2] to create the scores and contains around 14K words, (ii) the MPQA Lexicon [41] containing 8K words and (iii) the Bing Liu Lexicon [14] which contains around 6800 words. These lexicons give each word a single score: -1 for negative words, 1 for positive words and 0 for neutral words.

Automatically Created Lexicons The NRC Hashtag Sentiment Lexicon [19] was automatically generated based on a set of $775,000$ tweets in an unsupervised fashion. They used 78 seed words of which 32 were positive and 36 negative and computed score for a unigram or bigram using pointwise mutual information (PMI). Each unigram or bigram w receives the following score: PMI(w,positive seed-words) $-$ PMI(w,negative seed-words). Thus, the sign of the score indicates the association of word w to the positive or negative class and the magnitude of the score indicates the degree of the association. It contains 54K entries for unigrams and 316K entries for bigrams.

The Sentiment140 Lexicon [19] was created with the same method as above. The differnece is that it is based on the sentiment140 corpus [6] which contains 1.6M tweets. This lexicon contains 62K unigram entries and 677K bigram entries.

The automatically generated lexicons by [9] are generated from a subset of the tweets in the sentiment140 corpus [6]. The polarity of the emoticons in the tweets were used to infer the tweets' sentiment. They extracted 200K tweets of which 100K positive and 100K negative. Additionally they added 100K 'neutral' tweets randomly chosen from the thninknook.com dataset.
A second lexicon is generated from a set of 100K annotated movie-reviews from rotten-tomatoes. These lexicons were constructed by using unigrams and POS n-grams features, as described below, and trained an SVM classifier for the 3 classes. Each unigram and part-of-speech (POS) n-gram has 3 scores which indicate its affinity to each sentiment class.

4.1.3 Features

We used the following feature groups:

[2]https://www.mturk.com/mturk/welcome

- Bag of words: For each word in the dataset we note if the word occurred in the tweet or not.
- Word n-grams: We note for each contiguous subsequence of words whether it occurred in the tweet or not. We use $n \in \{2, 3, 4, 5, 6\}$.
- Non-contiguous word n-grams: These word n-grams are n-grams with one token removed. We note its presence in the tweet.
- Character n-grams: We note for each contiguous subsequence of characters if it occurred in the tweet or not. We use $n \in \{3, 4, 5, 6\}$.
- Number of hashtags.
- Number of negated words.
- Emoticons: We count the number of emoticons in the tweet as well as the number of positive and negative emoticons. We adopted the smiley parsing from Christopher Potts' tokenizer.
- Elongated words: We count the number of words that contain characters which are repeated twice or more (e.g. 'Hellooooo').
- Last token type: We note if the last token is an exclamation mark, a question mark or an emoticon.
- POS n-grams: These n-grams are obtained by creating word n-grams as done above in addition replacing all the tokens but d tokens by their corresponding POS-tag. We use $n \in \{3, 4, 5, 6\}$ and $d \in \{2, 3\}$.
- GloVe: We use the GloVe word embeddings [26] which are created by means of matrix factorization. We compute the average of the word embeddings corresponding to the words inside the tweet.
- Lexicons: For each lexicon introduced above we compute:
  - The number of positive/negative tokens
  - The sum of all the positive/negative tokens
  - The score of the last positive/negative token
  - The maximum score among all positive/negative tokens

  In case the word appears negated (i.e. has '_NEG' postfix) we multiply the score by -1.

4.2 Results

We conducted the experiments with an l_1-regularized squared loss SVM classifier. The F1-score (averaged macro F1 of positive and negative classes) was

used to quantitatively evaluate the results.
First we show the impact of each feature group. This is done by a leave-one-out method where we leave out one feature group at a time and compare the results to the baseline. For the baseline the SVM is trained using all the features presented above. Second we show the impact of different choices of class weights.
For all experiments we use the LibLinear package [4] for training the classifier. As in [9] we used $C = 0.055$ for regularization and $\varepsilon = 0.003$ as the optimization stopping criterion.
For the class weighting scheme we use the inverse class frequencies scaled so that the sum of the class weights is equal to 1.0, i.e. $C_{neg} + C_{neut} + C_{pos} = 1.0$ where $C_l = \frac{s}{F_l}$ is the class weight for class l, F_l is the frequency of class l and $s = \frac{1}{\sum_{l=0}^{L} \frac{1}{F_l}}$ is the scaling factor. The scaling factor s is needed to avoid numerical instability which occurs when the class frequencies are too high and thus the class weights are too close to 0. According to this formula we get following class weights.

- $C_{negative}$: 0.581
- $C_{neutral}$: 0.211
- $C_{positive}$: 0.206

4.2.1 Feature Comparison

We apply the leave-one-out technique to compare the impact of the various feature groups. We train the SVM classifier by leaving out one feature group and compare the result to the baseline. As stated above the baseline is the score achieved by the SVM classifier trained on all the features. To reduce the number of tests we group together following features in a group we call *Counting Features*: Number of hashtags, the emoticons counts and the number of elongated words.
Table 4.1 shows the overall results. The lower the score is compared to the baseline the higher is the discriminative impact of the left-out feature group. We also include the scores of *Webis* [7] which is the team that won SemEval 2015, Task 10b. In the 2016 edition of SemEval they doubled the training set. Thus, the *Webis* system was trained only on half the dataset.
The results show that the *GloVe*-and *Lexicon* features have the most impact on the performance. By removing the *GloVe* features the score drops by about 2 points in both the *Test 2016* and *Test 2015* sets. By removing the *Lexicon* features the score drops by 2 points on the *Test 2016* set and by almost 4 points on the *Test 2015* set. The results don't show similar drops in scores when other feature groups are removed. However the results show that this approach was not able to beat *Webis* on the *Test 2015* set. In fact the 'Baseline' score is 1.16 points below *Webis* score.

	Baseline	Character n-gram	Counting features	Glove	Word n-gram	Negation	Bag Of Words	Lexica	Last Token Type	Non Cont n-gram	POS n-gram	Swiss-Chocolate Lexicons	Webis
Test 2016	60.30	59.93	60.53	58.01	60.34	60.60	60.00	58.21	60.39	59.99	60.26	60.29	-
Test 2015	63.68	63.21	62.56	61.68	63.34	62.24	63.81	59.46	63.15	63.11	63.18	63.32	64.84
Test 2014	66.57	67.50	67.02	66.00	66.74	68.72	68.13	66.41	66.12	65.75	66.36	66.38	70.86
Test 2013	68.84	68.40	68.15	67.79	68.86	68.72	68.31	66.30	68.75	69.26	68.80	68.43	68.49
Test LiveJournal2014	71.61	70.46	71.46	70.79	70.89	71.35	70.89	65.62	70.68	71.62	71.12	70.67	71.64
Test Tw2014Sarcasm	50.41	50.64	52.98	48.96	53.69	50.00	50.45	54.73	52.59	53.74	51.89	52.62	49.33

Table 4.1: Overall results of the feature inspection. Underlined: the lowest score in the row.

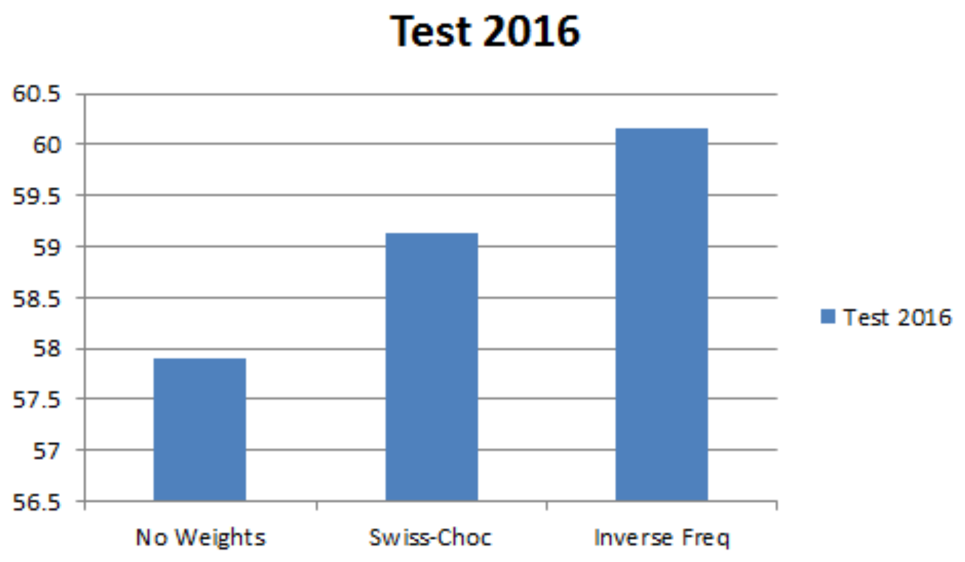


Figure 4.1: The scores of the SVM classifier with different class weighting schemes.

4.2.2 Class Weights

We tested the following three weighting schemes:

- No class weighting: all class weights are equal to 1.0
- Swiss-Chocolate: Based on [9] the class weight is computed as $\frac{2\times\text{average class frequency}}{\text{class frequency}}$.
- Inverse class frequencies scaled so that the sum of the class weights equals 1.0.

Figure 4.1 shows the results on the *Test 2016* set. We see that without class weights the score is about 2 points lower than the inverse frequency scheme. The Swiss-Chocolate [9] scheme scores higher than no class weighting but it performs worse than the inverse frequency scheme.

4.3 Class Weights

As we saw in the previous section the SVM is sensitive to how balanced the classes in the training set are. If a class is under-represented it will perform worse. This means that there will be a low recall for this class. To overcome this limitation we introduce a class weighting scheme which adapts the weights of a class. In the following we show how the structural SVM (SSVM) formulation changes when the class weighting is adopted. This work was integrated into the *Dissolve-struct* [3] framework which is a distributed solver for structured predictions.

