Inspiring Excellence

Aspect-Based Sentiment Analysis Using SemEval and Amazon Datasets

---

By:

Tamanna Hasib

ID: 17141017

Saima Ahmed Rahin

ID: 13301117

Advisor:

Mr. Moin Mostakim

BRAC University

Department of Computer Science and Engineering

Declaration

This is to certify that the research work titled "Aspect-Based Sentiment Analysis Using SemEval and Amazon Datasets" is submitted by Saima Ahmed Rahin and Tamanna Hasib to the Department of Computer Science and Engineering, BRAC University in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Engineering. We hereby declare that this thesis is based on results found from our own work. Materials of work found by other researcher are mentioned by reference. This Thesis, neither in whole or in part, has been previously submitted for any degree. We carried our research under the supervision of Mr. Moin Mostakim.

Supervisor's signature

_____

Mr. Moin Mostakim

Author's signature

_____

Saima Ahmed Rahin

Author's signature

_____

Tamanna Hasib

Abstract

Sentiment analysis has become one of the most important tools in natural language processing, since it opens many possibilities to understand people's opinions on different topics. Aspect-based sentiment analysis aims to take this a step further and find out, what exactly someone is talking about, and if he likes or dislikes it. Real world examples of perfect areas for this topic are the millions of available customer reviews in online shops.

There have been multiple approaches to tackle this problem, using machine learning, deep learning and neural networks. However, currently the number of labelled reviews for training classifiers is very small.

Therefore, we undertook multiple steps to research ways of improving ABSA performance on small datasets, by comparing recurrent and feed-forward neural networks and incorporating additional input data that was generated using different readily available NLP tools.

Keywords

Opinion Mining, Natural Language Processing, Recurrent Neural Network, Feed-Forward Neural Network, Part-of-Speech Tagging, Dependency Parsing, Word Vectors

Acknowledgements

First and foremost, we thank our thesis supervisor Mr. Moin Mostakim for his continuous support throughout the course of our work on this thesis. Moreover, we thank our families and friends for their encouragement and support.

Lastly, we thank BRAC University for giving us the opportunity to complete our Bachelor of Science in Computer Science and Engineering.

Table of Contents

Declaration 1
Abstract 2
Keywords 2
Acknowledgements 2
Table of Contents 3
List of Figures 4
List of Tables 4
Abbreviations 5
1. Introduction 6
2. Literature Review 7
2.1. Natural Language Processing 7
2.2. Related Work 8
2.3. Word Vectors 9
2.4. Part-of-Speech Tagging 10
2.5. Dependency Parsing 11
2.6. Recurrent Neural Network 11
2.7. Feed-Forward Artificial Neural Network 12
3. Methodology 13
4. System Implementation 14
4.1. Languages and Tools 14
4.2. Datasets 15
4.3. Text Preparation 16
4.4. Training and Preparing Word Vectors 17
4.5. Integrating Part-of-Speech Tags 18
4.6. Integrating Word Dependencies 18
4.7. Running the Neural Networks 19
5. Experiments and Result Analysis 23
5.1. Experiments 23
5.2. Result Analysis 32
6. Comparative Analysis 33
7. Conclusion and Future Work 34
8. References 35

List of Figures

Fig. 1 Basic aspect-based sentiment analysis example 6
Fig. 2 Model visualization of word vector clustering 9
Fig. 3 Simple example for a POS-tagged sentence 10
Fig. 4 Relevance of POS tags for aspects and sentiment 10
Fig. 5 Simple representation of a sentence with parsed dependencies 11
Fig. 6 Layers of a basic recurrent neural network 11
Fig. 7 Layers of a basic feed-forward neural network 12
Fig. 8 Screenshot of the labelling tool we built for this task 15
Fig. 9 Sample sentence from the SemEval laptop dataset 16
Fig. 10 DisplaCy Dependency Visualizer example output 19
Fig. 11 Aspect counting F1 score development using RNN 25
Fig. 12 Aspect counting training process using FF-ANN 25
Fig. 13 Aspect extraction F1 score development using RNN 26
Fig. 14 Aspect extraction F1 score development using FF-ANN 26
Fig. 15 Aspect sentiment prediction F1 score development using RNN 27
Fig. 16 Aspect sentiment prediction F1 score development using FF-ANN 27
Fig. 17 F1 score comparison of all experiments 32
Fig. 18 Aspect sentiment prediction F1 score development and trendline using FF-ANN. 34

List of Tables

Table 1: Results of experiments using RNN: only the review sentences 28
Table 2: Results of experiments using FF-ANN: only the review sentences 28
Table 3: Results of experiments using RNN: word vectors 29
Table 4: Results of experiments using FF-ANN: word vectors 29
Table 5: Results of experiments using RNN: word vectors & POS tags 30
Table 6: Results of experiments using FF-ANN: word vectors & POS tags 30
Table 7: Results of experiments using RNN: word vectors, POS tags & dep. 31
Table 8: Results of experiments using FF-ANN: word vectors, POS tags & dep. 31
Table 9: Comparison of the complete result set 32

Abbreviations

NLP – Natural Language Processing

ABSA – Aspect-Based Sentiment Analysis

POS – Part-of-Speech

WV – Word Vectors

WD – Word Dependencies

RNN – Recurrent Neural Network

LSTM – Long Short-Term Memory

FF-ANN – Feed-Forward Artificial Network

1. Introduction

Opinion mining is the task of gathering huge amounts of data that contain valuable information about people's views on different topics. Online shopping websites like Amazon.com are of specific interest for this task, as they host hundreds of thousands of user-reviews for tens of thousands of different products. However, those reviews are currently only of use to users who read them one by one.

Using aspect-based sentiment analysis it is possible to analyze these reviews and predict opinions not only for a whole review or sentence, but on an aspect-level [1]. This means, it is possible to find out that, for example, a sentence is praising a laptop's display, while simultaneously criticizing its battery

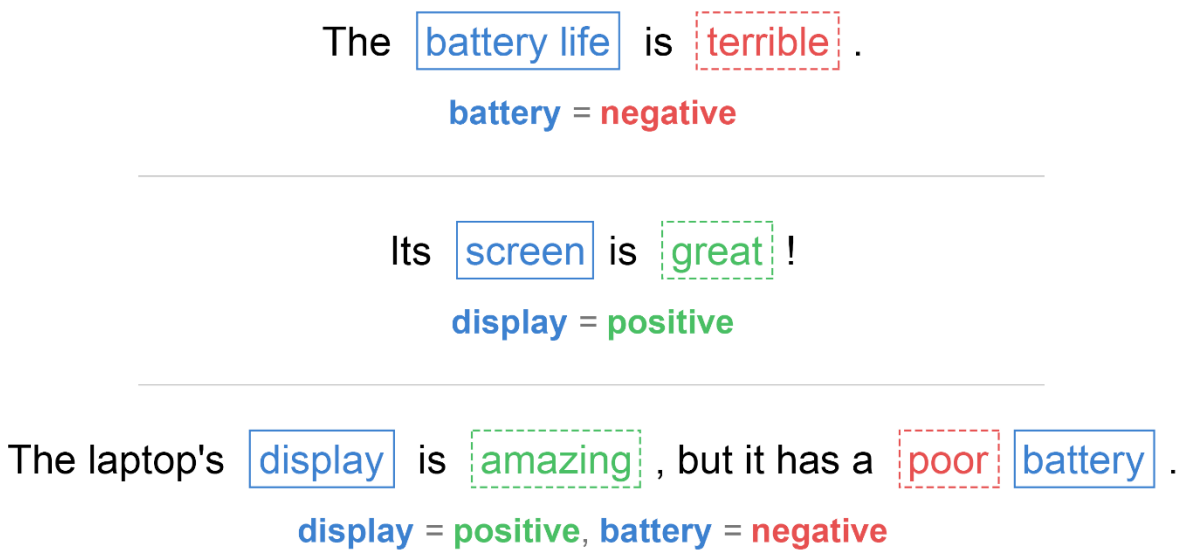

Fig. 1 Basic aspect-based sentiment analysis example

So far there have been multiple approaches and competitions about aspect-based sentiment analysis, which resulted in various solutions using machine learning algorithms, as well as a few implementations using neural networks [1][2][3][4].

Despite the many approaches, nearly all of them based their work on a relatively small dataset provided by the SemEval community for this specific task, which consists of review sentences for laptops, hotels and restaurants, labelled with their respective aspects and sentiment polarities [1]. Small datasets like this are a problem for classification tasks, since they can't cover a big variety of eventualities that appear in human language.

To offer a solution to this problem, we proposed that it is possible to support aspect-based sentiment analysis by generating additional structural data out of the existing text data. We wanted to prove that this can be achieved by using natural language processing models that are already trained on big corpora of text and can provide meta data about each review sentence up to the word-level. We intended to show that feeding this additional data into a neural network along with labelled text, can increase the accuracy in predicting aspect-based sentiments for reviews.

2. Literature Review

We researched various methods and technologies for processing written human language to achieve our goal. Data mining and natural language processing have been an important topic in the last years and scientists all around the world have achieved various levels of success. The relevant background studies and literature for our task are given below.

2.1. Natural Language Processing

Natural language processing is an umbrella term for all kinds of approaches in using computers to understand and manipulate human language. Research fields focus on working with written language, as well as actual auditory speech [5]. Scientists have been working on this field since capable computers were available, but with the rise of the internet as one of the most important ways of communication, NLP became more and more important.

Today, NLP has become a part of our daily lives. Google, for instance, tries to predict what exactly we mean, when we start typing a search term into its search field. Moreover, nearly all modern phones include software that tries to understand our speech to do certain tasks for us, and gadgets like Amazon's Alexa and Google Home are already being used as home assistants that understand what we ask them to do.

Over the years, many different out-of-the-box solutions have been developed, which can be used freely by anyone to work on various tasks in NLP. A big part of those solutions are classifiers, which provide a desired output given a certain textual input.

Since our goal was to implement an aspect-based sentiment analysis system, the already available tools were of great interest, to find out which of them can aide in realizing our classifier. These tools are explained in detail in this section.

2.2. Related Work

There have already been multiple attempts for aspect-based sentiment analysis, using different approaches, using machine learning [3][6][7], as well as neural networks [2].

