UFAL at SemEval-2016 Task 5: Recurrent Neural Networks for Sentence Classification

Aleš Tamchyna and **Kateřina Veselovská**
Charles University in Prague, Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics
Malostranské náměstí 25, Prague, Czech Republic
{tamchyna,veselovska}@ufal.mff.cuni.cz

Abstract

This paper describes our system for aspect-based sentiment analysis (ABSA). We participate in Subtask 1 (sentence-level ABSA), focusing specifically on aspect category detection. We train a binary classifier for each category. This year's addition of multiple languages makes language-independent approaches attractive. We propose to utilize neural networks which should be capable of discovering linguistic patterns in the data automatically, thereby reducing the need for language-specific tools and feature engineering.

1 Introduction

Aspect-based sentiment analysis (ABSA) refers to the identification of specific entities and their aspects (aspect terms, opinion targets) in text and to the classification of their polarity. Typically, ABSA is applied to user reviews from various fields, such as consumer electronics, hotels or restaurants. In ABSA, we assume that the general target of evaluation has several aspects (e.g. food quality for restaurants) and we attempt to identify users' opinions on these individual aspects. Unlike the more general task of sentiment analysis where the goal would be to classify the polarity of entire sentences (or even whole reviews), in ABSA, we need to take a more fine-grained approach and consider also the internal structure of the given sentences.

As for the sentence-level ABSA, the aim is to identify all opinion tuples present in the sentence, taking into account the context of the whole review. In our work, we focus on identifying aspect term categories; these are composed of entities (e.g. FOOD) and their attribute labels (e.g. QUALITY or PRICE). Both the entities and the attribute labels were assigned based on predefined inventories.

We apply our method on several languages covering the following domains:

- Arabic: hotels
- Dutch: restaurants
- English: consumer electronics and restaurants
- French: restaurants
- Russian: restaurants
- Spanish: restaurants
- Turkish: restaurants

Our submission was the best system for Russian and Turkish but did not achieve noteworthy results in other domains/languages.

2 Related Work

The previous results for different ABSA SemEval tasks are discussed in Pontiki et al. (2014) and Pontiki et al. (2015). So far, most researchers in the field have focused on traditional machine learning approaches, such as various probabilistic methods (see Agarwal and Mittal (2016)), and/or employed deterministic methods, e.g. subjectivity lexicons (Taboada et al., 2011). Because more languages were added this year, approaches such as neural networks, which require less language-specific data and engineering, become attractive.

Proceedings of SemEval-2016, pages 379–383,
San Diego, California, June 16-17, 2016.

Neural networks have been used for sentiment analysis. Particularly, in last year's SemEval Twitter sentiment classification task, several submissions applied convolutional networks (Toh and Su, 2015; Ebert et al., 2015). In our work, we use recurrent networks instead. Our motivation for this decision is that for aspect identification, syntactic relationships and long-distance dependencies may play a significant role and that such phenomena may be better modeled with a recurrent network. Furthermore, our recurrent network could easily be adapted to perform sequence labeling instead of sentence level classification – this would allow us to identify the exact position in the sentences where the aspect was mentioned.

3 System Description

Our system addresses Subtask 1 – sentence-level ABSA. Within the subtask, we focus on Slot 1, i.e. aspect category detection. For each sentence, our goal is to identify all aspect categories which are mentioned. Each category is composed of an entity E (e.g. FOOD or SERVICE) and its attribute A (e.g. QUALITY or PRICE). We do not decompose this definition and treat each category independently, effectively reducing the task to many binary classification subtasks (one for each E#A pair).

Each classifier in our system is a deep recurrent neural network with Long Short-Term memory cells (LSTM, Hochreiter and Schmidhuber (1997)). LSTMs have been designed to overcome the vanishing gradient problem present in standard recurrent neural networks. Their ability to remember information over many time steps should enable them to capture long-term dependencies in the data.

Our network encodes the input sentence word by word and at the end produces a binary classification decision based on the representation of the full sentence.

3.1 Word Representations

We represent each word (token) on the input by its pre-trained word embedding (and we do not further optimize the embeddings when training the network).

For each language, we use the current dump of Wikipedia[1] as training data for the word embeddings. We use WikiExtractor[2] to extract plain text from the dump. We run a sentence splitter and we tokenize the sentences. Table 1 shows statistics of the data for each language.

