FH Salzburg

Marshallplan-Jubiläumsstiftung
Austrian Marshall Plan Foundation

Marshall Plan Scholarship Report

Convolutional Neural Networks in Natural Language Processing

Name	Veronika Haaf
University	University of Massachusetts, Amherst
College	College of Information and Computer Sciences (CICS)
Research Center	Center for Intelligent Information Retrieval (CIIR)
Supervisor CIIR	Distinguished Professor W. Bruce Croft
Supervisor SUAS	FH-Prof. Univ.-Doz. Dr. Stefan Wegenkittl

Salzburg, November 2, 2017

Acknowledgement

Researching at the CIIR at the University of Massachusetts Amherst was an interesting and revealing time for me. I am very grateful to everyone, who made this memorable experience possible.

First of all, I would like to thank Distinguished Professor W. Bruce Croft for the great opportunity of being part of the CIIR research group and his valuable guidance throughout my research. At this point, I also would like to thank Hamed Zamani for his continuing patience, advice and support on my work.

I want to express my gratitude towards FH-Prof. Univ.-Doz. Dr. Stefan Wegenkittl, who initiated the collaboration between the CIIR and the Salzburg University of Applied Sciences, and thank him for his support and advise as my supervisor.

Furthermore, I thank the Austrian Marshall Plan Foundation for their generous financial support in this undertaking.

Contents

1. Introduction

In times of social media platforms like Twitter, where people openly comment and discuss any kind topic - be it public figures, newly presented products or a disastrious event; when the public reputation of people, companies and any other agenda is decisively influenced by a flood of online reviews and comments, the research field of Natural Language Processing (NLP) becomes increasingly relevant in the context of automatically sorting out public attitudes and opinions on specific topics. Only recently in the US election the influence of social media platforms on the election results has been intensively discussed.

The field of NLP includes two main research areas. The first one is related to understanding natural language, while the second area deals with the synthesis of natural language. A descriptive example for both fields are dialog systems, so-called chatbots, that interact with humans. An actual dialog between a machine and a human, can only take place, if both parties understand each other and have the means to communicate to one another. Thus, the machine has to have the capability to grasp what the interacting human says or writes in order to generate and give an appropriate response. Many currently used dialog systems are very basic in the sense, that the dialog is precisely specified. This means, that the chatbot states a question, as well as some predefined answers to a human. Research aims at more sophisticated systems, that do not rely on predefined answers and keywords, but are rather able to extract the relevant information from the humans answer and react flexibly.

Several sub-fields have formed up within NLP, like topic detection, studies on controversy and stands, and relationship extraction. Other areas focus on exposing fake news or analyzing the public opinion on some specific topic, which can be a useful tool in the context of politics and marketing.
The latter, which is also known as Opinion Mining or Sentiment Analysis (SA), is the research field of interest in this work. It plays an important role in all research related to opinionated text, which involves any content-related analysis of comments, reviews, Tweets, and discussions.

While early approaches employed sentiment lexicons and syntactic rules for predicting the sentiment of texts, most approaches nowadays focus on employing Machine Learning (ML) systems to solve this task. In this context, Neural Networks (NNs) are often the tool of choice. Thereby, especially two types of NNs, Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) are favored the most. Due to their sequential processing, RNNs have been an established tool in NLP for quite some time. CNNs, on the other hand, originated in the field of Image Processing and Computer Vision and have been in use for only a few years. By now, RNNs and CNNs represent the most common competing NN types in several NLP tasks.

This report is mainly concerned with the application of CNNs in SA. Thereby, a special focus lies on the performance comparison of CNNs and RNNs in sentiment classification. After a thorough investigation of state-of-the-art applications and performances of CNNs and RNNs in the field of SA, two exemplary models are compared with regard to their architecture, training process and classification performance. Later on, the benefits of using sentiment signals in various forms for document ranking in Information Retrieval (IR) are examined. Experiments on two corpora help to explore the potential benefits of sentiment features in a learning-to-rank approach. Finally, a summary of the conducted work and findings is given.

2. Neural Networks in Sentiment Classification

This chapter investigates the research efforts untertaken in combining the application of NNs in the field of SA. Beginning with the state-of-the-art, Section 2.1 focusses on the recent advances made in SA with respect to NNs. In particular, the progress achieved using RNNs and CNNs in various settings is discussed, leading to a final decision on two deputy NN architectures, that will be analysed and compared in more detail in Section 2.2. Therefore, one RNN and one CNN model will be trained and evaluated based on selected criteria.

2.1 State-of-the-art

In this section, recent advances made in the field of SA using NNs, especially RNNs and CNNs, will be presented. With regard to later experiments, only approaches favouring RNN or CNN architectures will be discussed.

2.1.1 RNNs in Sentiment Classification

Long Short-Term Memory (LSTM) architectures represent a special kind of RNNs. Contrary to RNNs, LSTMs learn long-term dependencies through special memory cells. Tai et al. [23] worked on a generalization of RNNs with LSTM units and developed a so-called Tree-LSTM, which extended the basic LSTM architecture, thereby enriching its possible network topologies. They introduced models for two use-cases: semantic relatedness of sentence pairs and sentiment classification. In fine-grained 5-class sentiment prediction the proposed architecture exceeded other systems marginally and was just outperformed on binary sentiment classification by a CNN architecture.
A very similar approach was presented by Zhu et al. [28], who also developed a tree-structured LSTM (S-LSTM) using special S-LSTM memory blocks. They applied the

S-LSTM in the context of semantic composition substituting parts of the Recursive Neural Tensor Network (RNTN) proposed by Socher et al. [22]. With their approach, Zhu et al. [28] achieved a slight improvement in sentiment prediction at phrase-level compared to the original RNTN model.

Le and Zuidema [15] combined Recursive Neural Networks (ReNNs) with LSTMs in order to improve capabilities with regard to compositionality. Instead of only using the ouput of child nodes, as it is quite common in ReNNs, they also considered the information stored in their memory cells to make more information of the lower levels in the parse tree available to upper levels. Applied on the task of sentiment classification, in a setting similar to Tai et al. [23], they achieved very similar results.

Tang et al. [24] applied RNNs for document modeling, which they used as a base for sentiment classification. In a first step they created sentence representations using either CNNs or LSTMs, which they encoded in a second step in document representations using a gated RNN. The document representations were then used as feature vectors for classification. They exceeded several chosen baseline systems like Support Vector Machines (SVMs) and a CNN model as proposed by Kim [12] using their approach. In other experiments they showed that their developed models, especially the bidirectional gated NN, outperformed regular simple RNN models.

Wang et al. [25] combined NNs with look-up tables, taking word embeddings as additional information for binary sentiment classification. They tested several combinations: RNNs with a fixed and trainable look-up table as well as a LSTM with a trainable look-up table. Escpecially, the RNN and the LSTM with trainable look-up tables performed well and exceeded other classifiers like a SVM. However, they did not outperform the Dynamic Convolutional Neural Network (DCNN) proposed by Kalchbrenner et al. [11], which will be discussed in more detail in the next section.

Radford et al. [20] investigated the properties, especially the representations, learned by multiplicative Long Short-Term Memory (mLSTM) architectures, which have been proposed by Krause et al. [13] for sequence modeling. By training an mLSTM model for text prediction on character-level, they achieved good performance results for sentiment classification using logistic regression. Further analysis of their trained model yielded the finding of a sentiment unit.

Kumar et al. [14] proposed a *dynamic memory network*, which can be applied to different NLP tasks. Their network consisted of four components: an input module, a question module, an episodic memory module and an answer module. The input module employed a gated RNN to encode text to vector representations, as did the question module. The episodic memory module itself consisted of a RNN and an attention mechanism, processing current and previous inputs from the input and question modules in order to update the episodic memory. Finally, the information of the episodic memory module was forwarded to the answer module, another gated RNN, which outputed an answer depending on the task type. Applied in the sentiment analysis task, this network structure outperformed several other systems like the Tree-LSTM of Tai et al. [23] and some CNN architectures.

As this selection of publications shows, the state-of-the-art research on RNNs in the field of SA pursues mainly two strategies in order to boost performances. One strategy is the exploration of different kinds of RNN structures such as the work on Tree-LSTMs of Tai et al. [23] or the experiments on combining ReNN and LSTM structures by Le and Zuidema [15]. Others focus on the search and development of supplementary components like Radford et al. [20], who experiment with mLSTM units, or Kumar et al. [14], who worked on a four-component network. Both strategies result in complex network architectures, whose achieved improvements in performance seems rather small. Most importantly, there is a noticable trend of applying and adapting LSTM networks for SA tasks.

2.1.2 CNNs in Sentiment Classification

CNNs have been popular in research fields like Computer Vision for quite some time. A few years back, they increasingly attracted attention from research in the area of NLP. Several approaches applying CNNs on different aspects of SA have been proposed over time.

Kim [12] applied CNNs on sentiment classification at sentence-level. The used models are slightly different variants of a one-layer CNN. His experiments cover static and non-static, random and pre-trained word embeddings as well as channels. On four out of seven different classification tasks, Kims models outperformed other state-of-the-art systems.

Kalchbrenner et al. [11] were concerned with a slightly different task: the modelling of sentences using so-called Dynamic Convolutional Neural Networks (DCNNs). Distinct properties of these networks were the wide convolutional layers followed by a new method called dynamic k-max-pooling and the varying widths of intermediate layers, which depended on the DCNNs length of input sentences. In experiments, the DCNN performed better than several other systems in terms of accuracy for both fine-grained and binary sentiment prediction.

Another kind of CNNs, a *Character to Sentence Convolutional Neural Network*, was developed by Dos Santos and Gatti [5]. Using the proposed CNN architecture words were represented by word-level and character-level embeddings, which were used further to extract a sentence-level representation and finally, also a sentiment score. By applying their method to fine-grained and binary sentiment prediction, Dos Santos and Gatti exceeded the results of the RNTN model on sentence-level (root node) reported by Socher et al. [22].

Severyn and Moschitti [21] applied a CNN architecture, which was based on the works of Kim [12] and Kalchbrenner et al. [11], to SA in the context of Twitter. They trained word embeddings as input representations on Tweets and refined them using a distant supervision approach (see Go et al. [6]). They used the trained parameters to initialize their CNN. Finally, they tested their network on sentiment classification at phrase- and message-level and showed that this kind of pre-training leads to a better classification performance compared to a random or skip-gram word embedding.

Johnson and Zhang [9] compared two CNN architectures, a seq-CNN and a bow-CNN, and investigated the influence of word order on the text classification performance. The input for the seq-CNN was given by a concatenation of one-hot encoded words, while the input for the bow-CNN corresponded to an aggregation of those. They also experimented with parallel layers, in order to train several word embeddings at once. They chose classifiers like a SVM and bag-of-n-gram features as baselines for sentiment classification. Their bow-CNN variant outperformed those baselines. In a later work, Johnson and Zhang [10] introduced semi-supervised CNNs with two-view-embeddings. Those embeddings have been learned in an unsupervised or partially supervised setting. Experiments showed that two-view-embeddings applied as additional input on CNNs improved the performance in sentiment classification.

CNNs have also been tested on text at character-level. Zhang et al. [27], for example, proposed a nine-layer CNN architecture with only one-dimensional convolution applied. They tested their architecture in various combinations like the use of word embeddings and different alphabets. Applying data augmentation and considering 70 different characters, they could outperform different bag-of-word variants as well as an LSTM approach for sentiment classification.

More recently, inspired by their success in the field of Computer Vision, Conneau et al. [3] presented a Very Deep Convolutional Neural Network (VDCNN) architecture for text classification. The proposed VDCNN worked on character-level and incorporated 29 convolutional layers. Among other classification tasks, they applied the VDCNN also to sentiment classification and exceeded the performance achieved by Zhang et al. [27].