SSVM standard setup [13] The goal is given a data-point x to predict the structured object y. The feature map $\Phi(x, y)$ maps the input/output pair into the feature space $\mathbb{R}^d$. Let $D = \{(x_i, y_i)\}_{i=1}^n$ be the labelled training set then the parameters w are estimated by solving:

$$\begin{aligned}
&min_{w,\xi} \quad \frac{\lambda}{2}||w||^2 + \frac{1}{n}\sum_{i=1}^{n}\xi_i \\
&s.t \quad \langle w, \psi_i(y)\rangle \geq L(y_i, y) - \xi_i \quad \forall i, \forall y \in Y \\
&\xi_i \geq 0
\end{aligned} \tag{4.1}$$

where λ is the regularization parameter, ξ_i is the slack variable, $L(y_i, y)$ is the loss for predicting y instead of y_i and $\psi_i(y) = \Phi(x_i, y_i) - \Phi(x_i, y)$.

Class Weighted SSVM We introduce the class weighting scheme. It applies class specific weights C_k to the slack variables. Thus the primal objective becomes:

$$\begin{aligned}
&min_{w,\xi} \quad \frac{\lambda}{2}||w||^2 + \frac{1}{n}\sum_{k=1}^{l} C_k \sum_{i, y_i = k} \xi_i \\
&s.t \quad \langle w, \psi_i(y)\rangle \geq L(y_i, y) - \xi_i \quad \forall i, \forall y \in Y
\end{aligned} \tag{4.2}$$

where l is the number of class labels and C_k denotes the weight associated with class k.

Dual Formulation After the application of standard Lagrangian methods we derive the following dual (see full derivation in Appendix A):

[3] http://dalab.github.io/dissolve-struct/

$$\begin{aligned} & min_{\alpha} \quad \frac{-\lambda}{2} \left\| \frac{1}{\lambda n} \sum_{i=1}^{n} \sum_{y \in Y_i} \alpha_i(y) \psi_i(y) \right\|^2 + \frac{1}{n} \sum_{i=1}^{n} \sum_{y \in Y_i} \alpha_i(y) L(y_i, y) \\ & s.t \quad \sum_{y \in Y_i} \alpha_i(y) = C_{y_i} \quad \forall i \in [n] \end{aligned} \tag{4.3}$$

Setting $C_{y_i} = 1 \quad \forall i \in [n]$ we get the same primal and dual as in the standard formulation.

Chapter 5

Deep Learning Method

Our model extends the CNN proposed by [31, 32] to construct sentence embeddings. We examine three different architectures by varying the number of convolutional layers from 1 to 3. As an example, the 2-layer[1] network variant is illustrated in Figure 5.1. In this chapter, we first describe the details of the 1-layer architecture and its parameters, and then present the deeper architectures considered in this work.

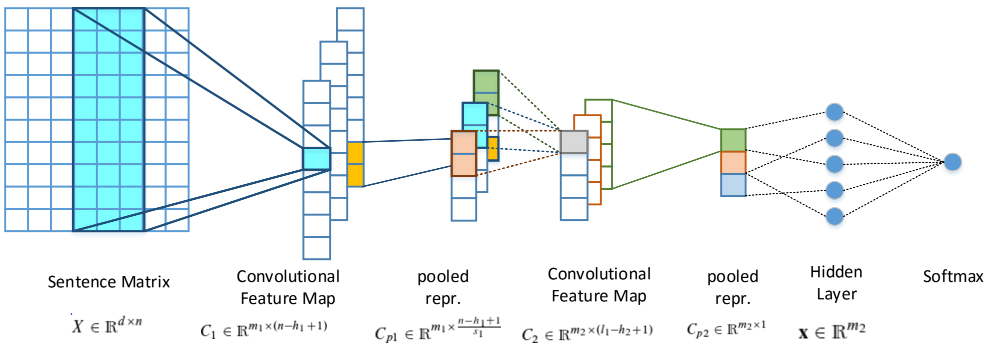

Sentence Matrix
$X \in \mathbb{R}^{d\times n}$
Convolutional Feature Map
$C_1 \in \mathbb{R}^{m_1\times(n-h_1+1)}$
pooled repr.
$C_{p1} \in \mathbb{R}^{m_1\times\frac{n-h_1+1}{s_1}}$
Convolutional Feature Map
$C_2 \in \mathbb{R}^{m_2\times(l_1-h_2+1)}$
pooled repr.
$C_{p2} \in \mathbb{R}^{m_2\times 1}$
Hidden Layer
$\mathbf{x} \in \mathbb{R}^{m_2}$
Softmax

Figure 5.1: The architecture of the CNN used in our approach, for the case of two convolutional+pooling layers.

5.1 Convolutional Neural Networks

Sentence Model Each word is represented by its d-dimensional word embedding. A sentence (or tweet) is represented by a concatenation of the representations of its n constituent words. This yields a matrix $\mathbf{X} \in \mathbb{R}^{d\times n}$, which is used as input to the convolutional neural network.

[1]We here refer to the number of convolutional layers instead of the total number of layers that also includes pooling and hidden layers.

Convolutional Layer In this layer, a set of m filters is applied to a sliding window of length h over each sentence. Let $\mathbf{X}_{[i:i+h]}$ denote the concatenation of word vectors $\mathbf{x}_i$ to $\mathbf{x}_{i+h}$. A feature c_i is generated for a given filter $\mathbf{F} \in \mathbb{R}^{d \times h}$ by:

$$c_i := \sum_{k,j} (\mathbf{X}_{[i:i+h]})_{k,j} \cdot \mathbf{F}_{k,j} \tag{5.1}$$

The concatenation of all vectors in a sentence defines a feature vector $\mathbf{c} \in \mathbb{R}^{n-h+1}$. The vectors $\mathbf{c}$ are then aggregated from all m filters into a feature map matrix $\mathbf{C} \in \mathbb{R}^{m \times (n-h+1)}$. The filters are learned during the training phase of the neural network, as described in Section 5.3.

Max Pooling The output of the convolutional layer is passed through a non-linear activation function $relu(x) = \max(0, x)$, before entering a pooling layer. The latter aggregates vector elements by taking the maximum over a fixed set of non-overlapping intervals. The resulting pooled feature map matrix has the form: $\mathbf{C}_{\mathbf{pooled}} \in \mathbb{R}^{m \times \frac{n-h+1}{s}}$, where s is the length of each interval. In the case of overlapping intervals with a stride value s_t, the pooled feature map matrix has the form $\mathbf{C}_{\mathbf{pooled}} \in \mathbb{R}^{m \times \frac{n-h+1-s}{s_t}}$. Depending on whether the borders are included or not, the result of the fraction is rounded up or down respectively.

Hidden Layer A fully connected hidden layer computes the transformation $\alpha(\mathbf{W}_{\mathbf{hidden}} * \mathbf{x} + \mathbf{b}_{\mathbf{hidden}})$, where $\mathbf{W}_{\mathbf{hidden}} \in \mathbb{R}^{m \times m}$ is the weight matrix, $\mathbf{b}_{\mathbf{hidden}} \in \mathbb{R}^m$ the bias, and α the rectified linear ($relu(x) = \max(0, x)$) activation function [21]. The output $\mathbf{x} \in \mathbb{R}^m$ of this layer is the vector of the sentence embeddings for each tweet.

Softmax Finally, the outputs of the previous layer $\mathbf{x} \in \mathbb{R}^m$ are fully connected to a soft-max regression layer, which returns the class $\hat{y} \in [1, K]$ with largest probability. i.e.

$$\begin{aligned} \hat{y} &:= \arg\max_j P(y = j \mid \mathbf{x}, \mathbf{w}, \mathbf{a}) \\ &= \arg\max_j \frac{e^{\mathbf{x}^\intercal \mathbf{w}_j + a_j}}{\sum_{k=1}^K e^{\mathbf{x}^\intercal \mathbf{w}_k + a_j}}, \end{aligned} \tag{5.2}$$

where $\mathbf{w}_j$ denotes the weights vector of class j, from which the dot product with the input is formed, and a_j the bias of class j.

Network Parameters Training the neural network consists of learning the set of parameters $\theta = \{\mathbf{X}, \mathbf{F}_i, \mathbf{b}_i, \mathbf{W}_{\mathbf{hidden}}, \mathbf{b}_{\mathbf{hidden}}, \mathbf{W}, \mathbf{a}\}$, with $\mathbf{X}$ the word embedding matrix, where each row contains the d-dimensional embedding vector for a specific word; $\mathbf{F}_i, \mathbf{b}_i$ the filter weights and biases of the i^{th} convolutional layer; $\mathbf{W_{hidden}}$ and $\mathbf{b_{hidden}}$ are the weights and biases of the hidden layer; $\mathbf{W}$ the concatenation of the weights $\mathbf{w}_j$ for every output class in the soft-max layer; and $\mathbf{a}$ the biases of the soft-max layer.

5.2 Deep Convolutional Neural Networks

In this Section, we describe the implementation of two deeper variants, namely with 2 and 3 convolutional layers. The 2-layer architecture, depicted in Figure 5.1, has two consecutive convolutional+pooling layers followed by a hidden layer and soft-max operation. The 3-layer architecture has an additional convolutional+pooling layer before the hidden and soft-max ones. For each convolutional+pooling layer, parameters to be defined are: (i) number of filters, (ii) filter width, (iii) length of intervals and (iv) stride in the max-pooling layer. For the last convolutional+pooling layer, only the number of filters and filter width are to be selected, since the max-pooling picks the maximum value of the feature map. We choose the same number of filters for each layer. Table 5.1 lists the details of all the experimented architectures.