Most approaches so far were based on machine learning. In fact, in the annual SemEval competitions on ABSA since in 2014 and 2015, most teams decided to use machine learning techniques like support vector machines (SVM) or conditional random field (CRF) classifiers and scored the best results with those approaches. The ABSA task was split into two tasks, aspect extraction and sentiment prediction. The winning team of 2015 in the aspect extraction task used CRF and modelled the aspect extraction as a multiclass classification problem and used n-grams and word clusters learnt from Amazon (laptop review task) and Yelp (restaurant review task). The winning team on the sentiment prediction task in 2015 used a maximum entropy classifier in its machine learning approach [1].

Our initial inspiration on using deep learning was mostly based on Wang's and Liu's work on aspect based sentiment analysis [2]. Using deep neural networks, they have provided a proof of concept, showing that deep learning algorithms are capable of potentially outperforming other implementations in aspect-based sentiment analysis. However, as mentioned before, they mainly used the small SemEval dataset. Their approach scored better than any of the winning teams in the 2015 SemEval competition.

Along with it, they also made use of the Google News 300-dimensional word vectors [16]. We have seen the practice of using word vectors to support machine learning and neural networks in multiple research papers [2][8][9][10], hence we also decided to adapt it for our work. However, like one team in [1] did with word clusters, we decided to train our word vectors on reviews rather than newspaper articles.

Wang and Liu used word vectors trained using the word2vec algorithm. While word2vec is a predictive model [11][12][13], there are other approaches like GloVe, which are count-based models. Baroni, Dinu and Kruszewski found out that predictive models are superior to count-based ones [14]. Using this information and the conclusions provided we chose word2vec as our training method for word vectors as well.

2.3. Word Vectors

Word vectors are word representations in form of high-dimensional vectors of real numbers. Each word in a given text corpus can be represented by one vector. The vectors of words which share a close relationship, because they often appear together in the text corpus, are clustered together in their vector representation as well. This way it is possible to, for example, obtain similar words or synonyms for a word simply by retrieving words with a close vector representation to a given word [15].

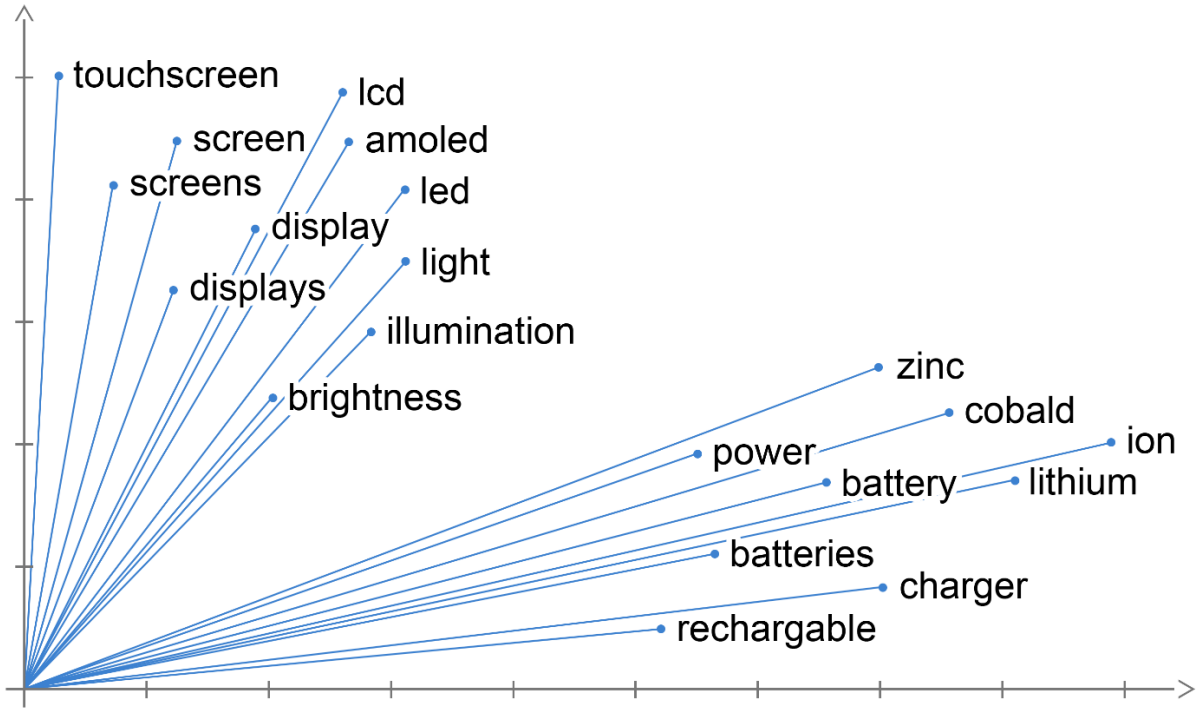

Fig. 2 Model visualization of word vector clustering

The behavior of word vectors was of particular interest for our aspect extraction subtask. If a review is talking about a laptop's display, the use of word vectors could help the network understand more quickly that the words "display" and "screen" are often used interchangeably and belong to the same aspect category [17].

Moreover, if the word vectors are trained using a high number of product reviews of the same domain as the reviews to be classified, it could also act as kind of a spell check, since common mistakes also cluster together with their correct forms [18].

2.4. Part-of-Speech Tagging

Part-of-speech tagging is a way of marking up a text with meta-information regarding the grammatical roles of the text's words. Common tags are word types (noun, verb, adjective, adverb etc.), but many POS-taggers provide even more information, like whether a word appears in plural form, signifies possession or even negations.

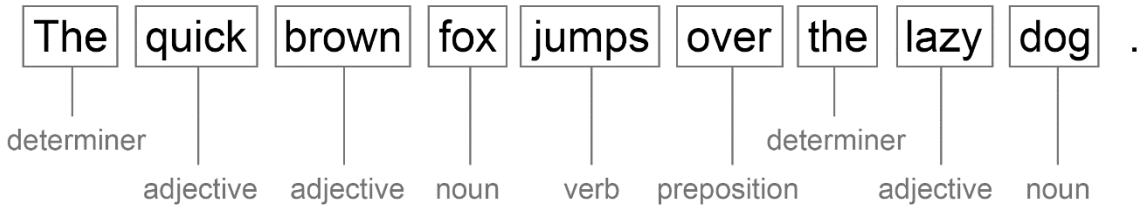

Fig. 3 Simple example for a POS-tagged sentence

There are multiple classifiers which accomplish this task successfully, one of the most popular ones being the Stanford Log-linear Part-Of-Speech Tagger [23][24]. There are different methods used for POS-Tagging, one of the most accurate algorithms now is the Hidden Markov Model (HMM) [26]. As shown in Fig. 3, POS taggers can determine which words are adjectives, which are nouns, and which are verbs.

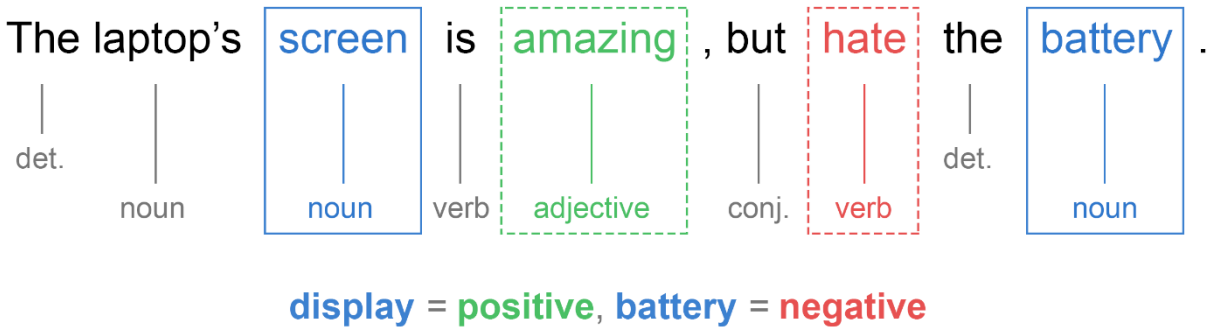

Fig. 4 Relevance of POS tags for aspects and sentiment

The relevance of this for our task lies in the fact that in many cases, sentence aspects are represented by nouns and sentiment can be detected using verbs and adjectives (Fig. 4). For this reason, we were positive that POS-Tagging should be a valuable addition to the data used to train the aspect-based sentiment analysis networks.

2.5. Dependency Parsing

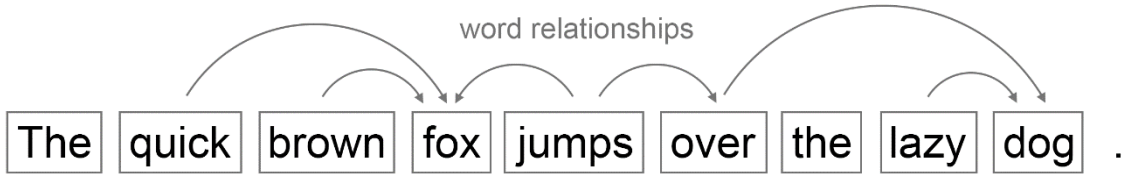

Fig. 5 Simple representation of a sentence with parsed dependencies

Just like part-of-speech tagging, dependency parsing is a way of finding out how text is structured. However, dependency parsing goes a step further and extracts the meaning of a text by analyzing the word relationships within it [22][28]. With a well-trained dependency-parser it is possible to know exactly which adjective describes which noun, or which noun belongs to which verb [27].

We hoped that the dependencies on the one hand improve the sentiment analysis, like proposed in [27] and on the other hand help the classifier understand, which aspect belongs to which sentiment [25].