The visible trends for CNNs in SA are similar to the previously mentioned developments of RNNs in Section 2.1.1. Compared to the first popular CNNs such as the one-layer CNNs proposed by Kim [12], later approaches like the VDCNN of Conneau et al. [3] or the *Character to Sentence Convolutional Neural Network* by Dos Santos and Gatti [5] are far more complex. Similar as before, the achieved improvements with regard to performance are rather small.

2.1.3 Discussion

In the previous sections, recent advances made in SA by applying NNs have been introduced. For both RNNs and CNNs, researchers tend to experiment on the overall system structure and supplementary components, thereby increasing the system's complexity. Regarding RNNs, many findings indicate the great potential of LSTM structures and show how they can contribute to performance improvements.

Since there exist multiple NN structures of varying complexity featuring different input formats and sentiment levels, two comparable deputy NN architectures - one RNN and one CNN architecture - will be chosen for further analysis and comparison. In order to keep the comparison of RNNs and CNNs purely dependent on the system types, plain and elementary architectures are chosen for both kinds of NNs. Based on such elementary systems, an analysis and comparison of both architectures can be conducted in a clear, comprehensive manner.

The most elementary CNN known in research is the single-channel CNN structure of Kim et al. [12]. It represents a one-layer CNN, which has already been trained and successfully applied on sentence-level SA. Kim [12] presented three different single-channel CNN variants; thus, the selection will be narrowed down further in a second step in order to adjust the overall set up with regard to the chosen RNN architecture.

Since LSTMs constitute a very promising category of RNNs, this kind of network has been chosen as a comparitive network architecture. Most state-of-the-art LSTMs do not represent a plain, basic RNN architecture anymore but contain supplementary, additional components, which make no fair comparison to the chosen CNN. Since the CNN network incorporates a plain one-layer structure, an equally basic and working RNN structure is desirable. A comparable adversary was finally found in the LSTM network structure used in the Deep Learning Tutorial of the LISA lab at the University of Montreal on "LSTM Networks for Sentiment Analysis"[1], which is also applicable on sentence-level SA.

Based on these two plain network architectures, a comprehensive comparison is conducted, which is primarily concerned with the architectural network types themselves and their performances in the context of sentiment classification. The experimental setup and implementation as well as the resulting outcomes and insights will be discussed in Chapter 2.2, after the selected RNN and the CNN deputy architectures have been introduced and explained in more detail.

[1]`http://deeplearning.net/tutorial/lstm.html`

2.2 RNNs versus CNNs

The investigation of state-of-the-art CNN and RNN network structures in the last section points out the various advances in the performance of sentiment classification that can be achieved by creating more complex network structures and adding various other components. While the additional structural complexity contributes to the classifaction performance in terms of slight improvements, it hides the potential of the plain NN architecture. These capabilities, that lie within the basic, plain NN structures itself, will be the focus of this section.

Therefore, basic networks have been chosen for the comparitive exmination of RNNs and CNNs in SA. As mentioned in Section 2.1.3, a one-layer CNN as proposed by Kim [12] and a LSTM as developed by the LISA lab at the University of Montreal[2] have been selected as deputy architectures in the comparison. These two network architectures will be explained in more detail in the upcoming section. Then, the experimental setup and execution will be discussed.

2.2.1 CNN Architecture

In this section, the basic architecture of the single-channel CNN variants (static-random, static-Word2Vec, non-static-Word2Vec) proposed by Kim [12] will be introduced.

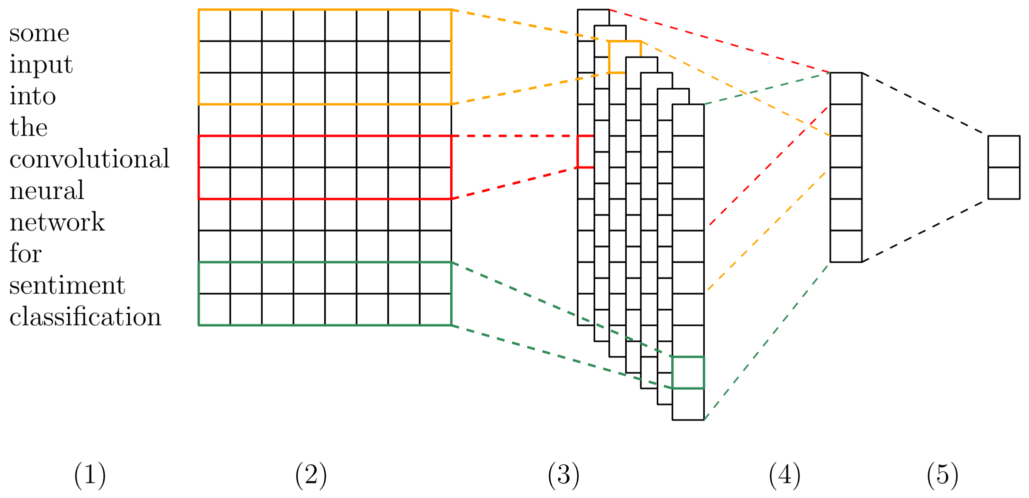


Figure 2.1: Single channel CNN architecture as described by Kim [12]: (1) input sentence with n words, (2) $(n \times m)$-dimensional word vector representation, (3) convolution with different filters, (4) max-pooling, (5) fully connected dropout

[2]http://deeplearning.net/tutorial/lstm.html

The single channel CNNs are structured as illustrated in Figure 2.1. The input to the CNN are single sentences (1), which are mapped to multi-dimensional word vector representations in a word-wise manner. If the dimension of the chosen word embedding was m and the sentence length n (including appropriate padding), this results in a sentence representation of a $(n \times m)$-dimensional matrix (2). In the next step, the sentence matrix is convoluted with differently sized filters, whose second dimension equals m. As mentioned by Kim [12], the first dimension was chosen to be 3, 4 and 5, respectively. Thus, the matrix is convoluted by three different filters. Since the second dimension of the matrix is equal to the second dimension of the filters, the resulting feature maps (3) are vector-shaped. There are 100 feature maps per filter size (see Kim [12]), implying that the input is convoluted by 300 different filters. The information contained in the feature maps is then processed by Rectified Linear Units (ReLUs) and sub-sampled by max-pooling over time to the prenulitmate layer (4). This layer in turn is fully connected to the final softmax layer (5), which predicts the sentiment of the sentence.

Model name	Word vector initialization	Adaptation in training
non-static-random	Random	yes
static-Word2Vec	Word2Vec	no
non-static-Word2Vec	Word2Vec	yes

Table 2.1: Single channel CNN variants (see Kim [12])

The difference between the three CNN variants lies not in their architecture but in their initialization and manipulation of word representations during training. As summarized in Table 2.1, the word embeddings are initialized randomly and adapted during training for the first variant, the non-static-random model. The other two variants, the static-Word2Vec and the non-static-Word2Vec models, are build upon pre-trained Word2Vec representations whereas the embeddings are kept static for the first and are being adapted for the other during training.

2.2.2 RNN Architecture

To conclude the exploration of basic NNs structures, the architecture and performance of the LSTM network will be explained in the following.

The model structure, as depicted in Figure 2.2, has been taken from the Deep Learning Tutorial of the LISA lab at the University of Montreal on "LSTM Networks for Sentiment Analysis"[3].

[3]http://deeplearning.net/tutorial/lstm.html

At first, word sequences are presented to the network in the form of sequence vectors. These vectors are built based on a dictionary, which contains terms ordered by frequency. The input to the network itself is then transformed using a randomly initialized look-up table, which maps each term to a multi-dimensional vector (1). This procedure is comparable to the previously introduced input for the CNN in the case of random vector initialization. The trainable, randomly initialized word vectors for the CNN are stored in the pre-compiled dictionary. The word representations in the LSTM model, on the other hand, are randomly initialized and trained in the form of a word embedding layer, respectively look-up table. The implementation of trainable word embeddings is different for both networks. The underlying concept is nonetheless the same.
The initially random word vectors are then subsequently fed to the LSTM layer and processed (2). The LSTMs output is subject to mean-pooling over time (3). The sentiment is predicted via the final softmax layer (4).

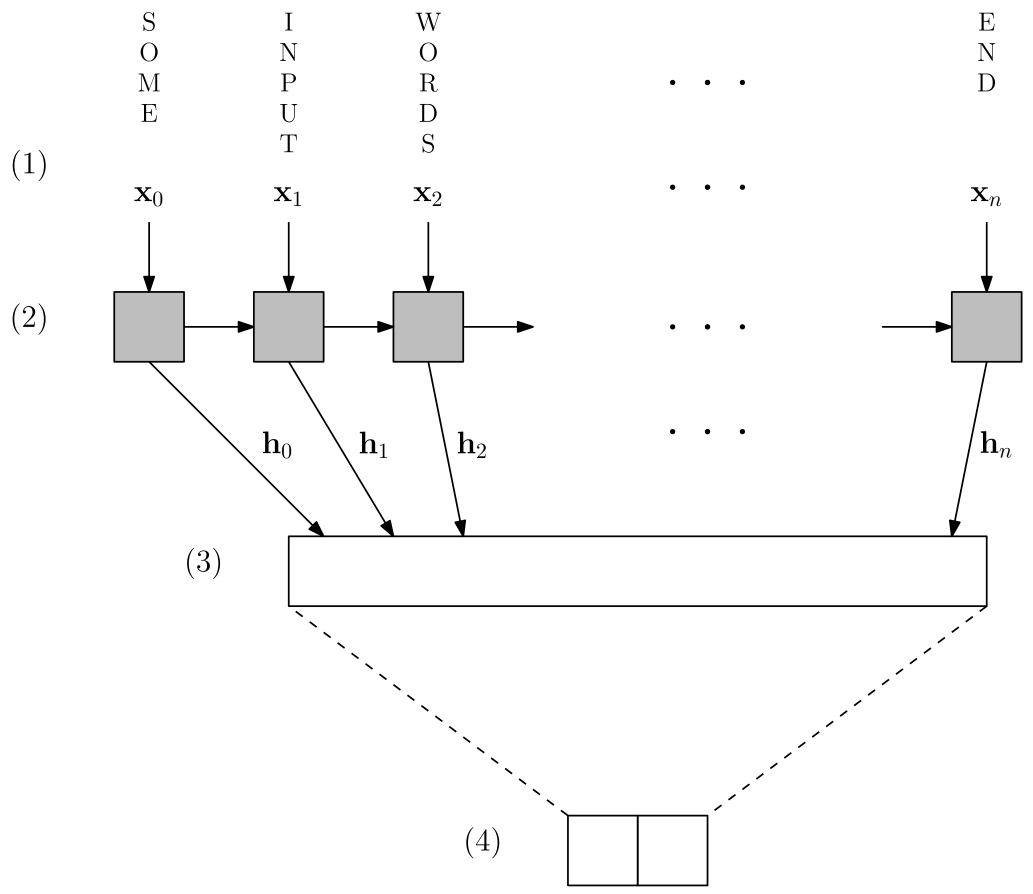


Figure 2.2: Unfolded LSTM network architecture: (1) input sentence represented as frequency vector, (2) LSTM layer unfolded over time, (3) mean-pooling, (4) logistic regression

2.2.3 Experimental Setup

The NN architectures chosen for the RNN and CNN have been introduced in the previous section. A small-scale experiment was conducted in order to compare the performances of the chosen RNN and CNN architectures with respect to SA. Besides the qualitative comparison of test accuracy, also other aspects like the training time or the selection of the best model will be of interest. In the following, the experimental setup with regard to datasets, tools and parameter choices are explained. The performance results will be presented in Section 2.2.4.

Training Data and Setup

The two NN architectures have been applied on the Sentence Polarity Dataset v1.0[4], which was introduced by Pang and Lee [19]. The data set contains 10,662 binary labeled sentences, whereas one half contains sentences with positive and the other negative sentiments.

The network training was conducted in form of a ten-fold cross validation. The splits into train and test set have been conducted according to the stratefied sampling strategy. Separately, for each of the two sentiment classes (positive and negative), the data samples have been assigned randomly to one of the ten folds. In a second step, 10% of the train set samples were assigned to the validation set.