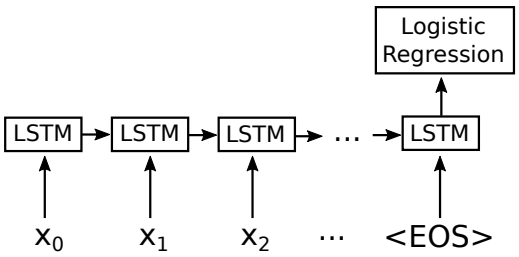

Figure 1: Architecture of our network (a single binary classifier).

Language	Sentences (M)	Tokens (M)
English	97.0	2103
French	26.2	633
Spanish	20.5	505
Russian	18.9	347
Dutch	15.4	252
Turkish	3.9	60
Arabic	3.1	63

Table 1: Sizes of Wikipedia dumps for each language.

We train the representations using `word2vec`[3] with continuous bag of words as the underlying model. We set the embedding size to 200.

3.2 Network Architecture

Our network is essentially a deep LSTM encoder followed by a logistic regression layer with two output classes (neurons). For each sentence, we go over the input word by word and provide the embeddings of the tokens to the input layer. The hidden LSTM layers maintain a state at each step which encodes the (partial) sentence. After the final token, we input an artificial end-of-sentence token which signals the network to output a classification decision on the final layer. Depending on the activation of the

[1] `http://wikipedia.org/`, we retrieved the current dumps on January 6, 2016.

[2] `http://medialab.di.unipi.it/wiki/Wikipedia_Extractor`

[3] `https://code.google.com/archive/p/word2vec/`

two output neurons, we either classify the instance as positive (i.e. containing the given E#A pair) or negative.

Figure 1 shows an illustration of the architecture. We initially experimented with a single LSTM layer but stacking several layers lead to improved accuracy. All of the networks used in the final submission share an identical architecture consisting of an input layer (word2vec embeddings, size 200) followed by three LSTM layers (64, 32 and 32 cells) and the final layer with two neurons. We did not experiment with tuning a decision threshold; we simply assign the class (positive or negative) which has the higher probability according to the network. We implement the networks in Chainer,[4] an open-source framework for neural networks.

3.3 Training

We evaluated several optimization algorithms and found that Adam (Kingma and Ba, 2014) lead to the fastest convergence. We also use gradient clipping as described in Pascanu et al. (2012): when the L2 norm of the gradients increases over a given threshold (which we set to 1), gradients are rescaled to fit within that norm. We also found dropout (Srivastava et al., 2014) to improve the results and we utilize it in all LSTM layers with the probability set to 0.5.

When training, we use cross-validation and measure held-out accuracy to detect overfitting. We found that even though training error (almost) monotonically decreases, held-out performance tends to be rather unstable. Moreover, many classes are rare and the held-out set may therefore only contain a handful of positive instances. We were not able to mitigate this instability which made it difficult to choose the final model.

Nonetheless, we decide how many training iterations to run based on cross-validation; we measure the average f-measure over the folds for each iteration and then choose the iteration with the highest average.

4 Results

In this section, we present and discuss the obtained results. We compare a number of systems for each language and domain.

[4] http://chainer.org/

Official Baseline. We report the results of the official baseline (provided by the task organizers). The official baseline is an SVM classifier with bag-of-words features which assigns a positive label to a sentence if the predicted probability is above a certain threshold.

Baseline. We also implement our own baseline. We use a simple logistic regression model and, similarly to the submitted system, we train a binary classifier for each category. We only use bag of words as features; we do not include any other information (such as morphological analysis or subjectivity lexicon features). We use L2 regularization with weight 1 for all models. Our motivation for including this baseline is to provide a direct comparison with a simpler model trained in a similar way as the neural network.

Submitted. The system as submitted for the official evaluation. Due to the number of networks that we had to train, we were not able to fully optimize all of them before the submission deadline. In some cases, we therefore use models trained only on a handful of iterations over the training data.

Optimized. We report the results of the fully optimized system separately. These results were obtained after the deadline. In this case, all models were selected based on cross-validation as described in Section 3.3.

Best. To put the performance of our system into context of the state of the art, we also report the scores of the best system for each language and domain.

4.1 Discussion

We evaluate our system using the toolkit provided by the organizers of the task. Table 2 shows all our results. The best result for each data set (excluding the winning system) is marked in bold. Asterisks mark the cases where our system won.