2.6. Recurrent Neural Network

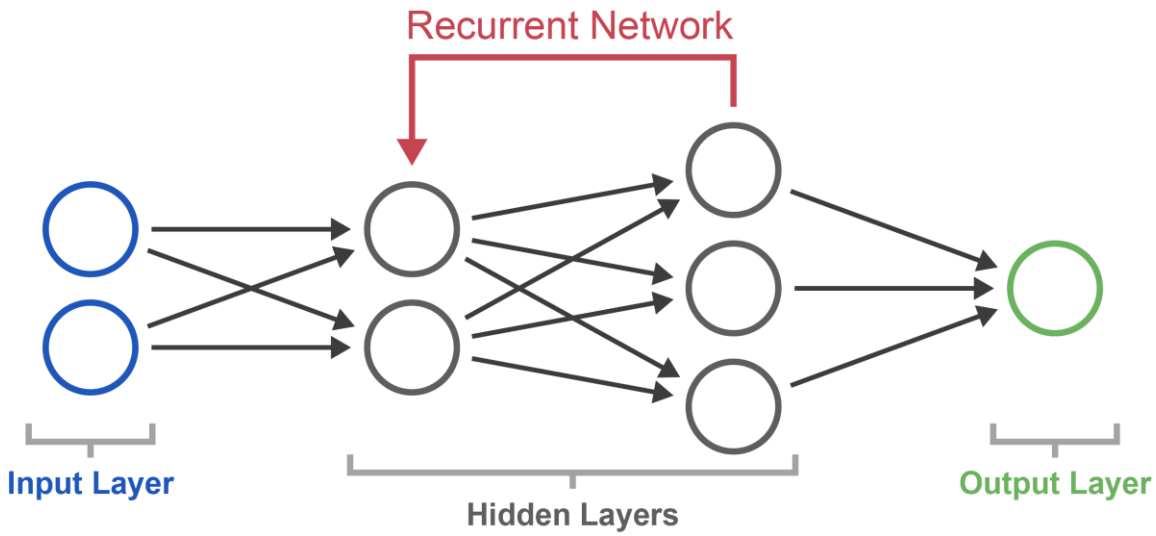

Fig. 6 Layers of a basic recurrent neural network

A standard RNN is a deep neural network, which maps sequences to sequences [21][20]. In comparison to other neural networks, like feed-forward or convolutional neural networks, the hidden states of the network's output in an RNN are fed back into the network. This way, inputs from earlier data points in a sequence still have an influence on later iterations, which closely

resembles the work process of the human memory on storing information. For this reason, recurrent neural networks are often used to classify or generate sequential data.

2.6.1. Long Short-Term Memory

Conventional recurrent neural networks used to make use of Back-Propagation Through Time (BPTT) or Real-Time Recurrent Learning (RTRL) for the error calculation. The problem with these approaches was that error signals tended to blow up or vanish, which lead to lack of efficiency and accuracy when training RNNs.

In [30] Hochreiter and Schmidhuber proposed a new recurrent network architecture – the LSTM. It was designed to overcome said problems by using memory cells and gates which learn to bridge time intervals. The gates define, if a memory is added, kept or removed from a cell. This way the LSTM learns, which memories should be kept, and which can be "forgotten".

The result was a recurrent neural network, which can offer drastically more efficiency and accuracy, and even keep improving the network in 1000 and more epochs of training.

2.7. Feed-Forward Artificial Neural Network

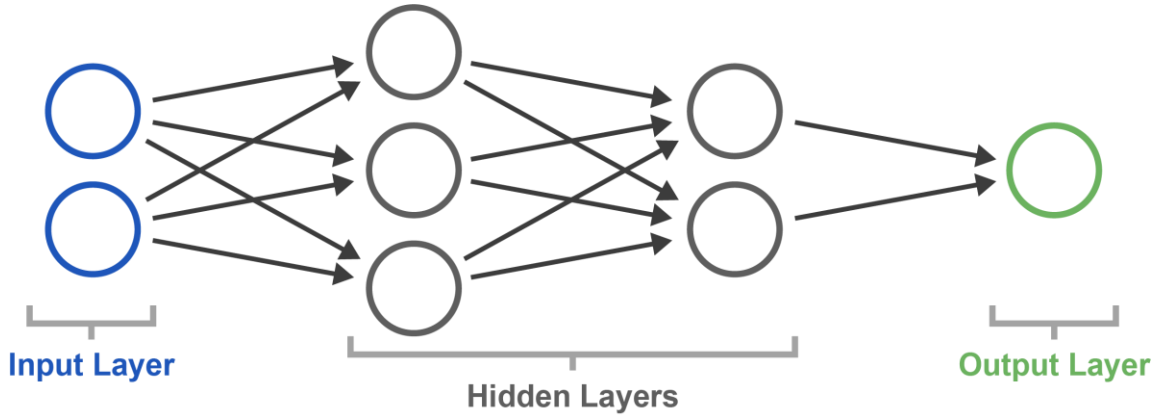

Fig. 7 Layers of a basic feed-forward neural network

In contrast to recurrent neural networks, feed-forward networks allow signals to only travel one way – previous outputs do not influence later states. Feed-forward ANNs are today the most widely used neural networks, since they are easy to implement and very versatile in their application. They consist of one input layer, a desired number of hidden layers and an output layer. The input size is defined by the desired input data and the output size by the target labels used for loss calculation [31][32]. Using hidden layers to make use of one or more nonlinear activation functions, many classification tasks can be implemented using this type of neural network.

3. Methodology

The ABSA classification task can be divided into three sub-tasks, namely:

1. Predicting how many aspects a sentence contains
2. Extraction of those aspects
3. Prediction of the sentiment based on each of the sentence's aspects

Since each sentence can potentially contain more than one aspect, the classifier of sub-task 2. had to return a probability distribution of aspects in a sentence. However, this probability distribution alone can hardly be used to define how many of the highest probabilities should be used to create the result set of aspects. Therefore, sub-task 1. was introduced to first define how many aspects the second classifier should use from the values it returns. Sub-task 3. then uses the found aspects to predict what sentiment is applied to each aspect inside a review sentence.

Because the available labelled SemEval dataset is very small, around one thousand additional amazon review sentences on the relevant product category were scraped and labelled by hand to increase the amount of training data. The resulting dataset contained about 2500 labelled sentences.

In addition to this dataset, around 100,000 more reviews were scraped from amazon to train the word vectors. The word vectors were trained using the popular word2vec algorithm [11][12][13].

The main intention of our work was to find out, if part-of-speech tags and semantic word dependencies obtained from the original datasets are suitable to improve the performance of an ABSA classifier. For this task, a POS tagger and a dependency parser were used to extract the needed information from each of the review sentences.

Three neural networks were used to take care of each of the three classification sub-tasks defined above. The first neural network takes a review sentence, the pre-trained word vectors and a list of POS tags to predict how many aspects the sentence contains.

The second neural network takes the same data as the first one; however, the sentences are split into sub-sentences, according to the obtained word dependencies. It returns a prediction of which aspects are most likely contained in the sentence and uses the result from the first network to return the correct number of aspects.

The third neural network again takes the same data as the second one; however, an aspect label is also fed into it, to determine the polarity for a specific aspect within the sentence.

All networks were implemented both as recurrent neural networks, as well as feed-forward neural networks, to compare their performances on the ABSA task.

4. System Implementation

The implementation of the classification system consisted of multiple parts. In the following sections, the necessary steps are explained in detail and the used tools are mentioned.

Firstly, the datasets had to be prepared and tokenized to be able to feed them into a neural network. During the preparation of the datasets, POS tags and dependencies were generated. Afterwards, word vectors were trained using the set of 100,000 amazon reviews. In the end, three neural networks were implemented for each of the three ABSA tasks: aspect count prediction, aspect extraction and polarity prediction.

4.1. Languages and Tools

The system is implemented in Python 3 and making use of the following Python modules:

4.1.1. SpaCy

SpaCy is an NLP library, offering multiple tools for various tasks. It is the fastest syntactic parser in the world and its accuracy lies within 1% of the best tools that are available [29]. Apart from dependency parsing, it also offers a full-fledged POS tagger. SpaCy was developed as an NLP tool that can be used in production environments.

4.1.2. PyTorch

PyTorch is a Python implementation of the Torch library written in the Lua programming language. Just like other deep learning libraries like TensorFlow, it uses Tensors to make its computations. Tensors are matrixes of real numbers, which own multiple extra functionalities that make using them in neural networks very easy. They can do various tasks for you, which you would have to do by hand using other approaches.

The main difference between PyTorch and other popular libraries like TensorFlow is that it uses a dynamic graph definition approach. This means you can define, change and execute nodes as you go which makes the programming of networks much more intuitive and integrates better into the Python programming environment. PyTorch offers many different types of

neural networks that can be implemented very quickly. Apart from that, you are also free to implement your own network from scratch.

4.1.3. Gensim

Gensim is an open source library specializing in unsupervised text modelling algorithms. We used it to easily train word vectors.

4.2. Datasets

The main datasets we used were the "Laptops Train Data" and "Laptops Trial Data" of SemEval's 2015 Task 12: Aspect Based Sentiment Analysis for training our neural networks and evaluating the results respectively. About 1000 review sentences were additionally labelled by hand to increase the dataset's size to about 2500 labelled sentences. For the task of labelling the additional sentences in an easy way, we implemented a web-based tool that gives the user a review sentence, pulled from a database of scraped reviews which were split into sentences before. The user can then choose which aspects are present in the sentence, and which polarity they have.

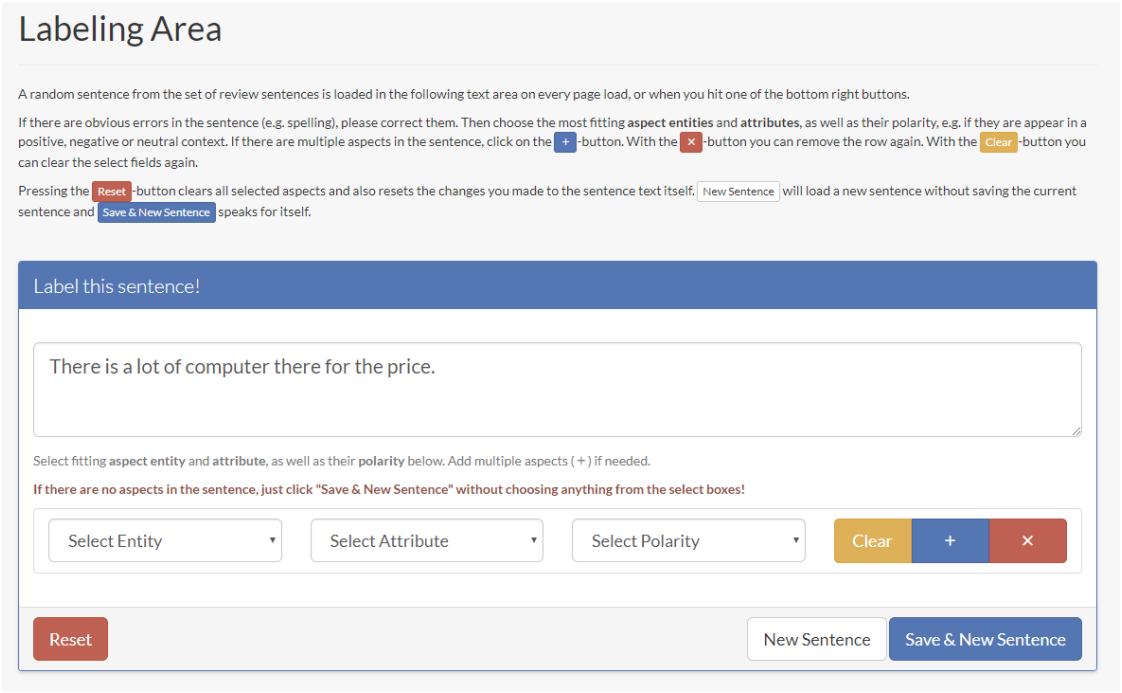

Fig. 8 Screenshot of the labelling tool we built for this task

The set of aspect labels in the original dataset consists of entities and attributes, separated by a "#" character. Our labelling tool used the same entities and attributes, to create a dataset with the same amount of information as in the SemEval dataset.