Implementation

In order to compare and analyze the chosen NN architectures, some actual models have been trained to investigate different aspects related to the model's architecture, training and performances. As discussed earlier, the single-channel CNNs will be compared to a basic LSTM architecture.

The code to train and test the single channel models of Kim [12] is publicly available on GitHub[5]. It is written in Python and employs Theano to implement the NNs structure. Some minor adaptations, that do not involve the networks' basic archtecture but its framework, have been conducted. Due to its comparability with regard to the input, the non-static-random single channel model has been chosen as adversary network to the LSTM.

The LSTM network implementation in the Deep Learning Tutorial on LSTMs for SA[6] represents a simplified variant of a LSTM layer. It was intended to extend the LSTM

[4]http://www.cs.cornell.edu/people/pabo/movie-review-data/
[5]https://github.com/yoonkim/CNN_sentence
[6]http://deeplearning.net/tutorial/lstm.html

units to incorporate three peepholes, one for each gate. Over the course of empirical experiments and a parameter grid search, it became clear that the inclusion of peepholes does not improve the performance compared to the original LSTM implementation.
Table 2.2 gives an examplary overview of the performance differences between the original and the adapted implementation. Both models have been trained using the same parameters (see Table 2.4) on the same dataset and the same allocation of data samples to the train and test set. The direct comparison shows an identical validation performance and a notable difference test accuracy and an increase in training time in terms of the needed number of epochs, as well as time per epoch. Such a fundamental change in the LSTMs implementation can certainly require a different set of parameters for optimum performance. Thus, a grid search was conducted on both, the original and the adapted implementation. Hereby, different values for the learning rate, the maximum number of words, the maximum sentence length, the batch size as well as the word embedding dimension have been examined.
The grid search showed in general, that the inclusion of peepholes prolongs the training time noticably. For a LSTM with peepholes each epoch in training took on average twice as long as an epoch for the simple LSTM implementation. The overall performances of each LSTM model structure are comparable. The best validation performance of 78.95% (on 4 models) was achieved with the original implementation. The parameters of the best models (best validation performances) for both implementations have been trained using a batch size of 4, a dictionary of 5 000 words, maximum sentence length of 50 words, word embedding dimensions of 32 and mostly 64 and a varying learning rate.

LSTM	Last epoch	Train	Valid	Test	Average time per epoch
original	36	99.90	86.66	78.39	188.7
adapted	46	100.00	86.66	72.38	223.5

Table 2.2: Comparison of original (simplified, no peepholes) and adapted (3 peepholes) LSTM implementations with regard to performance.

Note: The empirical comparison, was conducted based on the Large Movie Review Dataset v1.0[7], which was introduced by Maas et al. [18]. Among other data it contains 50,000 binary labeled movie reviews of IMDb, which are already separated in train and test sets. Both test sets contain 25,000 review with a balanced distribution positive and negative labels.
The grid search on both, the original and the adapted implementation was done based on the Sentence Polarity Dataset (see Section 2.2.4).

[7]http://ai.stanford.edu/~amaas/data/sentiment/

Parameter choices

Kims [12] recommendations on parameter settings are listed in Table 2.3 and given based on a grid search. Therefore, these parameters are applied for the forthcoming experiments, too.

Parameter name	Parameter value
Word embedding dimension	300
Image shape ($n \times k$)	64×300
Filter sizes ($h \times k$)	(3×300), (4×300), (5×300)
Number of filters	100 per filter size
Batch size	50
Batch shuffle	*true*
Learn decay	0.95
Learning method	AdaDelta
Dropout	0.5
l_2 contraint	3

Table 2.3: CNN parameter choices provided by Kim [12], whereas the first dimension of the image shape includes padding (maximum sentence length was 56)

Parameter name	Parameter value
Word embedding dimension	128 (300)
Maximum sentence length	100 (56)
Number of words	$10,000$
Batch size	16 (8)
Batch shuffle	*true*
Learn rate	0.0001
Learning method	AdaDelta
Dropout	0.5

Table 2.4: LSTM parameter choices used in the Deep Learning Tutorial. Parameter choices which differ in the context of the experiments are given in round brackets.

The Deep Learning Tutorial provided a set of parameters (see Table 2.4), too. Yet, there was no explanation given for the taken choices. During a grid search different values for the maximum sentence length, the batch size, the dimension projection, number of words and the learning rate have been tested. The decision on the final parameter choice was influenced by two factors. The results of the grid search as well as the comparability to the CNN model. Based on these aspects some of the provided parameters have been adapted. The adapted parameter values are listed in 2.4 next to the original parameter choice in round brackets.

Selecting the best model

During the initial analysis of the CNN and LSTM implementation, a difference in the selection process of the best model stood out. While the CNNs implementation would run for a given amount of epochs and then select the model with the best validation performance, the training of the LSTM would be finished early, depending on a patience parameter, which was set to 10. Using the patience parameter, the training is stopped early, if ten times in a row, the validation error is bigger than the minimum of all noted validation errors in the history buffer (excluding the last ten entries). The final model is then again the model with the best validation performance, only the overall number of trained models is smaller.
Within the following experiments, also the difference between those two methods of selecting the best model (global selection, patience criterion) will be discussed.

2.2.4 Experiments

In the following subsections, the performance results of the CNN and the RNN on the sentence polarity dataset is presented.

CNN Performance on Sentence Polarity Dataset

In this experiment the non-static-random CNN was trained on the earlier mentioned Sentence Polarity Dataset in a ten fold setting. The time needed for training and testing per fold (each 50 epochs) on a CPU (Intel Core i7) amounted to 5 hours and 45 minutes on average, but only took approximately 5 minutes per fold using a GPU (Titan X).

Table 2.5 and 2.6 summarize the best models per fold trained over the course of 50 epochs, respectively a variable number of epochs, with regard to previously mentioned patience criterion. Within the tables also the epoch, when the final model happened to be trained, is noted. Considering the chosen models for each epoch setting, it gets clear that only in four out of ten folds the pro-longed training resulted in an improved model.

For fold three, four, six and ten, the validation performance reached a higher peak at a later epoch than the patience criterion would have allowed.

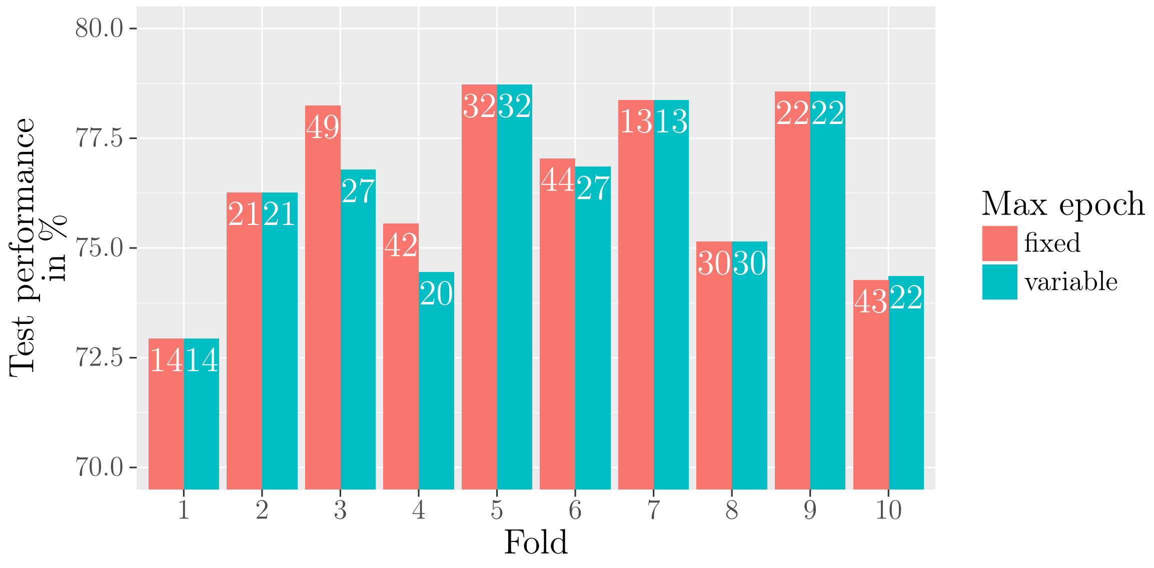
Fold	Epoch	Train	Valid	Test	Max epoch
1	14	100.00	**78.11**	72.93	fixed
2	21	100.00	76.42	76.26	fixed
3	49	100.00	76.11	78.24	fixed
4	42	100.00	77.26	75.55	fixed
5	32	100.00	76.84	**78.72**	fixed
6	44	100.00	77.58	77.04	fixed
7	13	100.00	76.84	78.37	fixed
8	30	100.00	75.58	75.14	fixed
9	22	100.00	76.00	78.56	fixed
10	43	100.00	77.58	74.27	fixed

Table 2.5: Top CNN model performances among 50 epochs per fold. The best test performance is achieved in the fifth fold, the best model according to validation performance was trained in fold one.

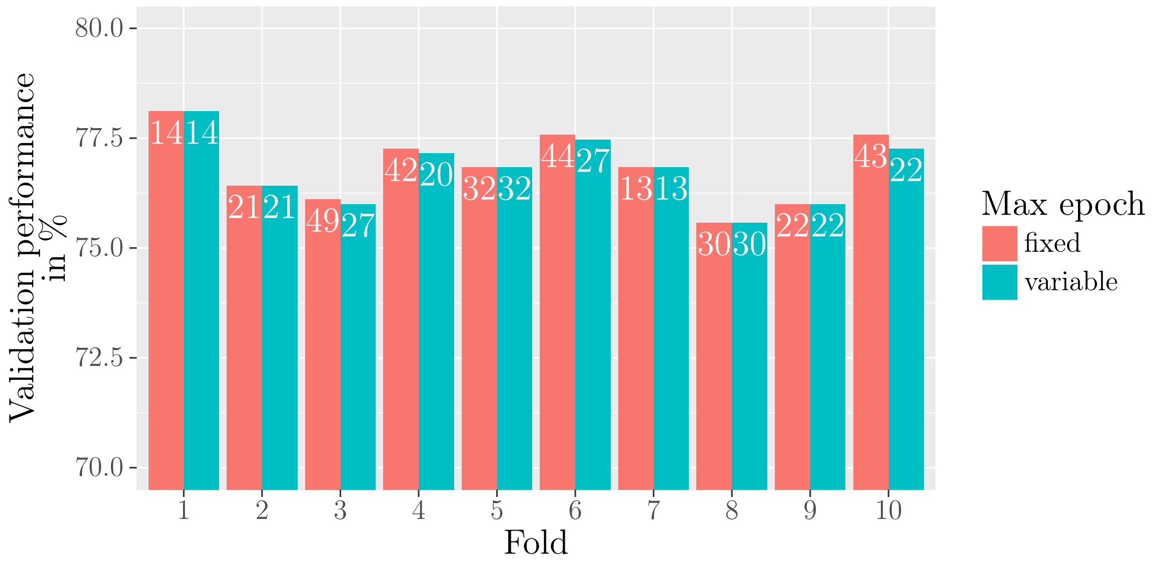
Fold	Epoch	Train	Valid	Test	Max epoch
1	14	100.00	**78.11**	72.93	variable
2	21	100.00	76.42	76.26	variable
3	27	100.00	76.00	76.78	variable
4	20	100.00	77.16	74.45	variable
5	32	100.00	76.84	**78.72**	variable
6	27	100.00	77.47	76.85	variable
7	13	100.00	76.84	78.37	variable
8	30	100.00	75.58	75.14	variable
9	22	100.00	76.00	78.56	variable
10	22	100.00	77.26	74.36	variable

Table 2.6: Top CNN model performances among a variably chosen number of epochs per fold. The best test performance is achieved in the fifth fold, the best model according to validation performance was trained in fold one.