	Number of Layers	Number of Filters	Filter window size h	Size of max-pooling intervals and striding
L1A	1	300	$h_1 = 5$	*None*
L1B	1	200	$h_1 = 5$	*None*
L2A	2	200	$h_1 = 4, h_2 = 3$	$w_1 = 4, st_1 = 2$
L2B	2	200	$h_1 = 6, h_2 = 4$	$w_1 = 6, st_1 = 2$
L3A	3	200	$h_1 = 4, h_2 = 3, h_3 = 2$	$w_1 = 4, st_1 = 2, w_2 = 3, st_2 = 1$
L3B	3	200	$h_1 = 3, h_2 = 3, h_3 = 3$	$w_1 = 3, st_1 = 1, w_2 = 3, st_2 = 1$
L3C	3	200	$h_1 = 3, h_2 = 4, h_3 = 5$	$w_1 = 3, st_1 = 2, w_2 = 4, st_2 = 1$
L3D	3	200	$h_1 = 3, h_2 = 5, h_3 = 7$	$w_1 = 3, st_1 = 1, w_2 = 5, st_2 = 3$
L3E	3	200	$h_1 = 7, h_2 = 5, h_3 = 3$	$w_1 = 7, st_1 = 4, w_2 = 5, st_2 = 2$
L3F	3	200	$h_1 = 6, h_2 = 3, h_3 = 3$	$w_1 = 6, st_1 = 3, w_2 = 4, st_2 = 2$
L3G	3	200	$h_1 = 8, h_2 = 6, h_3 = 3$	$w_1 = 3, st_1 = 2, w_2 = 4, st_2 = 1$
L3H	3	300	$h_1 = 4, h_2 = 3, h_3 = 2$	$w_1 = 4, st_1 = 2, w_2 = 3, st_2 = 1$

Table 5.1: Summary of the parameters. All systems used $\alpha = relu$.

5.3 Learning the Model Parameters

The model parameters are learned using the following three-phase procedure: (i) creation of word embeddings; (ii) distant-supervised phase, where the word embeddings and network weights are trained to capture the weak sentiment labels ; and (iii) final supervised phase, where the network is trained on the provided labelled training data.

Preprocessing and Word Embeddings The preprocessing is the same as in Chapter 4. The word embeddings are initialized using word2vec [17]. These are then trained on an unsupervised corpus of 218M tweets presented in Chapter 3. We apply a skip-gram model of window-size 5 and filter words that occur less than 5 times [32]. The dimensionality of the vector representation is set to $d = 52$.

Training During a first distant-supervised phase, we use emoticons as distant labels for the polarity of the tweets [27, 6]. Since the 218M dataset is highly unbalanced having only 45M negative tweets we randomly choose 45M positive tweets out of the 173M positive tweets. The resulting dataset contains 45M tweets for both the positive and negative classes. The neural network is trained on these 90M tweets for one epoch, before finally training for about 15 epochs on the supervised data provided in SemEval-2016. The word-embeddings $\mathbf{X} \in \mathbb{R}^{d \times n}$ are updated during both the distant and the supervised training phases, as back-propagation is applied through the entire network. The resulting effect for the embeddings of words are illustrated Figure 6.6, see the discussion in Chapter 6.3.

Optimization The CNN parameters are trained by minimizing the negative log-likelihood loss-function $\text{NLL}(\theta, D) = -\sum_{i=0}^{|D|} \log P(Y = y^{(i)}|x^{(i)}, \theta)$ where D is the training set. During both supervised training phases, the network parameters are learned using *AdaDelta* [44] which adapts the learning rate for each dimension using only first order information. It takes into account the gradients of previous iterations to decide the step size. This is done by accumulating an exponentially decaying average of the square gradient. *AdaDelta* takes two hyper-parameters: The first hyper-parameter is ε which is used to start off the first iteration and to ensure progress when previous updates become very small. The second hyper-parameter is ρ which is the decay constant.
AdaDelta updates the parameters θ as follows: $\theta_{t+1} = \theta_t + \Delta\theta_t$ by:

$$\begin{aligned}
&\text{Computing the gradient:}\quad g_t \\
&\text{Accumulate Gradient:}\quad E[g_t^2] = \rho E[g_{t-1}^2] + (1-\rho) * g_t^2 \\
&\text{Computing the Update:}\quad \Delta\theta_t = -\frac{\text{RMS}[\theta_{t-1}]}{\text{RMS}[g_t]} * g_t \\
&\text{Accumulating the Updates:}\quad E[\Delta\theta_t^2] = \rho E[\Delta\theta_{t-1}^2] + (1-\rho) * \Delta\theta_t^2
\end{aligned}$$

where $\text{RMS}[g_t] = \sqrt{E[g_t^2] + \varepsilon}$ is the root mean square of the exponentially decaying squared gradients and $\text{RMS}[\Delta\theta_{t-1}] = \sqrt{E[\Delta\theta_{t-1}] + \varepsilon}$ is the root mean square of the exponentially decaying updates. The latter is used to approximate $\Delta\theta_t$ to ensure the units of the update are matched to the units of the current parameters θ.
The default values of these hyper-parameters are: $\varepsilon = 1e^{-6}$ and $\rho = 0.95$

as stated in [44]. In Chapter 6.5.3 we will study the effect of the hyper-parameters on the supervised phase in more detail.

5.4 Computing Resources

The core routines of our system are written in `Python`, making heavy use of mathematical routines in `Theano` [1] that exploits GPU acceleration. For further performance improvement, we used the `CuDNN` library [2]. Our setup requires approximately 10 hours for the distant-supervised phase and only 20-30 minutes for the supervised phase.

Experiments were conducted on ’g2.2xlarge’ instances of *Amazon Web Services* (AWS), which offer a *GRID K520* GPU with 3072 CUDA cores and 8 GB of GDDR5 RAM.

Chapter 6

Experiments

We compared the different network architectures listed in Table 5.1. The F1-score (averaged macro F1 of positive and negative classes) was used to quantitatively evaluate the results, which are reported for all the datasets provided in the SemEval-2016 competition[1]. Furthermore, we provide an in-depth analysis based on the results achieved on the *Test 2016* set. If not stated otherwise the size of the mini batches are set to 1000 tweets per mini batch. We focus our discussion on the following critical aspects:

- The impact of the amount of weakly labelled data during the distant-supervised phase
- The impact of the unsupervised initialization of the word-embeddings
- The performance differences among architecture choices (number of layers, filter-widths and sizes of max-pooling intervals)
- The effect of the step size during the supervised phase
- The types of errors the classifier makes

6.1 Comparing Network Architectures

Results of all SemEval test sets of the different proposed approaches are summarized in Table 6.1. Furthermore, Figure 6.1 compares the F1-scores of the 1-, 2- and 3-layer CNNs as described in Table 5.1. On average, most of the 2 and 3-layer architectures perform better than the 1-layer architecture. On *Test 2016*, the lowest score is obtained by L3D, which uses a large window size in the last convolutional layer. Increasing the number of filters lowered the accuracy, see L1A vs. L1B and L3H vs. L3A in Figure 6.1. Overall among individual networks, the best results on the Twitter sets were achieved by

[1] http://alt.qcri.org/semeval2016/

	UNITN-2015	L1A	L1B	L2A	L2B	L3A	L3B	L3C	L3D	L3E	L3F	L3G	L3H	SemEval-2016
Test 2016	NA	62.93	63.07	63.58	62.95	63.61	63.13	63.69	62.80	63.58	63.11	63.11	63.51	63.30
Test 2015	64.59	66.08	65.5	66.79	65.50	67.45	66.46	67.46	66.73	65.69	65.77	65.19	66.58	67.10
Test 2014	73.6	74.61	74.13	74.67	73.89	73.55	75.25	74.93	74.72	74.37	75.50	73.49	74.08	74.40
Test 2013	72.79	71.66	71.18	71.46	71.39	72.14	71.63	72.49	72.02	71.78	71.32	71.35	71.72	70.60
Test LiveJournal2014	72.48	73.86	72.83	74.75	73.68	73.29	74.00	73.69	73.72	71.25	72.17	72.71	73.39	74.10
Test Tw2014Sarcasm	55.44	62.33	60.74	61.87	65.95	60.75	62.90	63.31	61.89	62.28	63.20	62.75	65.00	56.60

Table 6.1: Overall results of all network architectures on all SemEval-2016 test sets. The highest scores are highlighted in bold face. We compare our results to the best reported scores achieved on each dataset by all teams who participated in the SemEval-2016 competition [22] (last column). On the other end of the spectrum, we also compare to UNITN [32], the winner of the SemEval-2015 competition, as this system relies on the same 1-Layer convolutional architecture as our baseline systems L1A and L1B.

L3C. In Table 6.1, we also compared our results to the best scores achieved by all teams who participated to the SemEval-2016 competition, for each test dataset. For all sets, the accuracy of the best performing CNN exceeds the ones of these teams by 1-7%.

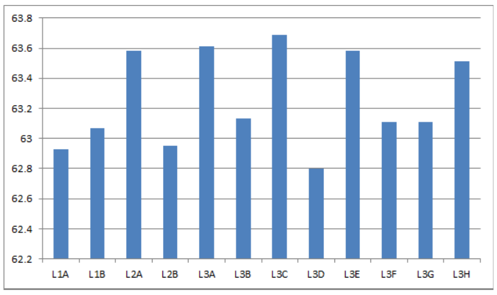


Figure 6.1: F1-scores of the different CNN architectures on the Test 2016 dataset using distant-supervised pre-training on 90M tweets.

6.2 Leveraging Large Amounts of Distant Training Data

The amount of data used in the distant-supervised phase has a substantial impact on the F1-scores on both the *Test 2016* and *Test 2015* data sets. Figure 6.3 shows the performance of three representative architectures, i.e. L1A, L2A and L3A, as a function of the number of tweets used during the distant-supervised phase. The test F1-score increases together with the amount of data. Both L2A and L3A achieve a higher score than L1A, if trained on the full 90M dataset. A large amount of data for distant-supervision is particularly relevant for the 3-layer architecture, which performs poorly when trained with less than 50M tweets, but exceeds the accuracy of all other architectures when the distant dataset is large enough. For the *Test 2016*

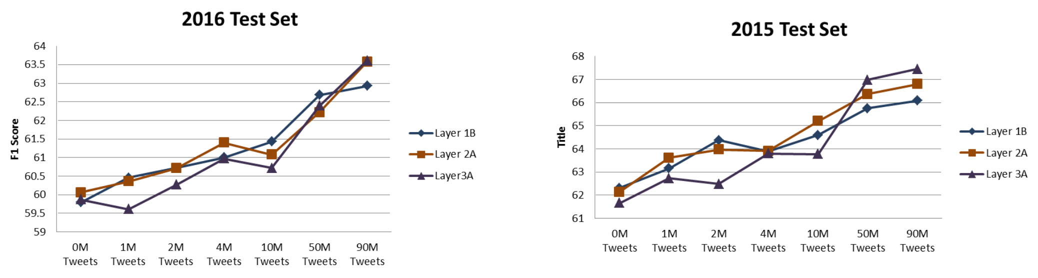


Figure 6.3: Results obtained by training the CNNs on different amounts of data during the distant-supervised phase. Each CNN was trained for one epoch.

results the graph of L1A seems to be saturated meaning that L1A is likely to not benefit from being trained with more data. In case L2A and L3A the scores are still rising when more data is used. However on the *Test 2015* set the graph hints that the scores for all three architectures are at a saturation point. Thus further experiments with more data are needed.