There are 22 different entities (e.g. LAPTOP, BATTERY, DISPLAY) and 9 different attributes (e.g. USABILITY, PRICE, OPERATION_PERFORMANCE). This leads to a total possible number of 198 different combinations, but the SemEval dataset uses 81 combinations, hence 81 individual aspect labels.

```
<sentence id="79:5">
    <text>
        This computer is really fast and I'm shocked as to how easy it is to get used to...
    </text>
    <Opinions>
        <Opinion category="LAPTOP#OPERATION_PERFORMANCE" polarity="positive"/>
        <Opinion category="LAPTOP#USABILITY" polarity="positive"/>
    </Opinions>
</sentence>
```

Fig. 9 Sample sentence from the SemEval laptop dataset

Using such a high number of possible labels to train a classifier on a small dataset, makes the system very prone to overfitting [19]. For this reason, we translated the entity-attribute-combinations into simpler labels, decreasing the total number of aspect labels to 30. As for the sentiment polarity, the labels “positive”, “neutral” and “negative” were used as intended in the SemEval dataset.

The second dataset used were around 100,000 amazon reviews on laptops, tablets and smartphones. This dataset was used to train the word vectors that were later used as word embeddings in the neural networks.

4.3. Text Preparation

In NLP, it is important to consider that there are different types of text corpora. An example is the difference between lyrical texts and scientific texts. Not only the choice of words, but also the used sentence structure and even punctuation can be very different. This is the reason why data scientists use different text corpora to train different algorithms for their classification tasks, depending on which kind of text they are working with.

The area of product reviews is in itself very diverse: some users use a language which is close to scientific texts; others are sometimes even hard for humans to understand, because they contain errors, wrong punctuation and even emoticons.

It may not be possible to change those reviews' sentence structures to bring them in line with each other, but there are certain steps that can be undertaken, to at least minimize huge differences.

These steps are:

1. Converting the whole text corpus to lower case text
2. Removing non-alphanumerical characters (except for apostrophes)
3. Spell-checking every word in the corpus

4.3.1. Preparing the Main Dataset

Our main dataset of 2500 sentences was in the first step pre-processed. Firstly, the three steps above were applied to the text. Next, the text was tokenized, meaning it was split into its separate words. In this step, it was important to consider the expected format for the external tools we used, like the POS-tagger and dependency parser. For example, some tools expect the sentence "i'm fine." to be split as ["i'm", "fine"], others expect ["i", "'m", "fine"]. Since we were using the python library SpaCy which provides tokenization, POS tagging and dependency parsing, the sentences were already tokenized in the correct form.

After tokenization, the words of all review sentences were added to a global vocabulary, mapping each word to exactly one distinct integer value.

4.3.2. Preparing the Word Vector Dataset

Gensim's algorithm for training word vectors automatically takes care of tokenization and other necessary tasks. However, it expects a text file containing one long line of text. The reviews which were scraped from amazon therefore needed be concatenated into one file. The resulting text corpus was converted to lowercase text, since only lowercase text was relevant for the implemented network. Moreover, punctuation was also stripped from the text corpus.

4.4. Training and Preparing Word Vectors

The word vectors were trained using Gensim's implementation of Google's word2vec algorithm. A dimensionality of 50 was used for the resulting vectors and the vectors were trained using the text file lined out in the section above.

After the word vectors were ready, they were matched with the dictionary obtained from the 2500 review sentences. Since words that weren't present in the review sentences which were

to be used to later train the networks, all word vectors which represented words not in the vocabulary, were removed.

4.5. Integrating Part-of-Speech Tags

As mentioned before, the POS tags were generated using SpaCy. The tagger provides two types of POS tags for each word: a simple POS tag (e.g. VERB, NOUN etc.), which only provides the simple word type, and a detailed tag (e.g. VBZ, VBG, NN), which provides a more sophisticated explanation of the word's role, based on its context and form. Both types of tags were used to later determine which one leads to better results.

Since POS tags can't simply be fed into the network as words, a POS vocabulary mapping each tag to a distinct integer value was generated, very similar to how it was done with the main dataset. Moreover, the vocabulary was then used to generate embeddings, which were concatenated to the word vectors during training to form a Tensor, which contains the sentence's text information as well as POS tags.

4.6. Integrating Word Dependencies

Bringing the word dependencies into a form which can be fed to the neural network was more complex, since they describe relationships rather than one-on-one tags such as, the POS-tagging process does.

We decided to use the word dependencies which were provided by SpaCy, to split sentences into multiple sub-sentences which were then fed into the network separately. This leads to the situation that parts of sentences are trained on wrong aspects, since, for example, a sentence with two aspects will be split, and both parts will be trained using each of the two aspects. However, given other occurrences of sentences and sub-sentences which contain the same topics and aspects, this problem should be negligible, and predictions should still tend to point to the correct aspect.

4.6.1. Recognizing Sentence Parts

SpaCy's dependency parser attaches a parent node to each of a sentence's words. This parent can either have another parent itself or be the root word of the whole sentence. Each edge in the resulting dependency tree is described by a dependency descriptor that shows which kind of connection two words share. A useful dependency visualizer called DisplaCy is provided by SpaCy and was used to understand the dependencies more easily during development.

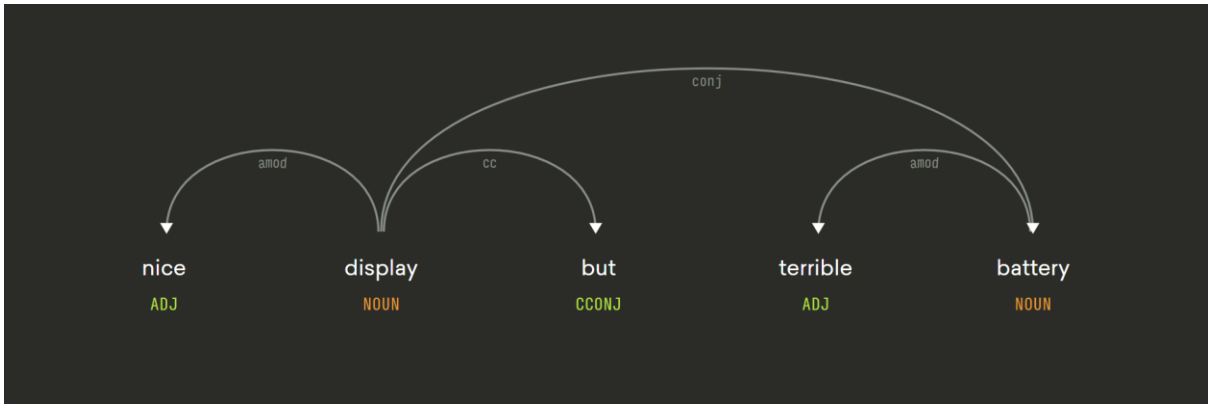

Fig. 10 DisplaCy Dependency Visualizer example output (source: demos.explosion.ai/displacy/)

To split sentences into multiple semantic parts, each word in the sentence was analyzed in a loop. For each item, the dependency tree was traversed from the leaves upwards to the root word. During the traversal, it was checked if the current word is the main verb of a sub-sentence, meaning a verb that has its own subject. If that was the case, every connection leading to this verb described an own semantic unit within the given sentence, hence a separable sub-sentence.

The procedure above worked well, if the sentence contained a verb. However, it failed, when there were no verbs, like for example in the sentence “Nice display, but terrible battery.”

If in the first loop no verb was found, a second loop went over the sentence again, this time looking for nouns. If a noun was found, which wasn’t part of a compound, signifying it is the main subject of a sub-sentence, every connected word leading to this noun was treated as a sub-sentence.

Using these procedures, sentences were split into multiple semantic parts, which were likely talking about different aspects.

4.7. Running the Neural Networks

The first two steps of the proposed classifier were predicting the number of aspects and extracting the aspects from each of the reviews’ sentences. The third step was then to use this information and predict the sentiment applied to each of the found aspects in the given sentence. As this three-step approach is a logical process of determining the sentiment of an aspect in a sentence, we split our classifier into these steps and implemented one neural network for each of them. To find out which type of neural network suits our needs best, we implemented one

RNN and one feed-forward ANN for each of the three tasks. However, they were trained on predicting different labels.

The first two networks (RNN + FF-ANN) were optimized to predict how many aspects are contained in one sentence. The second two were trained to then extract those aspects, while the last two networks were trained to predict a sentiment given one of the aspects.

Lastly, the results of the different types of networks were compared, as well as the results using different combinations of input data.

4.7.1. Calculation of Loss and Optimization

All our networks used the mean square error loss function for calculating the loss, using the output of the neural network and the expected targets (e.g. number of aspects, aspect and aspect polarity).

After backpropagating to calculate the gradients, which was easily done in PyTorch, since the network automatically remembers the necessary calculations, the network was optimized using stochastic gradient decent as the optimization function and a learn rate of 0.005, which we found to provide the best possible results in our test cases.

4.7.2. Common Setup of the Recurrent Neural Networks

All three recurrent networks were implemented using PyTorch's nn.LSTM class and initialized using the generated word vectors as embeddings. They consisted of four layers, the first one being the input layer. The second two layers were provided by the LSTM, which computes the following function for each element in the input sequence (e.g. word representations for each word in a sentence) for those two layers:

$$
\begin{aligned}
i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{(t-1)} + b_{hi}) \\
f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{(t-1)} + b_{hf}) \\
g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hc}h_{(t-1)} + b_{hg}) \\
o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{(t-1)} + b_{ho}) \\
c_t &= f_t * c_{(t-1)} + i_t * g_t \\
h_t &= o_t * \tanh(c_t)
\end{aligned}
$$

Equation 1 LSTM layer calculations

h_t is the hidden state at time t, c_t is the cell state at time t, x_t is the hidden state of the previous layer at time t or the input in case of the first layer. i_t, f_t, g_t, and o_t are the input, forget, cell and

out gates. σ is the sigmoid function. A general explanation of LSTM can be found in section *2.6.1 Long Short-Term Memory*.