Figure 2.4 exemplary displays the course of the train, validation and test performance for the CNN model training over 50 epochs in the fifth fold, which showed the highest test accuracy. Starting with 60% at the end of the first epoch, the performance of the CNN model on the train set reached 100% within the first ten epochs. The performances of the validation and test set, also grew rapidly within these epochs, but ran lower. The accuracy on the validation set ranges marginally above 75%. The test set accuracy runs slightly higher. The curvature of the depicted curves is similar for all other folds. The only differences are given with regard to the performance values. In some cases, the test and validation performances ran not that clearly separated .



(a) CNN test set classification performance over ten folds.



(b) CNN validation set classification performance over ten folds.

Figure 2.3: CNN performance on MR: Comparing the best models chosen among a fixed set of 50 epochs, respectively variable set of epochs. The white labels mark the epoch, where the final model was achieved.

Parts of the of the information displayed in Table 2.5 and 2.6 is visualized in Figure 2.3. The focus of these diagrams lies in the performance comparisons caused by the

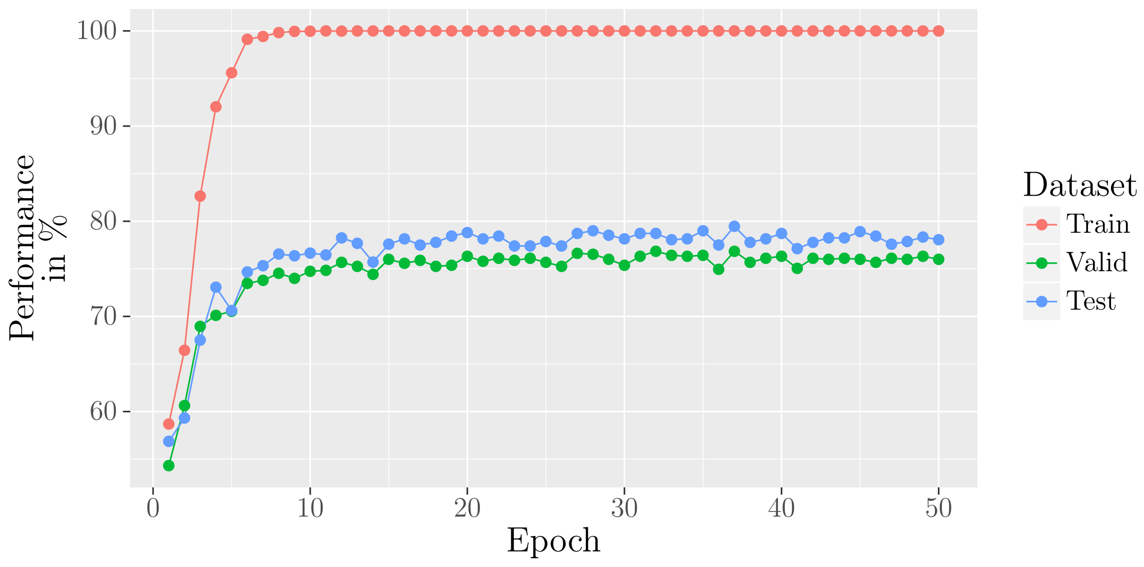


Figure 2.4: CNN model performances of train, validation and test sets over 50 epochs in the fifth fold.

different epoch regulations. The best model performances achieved over a fixed number of 50 epochs and a variable number of epochs due to the early patience criterion are compared for the test and the validation set. Figure 2.3b illustrates clearly the small differences in the validation performances of both strategies, which vary more extreme with regard to the test performances, as Figure 2.3a shows.
On average, over 50 epochs of training the validation performance of the CNN model amounts to 76.83% and the test accuracy to 76.51%. The standard deviation amounts to 0.77% for the validation and to 1.91% for the test performance. With regard to the patience criterion the validation performance is 76.77% and the test performance 76.24%, which represents a rather small difference to the average performances of the fixed runs. The standard deviation is 0.74% for validation and 1.89% for the test performance.

LSTM Performance on Movie Reviews

In this experiment the LSTM network (original implementation) was trained on the Sentence Polarity Dataset in a ten fold setting. The time needed for training and testing per fold (each 50 epochs) on a CPU (Intel Core i7) amounted to 2 hours and 55 minutes on average, but could be reduced using a GPU (Titan X) to half an hour per fold.

Table 2.7 contains the best models per fold according to validation performance. The models with the best validation and test performance are highlighted in gray. In case of the LSTM there is no difference in selecting best performing model on a global scale compared to applying the patience criterion. Thus, running the the training for a fixed number of epochs did not result in better models.

Fold	Epoch	Train	Valid	Test	Max epoch
1	8	97.47	76.53	72.44	fixed/variable
2	6	94.35	**78.79**	75.44	fixed/variable
3	6	93.37	77.07	76.10	fixed/variable
4	5	92.10	76.59	75.37	fixed/variable
5	7	95.66	77.29	74.29	fixed/variable
6	8	96.94	76.10	**77.41**	fixed/variable
7	4	87.36	76.10	75.51	fixed/variable
8	5	92.72	77.32	76.00	fixed/variable
9	4	87.89	76.18	76.92	fixed/variable
10	5	91.99	75.83	75.59	fixed/variable

Table 2.7: Top LSTM model performances among 50 epochs, respectively a variably chosen number of epochs per fold. The best test performance is achieved within the sixth fold, the best model according to validation performance was trained in fold two.

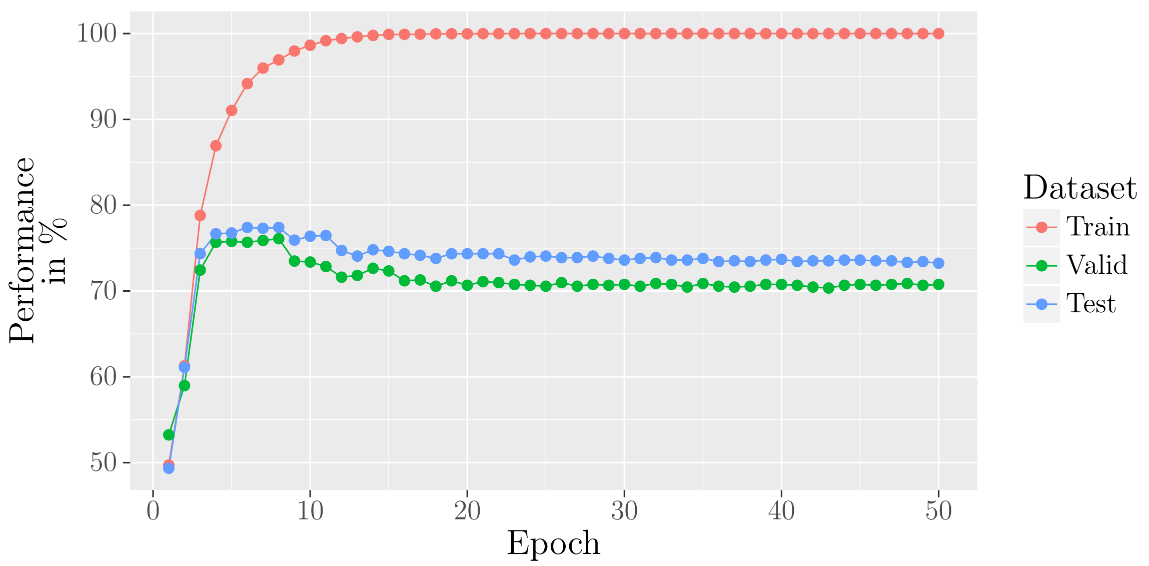
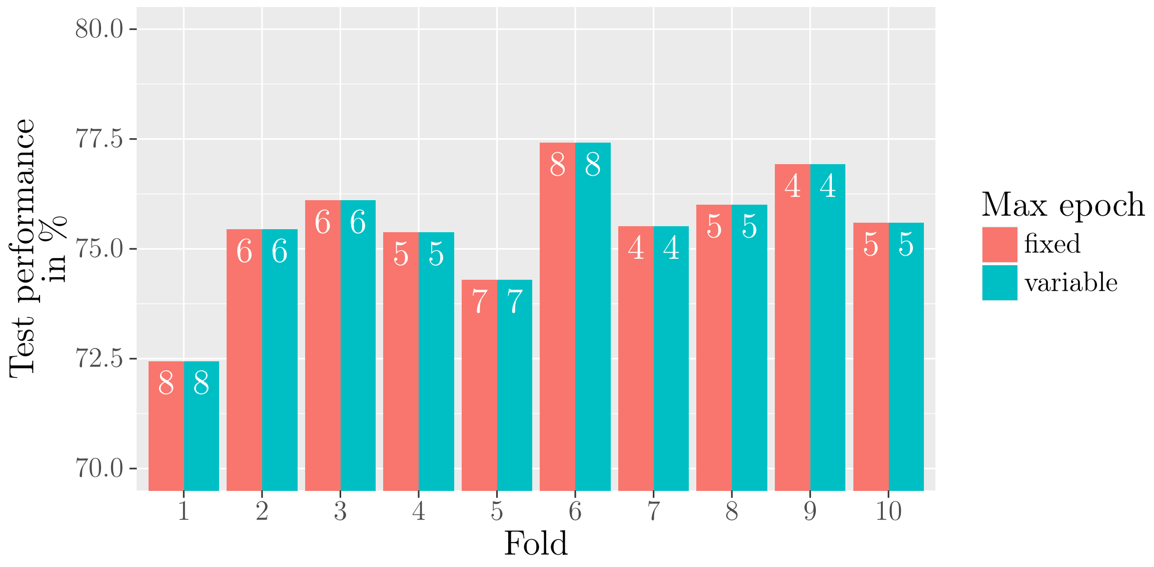
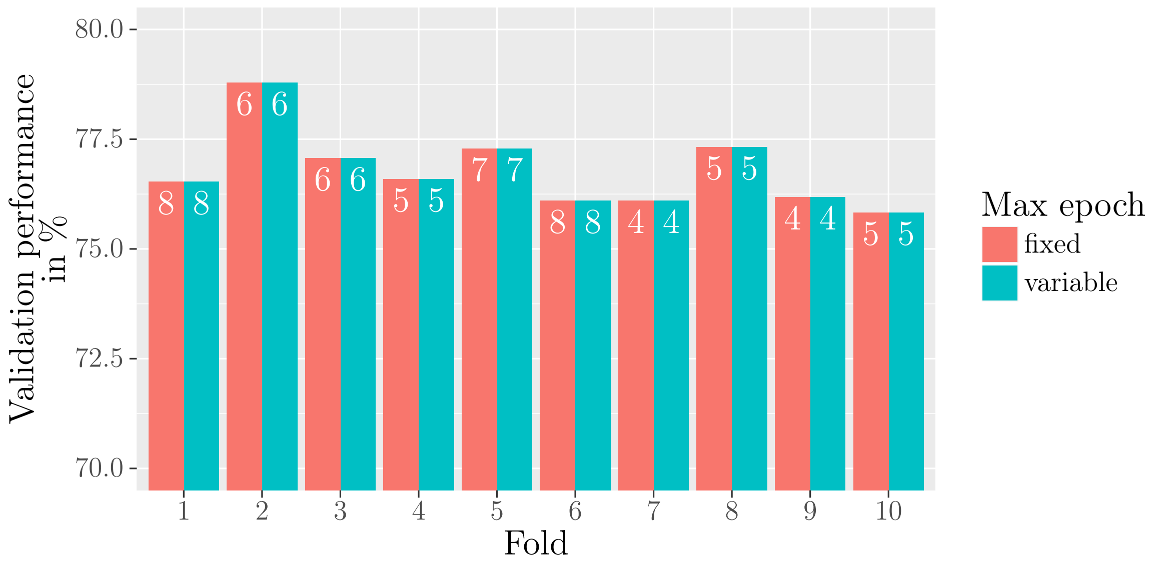


Figure 2.5: LSTM model performances of train, validation and test sets over 50 epochs in the sixth fold.

Figure 2.5 exemplary illustrates the the course of the train, validation and test performance for the LSTM model training over 50 epochs in the sixth fold, which showed the highest test accuracy. Within the first ten epochs the train performance reaches 100%. The validation and test performances increase equally strong, but only up to slightly over



(a) LSTM test set classification performance over ten folds.