6.3 Word Embeddings

In order to investigate the importance of initializing the word embeddings with high-quality embeddings, we compared the performance of three CNNs (L1A, L2B and L3A) in three cases: (i) using the custom word2vec modified by distant-supervised training, (ii) not updating the word embeddings during the distant-supervised phase, and (iii) using randomly initialized word embeddings. Results are shown in Figure 6.4. It is clear that in all cases, the quality of the initialization is crucial. In fact, the difference in F1-score between the randomly initialized system and the system initialized with the custom word2vec embeddings is approximately 7 to 9 points in F1. Updating the word vectors during the optimization of the network parameters yields further improvements. When initialized randomly, the deeper network architecture yields higher scores than shallower ones. On the contrary, the scores among the architectures are closer when initialized with word2vec embeddings.

Figure 6.6 shows the effect of the distant-supervised phase on the word embeddings. For visualization purposes, principal component analysis (PCA) was used to reduce the dimensions of the word embeddings to two. We see that the distant-supervised training phase significantly changes the geometry of the word embeddings, creating a space where distance reflects the distance in sentiment of each word. The upper figure shows the initial word embeddings created by word2vec, before the distant-supervised phase. For example, the words ’good’ and ’bad’ are very close, as expected, since

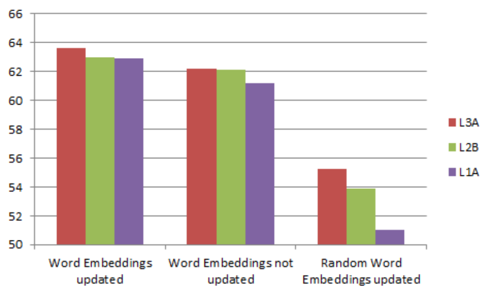


Figure 6.4: Results obtained by initializing the CNNs with different word embeddings.

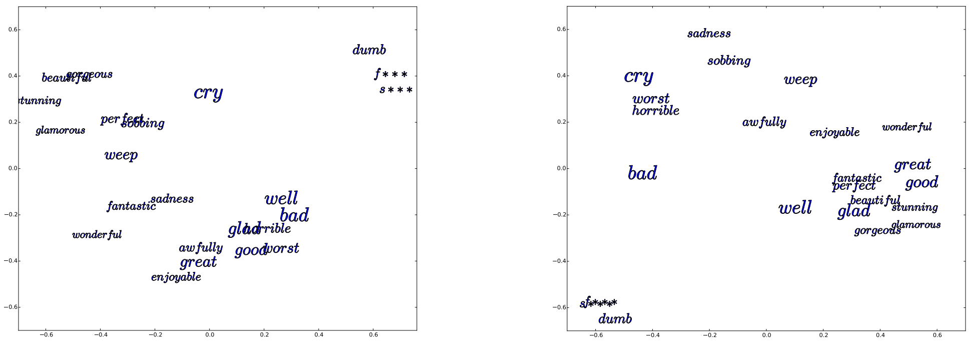


Figure 6.6: Word embeddings projected onto a 2-dimensional space using PCA initially before (left) and after (right) the distant-supervised training phase. We see that the training process significantly changes the word embeddings towards a geometry which better reflects the sentiment of each word.

word2vec puts words close to each other that appear in the same context. The similarity score of the two corresponding vectors of the words 'good' and 'bad' is 0.785. After the distant-supervised phase, the word embeddings is able to incorporate sentiment. As shown in the lower figure, negative and positive words are neatly separated into two clusters. In this case, the similarity score between the word embeddings of 'good' and 'bad' becomes -0.055. Finer grained clusters are also revealed in the second embedding. For example, words that convey sadness are close together.

6.4 Distant and Supervised Phases

We here focus on the supervised phase on the SemEval training and development data consisting of a total of 18K labelled tweets, see Table 3.1. We first study the effect of the distant-supervised phase on the final train & test

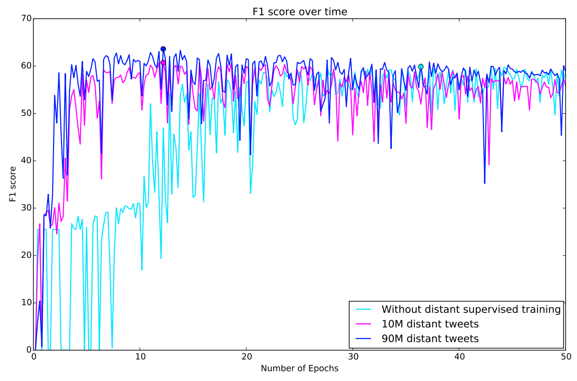
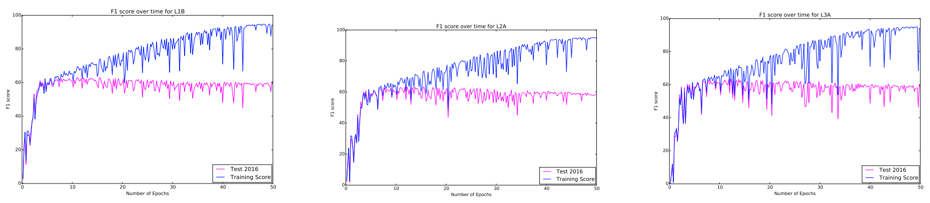


Figure 6.7: Test F1-score of the L3A *architecture vs. number of epochs during the final supervised phase, started after different variants of distant supervised training. Dots indicate the location of the maximum score.*

scores. We trained the L3A CNN for 50 epochs and computed the F1-score at regular intervals (5 times per epoch). We show the results on the *Test 2016* set in Figure 6.7 for different variants of distant pre-training.We show the results without distant pre-training as well as for varying sizes of the training set used in the distant-supervised phase. The point-markers show where the maximum score values were obtained. Note that the time it takes for a method to get to a regime of good predictive performance highly depends on the use of the distant-supervised pre-training. The network trained with the whole 90M distant tweets achieves its best performance after 12 epochs, but it takes three times as many epochs for a network without distant supervision to get to its best (weaker overall) performance. The network without distant supervision stays on a plateau for 10 epochs not exceeding 25 points before jumping to a higher-score region. The pre-trained networks jump to higher scores after the first couple of epochs.

In Figure 6.8 we show the results on the *Test 2016* set and the training set for the pre-trained L1A, L2A and L3A networks. The plots for the other networks can be found in Appendix B. There are two observations to be made. First the network overfits on the training set: After 40 epochs they achieve an F1-score of 95 on the training set. Simultaneously the score on the *Test 2016* set decreases slightly over time. The second observation is the high variance in score over time. We see that there are sudden drops in scores e.g. the score of the L3A network drops by almost 10 points in epoch 49 before rising by the same amount. In the following section we will study these spikes in more detail and show how to smoothen the scores.



(a) L1A (b) L2A (c) L3A

Figure 6.8: Training and test scores for the different 1,2 and 3-layer convolutional architectures, during the final supervised phase.

6.5 On Variance Reduction

In this section we will address the high variance observed during the supervised phase. We investigated the following ways to decrease the variance: using vanilla SGD, reducing the mini batch size to 100 instead of a 1000 and tuning the hyper-parameters of *AdaDelta*. The intuition being based on reducing the step size to avoid large changes in the parameters. For these experiments we used system L3A which was pre-trained on 90M tweets.

6.5.1 Vanilla SGD

Instead of using *AdaDelta* updates we use the vanilla SGD updates: $x_{t+1} = x_t + \eta g_t$. Where x is a set of parameters, g_t is the gradient and η is the step size. We chose $\eta = 0.1$ for the following experiment. Figure 6.9 shows the training and test scores. We see that the variance is greatly reduced. This behaviour is expected as we use a fixed step-size $\eta = 0.1$. The drawback is that the accuracy on the Test 2016 set is 62.86 as opposed to the 63.61 when *AdaDelta* is used.

6.5.2 Mini Batch Size

Figure 6.10 shows the results when training L3A with the default setting using a mini batch size of 100 instead of 1000 tweets. As we see the variance is also greatly reduced. However it only achieves a score of 61.59 on the Test 2016 set as opposed to the 63.61 achieved by using a mini batch size of 1000.

6.5.3 AdaDelta Hyperparameters

As mentioned above *AdaDelta* has two hyper-parameters: ε and ρ. Figure 6.11 shows the training and test scores depending on the vaules of ε. We see clearly that high values of ε create a very high variance whereas choosing

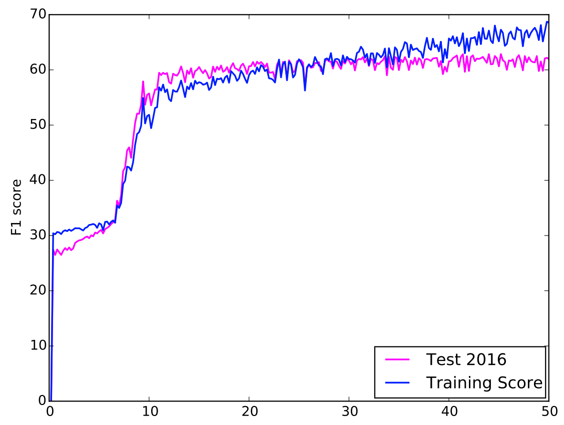


Figure 6.9: Training and test scores for system L3A *trained using vanilla SGD.*

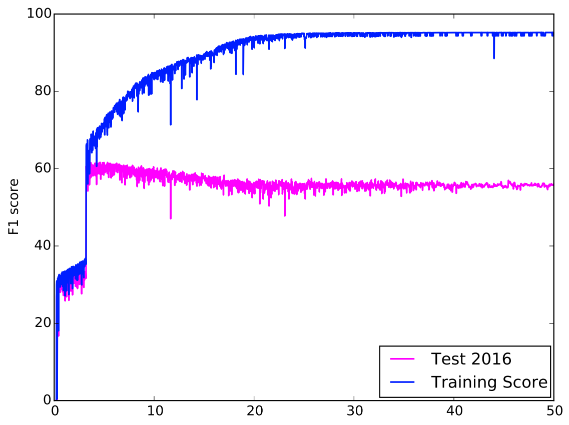
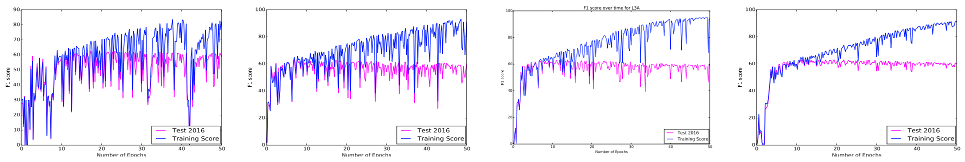


Figure 6.10: Training and test scores for system L3A *trained with a smaller mini batch size.*

smaller values reduce this variance. When choosing $\varepsilon = 1e^{-8}$ the curve is very smooth in contrast to the others. As with the other approaches even though the curve is smoother the score is at 63.14 which is smaller than the 63.61 achieved using the default values. See Appendix C for more plots.

