

Aspect-Based Sentiment Analysis Using SemEval and Amazon Datasets

Saima A. Rahin

Dept. of CSE

BRAC University

Dhaka, Bangladesh

saima.ahmed.rahin@g.bracu.ac.bd

Tamanna Hasib

Dept. of CSE

BRAC University

Dhaka, Bangladesh

tamannahasib70@gmail.com

Mohibul Hassan

Dept. of CSE

BRAC University

Dhaka, Bangladesh

insular55@gmail.com

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 labeled 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

I. 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. This means, it is possible to find out that, for example, a sentence is praising a laptop's display, while simultaneously criticizing its battery. 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. 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. 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 a big corpus of text and can provide metadata about each review sentence up to the word-level. We intended to show that feeding this additional data into a neural network along with labeled text, can increase the accuracy in predicting aspect-based sentiments for reviews.

II. SYSTEM IMPLEMENTATION

A. 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 labeled by hand to increase the dataset's size to about 2500 labeled sentences. For the task of labeling 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. The set of aspect labels in the original dataset consists of entities and attributes, separated by a "#" character. Our labeling 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.

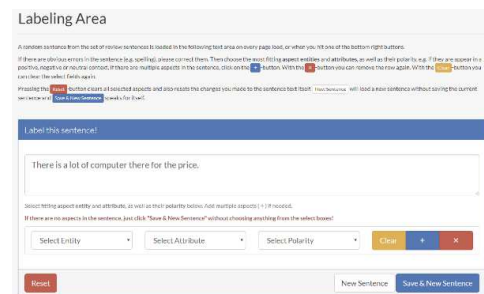


Fig. 1 Screenshot of the labeling tool we built for this task

```

<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. 2 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. 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.

B. Text Preparation

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-alphanumeric characters (except for apostrophes), 3) Spell-checking every word in the corpus

- a. 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.
- b. 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 to 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

C. 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.

D. 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.

E. 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, like 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.

a. 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.

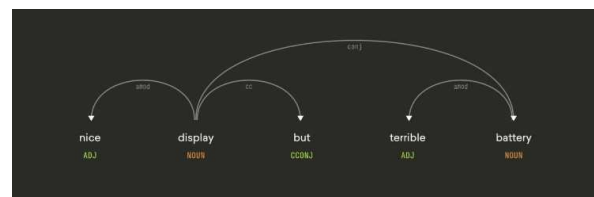


Fig. 3. DisplaCy Dependency Visualizer example output

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.

F. Running the Neural Networks

- **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.
- **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 matrices 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 labeled 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.
- **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.

III. EXPERIMENTS AND RESULT ANALYSIS

A. 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 in 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.

a. 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. As 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 descend 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.

- **Aspect Counting Accuracy Increase during Training**

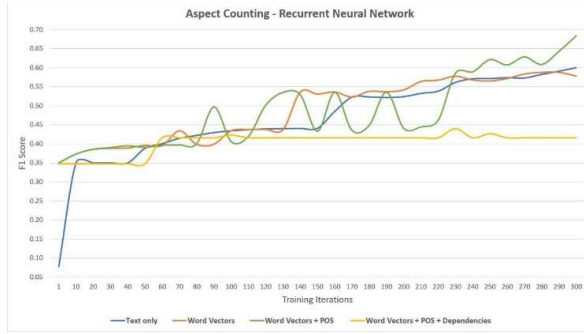


Fig. 4. Aspect counting F1 score development using RNN
The comparisons show that the feed-forward network overall seemed to train much faster and overall reached a higher score on the test data in all cases of different input data. POS tags looked promising especially on the RNN.

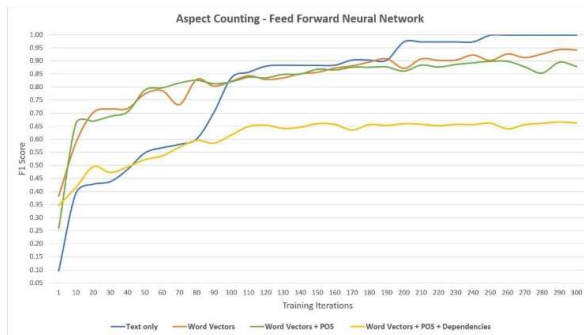


Fig. 5. Aspect counting training process using FF-ANN

- Aspect Extraction Accuracy Increase during Training

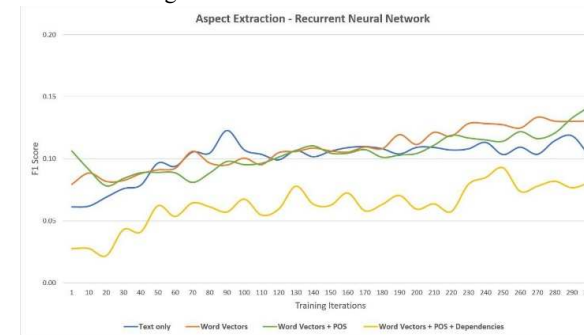


Fig. 6. Aspect extraction F1 score development using RNN

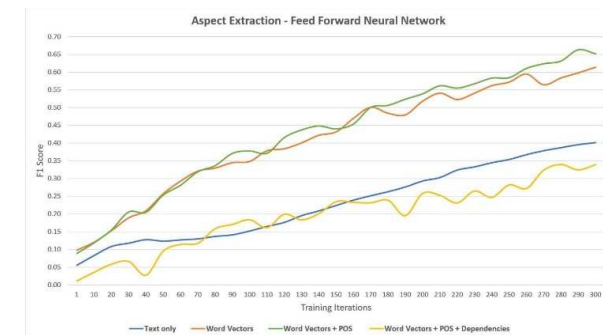


Fig. 7. Aspect extraction F1 score development using FF-ANN

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.

- Aspect Sentiment Prediction Accuracy Increase during Training

The most notable curves in the sentiment networks were the text-only training processes, since their accuracy increased drastically right from the start. If this was a case of either very good training or of overfitting had to be checked using the test set.

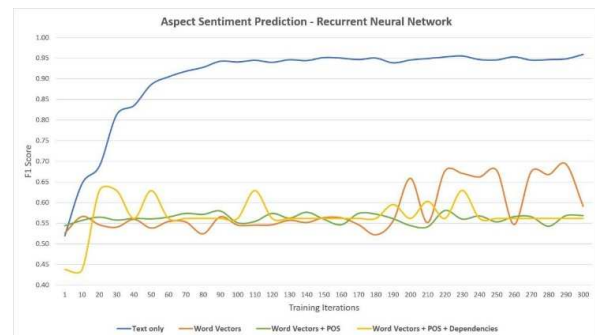


Fig. 8. Aspect sentiment prediction F1 score development using RNN

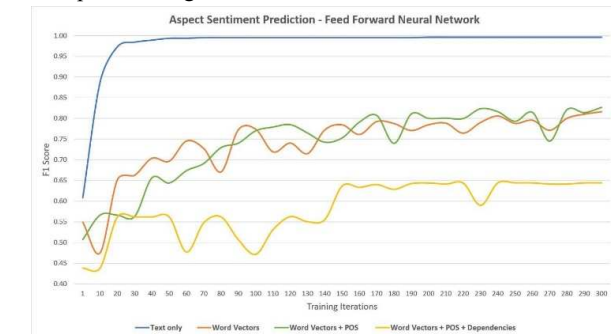


Fig. 9. Aspect sentiment prediction F1 score development using FF-ANN

b. Evaluation Using Test Dataset

- 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 introducing 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 is slightly overfitted to the training data.

- 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.

- 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 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. 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.

- 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.

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.

IV. 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.

V. 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 of 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.