(b) LSTM validation set classification performance over ten folds.

Figure 2.6: LSTM performance on MR: Comparing the best models chosen among a fixed set of 50 epochs, respectively variable set of epochs. The white labels mark the epoch, where the final model was achieved.

75%. For a few epochs the accuracy remains roughly the same, then, it slowly decreases again. These examplary performance runs clearify, why using the global selection gives the same models as the patience criterion. The best validation and test performances are achieve within the first roughly 15 epochs.

Furthermore, the information displayed in Table 2.7 is also illustrated using the diagrams provided in Figure 2.6. The average validation performance of the LSTM amounts to 76.78%, the test accuracy to 75.51%. Thereby, the standard deviation for the validation set amounts to 0.83% and for the test set to 1.31%.

2.2.5 Evaluation

The CNN and LSTM architectures have been investigated under adjusted conditions. For both network types basic, lightweight architectures have been chosen in order to draw a comprehensive comparison on the architectural typologies themselves. The networks have been trained in a 10 fold setting on the Sentence Polarity Dataset. The input was presented to both architectures in form of randomly initialized 300-dimensional word embeddings; each network was trained up to 50 epochs. A main difference was the used vocabulary. While the CNN considered all words, the LSTM vocabulary was restricted to 10,000 words with regard to the grid search results, which showed that the LSTM performed better on a smaller vocabulary.
Given the fact, that no pre-trained word vectors or other components that knowingly boost classification performances have been applied, no state-of-the-art accuracy has been achieved.

Both kinds of networks perform approximately equally well on the given binary sentiment classification task. Table 2.8 summarizes the averages of several performance parameters for the LSTM and the CNN per fold, based on the earlier conducted 10 fold training. The differences for the validation performance are rather small. For the global selection the CNN model performs marginally better, for the patience criterion the LSTM. With regard to the test performances, the CNNs average accuracy is slightly higher.

Architecture	Max epoch	Epochs	Valid	Test
CNN	fixed	50	76.83	76.51
CNN	variable	34	76.77	76.24
LSTM	fixed	50	76.78	75.51
LSTM	variable	24	76.78	75.51

Table 2.8: Comparing average performances over all folds

As Figure 2.4 and 2.5 examplary illustrated, the validation and test performance of the CNNs and the LSTM proceeded differently over the course of 50 epochs. These different courses of the performance values relate to the differences discovered in the selecting the best model. Here, two methods have been tested: global selection and local selection based on the patience criterion. Both methods resulted in the same selected models for the LSTM network (see Figure 2.6b). For the CNN in four out of ten folds better models (in terms of validation performance) have been elected using the global approach.

Thus, for LSTM models the patience criterion is a useful tool to reduce the training time without loosing the chance of models with better validation performance. In fact, applying the patience criterion on the LSTM training would have halved the training time in terms of epochs in the experiments above, hence the average training time per fold. For CNNs, on the other hand, it makes more sense to apply the global selection considering the course of the validation performance during training.

Based on the elementary architectures and the performances measured during this experiment, it is not possible, to decide on the superiority of one architecture over another. Both networks performed equally. Yet, regarding the training time the CNNs hold the clear advantage of parallelization. Their training per fold took a long time on a CPU (hours), but only a few minutes using a GPU. On the CPU the training of the LSTM could be conducted faster than the CNNs. This was not the case for GPUs and can not even be compensated by using the patience criterion.

Nowadays, researchers have two consider two challenges in processing data: size and speed. Big Data is one of the recently growing research challenges. In order to efficiently process large amounts of data, the processing speed is a key factor. In this context the CNN in combination with GPU training is clearly the better choice to satisfy the given need for fast processing.

2.3 Future Work

Future investigations should consider various other datasets. Valuable insights could be gained by considering data that does not belong to the movie domain, by considering more sentiment classes, or a dataset which is not in English. In that way, the comparison of the plain LSTM and CNN architectures could be elaborated.

The influence of pre-trained word embeddings on CNNs was already investigated by Kim [12], but has not been explored for the basic LSTM architecture. In this context, the comparison and interchanging of the trained word embeddings might lead to interesting conclusions about likewise learned aspects of sentiments.

Although, the inclusion of peepholes did not seem to contribute to improving the LSTMs classification performances, it might still be interesting to investigate their effect in more detail, for example by only adding a single peephole or two. Further grid searches might help to adjust parameters and analyse the potential performances.
Also, additional experiments on the CNNs parameters like its filters might yield interesting insights.

3. Sentiments in Ranking Tasks

This chapter will introduce a new perspective on sentiments with respect to the ranking task in IR. In Section 3.1, the basic problem definition of ranking in IR and some approaches to solving this task will be discussed. Subsequently, in Section 3.2, the idea of using sentiment information as a feature in ranking is presented and finally emphasized by a small scale ranking experiment, which will be explained and evaluated in detail in Section 3.3. The following content has been developed in collaboration with Hamed Zamani and Bruce Croft (CIIR, UMass Amherst).

3.1 Ranking in Information Retrieval

Search engines are an omnipresent application for the core problem in Information Retrieval (IR): ranking. In search for specific information a search engine is the tool of choice to satisfy ones information need. Based on the given query, the ranking algorithm, which the search engine is build upon, produces a list of ranked documents matching the query. The main concept involved in the ranking procedure is *relevance*, which denotes that the retrieved results should likely be relevant to the searching person (see Croft et al. [4, pp. 1-11]).
Much research is focused on constructing ranking models using learning-to-rank approaches, which have been developed in the context of Machine Learning. Following the ML methodology, documents are represented as feature vectors and used for discriminative training (see Liu [17, pp. 225-246]). In general learning-to-rank methods are sub-divided into three kinds of approaches: pointwise, pairwise and listwise approaches. In pointwise approaches (see Liu [17, Chapter 3]), a ranking model aims at the prediction of the correct relevance degree of a document, while pairwise approaches (see Liu [17, Chapter 4]) focus on the relative order of two documents. Listwise approaches (see Liu [17, Chapter 5]) either optimize with regard to a common evaluation measure or minimize listwise ranking losses. During the forthcoming experiments a ranking SVM, which is considered a pairwise learning-to-rank approach, will be employed (see Liu [17, pp. 262-263]).

3.2 Sentiments in Information Retrieval

Since the processing and analysis of textual data is fundamental for many IR tasks, the concept of sentiments posed an important part in some previous IR research. In the following a few examples of research incorporating sentiments into IR tasks will be given.

Huang and Croft [8] developed an *opinion relevance model*, a framework to represent an infomation need fo opinion retrieval. This model represented a unified model of topic relevance and sentiments. It resulted from extending a relevance model approach by incooperating the additional information needs regarding opinion retrieval. Sentiments were included by query expansion and different approaches have been tested: query-independent and query-dependent sentiment expansions, as well as a mixture relevance model, which associated the preceeding approaches. By applying the opinion relevance model the results of several opinion retrieval experiments could be improved significantly.

Aktolga and Allan [1] applied sentiments in the context of diversification, which aimed at improving topical variety in search results. Their main focus was on the diversification of opinionated content, for which they presented two different models. By using sentiments and an explicit bias they were able to emphasize search results in three ways: equal diversification, meaning that the representation of all sentiments on a topic is balanced, and diversification towards, respectively against a topic sentiment. Conducted experiments showed, that these models improved the variety in search results significantly for minority sentiments by applying a corresponding bias for diversification.

3.3 Sentiments in Ranking

Over the course of several experiments the question whether sentiment information can improve the ranking performance will be discussed. Sentiment signals already proved to be helpful on some IR tasks, as shown in Chapter 3.2, but have not been applied in the context of document ranking. In the following sections the setup and execution of a series of experiments concerned with different sentiment signals will be described. A subsequent evaluation will not only be concerned with the question if sentiments improve ranking, but also how and why.

3.3.1 Experimental Setup

As discussed in Chapter 3.1, document ranking aims - like many IR tasks - at providing relevant information to a given query. In order to investigate the potential improvements gained by the application of sentiment information in ranking, a series of experiments has been conducted. In the following, the basic structure of the experiments will be explained in more detail starting with the general course of the experiments. Subsequently, the used tools and data will be discussed as well as the chosen features and performance measures.

Procedure

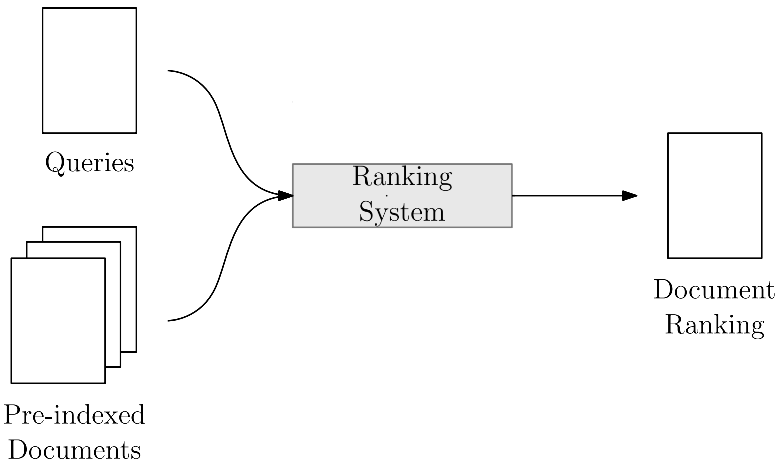


Figure 3.1: Given a set of queries and a pre-indexed corpus of documents, the ranking system generates ranked lists of documents for each query.

The basic experimental setup to test whether sentiment information can improve the performance in the context of a ranking task consists of two main steps: the generation of a baseline ranking, and a re-ranking based on selected document features. The baseline ranking is generated using an established ranking system, meaning that an existing ranking algorithm is employed in order to retrieve a number of potentially relevant documents from a given corpus for each query in a given query set, as depicted in Figure 3.1. These initial rankings of documents serve as a baseline and state a reference and minimum requirement in the exploration of new features.

Then, as sketched in Figure 3.2, a set of features based on text statistics and in this case also sentiment information is extracted for each document listed in the initial ranking. Those features in combination with manually constructed relevance judgements as labels build the foundation to train a ranking model that will rank documents by relevance given their features. Depending on the choice of features and the corpus, this procedure

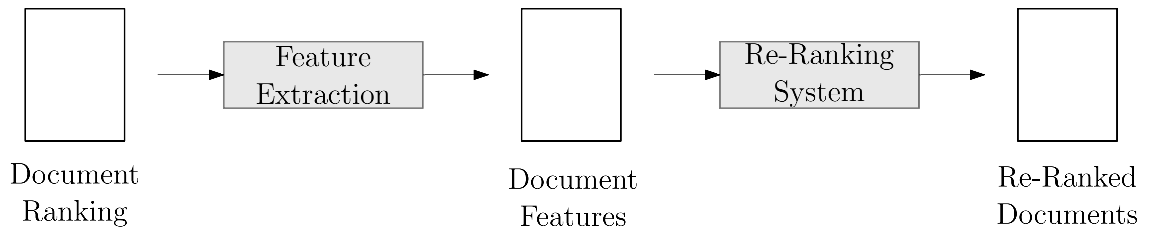


Figure 3.2: Re-ranking procedure

ideally leads to a new ranking model that exceeds the baseline results. The performance of the re-ranking is evaluated using common ranking heuristics like Mean Average Precision (MAP) or Normalized Discounted Cumulative Gain at Position 20 (NDCG@20)

Tools

Realizing the previously described experiment procedure involves several different tools. For once, an established ranking system is needed in order to generate a baseline ranking. Therefore, the Java-based search engine Galago[1] was used, which was developed by Croft et al. [4]. Utilizing the flexibility provided by the Galago search engine, the probablilistic BM25 ranking algorithm (see Croft et al. [4, pp. 250–252]) was used to retrieve 1000 documents per query. For each of those documents, in turn, several features have been computed. Those features were used to train several SVM$^{\text{rank}}$ [2] models, whereas the margin C was fixed to $C = 35$. Then, the trained SVM$^{\text{rank}}$ models were used to re-rank the test data. Finally, the resulting rankings were evaluated using the official Trec evaluation tool `trec_eval`[3]. Depending on the corpus, different heuristics have been selected to assess ranking performances.