Concerning baselines, we observe that our implementation is often substantially better than the official baseline. This does not hold for the restaurants domain in English and Turkish, and for laptops. For Turkish, we suspect that the size of the test set (144 sentences) plays a role – the scores can be unstable when test data is small. For laptops, we believe this can be attributed to the large number of possible categories in this domain; our classifiers do not use a

Domain,	Restaurants						Hotels	Laptops
Language	English	Spanish	French	Dutch	Russian	Turkish	Arabic	English
Off. Baseline	59.93	54.69	52.61	42.82	55.88	58.90	40.34	37.48
Baseline	58.05	**62.17**	**54.81**	54.71	60.75	34.41	49.43	35.08
Submitted	**59.30**	58.81	49.93	53.88	**64.83**	**61.03**	47.30	26.98
Optimized	58.40	58.54	50.84	**55.03**	60.19	56.54	**52.59**	**38.26**
Best	73.03	70.59	61.21	60.15	64.83*	61.03*	52.11	51.94

Table 2: F-measure of the baselines, the submitted system, the fully optimized system and the winning system for all domains and languages.

tuned threshold.

Our submitted systems do not always outperform the baselines. While we did win in Russian and Turkish, results in other data sets are less promising, with scores similar to the (stronger) baseline or even lower. This is a discouraging finding, especially considering the amount of additional data used in network training – word embeddings were trained on millions of sentences from Wikipedia.

However, we do observe some generalization in the outputs of the deep-learning models which is beyond the capabilities of the baseline models. For instance, our model correctly identifies the category FOOD#QUALITY in the sentence "Green Tea creme brulee is a must!", even though this dish is not mentioned in the training data.

Our *optimized* networks do not always perform better than the submitted systems. Considering that we trained the submitted networks with fewer iterations (due to time constraints), we suspect that even with our model selection based on cross validation, overfitting may still have affected the results.

Overall, there are several possible explanations for the weak performance. Due to significant domain mismatch between Wikipedia and the data sets for the task, the trained word embeddings may not be suitable.[5]

Overfitting, and more generally, suboptimal setting of model (hyper)parameters, also most likely play a role.

The system design may also be problematic: we build a relatively large number of completely independent models (each doing binary classification for a single category) even though it seems clear that some parameter sharing should be possible. This problem is particularly prominent in data sets with a large number of possible classes, such as the laptop domain. In these cases, many E#A pairs are very rare. Because we do not decompose the entity and attribute and we train separate models, our classifiers only observe a handful of positive training instances, which results in very unreliable models.

5 Conclusion

We described our submission to the ABSA shared task. Our system won in two categories but overall does not outperform a simple baseline solution. We believe that more careful training of the networks is required and that we may need to revise the system design. On the other hand, the deep-learning model does show some generalization power, so this direction seems promising.

5.1 Future Work

In the future, we are planning to try other possible network architectures. In particular, instead of our many binary classifiers we could train a single network (for each language/domain) where the final layer would have as many neurons as there are categories. This could simplify the training and perhaps make the model more robust. Especially for domains with many possible sentence labels (such as laptops), this could improve the system performance.

A natural extension of this approach would be to design the network to predict the entity and attribute separately; this could also allow for some parameter sharing between different classes.

We would also like to further investigate the issue of hyperparameter tuning and model selection.

Finally, a direct comparison with convolutional networks could also be interesting.

[5] We would like to thank the reviewers for this observation.

Acknowledgements

This work was supported by the grant no. GA15-06894S of the Grant Agency of the Czech Republic.

References

Basant Agarwal and Namita Mittal. 2016. Machine learning approach for sentiment analysis. In *Prominent Feature Extraction for Sentiment Analysis*, pages 21–45. Springer.

Sebastian Ebert, Ngoc Thang Vu, and Hinrich Schütze. 2015. Cis-positive: A combination of convolutional neural networks and support vector machines for sentiment analysis in twitter. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 527–532, Denver, Colorado, June. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2012. Understanding the exploding gradient problem. *CoRR*, abs/1211.5063.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*, pages 27–35.

Maria Pontiki, Dimitrios Galanis, Haris Papageorgiou, Suresh Manandhar, and Ion Androutsopoulos. 2015. Semeval-2015 task 12: Aspect based sentiment analysis. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), Association for Computational Linguistics, Denver, Colorado*, pages 486–495.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, January.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. 2011. Lexicon-based methods for sentiment analysis. *Comput. Linguist.*, 37(2):267–307, June.

Zhiqiang Toh and Jian Su. 2015. Nlangp: Supervised machine learning system for aspect category classification and opinion target extraction. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 496–501, Denver, Colorado, June. Association for Computational Linguistics.