The last layer first computed a linear transformation

$$y = Ax + b$$

Equation 2 Linear transformation

to transform the dimensionality of the data coming from the LSTM back into a desired output dimension, which is defined by the corresponding training label vector. Afterwards, the nonlinear Softmax function was used to convert the output layer to a probability distribution:

$$f_i(x) = \frac{exp(x_i - shift)}{sum_j exp(x_j - shift)}$$ where *shift* is $\max_i * x_i$

Equation 3 Softmax function

The loss function and optimization approach described in section *4.7.1. Calculation of Loss and Optimization* was ultimately used, to compute the loss and optimize the network in each epoch.

4.7.3. Common Setup of the Feed-Forward Networks

Like the three RNNs we used, the three feed-forward neural networks also share the same setup. Each of them consists of one input layer, one hidden layer and one output layer.

The first hidden layer applies a linear transformation to the input:

$$y = Ax + b$$

Equation 4 Linear transformation

Again, we used it to transform the data to a 150-dimensional hidden layer, which we found to be a good value for best results in our test cases. Next, the first nonlinearity – the ReLU function – was applied to each element the layer:

$$ReLU(x) = max(0, x)$$

Equation 5 Rectified Linear Units (ReLU) function

After that, another linear transformation was applied to transform the data to the target dimension, which was defined by the desired label dimensionality. Afterwards, the nonlinear Softmax function was used, to convert the vector values into a probability distribution for the loss function:

$$f_i(x) = \frac{exp(x_i - shift)}{sum_j exp(x_j - shift)}$$ where *shift* is $max_i * x_i$

Equation 6 Softmax function

The loss function and optimization approach described in section *4.7.1. Calculation of Loss and Optimization* was ultimately used, to compute the loss and optimize the network.

4.7.4. Network Training Specification for Each Classification Task

The data that was used to train the RNNs and feed-forward networks was essentially the same, hence the approaches described in the following three sections apply to both types of networks.

4.7.4.1. Aspect Count Prediction Networks

The main difference between the three classification networks was the data which was fed into them. For the first network, which was trained to predict the number of aspects in a given sentence, the network was provided with a review sentence – or sub-sentences, if it was split using word dependencies – using its word vectors and POS tags, which were translated into word embeddings.

For the training label in this network, we used a one-hot encoding. We first determined the maximum number of aspects in one sentence among the whole dataset and created an array of this length, containing only zeros. For each sentence we then placed a "1" at that place in the array, which represented the number of aspects in that sentence, e.g. for a sentence with two aspects, the "1" was placed at the array index two.

The network was set up to return a prediction in line with this target array, so that the loss function could be used to backpropagate and then optimize the network.

4.7.4.2. Aspect Extraction Networks

For the second network, which was trained to extract the aspects within a given sentence, the network was fed one review sentence or sub-sentence, along with its POS tags. Each token in

the word sequence, as well as POS tag sequence was translated into word embeddings. The resulting matrixes were then concatenated and fed into the first layer of the network.

Each sentence in the dataset was fed into the network, according to how many aspects it had. For example, a sentence that was labelled with two aspects, was fed into the network two times, each time using one of the two aspects as the target.

The target labels for this network were the correct aspects for each iteration. Since only one aspect was fed into the network per iteration, the target also contained only one aspect. Again, we used a one hot encoding to represent the correct aspect among the set of all aspects in an array of zeros.

4.7.4.3. Polarity Prediction Networks

For the third task, the word embeddings and POS tag embeddings were generated in the same way like in the first two networks. To incorporate the aspect, another vector had to be introduced as well, which let us feed the desired aspect into the network. The simple approach was to use another embedding representation for this aspect. When the training algorithm was iterating over the input sets, the three inputs – sentence, POS tags and aspects – are translated into embeddings, which are then concatenated and fed into the first network layer.

Each sentence or sub-sentence was again fed into the network according to how many aspect labels it contained. The target label used was a one hot encoded vector, like in the previous two training algorithms. It consisted of the three polarities, represented by indexes in the vector and the correct polarity for the given aspect was encoded using a “1” inside a vector of zeros.

5. Experiments and Result Analysis

5.1. Experiments

Experiments with the six networks were run using different parameters for the number of training epochs, hidden layer dimensions and learn rates, to find out which parameters give best results. All parameter combinations were first run using only the sentences as inputs, then using word vectors, then using word vectors and POS tags, and then using word vectors, POS tags and word dependencies. This was done for each of the six recurrent and feed-forward networks, resulting on a total of 24 training and evaluation rounds for each time we tested the networks.

This way we could analyze which parameters lead to the best results at which combination of input data. In the following sections we present and discuss the results we gained, by using the best precision, recall and F1 measure we achieved during testing.

Since our classification problems were using multiple classes, and the used precision, recall and F1 measure are generally used for binary classification problems, we calculated each of the measures for each class in our classification classes and took the average precision, recall and F1 measure from all classes.

5.1.1. Training Process

In every epoch of the training process, we let the algorithm output the current average precision, recall and F1 score on the training set for each task and network respectively. Like explained above, firstly, all the networks were trained only on text, then with additional word vectors, then with additional POS tags and lastly using the split sentences. To compare training speed, we compared the development of the F1 score between the networks trained on different input data.

After running the training process multiple times using different settings, we committed on using 150 dimensions for the hidden layers, stochastic gradient descent as the optimization function, a learn rate of 0.005 for the optimizer and mean square error as the loss function, since these settings overall gave the best results in our tests.

5.1.1.1. Aspect Counting Accuracy Increase during Training

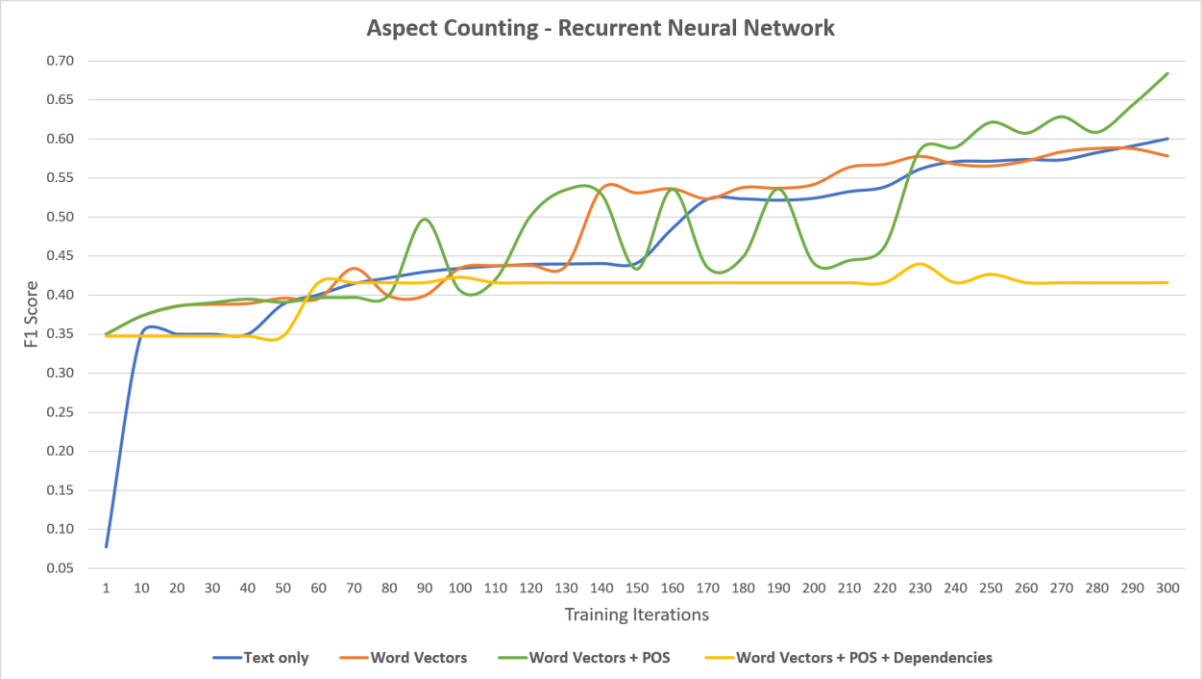

Fig. 11 Aspect counting F1 score development using RNN

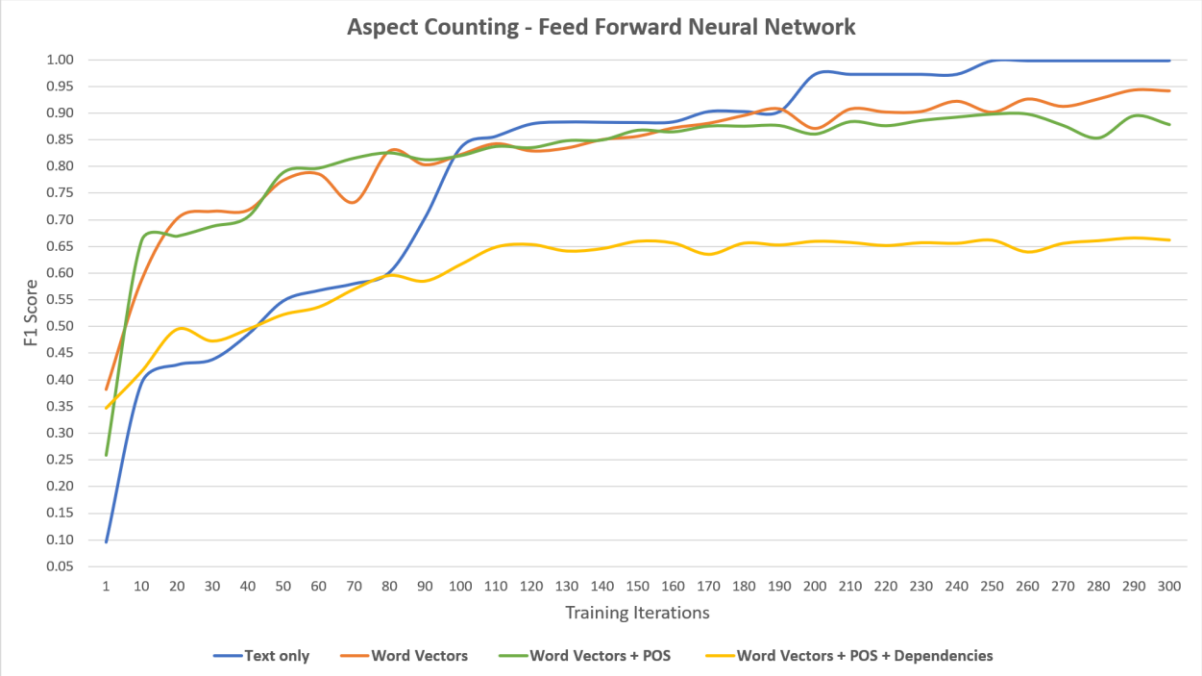

Fig. 12 Aspect counting training process using FF-ANN

The comparisons show that the feed-forward network overall seemed to train much faster and overall reached a higher score on the test data in all cases of different input data. POS tags looked promising especially on the RNN.