Corpora

The experiments have been run on two different corpora: the .GOV2 collection[4] and the ClueWeb09 dataset[5]. The former comes along with 150 different queries and binary relevance judgements, the latter with 200 queries and relevance judgements ranging from -2 and 0 to 4. -2 and 0 mark non-relevant documents with regard to a specific query, whereas a relevance degree of -2 marks documents that seem to be spam or junk. Judgements ranging from 1 to 4 refer to different relevance degrees: a higher judgement value denotes a higher relevance to a given query (see Clarke et al. [2, page 3]).

[1]http://www.lemurproject.org/galago.php
[2]https://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html
[3]http://trec.nist.gov/trec_eval/
[4]http://ir.dcs.gla.ac.uk/test_collections/gov2-summary.htm
[5]http://lemurproject.org/clueweb09/

Performance Measures

As mentioned before, the experiments will be run on two corpora: the .GOV2 corpus and the ClueWeb09 corpus. For each of the corpora a different IR evaluation measure will be applied, due to their different relevance degrees. The ranking performance of the .GOV2 corpus with its binary relevance judgements will be evaluated by the Mean Average Precision (MAP), while the Normalized Discounted Cumulative Gain at Position 20 (NDCG@20) will be employed for the ClueWeb09 corpus. The differences in performance will be analyzed with regard to their statistical significance by a two-sided, paired T-test.

Basic Features

Feature	Formula
1	$\sum_{w_i \in q \cap d} ln(tf(w_i, d) + 1)$
2	$\sum_{w_i \in q \cap d} ln(\frac{\lvert C \rvert}{tf(w_i, C)} + 1)$
3	$\sum_{w_i \in q \cap d} ln(idf(w_i))$
4	$\sum_{w_i \in q \cap d} ln(\frac{tf(w_i, d)}{\lvert d \rvert} + 1)$
5	$\sum_{w_i \in q \cap d} ln(\frac{tf(w_i, d)}{\lvert d \rvert} \cdot idf(w_i) + 1)$
6	$\sum_{w_i \in q \cap d} ln(\frac{tf(w_i, d) \cdot \lvert C \rvert}{\lvert d \rvert \cdot tf(w_i, C)} + 1)$
7	$ln(BM25\ score)$

Table 3.1: Document features, as proposed by Xu and Li [26], calculated for a document d in the document collection C with respect to query q, consisting of the terms $w_1, ..., w_i, ..., w_n$, whereas n denotes the number of terms in q. In this context $tf(w, d)$ and $tf(w, C)$ refer to the term frequency of a word w in a document d and the collection C, respectively, while $|d|$ and $|C|$ denote their respective sizes (number of terms and documents, respectively). $idf(w)$ refers to the inverse document frequency of a word w.

The features proposed by Xu and Li [26] (see Table 3.1) were chosen as a base feature set. Using those features in a learning-to-rank approach Xu and Li [26] achieved significant improvements in ranking performances on several corpora. Thus, this feature set builds a solid foundation for further feature explorations. Applying the features on the .GOV2 and the ClueWeb09 corpus, the findings of Xu and Li [26] could be verified. The corresponding ranking performances and significance tests are documented in Section 3.3.2.

Sentiment Features

Feature	Formula
8	$count_{d,pos}$ / $\|d\|$
9	$count_{d,neg}$ / $\|d\|$

Table 3.2: Simple count based sentiment features calculated for a document d in the document collection C with respect to query q. The variables $count_{pos}$ and $count_{neg}$ refer to the counted number of positive and negative terms in d according to the Opinion Lexicon. $|d|$ denotes the length of the document.

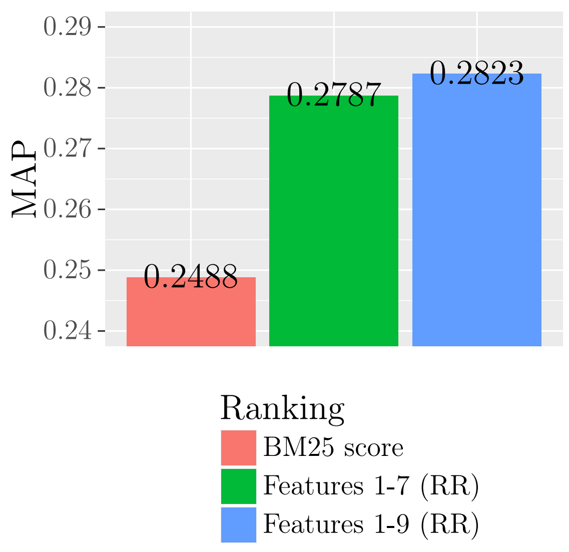
Feature	Formula
10 and 14	$count_{N,d,pos}$ / $\|d\|_s$
11 and 15	$count_{N,d,pos}$ / $\|d\|_s$
12 and 16	$countTerms_{N,d,pos}$ / $\|d\|$
13 and 17	$countTerms_{N,d,pos}$ / $\|d\|$

Table 3.3: Sentiment features based on binary sentiment prediction are calculated for a document d in the document collection C with respect to a query q. The variables $count_{N,d,pos}$ and $count_{N,d,neg}$ refer to the number of positively and negatively N-judged sentences within a d. $countTerms_{N,d,pos}$ and $countTerms_{N,d,neg}$ on the other hand denote the number of terms contained in positively and negatively N-judged sentences. $|d|$ describes the length of the document in by means of terms, while $|d|_s$ is the length of the document by means of sentences. For features 10-13 N refers to the judgements made by the LSTM network; for features 14-17 it denotes CNN-judgements.

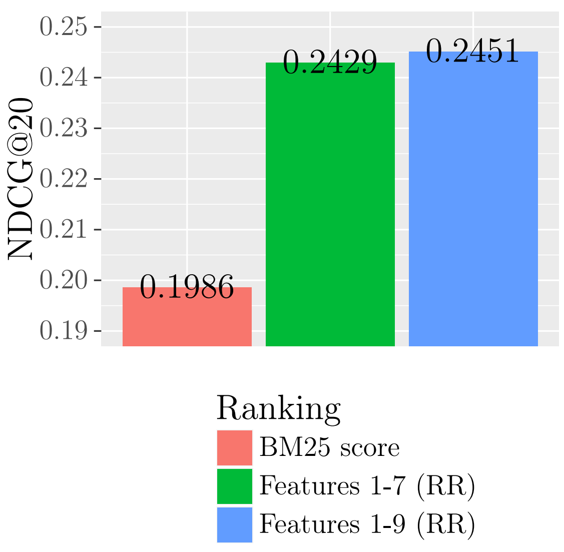
In a first test run, a simple sentiment signal resulting in two additional features was used. The features were given by the number of positive, respectively negative words occurring in a document, normalized by the document length (see table 3.2). The word counting was done based on the positive and negative terms listed in the Opinion Lexicon[6], which was introduced by Hu and Liu [7] and Lui et al. [16].
Later on, more sophisticated sentiment signals have been considered. Therefore, the previously mentioned CNN and LSTM have been applied on the copora. For each document the sentence-wise preditions have been summarized as described in Table 3.3.
It is important to note, that the LSTM features, are based on the original LSTM structure and training data, as used in the Deeplearning Tutorial[7].

3.3.2 Experiments

Baseline Ranking Performances



(a) Ranking performance for .GOV2 measured by MAP



(b) Ranking performance for ClueWeb09 measured by NDCG@20

Figure 3.3: Comparison of initial ranking with BM25 score, re-ranking with Dirichlet score, re-ranking based on Features 1-7, and Features 1-9

[6]http://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html
[7]http://deeplearning.net/tutorial/lstm.html

In order to assess the basic ranking performances and potential approvements gained by sentiment information, the initial ranking achieved by Galago using the BM25 score was compared to rankings attained through re-ranking by the Features 1-7, as well as Features 1-9 (see Table 3.1 and 3.2). As shown in Figure 3.3 the baseline BM25 ranking performs worse than the re-ranking by Features 1-7 for both corpora. A paired, two-sided T-test also confirmed a statistical significance difference in both ranking performances, which supports the findings of Xu and Li [26].
Extending the feature set consisting of Features 1-7 by sentiment information (Features 8 and 9) improves the ranking performance for both corpora even further. Yet, a statistical significance could only be confirmed for the ranking of the .GOV2 corpus.
The precise T-test results for performance differences between BM25 score, Features 1-7 and Features 1-9 for both corpora are listed in Table 3.4.

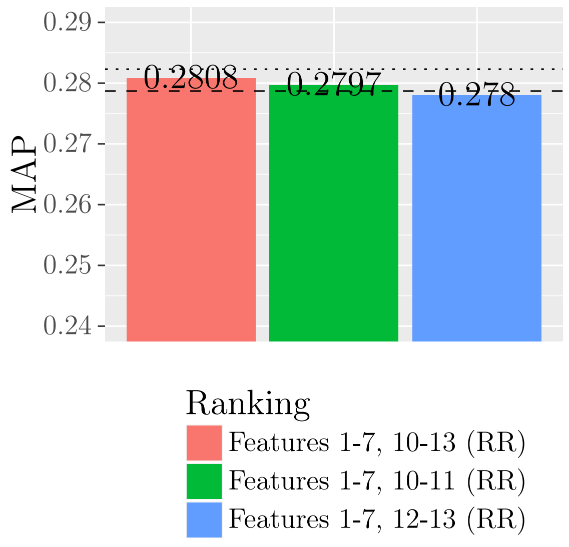
Corpus	Ranking 1	Ranking 2	H_0 rejected	P-value
.GOV2	BM25 score	Features 1-7	yes	1.405×10^{-8}
.GOV2	BM25 score	Features 1-9	yes	4.564×10^{-9}
.GOV2	Features 1-7	Features 1-9	yes	1.925×10^{-2}
ClueWeb09	BM25 score	Features 1-7	yes	9.477×10^{-5}
ClueWeb09	BM25 score	Features 1-9	yes	8.161×10^{-5}
ClueWeb09	Features 1-7	Features 1-9	no	4.285×10^{-1}

Table 3.4: T-test results: Analyzing statistical significance with regard to differences in ranking performances using BM25 score, Features 1-7 and Features 1-9.

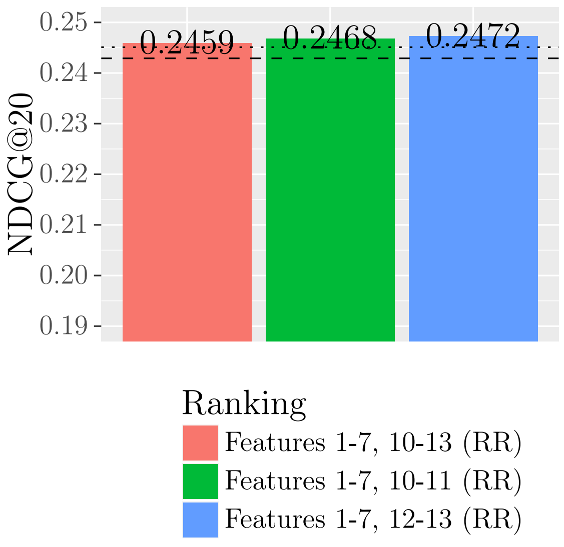
Based on these insights on the potential improvement lying in sentiment information, more sophisticated sentiment signals in form of Features 10-16 (see Table 3.3) have been tested. In the next section, the ranking performance with respect to different sentiment feature combinations will be investigated for the .GOV2 corpus and the ClueWeb09 corpus in more detail. Thereby, the ranking performances achieved using Features 1-7 and Features 1-9 serve as two baselines. *Baseline 1* (Features 1-7) will be marked in graphs by a dashed black line, *Baseline 2* (Features 1-9) by a dotted black line.