6.6 Error Inspection

The confusion matrix shown in table 6.2 shows that the main source of errors is with the neutral class. In fact only 58.25% of the neutral tweets are predicted correctly compared to the 71.8% of correctly predicted positive tweets and the 70.1% correctly predicted negative tweets.



(a) $\varepsilon = 1e^{-2}$ (b) $\varepsilon = 1e^{-4}$ (c) $\varepsilon = 1e^{-6}$ (d) $\varepsilon = 1e^{-8}$

Figure 6.11: Training and test scores for different values of ε *and* $\rho = 0.95$.

	Negative (T)	Neutral(T)	Positive(T)
Negative (P)	2265	1915	347
Neutral (P)	755	6025	1643
Positive (P)	211	2402	5069

Table 6.2: Confusion matrix. The rows show the predictions (P) and the columns are the true (T) value.

Tweets misclassified as positive There are 2613 tweets falsely classified as positive. Table 6.3 shows some examples of the tweets falsely classified as positive together with the confidence gathered from the softmax distribution. The tweets with high confidence all contain positive words (e.g. 'good', 'love', 'lol', 'happy', 'greatest' and 'excited'). The tweets with low confidence do not contain many words that convey sentiment.

Confidence	True Label	Tweet
0.91	Negative	@religulous @ATHE1STP0WER @chrispyosh I love how so many idiot Christians still think Sunday is the Sabbath! Ah, Constantine...lol
0.92	Negative	Seth Rollins is much less than zero percent happy. He wanted this past Monday to be the greatest night of his career! #Smackdown
0.81	Negative	I wish I was excited right now for Sharknado 3. But I'm not. I'll catch it later on today, or tomorrow on my DVR.
0.96	Neutral	Good Monday Rockers! Today we are going to discuss the best robots in TV/Movie History. Rolling Stone magazine... http://t.co/VNf5ip30dm
0.86	Neutral	@WWE Seth Rollins should put his title on the line every Monday. That what's best for business!
0.36	Negative	@mynamesnotmilan @maarebear "oh Milan just texted me. OHHH aw no she can't come ova Monday!" Literally a direct quote
0.37	Negative	"Kanye West to run for US president" No guys c'mon. POTUS is the most powerful man in the world. Get serious. Tho he'd be bettern tham Trump
0.34	Neutral	Still hungover and what not with no plans on this Saturday evening so this means I should see Paper Towns again, right?
0.35	Neutral	Cool, Ant-Man isn't working. Beauty sleep for me then seeing as it's Monday.

Table 6.3: Examples of tweets misclassified as positive.

Tweets misclassified as negative There are 2262 tweets falsely classified as negative. Table 6.4 shows some examples of tweets misclassified as negative. The most striking observation is the fact that every tweet about the terror organization 'Boko Haram' is classified as negative with very high confidence. The reason being that both words 'boko' and 'haram' are similar to the word 'grief' (0.48 similarity score).
This hints at the fact that the CNN is very sensitive to the sentiment con-

veyed by the words inside the tweet. The tweets misclassified as negative with low confidence contain both slightly positive and negative words (e.g. ’guilty’, ’pleasure’, ’die’, ’funniest’). In these cases the negative words were weighted slightly more than the positive ones.

Confidence	True Label	Tweet
0.90	Positive	Relatives of the 200 Nigerian schoolgirls kidnapped by Boko Haram will hold a march & candle-lit vigil to mark 500 days since the abductions
0.86	Positive	An update on the surprisingly effective fight against Boko Haram in Nigeria. Great reporting.
0.85	Positive	Foo Fighters opening with everlong tomorrow is gonna be a hot shitty mess of emotions
0.95	Neutral	The Sultan of Sokoto Alhaji Sa’ad Abubakar has said Boko Haram terrorist are out to spoil the name of Islam as a whole. May they be exposed
0.90	Neutral	@KaraMuneer @StandupMary @Lrihendry Destroy ISIS, Hezbollah, Boko Haram, Hamas, Al Qaeda, just to name a few, & peace may come to the world
0.37	Positive	Bad Blood by Kendrick Lamar and Taylor Swift may be my guilty pleasure of the year. I wish more mainstream pop was like this.
0.36	Positive	honestly i will die if Briana gives birth on February 1st that would be the funniest thing ever
0.39	Neutral	Great way to end my 21st: watch the Miss USA pageant and cry because fat and ugly
0.36	Neutral	I think it’s nice Shawn is trying. As annoying as it may be

Table 6.4: Examples of tweets misclassified as negative.

Tweets misclassified as neutral There are 2398 tweets that are misclassified as neutral. Table 6.5 shows several examples. The tweets misclassified as neutral do not contain words that convey sentiment. The sentiment in the tweet is conveyed by more subtle expressions and domain-specific knowledge is required to infer the sentiment. E.g. the CNN was not able to learn that ’throwing someone in the sun’ is conveys a negative sentiment. The tweets about David Wright requires knowledge about Baseball and on the current events in Baseball. As we mentioned in the introduction it is difficult to incorporate this knowledge into sentiment classifiers. These examples of tweets that are falsely classified as neutral emphasize this point.

Confidence	True Label	Tweet
0.70	Positive	Go Red Sox. Somehow beat the jays tomorrow.
0.71	Positive	Raise your hand if you thought David Wright would return on August 24 with the #Mets five games up on the #Nats...
0.85	Positive	Massimo is showing me every John Cena memes under the sun
0.74	Negative	David Wright grounds out in the 9th. Was 2-for-5 today.
0.72	Negative	How about we fire John Kasich into the sun instead? https://t.co/W87GgtIQJj
0.38	Positive	Ed Sheeran did a song with @theweeknd CLASS IS CANCELLED TOMORROW!
0.38	Positive	I may be tired, but at least the bags under my eyes are Gucci.
0.37	Negative	Hulk Hogan. More like Bulk Fooligan. Some folks never know when it’s time to leave the room. He’s had his day in the sun. Time to go away!
0.37	Negative	I want to see Ed Sheeran tomorrow but I can’t

Table 6.5: Examples of tweets misclassified as neutral.

6.6.1 On Words and Phrases

The above discussion gives two hypotheses: First that the CNN is sensitive to words conveying sentiment inside the tweet. Second that positive words give a stronger signal than negative words.
To further investigate these hypothesis we tested the impact of certain words and phrases on the results. We randomly generated sentences of length 10 which contain the word or phrase to be tested. For each word or phrase to be tested we generated 250 sentences to reduce the noise generated by the randomly chosen words. We then let our CNN predict the sentiment of these sentences and use the distribution of the softmax as a measure of confidence. Table 6.6 shows the results by showing for each word or phrase the most common label assigned across the 250 sentences, the percentage of sentences which said label was assigned to and the average confidence of the classifier.
In case of ′good′ we see that almost all sentences are classified as positive with high confidence. The word ′better′ scores lower only having 80% of sentences positive with lower confidence which is expected as it is a comparative which is used in non-positive sentences. The word ′best′ scores high in positivity with a high confidence. We do not see the same pattern with ′bad′: in fact bad is classified as neutral in 72.5% of the cases with very low confidence. The words ′worse′ and ′worst′ both are mostly classified as negative with higher confidence. However the confidence of negative words is lower than the confidence of positive words.

To test the CNNs capability to recognize more complex patterns we used small phrases composed of 2 or 3 words. The phrases ′very good′ and ′very best′ both score positive in all cases with 96% confidence which is higher than for ′good′ and ′best′ . The phrase ′very better′ has lower confidence as it is malformed syntax. The phrase ′very bad′ scores negative on 86% of cases however with a confidence of just 55%.
The CNN is capable of recognizing negation. The phrase ′not good′ is classified as negative in 96% of cases as is the phrase ′not happy′. We see the same pattern with the phrase ′not bad′ being classified as positive in 88% of cases.
The CNN thus is capable to recognize small phrases as well as correctly recognizing the sentiment of words. The results also show that the CNN is more sensitive to positive words as it is to negative since the confidence for positive words or phrases is higher than for negative.
The recurring example of the phrase ′boko haram′ is classified as negative in 98.8% of cases with a very high confidence of 79.9%. In fact ′boko haram′ has the highest confidence of all the tested negative phrases. This explains why all tweets in the *Test 2016* set containing ′boko haram′ were classified as negative.

Phrase	Most common label	Percentage	Average Confidence
good	positive	97.6%	0.837
better	positive	80.8%	0.621
best	positive	97.6%	0.808
very good	positive	100%	0.962
very better	positive	98.4%	0.806
very best	positive	100%	0.956
not very good	negative	90.8%	0.635
not good	negative	96%	0.65
not best	negative	91.6%	0.579
happy and good	positive	100%	0.984
happy	positive	99.6%	0.915
not happy	negative	87.6%	0.567
bad	neutral	72.5%	0.477
worse	negative	96.4%	0.672
worst	negative	96.4%	0.732
very bad	negative	86%	0.555
not bad	positive	88.4%	0.522
boko haram	negative	98.8%	0.799

Table 6.6: The impact of words and phrases on the score. Shows for each phrase the label which was most commonly assigned by the classifier, the percentage of sentences which said label was assigned to and the average confidence across the 250 sentences.