5.1.1.2. Aspect Extraction Accuracy Increase during Training

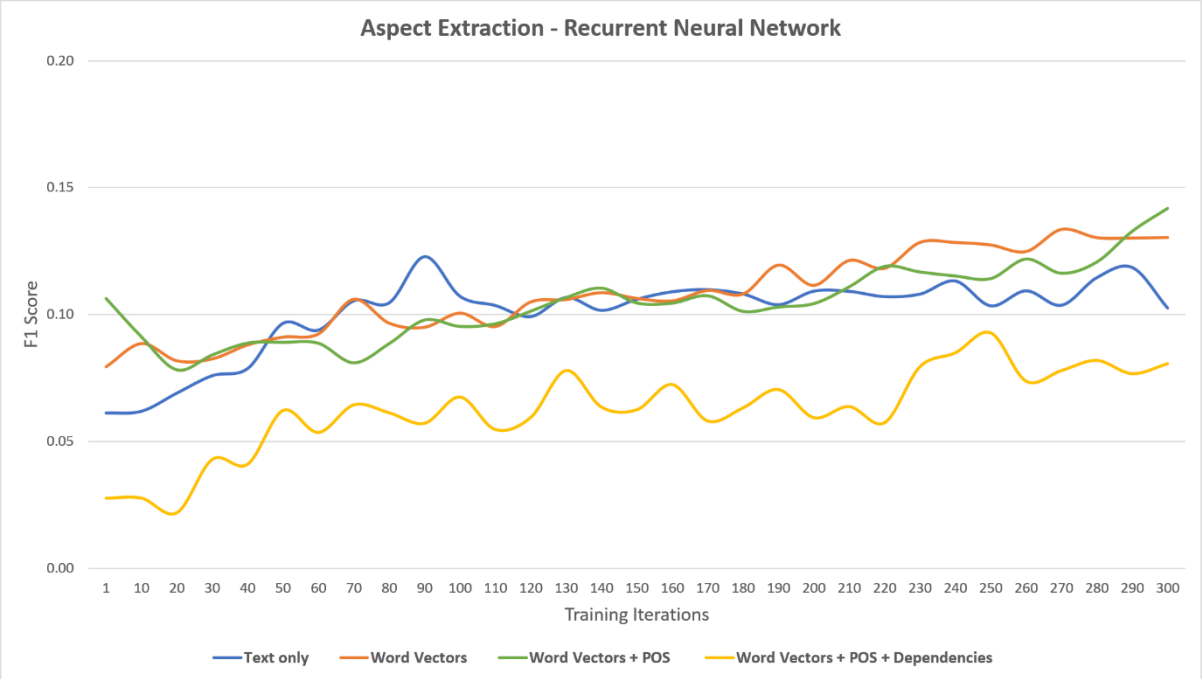

Fig. 13 Aspect extraction F1 score development using RNN

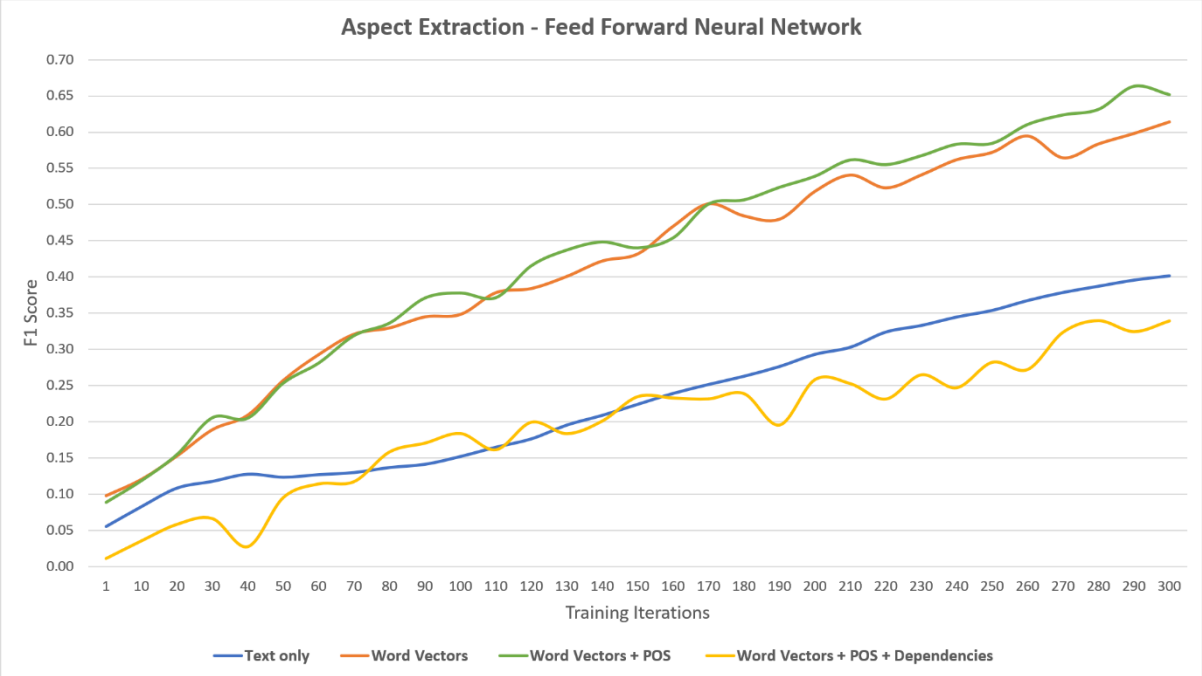

Fig. 14 Aspect extraction F1 score development using FF-ANN

The RNN's performance on the training set during training seemed disappointing, however the performance on the test set proved to be slightly better later. The feed-forward network, however, showed solid learning curves during training.