CNN- and LSTM-based Sentiment Features

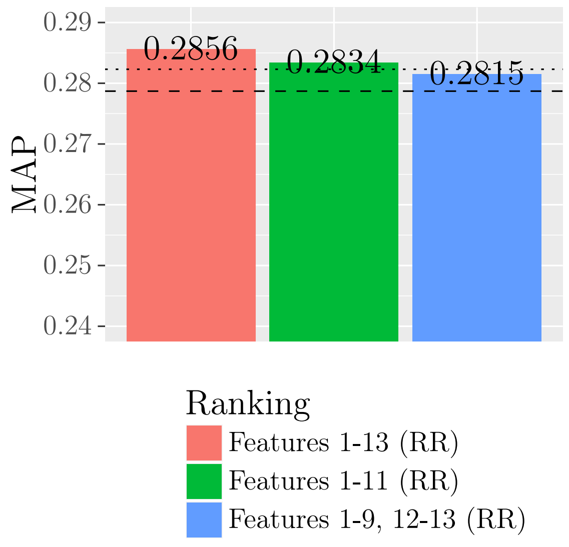
At first, CNN, respectively LSTM sentiment features and their combinations with the simple count based sentiment features have been investigated. The resulting ranking results measured by MAP, respectively NDCG@20 are visualized in Figure 3.4 and 3.5. In a second step combinations of CNN and LSTM features have been used for ranking. The results are shown in Figure 3.6. For the CNN, the non-static-Word2Vec model was used, due to its superior accuracy.



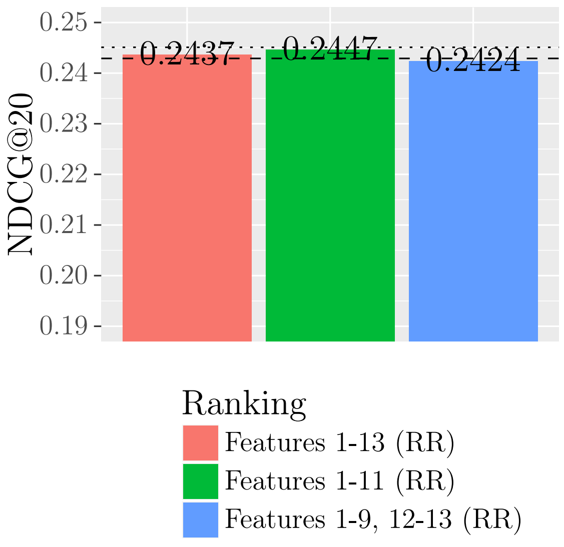
(a) Ranking performance for .GOV2 measured by MAP; Sentiment features 10-13



(b) Ranking performance for ClueWeb09 measured by NDCG@20; Sentiment features 10-13

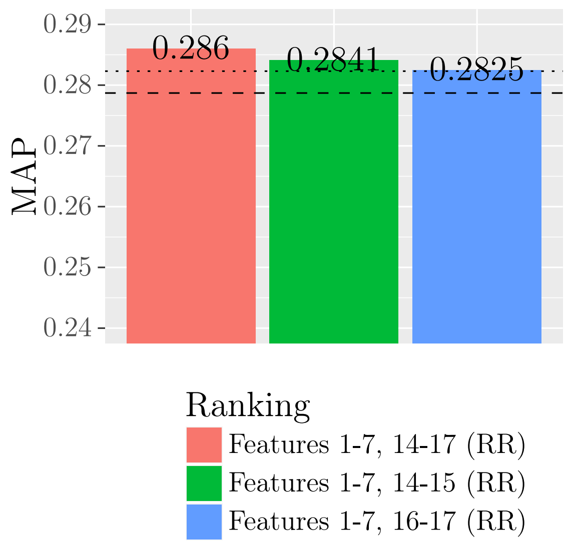


(c) Ranking performance for .GOV2 measured by MAP; Sentiment features 8-13

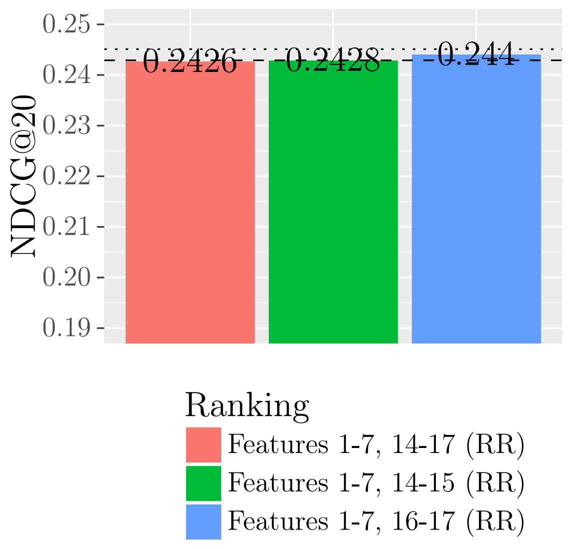


(d) Ranking performance for ClueWeb09 measured by NDCG@20; Sentiment features 8-13

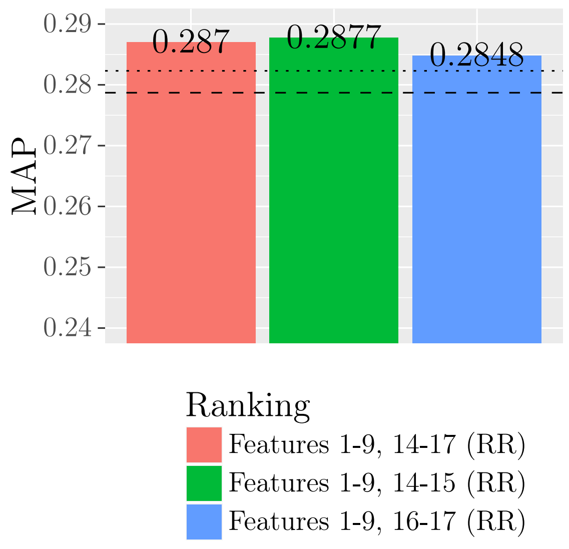
Figure 3.4: Exploring CNN sentiment features: Comparison of re-ranking performance regarding different sentiment feature combinations. The performance of Baseline 1 (Features 1-7) is marked by a dashed line. The performance of Baseline 2 (Features 1-9) is marked by a dotted line.



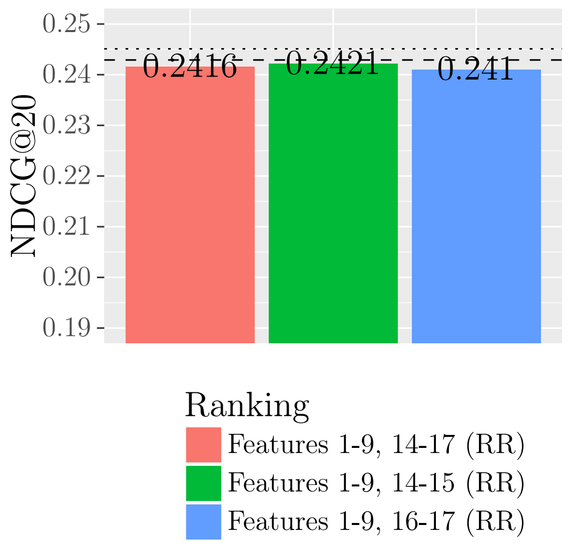
(a) Ranking performance for .GOV2 measured by MAP; Sentiment features 14-17



(b) Ranking performance for ClueWeb09 measured by NDCG@20; Sentiment features 14-17

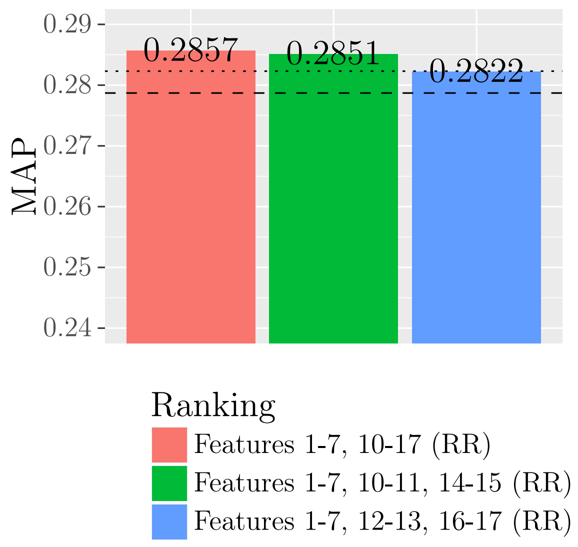


(c) Ranking performance for .GOV2 measured by MAP; Sentiment features 8-9 & 14-17

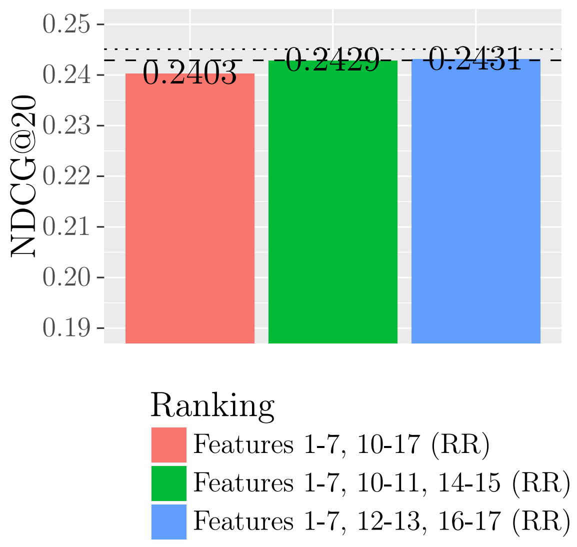


(d) Ranking performance for ClueWeb09 measured by NDCG@20; Sentiment features 8-9 & 14-17

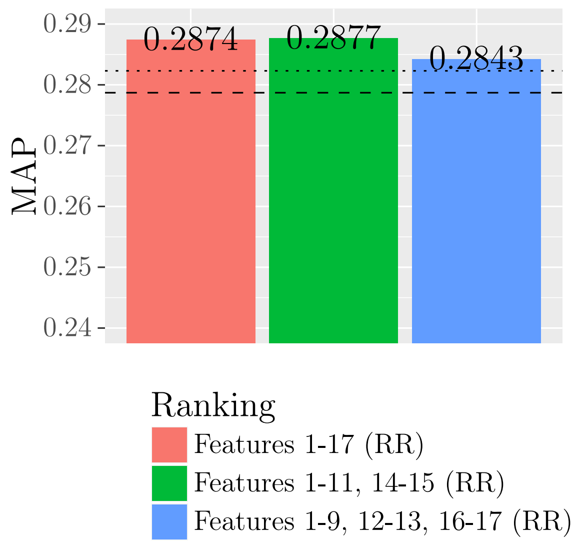
Figure 3.5: Exploring LSTM sentiment features: Comparison of re-ranking performance regarding different sentiment feature combinations. The performance of Baseline 1 (Features 1-7) is marked by a dashed line. The performance of Baseline 2 (Features 1-9) is marked by a dotted line.