6.6.2 Word Pairs

We further investigate the interplay of positive and negative words by testing the score if the tweet contains both positive and negative words. We randomly generate sentences with the same method as stated above. The only difference is that the word pairs to be tested are inserted randomly inside the tweet instead of co-occurring.
Table 6.7 shows the results for each word pair. It shows the percentage of sentences classified as positive, negative or neutral as well as the average confidence. The overall results show that with two words that convey opposite sentiments the sentences are classified as either positive or negative only a few are classified as neutral. The average confidence is not very high. The word pair 'happy,hate' has most of the sentences classified as positive. However, almost a third are classified as negative. In the case of 'love,hate' the results are more balanced, almost half are classified as positive and the other half as negative. This word pair has a lower confidence than the other pairs. The word pair 'happy,sad' has most sentences classified as positive. On the contrary the word pair 'good,worst' has most sentences classified as negative.

The results show that the CNN reacts to the signals of multiple words. It also shows that the signals are not averaged, as only a few tweets are classified as neutral, but it rather chooses the stronger signal. It also shows that the signal is not only dependent on the word itself as for each word pair the sentences are distributed on the positive and negative classes.

Word Pair	Negative	Neutral	Positive	Avg. Confidence
happy, hate	28.8%	2.4%	68.8%	0.55
love, hate	54.4%	3.2%	42.4%	0.424
happy, sad	12.0%	1.2%	86.8%	0.578
good, worst	77.6%	2%	20.4%	0.578

Table 6.7: Results of the word pair test. For each word pair we show the percentage of sentences classified as negative, neutral and positive. We show the average confidence of the classifier in its decisions.

6.7 Conclusion

In this chapter we presented the results of our in-depth analysis. We learned that deeper architectures can achieve higher scores than shallower architectures. However, not all the deeper architectures work well as some 3-layer architectures achieved lower scores than the 1-layer architecture. The results showed the importance of the distant supervised phase as it is able to raise the F1-scores for all architectures by 2-3 points. Especially the 3-layer architecture is dependent on a large dataset during the distant supervised phase since it performs poorly when trained with less than 50M tweets. We showed the immense importance of high quality word embeddings. The difference in score when using randomly initialized word embeddings vs. using word embeddings trained with word2vec is approximately 7 to 9 points in F1.

We showed that with a large dataset used during the distant supervised training the supervised training finds the maximum score three times earlier than without the distant supervised phase. We saw the importance the choice of step size has on the supervised phase. If the step size is to large the score shows a high variance over the epochs.

Finally we presented an in-depth analysis of the errors of our CNN. We learned that it reacts strongly to words that convey sentiment but at the same time learned to recognize small patters like negation.

Chapter 7

Future Work & Conclusion

Conclusion In this thesis we described a CNN system for automatic sentiment prediction and presented an in depth analysis of said system. We investigated deeper network architectures. Furthermore we showed the importance of high quality word-embeddings as well as the impact of the distant-supervised phase. We empirically demonstrated that deeper network architectures need a significant amount of training data in the distant-supervised phase. We discussed the importance of the distant-supervised phase with respect to the supervised phase. We presented an in-depth analysis of the mistakes the classifier makes hinting at how to further improve the system. We showed that our CNN outperforms the system based on hand-crafted features by 3.39 points. This indicates that the trend in sentiment analysis shifts towards the use of deep learning methods. The system we presented in this thesis was part of an ensemble that won the SemEval 2016 competition, Task 4a.

Future Work There are many different directions in which this thesis can be extended. We conclude the discussion by presenting three ideas for future work.

Hashtag Prediction It has been shown that CNNs can be used for a variety of tasks. We performed some experimental work on automatic hashtag prediction. We partially recreated the system proposed by [40] by replacing the softmax-layer in the 1-layer architecture. The new layer is based on a scoring function which computes the dot-product of the sentence-embedding with the hashtag-embeddings and returns the highest scoring hashtag. The gradient optimizes the following loss function: $L = \max\{0, m - f(w, t^+) - f(w, t^-)\}$ where the function $f(w, t)$ returns the dot-product between the sentence embedding of w and the embedding of hashtag t, m is a margin, w is the tweet, t^+ is the embedding of the correct hashtag and t^- is the embed-

ding of a randomly chosen incorrect hashtag . Thus the aim is to embed the sentences close to their respective hashtag-embedding.
For our experiments we focused on the 100 most used hasthags. We trained the CNN on 3M tweets and as a test set used 100K tweets so that each class was represented 1000 times. We randomly initialized the hashtag embeddings. Our preliminary results returned an accuracy of 35%.
For future work the system can be trained on the full dataset containing 481M tweets as well as using more than 100 hashtag-classes. The main challenge being in the scalability of the system regarding the number of classes. Furthermore the CNN can be extended to use more layers as we did with the CNN used for automatic sentiment analysis.

CNN inspection & improvement The discussion in Chapter 6 on the errors of our CNN showed the need to further investigate its inner workings. In the field of computer vision many approaches have been researched to interpret the neural networks [36, 43, 34]. Designing similar approaches for text analysis would be beneficial for helping to understand the source of the errors.
Further experiments on how well the CNN recognizes structures in the sentence are needed as well. The results indicate that the CNN is very sensitive to words that convey sentiment. However, we did not find indications of the CNN finding long-ranging patterns.
Based on [34] we did some preliminary results. The goal was to find out which filters got activated by which tweets or by which words inside a tweet. The approach is to compute the magnitude of the gradient for a specific filter in a specific layer given a tweet. From the results we got two observations: (i) The magnitudes of the gradients in the filters of layer 2 and 3 were much lower than those in layer 1. In fact the maximum magnitude in layer 2 and 3 is at 2.07 in contrast the maximum magnitude in layer 1 is at 19.8. This indicates that the signals in the deeper layers are weaker then the signals in the first layer. This could be a hint as to why the deeper networks did not increase the score by more than 1 point. (ii) The magnitudes of tweets that are misclassified are larger, i.e. a positive tweet that was falsely classified as negative has a larger magnitude than a correctly classified tweet.
In another attempt to analyse the filter we visualized the filters of the first layer. This technique is often applied in the field of vision as it results in visually pleasing images. Figure 7.1 shows all the 200 filters of the first layer. Note that the black areas are for padding purposes. However, there are no visible patterns to be found inside this plot. We tried to match each of the 4 vectors in a filter to the nearest word embedding which also did not lead to any notable findings.

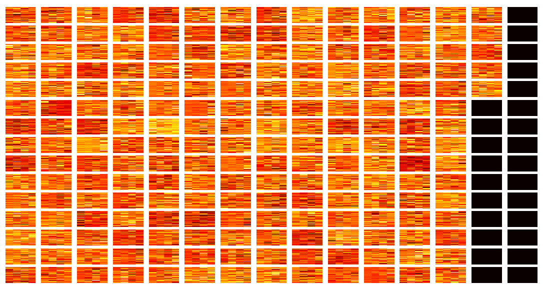


Figure 7.1: Visualization of all the filters inside the first layer.

Sentiment Analysis The system presented in this thesis is specialized to automatically classify a tweet as either positive, negative or neutral. Tweets might also be classified on a 5-point scale with the positive and negative class split into very positive and very negative respectively. Furthermore tweets can be classified on the sentiment they express towards a specific topic. Another task is given a set of tweets of a certain topic, predict the distribution of the tweets across the classes. The task would then be a quantification task instead of classification. As future work our CNN could be adapted to tackle above tasks as well.

Appendix A

SSVM Class Weight Derivation

A.1 Introduction

The SVM is sensitive to how balanced the classes in the training set are. If a class is under-represented it will perform worse. This means that there will be a low recall of this class. To overcome this limitation we introduce a class weighting scheme which adapts the weights of a class. In the following we show how the structured SVM (SSVM) formulation changes when the class weighting is adopted.

A.2 Standard setup for SSVM

The goal is given a data-point x to predict the structured object y. The feature map $\Phi(x, y)$ maps the input/output pair into the feature space $\mathbb{R}^d$. Let $D = \{(x_i, y_i)\}_{i=1}^n$ be the labelled training set then the parameters w are estimated by solving [13]:

$$\begin{aligned} & min_{w,\xi} \quad \frac{\lambda}{2}||w||^2 + \frac{1}{n}\sum_{i=1}^{n}\xi_i \\ & s.t\langle w, \psi_i(y)\rangle \geq L(y_i, y) - \xi_i \quad \forall i, \forall y \in Y \\ & \varepsilon_i \geq 0 \end{aligned} \tag{A.1}$$

A.3 Class Weighted SSVM

We introduce the class weighting scheme that applies class specific weights to the slack variables.
The primal objective thus becomes:

$$\begin{aligned} &min_{w,\xi} \quad \frac{\lambda}{2}||w||^2 + \frac{1}{n}\sum_{k=1}^{l} C_k \sum_{i,y_i=k} \xi_i \\ &s.t \quad \langle w, \psi_i(y)\rangle \geq L(y_i, y) - \xi_i \quad \forall i, \forall y \in Y \end{aligned} \tag{A.2}$$

where l is the number of class labels and C_k denotes the weight associated with class k which can be rewritten as:

$$\begin{aligned} &min_{w,\xi} \quad \frac{\lambda}{2}||w||^2 + \frac{1}{n}\sum_{i=1}^{n} \xi_i C_{y_i} \\ &s.t \quad 0 \geq (\langle w, -\psi_i(y)\rangle + L(y_i, y)) - \xi_i \quad \forall i, \forall y \in Y \end{aligned} \tag{A.3}$$

where C_{y_i} denotes the class weight which is associated to the class-label y_i.