5.1.1.3. Aspect Sentiment Prediction Accuracy Increase during Training

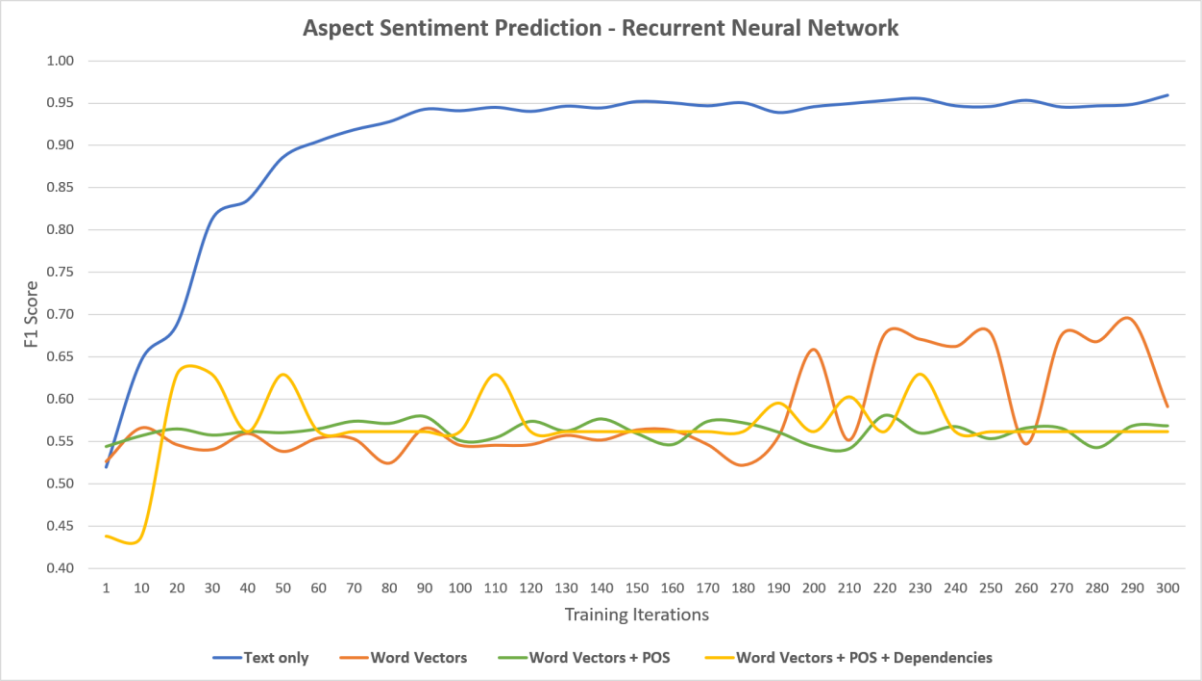

Fig. 15 Aspect sentiment prediction F1 score development using RNN

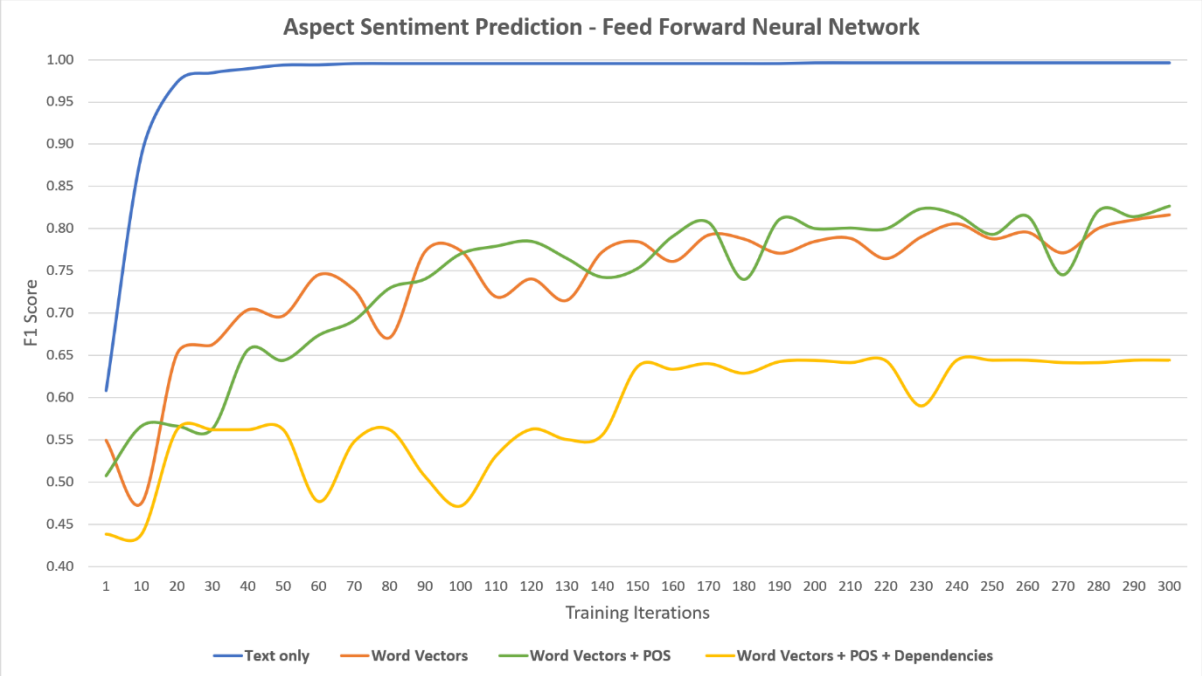

Fig. 16 Aspect sentiment prediction F1 score development using FF-ANN

The most notable curves in the sentiment networks were the text-only training processes, since their accuracy increased drastically right from the start. It was very hard to deduce whether it was due to very good training or overfitting. Thus, it had to be checked using the test dataset.

5.1.2. Evaluation Using Test Dataset

5.1.2.1. Without Additional Data

The first experiment used only the bare sentences as inputs. Sentence tokens were translated into random embeddings with a dimensionality of 50. This means the embeddings were only used to uniquely identify words in the sentence, while not introduce any semantic data into the network, which would be the case with word vectors.

We used this test's results as a baseline, to which the results of the following experiments could be compared to see, if POS tags and word dependencies can improve the accuracy of our neural network implementations.

Results using recurrent networks:

	Precision	Recall	F1
Aspect count	84.46%	84.12%	84.29%
Aspects	46.85%	50.08%	48.41%
Polarities	58.06%	57.70%	57.88%

Table 1: Results of experiments using RNN: only the review sentences

Results using feed-forward networks:

	Precision	Recall	F1
Aspect count	84.96%	84.87%	84.91%
Aspects	54.20%	56.96%	55.55%
Polarities	59.49%	59.83%	59.66%

Table 2: Results of experiments using FF-ANN: only the review sentences

The results on the test set using only the sentence text input were good, but didn't quite live up to the expectations from the accuracy increase seen during the training process. This is very apparent in the polarity prediction task, where the feed-forward network reached nearly 100% accuracy during training, but just scored about 60% on the test data. Hence, it is apparent that it slightly overfitted to the training data.

5.1.2.2. With Word Vectors

In the second experiment, the sentence tokens were translated into the generated word vectors and then fed into the networks. Using word vectors is already common practice to support neural networks on NLP tasks. Hence, testing the performance of using word vectors separately from POS tags and word dependencies, allows a comparison between this often-used practice, to our proposed techniques.

Results using recurrent networks:

	Precision	Recall	F1
Aspect count	84.60%	84.73%	84.66%
Aspects	49.86%	52.95%	51.36%
Polarities	64.57%	62.49%	63.51%

Table 3: Results of experiments using RNN: word vectors

Results using feed-forward networks:

	Precision	Recall	F1
Aspect count	93.16%	93.84%	93.50%
Aspects	61.90%	56.61%	59.14%
Polarities	65.70%	65.49%	65.59%

Table 4: Results of experiments using FF-ANN: word vectors

As expected, performance on both the recurrent and feed-forward networks increased using word vectors compared to the text-only experiments. The highest rise can be seen in the results of the aspect count task using the feed-forward network.

5.1.2.3. With Word Vectors and POS Tags

The third experiment used the POS tags, along with word vectors applied on the sentence tokens. Since POS tags in contain semantic information which is similar, but not as detailed as the one contained in word dependencies, they were tested separately as well.

Results using recurrent networks:

	Precision	Recall	F1
Aspect count	84.45%	84.58%	84.51%
Aspects	50.81%	53.32%	52.03%
Polarities	62.72%	65.00%	63.84%

Table 5: Results of experiments using RNN: word vectors & POS tags

Results using feed-forward networks:

	Precision	Recall	F1
Aspect count	94.57%	94.57%	94.57%
Aspects	63.56%	55.39%	59.19%
Polarities	66.90%	68.92%	67.89%

Table 6: Results of experiments using FF-ANN: word vectors & POS tags

Again, the results in nearly all experiments increased, compared to the word vectors and the text-only approaches. The results are overall in line with the expected results when taking the training process into account.

However, looking at the training accuracy increase on the aspect counting RNN, the score on that task in the test set was not significantly higher than the tests before, like expected. In fact, the accuracy is even slightly lower in this case.

5.1.2.4. With Word Vectors, POS-Tags and Word Dependencies

In the final experiment, all types of the proposed input data were used: input sentence, word vectors, POS tags and word dependencies, to see if word dependencies can increase the networks' accuracy even more than POS tags alone.

Results using recurrent networks:

	Precision	Recall	F1
Aspect count	79.71%	81.82%	80.75%
Aspects	54.32%	58.36%	56.27%
Polarities	52.87%	53.79%	53.33%

Table 7: Results of experiments using RNN: word vectors, POS tags & dependencies

Results using feed-forward networks:

	Precision	Recall	F1
Aspect count	91.84%	85.77%	88.70%
Aspects	69.61%	73.54%	71.52%
Polarities	64.17%	50.51%	56.53%

Table 8: Results of experiments using FF-ANN: word vectors, POS tags & dependencies

The results using word dependencies to train the networks on sub-sentences lead to an interesting result. The accuracy compared to the tests with less input data decreased on the aspect counting and polarity prediction tasks, but increased significantly on the aspect extraction task using the feed-forward network.

5.2. Result Analysis

		Precision			Recall			F1		
		Count	**Aspect**	**Polarity**	**Count**	**Aspect**	**Polarity**	**Count**	**Aspect**	**Polarity**
RNN	-	84.46%	46.85%	58.06%	84.12%	50.08%	57.70%	84.29%	48.41%	57.88%
	WV	84.60%	49.86%	64.57%	84.73%	52.95%	62.49%	84.66%	51.36%	63.51%
	WV+P	84.45%	50.81%	62.72%	84.58%	53.32%	65.00%	84.51%	52.03%	63.84%
	WV+P+WD	79.71%	54.32%	52.87%	81.82%	58.36%	53.79%	80.75%	56.27%	53.33%
FF-ANN	-	84.96%	54.20%	59.49%	84.87%	56.96%	59.83%	84.91%	55.55%	59.66%
	WV	93.16%	61.90%	65.70%	93.84%	56.61%	65.49%	93.50%	59.14%	65.59%
	WV+P	94.57%	63.56%	66.90%	94.57%	55.39%	68.92%	94.57%	59.19%	67.89%
	WV+P+WD	91.84%	69.61%	64.17%	85.77%	73.54%	50.51%	88.70%	71.52%	56.53%

Table 9: Comparison of the complete result set
(WV = word vectors, P = POS tags, WD = word dependencies)

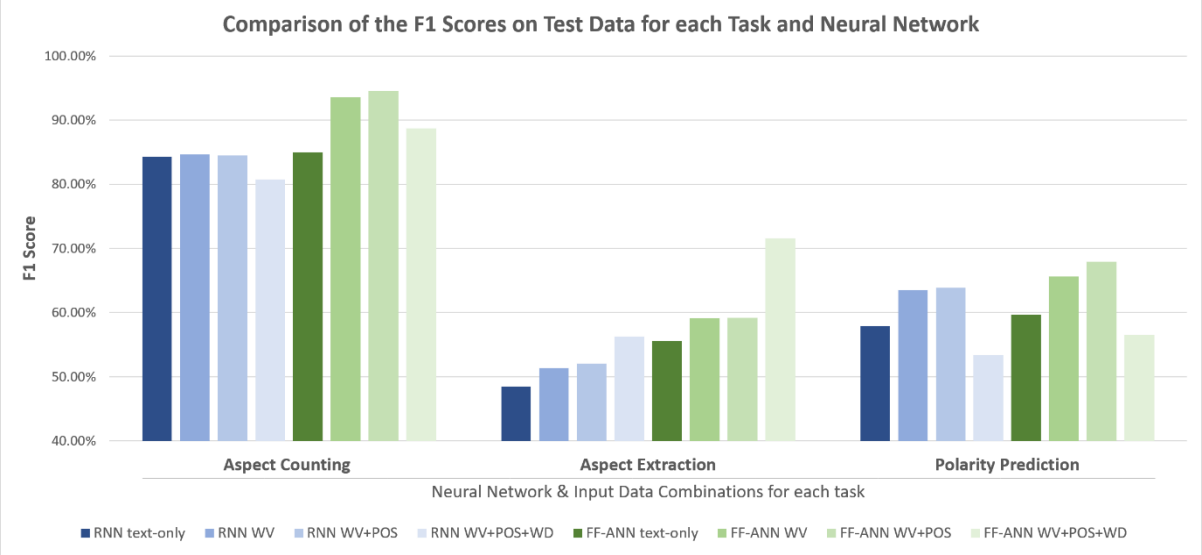

Fig. 17 F1 score comparison of all experiments

As expected, the use of word vectors, which is already common practice in many NLP tasks, increased the accuracy compared to only using the input sentences alone in all the implemented neural networks.