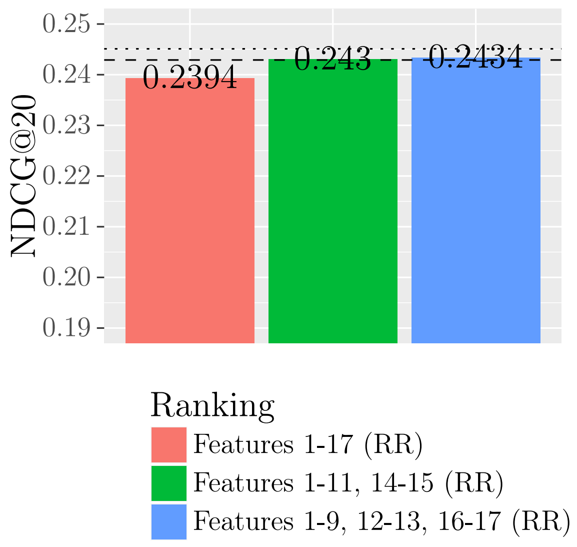
(a) Ranking performance for .GOV2 measured by MAP; Sentiment features 10-17



(b) Ranking performance for ClueWeb09 measured by NDCG@20; Sentiment features 10-17



(c) Ranking performance for .GOV2 measured by MAP; Sentiment features 8-17



(d) Ranking performance for ClueWeb09 measured by NDCG@20; Sentiment features 8-17

Figure 3.6: Exploring LSTM and CNN sentiment features: Comparison of re-ranking performance regarding different sentiment feature combinations. The performance of Baseline 1 (Features 1-7) is marked by a dashed line. The performance of Baseline 2 (Features 1-9) is marked by a dotted line.

For the .GOV2 corpus the ranking performance using LSTM sentiment features excels the performance of Baseline 2, as Figure 3.5a shows. An even better performance can be accomplished by combining LSTM- and count based sentiment features (see Figure 3.5c).
CNN sentiment features alone on the other hand can not outperform the Baseline 2 (see Figure 3.4a). Only by combining CNN and count based sentiment features an improvement over Baseline 2 can be accomplished (see 3.4c).
Combining LSTM and CNN features outperforms the Basline 2 performance in any, but the term-based case. Combinations of LSTM, respectively CNN and count based sentiment features in the context of .GOV2 lead mostly to better performances than Baseline 2. Also, term-based LSTM and CNN features always perform worse than their corresponding sentence-based variants.
The best performance for .GOV2 with MAP = 0.2877 can be achieved by using features 1-9 and 14-15, meaning it considers the simple count based sentiment features, as well as the sentence-based LSTM sentiment features. Adding the sentence-based CNN sentiment features to this combination, leads to the same performance. Anyway, the improvements in terms of ranking performance can be considered statistically significant with regard to Baseline 1 and 2 (see Table 3.5).

With regard to the ClueWeb09 corpus the different kinds of sentiment features lead to different outcomes. Contrary to .GOV2 the ranking performances for many of the sentiment feature combinations perform rather poorly. The combination of count based and CNN, respectively LSTM features leads to worse performances compared to only using CNN or LSTM sentiment features (see Figure 3.4b and 3.4d, respectively Figure 3.5b and 3.5d). In many cases there is not even improvement over Baseline 1. Among the more sophisticated sentiment features, the use of CNN features leads to better performances, than the use of LSTM features. By applying only CNN sentiment features small improvements over Baseline 2 can be achieved.
For the ClueWeb09 corpus the application of term-based CNN sentiment features gives the best performance with NDCG@20 = 0.2472, which proves to be a significant difference compared to Basline 1, but not Baseline 2 (see Table 3.5).

Corpus	Ranking 1	Ranking 2	H_0 rejected	P-value
.GOV2	Features 1-7	Features 1-9,14-15	yes	1.803×10^{-5}
.GOV2	Features 1-9	Features 1-9,14-15	yes	1.491×10^{-5}
.GOV2	Features 1-7	Features 1-11,14-15	yes	1.793×10^{-5}
.GOV2	Features 1-9	Features 1-11,14-15	yes	1.588×10^{-5}
ClueWeb09	Features 1-7	Features 1-7,12-13	yes	2.748×10^{-2}
ClueWeb09	Features 1-9	Features 1-7,12-13	no	4.210×10^{-1}

Table 3.5: T-test results: Analyzing statistical significance with regard to differences in ranking performances by applying CNN and LSTM sentiment features.

3.3.3 Discussion/Evaluation

Starting from very simple, count based sentiment signals and later on, more sophisticated sentiment signals using CNN and LSTM sentiment predictions, the effect of sentiment information in ranking tasks has been explored.

The conducted experiments showed, that even simple sentiment signals, in form of counting positively or negatively annotated words within documents, improve the ranking performance of Baseline 1 for the .GOV2 corpus significantly. In case of the ClueWeb09 corpus there was also an improvement, but no significant one.

Since the use of simple, most likely semi-accurate sentiment signals proved to be helpful already, NNs have been employed in order to gain more sophisticated sentiment signals. Based on the binary sentence-level sentiment predictions provided by a CNN and LSTM, which have been trained on movie reviews, more sentiment features have been generated.

For both corpora, the effect of NN based sentiment signals is quite different.
The sophisticated NN features show very promising results with regard to the .GOV2 corpus. LSTM features work well for the .GOV2 corpus and perform better than the simple sentiments only. Also, combinations of LSTM and CNN features perform well.
However, this is not the case for the ClueWeb09 corpus. Most of the feature combinations incorporating sophisticated sentiment signals cannot outperform the baseline set by the simple sentiment features. Often not even the baseline given by the basic features without sentiment signals can be matched. CNN based features performed best.

3.4 Future Work

Reviewing the presented data, it gets clear, that sentiment information can help improving the performance in ranking tasks, but it also raises some questions. Why exactly does sentiment information help in ranking tasks and how? Which queries perform especially well with sentiment features? Which queries do not?

First investigations on how sentiments work in ranking, revealed no common topic among good or bad performing queries. Yet, first feature analysis' revealed clear differences in the feature distributions of their matching documents, which still have to be investigated in more detail.

The key of improving ranking performances further, is to understand how and why sentiments help in ranking tasks. Therefore, follow-up investigations should focus on the analysis of feature distributions. A thorough investigation on the differences in distributions between top and bottom ranked documents might provide further insights.

Also, the exploration of alternative sentiment feature definitions and differently trained NN models has to be considered. Movie reviews are most certainly not the optimal choice for training NNs, that will be applied on general, most likely non-movie-related texts.

Referring to Chapter 2.2, the sentiment signals and their applications might also be of value in the comparing of LSTM and CNN performances. A thorough investigation on the annotation differences on both corpora might give valuable insights on learned contents of both networks. In order to estimate the accuracy and applicability of the trained networks, a qualitative evaluation of the sentiment predicitons of the documents should be conducted. The correlation between the sentiment predictions of both networks could also give additional hints about the commonly learned contents.

4. Final Conclusion

This report summarized the work and research conducted on CNNs in NLP during a five month stay at the Center of Intelligent Information Retrieval at the University of Massachusetts, Amherst. In particular the perfomance of CNNs has been compared to LSTMs in the field of SA. Also, the effect of sentiment signals in ranking has been analyzed.

A thorough investigation of the state-of-the-art with regard to CNNs and RNNs in the field of SA revealed two mainly pursued strategies for improving performances: the exploration of variations in basic network architectures and the search for supplementary compontents. In order to keep the comparison of CNNs and RNNs purely dependent on the architectural performances, two plain and elementary network structures have been selected as representatives - one CNN and one LSTM architecture.
Using these selected lightweight architectures a comprehensive comparison has been drawn. Considering the classification performance, the CNN and LSTM architectures performed equally. Differences have been detected with regard to the performance runs over the course of training. In this context the patience criterion proved to be a valuable tool for reducing the LSTMs training time to the actual necessary training time.
In light of nowadays computational challenges (big data and processing speed), CNNs proved to be at a clear advantage, due to the short training time achieved by parallelized computations on GPUs.

Later on, the potential of sentiment signals in the IR task of ranking has been investigated. Among other heuristics, the sentiment signals based on binary CNN and LSTM sentiment predictions have been used as features for ranking documents. Experiments testing different feature combinations on two corpora, the .GOV2 corpus and the ClueWeb09 corpus, showed promising results. Especially, for the .GOV2 corpus several feature combinations lead to significant improvements over the chosen baselines. With regard to both corpora no best performing feature set could be determined.
These initial results, raised several questions, that need to be answered over the course of future analysis' and experiments.

List of Figures

List of Tables

List of Abbreviations

CNN	Convolutional Neural Network	2
DCNN	Dynamic Convolutional Neural Network	4
IR	Information Retrieval	2
LSTM	Long Short-Term Memory	3
MAP	Mean Average Precision	25
ML	Machine Learning	2
mLSTM	multiplicative Long Short-Term Memory	4
NDCG@20	Normalized Discounted Cumulative Gain at Position 20	25
NLP	Natural Language Processing	1
NN	Neural Network	2
ReLU	Rectified Linear Unit	9
ReNN	Recursive Neural Network	4
RNN	Recurrent Neural Network	2
RNTN	Recursive Neural Tensor Network	4
SA	Sentiment Analysis	1
SVM	Support Vector Machine	4
VDCNN	Very Deep Convolutional Neural Network	6

Bibliography

[1] Elif Aktolga and James Allan. Sentiment diversification with different biases. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '13, pages 593–602, New York, NY, USA, 2013. ACM.

[2] Charles L. A. Clarke, Nick Craswell, and Ellen M. Voorhees. Overview of the TREC 2012 web track. In *Proceedings of The Twenty-First Text REtrieval Conference, TREC 2012, Gaithersburg, Maryland, USA, November 6-9, 2012*, 2012.

[3] Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann LeCun. Very deep convolutional networks for natural language processing. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1107–1116, Valencia, Spain, April 2017. Association for Computational Linguistics.

[4] Bruce Croft, Donald Metzler, and Trevor Strohman. *Search Engines: Information Retrieval in Practice.* Addison-Wesley Publishing Company, USA, 1st edition, 2009.

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

[6] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12), 2009.

[7] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 168–177, New York, NY, USA, 2004. ACM.

[8] Xuanjing Huang and W. Bruce Croft. A unified relevance model for opinion retrieval. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 947–956, New York, NY, USA, 2009. ACM.

[9] Rie Johnson and Tong Zhang. Effective use of word order for text categorization with convolutional neural networks. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 103–112, Denver, Colorado, May–June 2015. Association for Computational Linguistics.

[10] Rie Johnson and Tong Zhang. Semi-supervised convolutional neural networks for text categorization via region embedding. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 919–927. Curran Associates, Inc., 2015.

[11] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[12] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October 2014. Association for Computational Linguistics.

[13] Ben Krause, Iain Murray, Steve Renals, and Liang Lu. Multiplicative LSTM for sequence modelling. *ICLR Workshop track*, 2017.

[14] Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. Ask me anything: Dynamic memory networks for natural language processing. *CoRR*, abs/1506.07285, 2015.

[15] Phong Le and Willem Zuidema. Compositional distributional semantics with long short term memory. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, pages 10–19, Denver, Colorado, June 2015. Association for Computational Linguistics.

[16] Bing Liu, Minqing Hu, and Junsheng Cheng. Opinion observer: Analyzing and comparing opinions on the web. In *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, pages 342–351, New York, NY, USA, 2005. ACM.

[17] Tie-Yan Liu. Learning to rank for information retrieval. *Found. Trends Inf. Retr.*, 3(3):225–331, March 2009.

[18] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[19] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of ACL*, pages 115–124, 2005.

[20] A. Radford, R. Jozefowicz, and I. Sutskever. Learning to Generate Reviews and Discovering Sentiment. *ArXiv e-prints*, April 2017.

[21] Aliaksei Severyn and Alessandro Moschitti. Twitter sentiment analysis with deep convolutional neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 959–962, New York, NY, USA, 2015. ACM.

[22] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

[23] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China, July 2015. Association for Computational Linguistics.

[24] Duyu Tang, Bing Qin, and Ting Liu. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1422–1432, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[25] Xin Wang, Yuanchao Liu, Chengjie SUN, Baoxun Wang, and Xiaolong Wang. Predicting polarities of tweets by composing word embeddings with long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1343–1353, Beijing, China, July 2015. Association for Computational Linguistics.

[26] Jun Xu and Hang Li. Adarank: A boosting algorithm for information retrieval. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 391–398, New York, NY, USA, 2007. ACM.

[27] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, NIPS'15, pages 649–657, Cambridge, MA, USA, 2015. MIT Press.

[28] Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. Long short-term memory over recursive structures. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 1604–1612. JMLR.org, 2015.