To derive the dual form we set the Lagrangian as follows:

$$\begin{aligned} L(w,\xi,\alpha) = \min_{w,\xi} \quad \frac{\lambda}{2}||w||^2 + \frac{1}{n}\sum_{i=1}^{n} \xi_i * C_{y_i} \\ +\frac{1}{n}\sum_{i=1}^{n}\sum_{y\in Y_i} \alpha_i(y) * [(\langle w, -\psi_i(y)\rangle + L(y_i, y)) - \xi_i] \end{aligned} \tag{A.4}$$

We take the derivative of the Lagrangian w.r.t to w and ξ_i and set them to 0:

$$\frac{\partial L(w,\xi\alpha)}{\partial w} = \lambda w - \frac{1}{n}\sum_{i=1}^{n}\sum_{y\in Y_i} \alpha_i(y)\psi_i(y) \Rightarrow w = \frac{1}{\lambda n}\sum_{i=1}^{n}\sum_{y\in Y_i} \alpha_i(y)\psi_i(y) \tag{A.5}$$

$$\frac{\partial L(w,\xi\alpha)}{\partial \xi_i} = \frac{C_{y_i}}{n} - \frac{1}{n}\sum_{y\in Y_i} \alpha_i(y) \Leftrightarrow \sum_{y\in Y_i} \alpha_i(y) = C_{y_i} \tag{A.6}$$

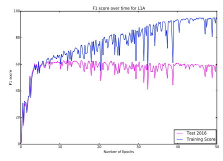
We plug these back into the Lagrangian and get the following dual formulation:

$$\begin{aligned} &min_{\alpha} \quad \frac{-\lambda}{2}\left|\left|\frac{1}{\lambda n}\sum_{i=1}^{n}\sum_{y\in Y_i} \alpha_i(y)\psi_i(y)\right|\right|^2 + \frac{1}{n}\sum_{i=1}^{n}\sum_{y\in Y_i} \alpha_i(y) L(y_i, y) \\ &s.t \quad \sum_{y\in Y_i} \alpha_i(y) = C_{y_i} \quad \forall i \in [n] \end{aligned} \tag{A.7}$$

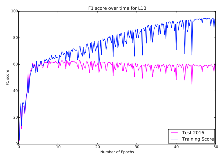
As one may notice the primal and the dual are the same as in the standard formulation if we set $C_{y_i} = 1 \quad \forall i \in [n]$

Appendix B

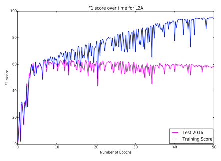
Full Architecture Plots



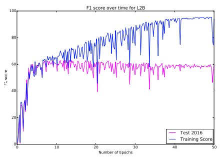
(a) L1A



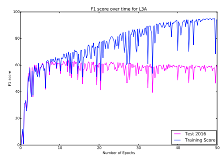
(b) L1B



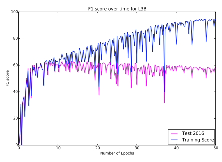
(c) L2A



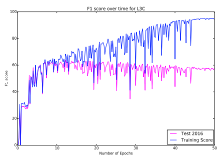
(d) L2B



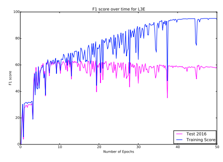
(e) L3A



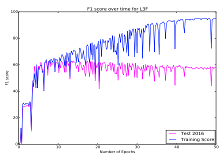
(f) L3B



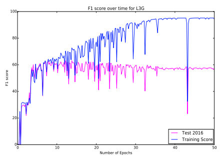
(g) L3C



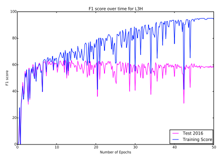
(i) L3E



(j) L3F



(k) L3G

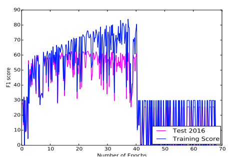


(l) L3H

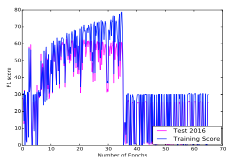
Figure B.1: Training and test scores for the different 1, 2 and 3-layer convolutional architectures, during the final supervised phase.

Appendix C

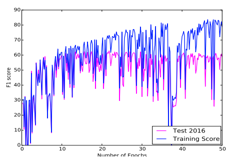
Full Adadelta Plots



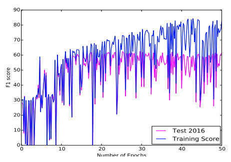
(a) $\epsilon = 1e^{-1}$ $\rho = 0.05$



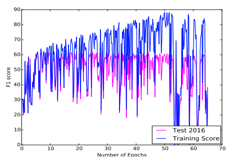
(b) $\epsilon = 1e^{-1}$ $\rho = 0.25$



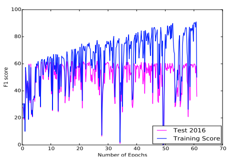
(c) $\epsilon = 1e^{-1}$ $\rho = 0.75$



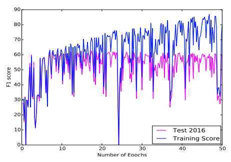
(d) $\epsilon = 1e^{-1}$ $\rho = 0.95$



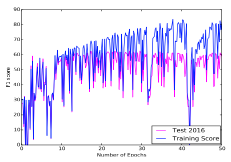
(a) $\epsilon = 1e^{-2}$ $\rho = 0.05$



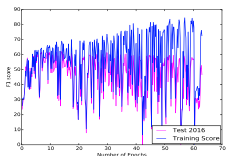
(b) $\epsilon = 1e^{-2}$ $\rho = 0.25$



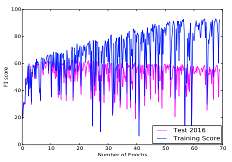
(c) $\epsilon = 1e^{-2}$ $\rho = 0.75$



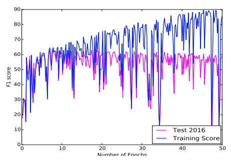
(d) $\epsilon = 1e^{-2}$ $\rho = 0.95$



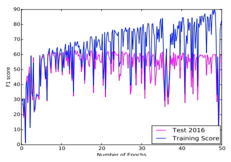
(a) $\epsilon = 1e^{-3}$ $\rho = 0.05$



(b) $\epsilon = 1e^{-3}$ $\rho = 0.25$

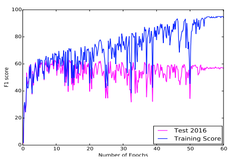


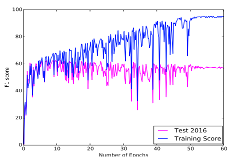
(c) $\epsilon = 1e^{-3}$ $\rho = 0.75$

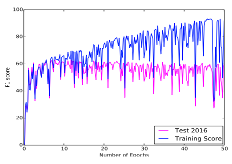


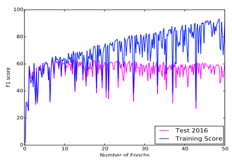
(d) $\epsilon = 1e^{-3}$ $\rho = 0.95$

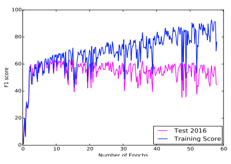
Figure C.1: The training and test scores for $\epsilon = 1e^{-1}, .., 1e^{-3}$ *and* $\rho = 0.05, 0.25, 0.75, 0.95$