Moreover, the results show that in all cases the results of the feed-forward neural networks are superior to those from the RNNs. In fact, in case of the aspect counting networks, the highest F1 score of the RNN is still lower than the lowest score of the feed-forward networks. In case of the aspect extraction task, the best RNN F1 score is only slightly better than the lowest score among the feed-forward networks.

Overall the networks trained using POS tags performed best in the tasks of aspect counting and polarity prediction. The task of aspect extraction however, was by far best solved by the feed-forward network that additionally was trained on word dependencies.

What can also be seen in the result set is that the proposed way of integrating word dependencies into the network noticeably decreased the accuracy in the polarity prediction task both in the RNN and FF-ANN, which leads to the assumption that the information contained in word dependencies is more relevant to the aspect recognition, than to the polarity recognition given the aspect.

Given these insights, we can conclude that feed-forward neural networks are much better suited for the task of aspect-based sentiment analysis and all its sub-tasks, namely aspect counting, aspect extraction and aspect polarity prediction. We can also conclude that feeding POS tags and word dependencies into the networks as additional inputs is indeed capable of increasing the accuracy in all the sub-tasks.

6. Comparative Analysis

In this section, we want to shortly compare our results to the ones from Wang and Liu [2], to deepen our understanding and find out possible learnings. Wang and Liu used an approach using only two steps: aspect extraction and aspect sentiment prediction. Hence, we can only compare those two tasks to their work.

Aspect Extraction Task:

Their best result in the task of aspect extraction was an F1 score of 51.3%, which we outperformed in our approach with an F1 score of 71.5%. Our result was achieved by implementing POS tags and word dependencies.

Sentiment Prediction Task:

The sentiment prediction task was solved by Wang and Liu with an F1 score of 78.3%, while our best result, using POS tags, resulted in only 67.9%. This can very likely be improved using different network parameters, and especially using a higher number of epochs. The trendline of our training process on sentiment prediction using POS tags (Fig. 18) shows, that the network will keep improving well over an epoch number of 300. This should be researched in future attempts.

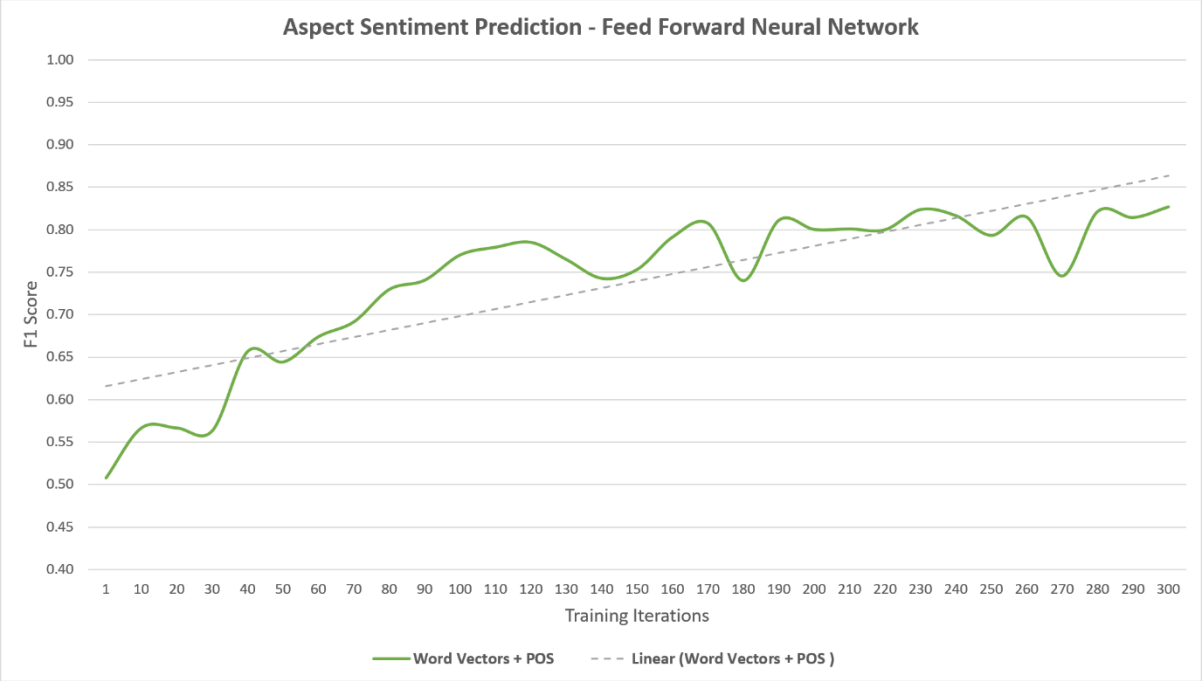

Fig. 18 Aspect sentiment prediction F1 score development and trendline using FF-ANN and POS tags

7. Conclusion and Future Work

In this work, we have shown that our proposed approach of improving the performance of neural networks for aspect-based sentiment analysis by using part-of-speech tags and word dependencies is indeed possible. It was also shown, in direct comparison that feed-forward neural networks are performing considerably better in the given tasks of ABSA than recurrent neural networks do.

Our hope is that these results can be an inspiration to use additional data, which is easily gathered using readily available NLP tools, to increase the performance of neural networks for different tasks.

While our results are very promising, future work should follow up on increasing the accuracy even more, by tweaking different parameters and trying out other layer-combinations inside the feed-forward neural networks. Additionally, other tasks than ABSA should be tested on the applicability of our learnings concerning the integration of POS tags, word dependencies and possibly other NLP tools.

8. References

[1] M. Pontiki, D. Galanis, H. Papageorgiou, S. Manandhar, I. Androutsopoulos (2015) Semeval-2015 task 12: Aspect based sentiment analysis. In Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), Denver, Colorado, USA

[2] B. Wang, M. Liu (2015). Deep Learning for Aspect-Based Sentiment Analysis.

[3] S. Brody, N. Elhadad (2010). An Unsupervised Aspect-Sentiment Model for Online Reviews. Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the ACL, pages 804–812, Los Angeles, California, USA

[4] D. Zhang, T. Luo, D. Wang, & R. Liu (2015). Learning from LDA using Deep Neural Networks.

[5] G. Chowdhury (2003). Natural language processing. Annual Review of Information Science and Technology, 37. pp. 51-89. ISSN 0066-4200, http://dx.doi.org/10.1002/aris.1440370103.

[6] M. Hu, B. Liu (2004). Mining and Summarizing Customer Reviews, KDD'04, August 22–25, 2004, Seattle, Washington, USA

[7] X. Ding, B. Liu, P. S. Yu (2008). A Holistic Lexicon-Based Approach to Opinion Mining, WSDM'08, February 11-12, 2008, Palo Alto, California, USA

[8] H. Shirani-Mehr (2015). Applications of Deep Learning to Sentiment Analysis of Movie Reviews.

[9] Z. Tu, W. Jiang, Q. Liu, S. Lin (2012). Dependency Forest for Sentiment Analysis. Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, CAS, Beijing, China. Natural Language Processing and Chinese Computing pp. 69-77.

[10] A. L. Maas, R. E. Daly, P. T. Peter, D. Y. Huang, A. Ng, Ch. Potts (2011). Learning Word Vectors for Sentiment Analysis.

[11] T. Mikolov, K. Chen, G. Corrado, J. Dean (2013). Efficient Estimation of Word Representations in Vector Space. In Proceedings of Workshop at ICLR.

[12] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean (2013). Distributed Representations of Words and Phrases and their Compositionality. In Proceedings of NIPS.

[13] T. Mikolov, W.-t. Yih, G. Zweig (2013). Linguistic Regularities in Continuous Space Word Representations. In Proceedings of NAACL HLT.

[14] M. Baroni, G. Dinu, G. Kruszewski (2014). Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. Center for Mind/Brain Sciences, University of Trento, Italy.

[15] T. Mikolov, K. Chen, G. Corrado, J. Dean (2013). Efficient Estimation of Word Representations in Vector Space.

[16] T. Mikolov et al. (2013). Distributed Representations of Words and Phrases and their Compositionality. Conference on Neural Information Processing Systems (NIPS2013).

[17] K. Kasahara, T. Kato, C. Manning. Synonym Retrieval Using Word Vectors from Text Data, https://nlp.stanford.edu/~manning/xyzzy/acl30224.pdf

[18] Y.-R. Wang, Y.-F. Liao (2015) Word Vector/Conditional Random Field-based Chinese Spelling Error Detection for SIGHAN-2015 Evaluation. 46-49. 10.18653/v1/W15-3108.

[19] P. Domingos (2012). A few useful things to know about machine learning. University of Washington, Seattle, USA.

[20] H. Sak, A. Senior, F. Beaufays (2014). Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition.

[21] I. Sutskever (2013). Training Recurrent Neural Networks. University of Toronto.

[22] D. Chen, C. D. Manning (2014). A Fast and Accurate Dependency Parser using Neural Networks. Proceedings of EMNLP 2014

[23] K. Toutanova, C. D. Manning (2000). Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger. In Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000), pp. 63-70.

[24] K. Toutanova, D. Klein, C. Manning, Y. Singer (2003). Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In Proceedings of HLT-NAACL 2003, pp. 252-259.

[25] T. Thura Thet, J.-Cheon Na, C. S. G. Khoo (2010). Aspect-based sentiment analysis of movie reviews on discussion boards, Journal of Information Science, 36 (6) 2010, pp. 823–848.

[26] F. M. Hasan, N. UzZaman, M. Khan (2007). Comparison of different POS Tagging Techniques (-Gram, HMM and Brill's tagger) for Bangla. BRAC University, Bangladesh.

[27] H. Pouransari, S. Ghili (2014). Deep Learning for Sentiment Analysis of Movie Reviews.

[28] M.-C. de Marneffe, C. D. Mannin (2008-2016). Stanford typed dependencies manual.

[29] J. D. Choi et al. (2015). It Depends: Dependency Parser Comparison Using A Web-based Evaluation Tool. ACL.

[30] S. Hochreiter, J. Schmidhuber (1997). Long Short-Term Memory. Neural Computation 9(8):1735-1780.

[31] D. Dai, W. Tan, H. Zhan (2017). Understanding the Feedforward Artificial Neural Network Model From the Perspective of Network Flow. arxiv.org/abs/1704.08068.

[32] R. Amardeep, Dr. K. T. Swamy (2017). Training Feed forward Neural Network With Backpropogation Algorithm. IJECS Volume 6 Issue 1 Jan. 2017 Page No.19860-19866