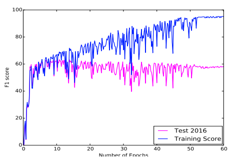

(a) $\varepsilon = 1e^{-4}$ $\rho = 0.05$

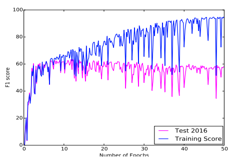

(b) $\varepsilon = 1e^{-4}$ $\rho = 0.25$

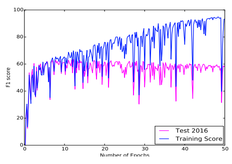

(c) $\varepsilon = 1e^{-4}$ $\rho = 0.75$

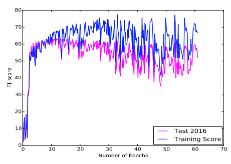

(d) $\varepsilon = 1e^{-4}$ $\rho = 0.95$

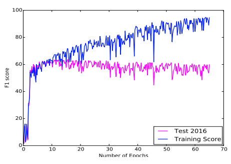

(a) $\varepsilon = 1e^{-5}$ $\rho = 0.05$

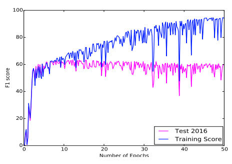

(b) $\varepsilon = 1e^{-5}$ $\rho = 0.25$

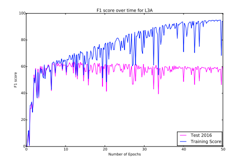

(c) $\varepsilon = 1e^{-5}$ $\rho = 0.75$

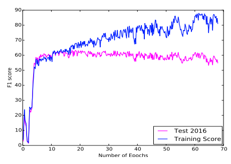

(d) $\varepsilon = 1e^{-5}$ $\rho = 0.95$

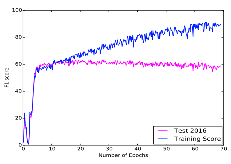

(a) $\varepsilon = 1e^{-6}$ $\rho = 0.05$

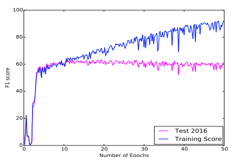

(b) $\varepsilon = 1e^{-6}$ $\rho = 0.25$

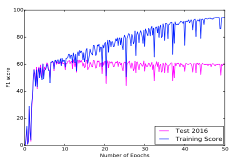

(c) $\varepsilon = 1e^{-6}$ $\rho = 0.75$

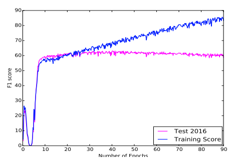

(d) $\varepsilon = 1e^{-6}$ $\rho = 0.95$

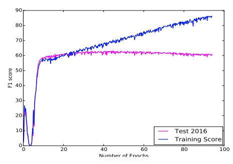

(a) $\varepsilon = 1e^{-7}$ $\rho = 0.05$

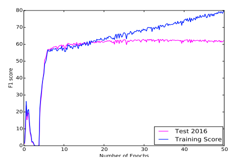

(b) $\varepsilon = 1e^{-7}$ $\rho = 0.25$

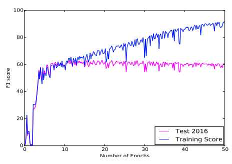

(c) $\varepsilon = 1e^{-7}$ $\rho = 0.75$

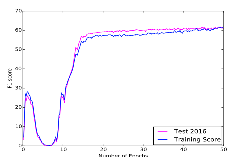

(d) $\varepsilon = 1e^{-7}$ $\rho = 0.95$

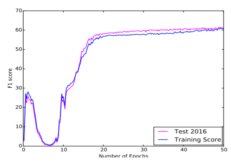

(a) $\varepsilon = 1e^{-8}$ $\rho = 0.05$

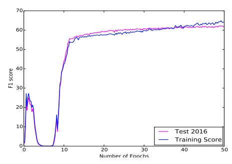

(b) $\varepsilon = 1e^{-8}$ $\rho = 0.25$

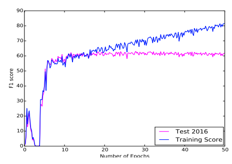

(c) $\varepsilon = 1e^{-8}$ $\rho = 0.75$

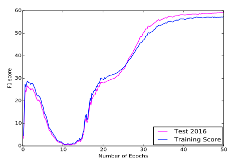

(d) $\varepsilon = 1e^{-8}$ $\rho = 0.95$

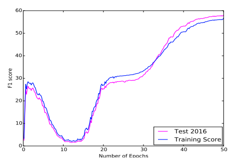

(a) $\varepsilon = 1e^{-9}$ $\rho = 0.05$

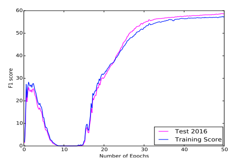

(b) $\varepsilon = 1e^{-9}$ $\rho = 0.25$


(c) $\varepsilon = 1e^{-9}$ $\rho = 0.75$


(d) $\varepsilon = 1e^{-9}$ $\rho = 0.95$


(a) $\varepsilon = 1e^{-10}$ $\rho = 0.05$


(b) $\varepsilon = 1e^{-10}$ $\rho = 0.25$


(c) $\varepsilon = 1e^{-10}$ $\rho = 0.75$

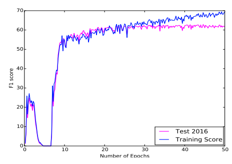

(d) $\varepsilon = 1e^{-10}$ $\rho = 0.95$

Figure C.2: The training and test scores for $\varepsilon = 1e^{-4}, .., 1e^{-10}$ *and* $\rho = 0.05, 0.25, 0.75, 0.95$

Bibliography

[1] James Bergstra, Olivier Breuleux, Frederic Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a CPU and GPU math compiler in Python. *Proceedings of the Python for Scientific Computing Conference (SciPy)*, pages 1–7, 2010.

[2] Sharan Chetlur and Cliff Woolley. cuDNN: Efficient Primitives for Deep Learning. *arXiv preprint arXiv: . . .* , pages 1–9, 2014.

[3] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *The Journal of Machine Learning Research*, 11:625–660, 2010.

[4] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A Library for Large Linear Classification. *JMLR*, 9:1871–1874, 2008.

[5] Jianfeng Gao, Xiaodong He, Wen-tau Yih, and Li Deng. Learning continuous phrase representations for translation modeling. In *In ACL*. Citeseer, 2014.

[6] Alec Go, Richa Bhayani, and Lei Huang. Twitter Sentiment Classification using Distant Supervision. Technical report, The Stanford Natural Language Processing Group, 2009.

[7] Matthias Hagen, Martin Potthast, Michel Büchner, and Benno Stein. Webis: An ensemble for twitter sentiment detection. SemEval, 2015.

[8] Hatzivassiloglou and McKeown. Predicting the semantic orientation of adjectives. In *Proceedings of the 35th Annual Meeting of the ACL and the*

8th Conference of the European Chapter of the ACL, pages 174–181. ACL, 1997.

[9] Martin Jaggi, Fatih Uzdilli, and Mark Cieliebak. Swiss-Chocolate: Sentiment Detection using Sparse SVMs and Part-Of-Speech n-Grams. In *SemEval 2014 - Proceedings of the 8th International Workshop on Semantic Evaluation*, pages 601–604, Dublin, Ireland, August 2014.

[10] Rie Johnson and Tong Zhang. Semi-supervised Convolutional Neural Networks for Text Categorization via Region Embedding. In *NIPS 2015 - Advances in Neural Information Processing Systems 28*, pages 919–927, 2015.

[11] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A Convolutional Neural Network for Modelling Sentences. In *ACL - Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 655–665, Baltimore, Maryland, USA, April 2014.

[12] Yoon Kim. Convolutional Neural Networks for Sentence Classification. In *EMNLP 2014 - Empirical Methods in Natural Language Processing*, pages 1746–1751, August 2014.

[13] Simon Lacoste-Julien, Martin Jaggi, Mark Schmidt, and Patrick Pletscher. Block-Coordinate Frank-Wolfe Optimization for Structural SVMs. In *ICML 2013 - Proceedings of the 30th International Conference on Machine Learning*, 2013.

[14] Bing Liu. Sentiment Analysis and Subjectivity . In *Handbook of Natural Language Processing*, 2010.

[15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[16] Ryan McDonald, Keryy Hannan, Tyler Naylon, Mike Wells, and Jeff Reynar. Structured Models for Fine-to-Coarse Sentiment Analysis. June 2007.

[17] Tomas Mikolov, Quoc V Le, and Ilya Sutskever. Exploiting Similarities among Languages for Machine Translation. *arXiv*, September 2013.

[18] 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.

[19] Saif M Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. NRC-Canada: Building the State-of-the-Art in Sentiment Analysis of Tweets. In *SemEval-2013 - Proceedings of the International Workshop on Semantic Evaluation*, pages 321–327, Atlanta, Georgia, USA, August 2013.

[20] Saif M Mohammad and Peter D Turney. Emotions Evoked by Common Words and Phrases: Using Mechanical Turk to Create an Emotion Lexicon. In *Proceedings of the NAACL HLT 2010 - Workshop on Computational Approaches to Analysis and Generation of Emotion*, pages 26–34, Los Angeles, California, USA, June 2010.

[21] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010.

[22] Preslav Nakov, Alan Ritter, Sara Rosenthal, Veselin Stoyanov, and Fabrizio Sebastiani. SemEval-2016 task 4: Sentiment analysis in Twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval '16, San Diego, California, June 2016. Association for Computational Linguistics.

[23] Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A Smith. Improved Part-Of-Speech Tagging for Online Conversational Text with Word Clusters. In *Proceedings of NAACL-HLT*, pages 380–390, 2013.

[24] Bo Pang and Lillian Lee. Mining the Peanut Gallery:Opinion Extraction and Semantic Classification of Product Reviews. In *WWW2003*, 2003.

[25] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? Sentiment Classification using Machine Learning Techniques. In *the ACL-02 conference*, pages 79–86, Morristown, NJ, USA, 2002. Association for Computational Linguistics.

[26] Jeffrey Pennington, Richard Socher, and Christopher D Manning. GloVe: Global Vectors for Word Representation. In *Conference on Empirical Methods in Natural Language Processing*, 2014.

[27] Jonathon Read. Using emoticons to reduce dependency in machine learning techniques for sentiment classification. In *Proceedings of the ACL student research workshop*, pages 43–48. Association for Computational Linguistics, 2005.

[28] Sara Rosenthal, Preslav Nakov, Svetlana Kiritchenko, Saif M Mohammad, Alan Ritter, and Veselin Stoyanov. SemEval-2015 Task 10: Sentiment Analysis in Twitter. In *Proceedings of the 9th International Workshop*

on Semantic Evaluation, Denver, Colorado, June 2015. Association for Computational Linguistics.

[29] Sascha Rothe, Sebastian Ebert, and Hinrich Schutze. Ultradense Word Embeddings by Orthogonal Transformation. *arXiv*, February 2016.

[30] Aliaksei Severyn and Alessandro Moschitti. On the Automatic Learning of Sentiment Lexicon. NAACL, 2015.

[31] Aliaksei Severyn and Alessandro Moschitti. Twitter Sentiment Analysis with Deep Convolutional Neural Networks. In *38th International ACM SIGIR Conference*, pages 959–962, New York, USA, August 2015. ACM.

[32] Aliaksei Severyn and Alessandro Moschitti. UNITN: Training Deep Convolutional Neural Network for Twitter Sentiment Classification. In *SemEval 2015 - Proceedings of the 9th International Workshop on Semantic Evaluation*, 2015.

[33] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 101–110. ACM, 2014.

[34] Karen Simonyan, Andrea Vedaldi, and Andrew Zissermann. Deep inside convolutional networks: Visualising image classification models and saliency maps. 2014.

[35] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer, 2013.

[36] Christian Szegdy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. 2014.

[37] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[38] Peter D Turney. Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. 2002.

[39] Peter D Turney and Michael L Littman. Unsupervised learning of semantic orientation from a hundred-billion-word corpus. In *orientation from a hundred-billion-word corpus. Technical Report EGB-1094*, 2002.

[40] Jason Weston, Sumit Chopra, and Keith Adams. Tagspace: Semantic embeddings from hashtags. In *Proceedings of the 2014 Conference on EMNLP*, pages 1822–1827. ACL, 2014.

[41] Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proc. of HLT-EMNLP-2005*, 2005.

[42] Yuanbin Wu, Qi Zhang, Xuanjing Huang, and Lide Wu. Structural opinion mining for graph-based sentiment representation. In *Proceedings of the 2011 COnference on Empirical Methods in NLP*, pages 1332–1341. ACL, 2011.

[43] Mattew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. 2013.

[44] Matthew D. Zeiler. ADADELTA: An Adaptive Learning Rate Method. *arXiv*, page 6, 2012.

[45] 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.

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Sentiment Analysis using Deep Convolutional Neural Networks with Distant Supervision

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
Deriu	Jan

With my signature I confirm that
- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
Olten, 22.04.2016	Deriu Jan

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.