Aspect-based sentiment analysis

Miha Bizjak, Anže Gregorc, Rok Grmek
University of Ljubljana
Faculty for computer and information science
Večna pot 113, SI-1000 Ljubljana
mb9232@student.uni-lj.si, ag9497@student.uni-lj.si, rg6954@student.uni-lj.si

Abstract

When performing sentiment analysis on texts mentioning multiple entities, the sentiment towards each of them in not necessarily the same, and it is important to determine what sentiment applies to each entity separately. In this paper, we study aspect-based sentiment analysis approaches applied to texts in Slovene. We implement three models. In the first model we use a sentiment lexicon to determine sentiment of words close to an entity, in the same sentence and in the same document and use those as features for a random forest classifier. In the second model, we add a neural model for dependency parsing to the pipeline and construct features based on words close in a sentence's dependency tree instead of sequentially. The third model uses BERT embeddings with a neural classifier to construct embeddings. We evaluate the approaches on the SentiCoref 1.0 corpus of Slovene texts for aspect-based sentiment analysis.

1 Introduction

Online news, forums and social media are a place for everyone to read and write articles and posts across various domains. People can also leave comments and giving their opinion and express their feelings about the topics. That leads to a huge amount of text content. That is probably why natural language analysis is currently a hot topic around the world. We wanted to extract useful information out of large amount of text data.Since we have no time for reading all the words that are written nowadays, we hope to build a good computer program to do that for us. In this project we chose to do aspect-based sentiment analysis. Our task is to get the subjective information from text material that refer to a entity with the use of natural language processing and other methods. An entity is considered as a person, organization or a location and can be represented multiple times in one document or a sentence and there could be more entities in one document.

For the given task, we decided to test multiple approaches and develop different models for predicting the sentiment for each entity. We will first define some really simple models as a starting point, and then we will try to derive some more complex models. All of them will be targeting the Slovene language, and we will evaluate each of the models on the Slovene corpus for aspect-based sentiment analysis - SentiCoref 1.0 (Žitnik, 2019).

2 Related work

The main challenge of entity-based analysis is how to find words that describe the entity and identify if contributes to positive or negative sentiment to a given entity. A lot of related work tried to predict sentiment of the whole document. But in many cases, a text can describe the polarity of more entities. That is why we suggest that sentiment analysis is done on entity level. Since our task is more specific we focused more on methods that identify entities.

In the paper (Ding et al., 2018) they developed an entity-based sentiment analysis SentiSW and tested it on issue comments from GitHub. SentiSW can classify issue comments into <*sentiment, entity*> tuples. They evaluate the entity recognition by manually annotation and it achieves 75% accuracy. The main pipeline of this tool is preprocess (words removal, words replacing, stem), feature vectorize (TF-IDF, Doc2vec), classifier (random forest, bagging and other supervised machine learning methods) and entity recognition (rule-based method).

The use of word embeddings provide powerful methods for semantic understanding without the need of creating large amounts of annotated test data. The paper (Sweeney and Padmanabhan, 2017) enhanced the word embeddings approach with the deployment of a sentiment lexicon-based technique to appoint a total score that indicates the polarity of opinion in relation to a particular entity. They associate a given entity with the words describing it and extracting the associated sentiment to try to infer if the text is positive or negative in relation to the entity.

As stated in the paper (Song et al., 2019), a lot of the existing approaches are modelling context and target words with RNNs and attention. This paper addresses the issues with RNNs and proposes an Attentional Encoder Network for modeling the semantic interactions between target and context words. The paper also addresses the label unreliability issue. The proposed model with the use of pre-trained BERT embedding achieved state-of-the-art results while still being a relatively lightweight model.

Another successful approach that utilizes the BERT model is described in the paper (Sun et al., 2019). In this paper, the authors tried a few methods, where they were generating an auxiliary sentence for each prediction. They basically converted the aspect-based sentiment analysis problem into a sentence-pair classification task.

3 Methods

3.1 Dataset

The SentiCoref 1.0 dataset contains 837 documents with annotations of named entities (31,419 entities in total) and sentiment annotations for each entity. For each entity, a sentiment value from 1 to 5 is assigned. The distribution of sentiment labels in the dataset is shown in Table 1.

Sentiment label	Entity count
1 - Very negative	30
2 - Negative	1801
3 - Neutral	10869
4 - Positive	1705
5 - Very positive	24

Table 1: Entity sentiment distribution in the SentiCoref 1.0 dataset.

3.2 Models

All developed models are described in this section. First two models are really simple and have no practical use - we implemented them as a starting point in order to test the entire evaluation framework, and to have a reference performance for comparing other more complex models.

3.2.1 Random model

The Random model simply assigns random sentiment (1 - 5) to each entity, without knowing anything about the actual sentiment. Assigned sentiments are uniformly distributed, and even the distribution of the sentiments in the dataset was not taken into account. We are expecting that this model would serve as a reference for the lowest possible evaluation performance.

3.2.2 Majority model

The Majority model assigns neutral sentiment (3) to all entities. The idea for this model came from a fact that around 75% of the entities in the dataset are labeled with the neutral sentiment. Therefore, we are expecting decent performance because of the distribution of the sentiments in the dataset. We will also use this performance as a reference for comparing other models. More complex models are expected to extract more knowledge from the given data, and they should be performing better than this one.

3.2.3 Lexicon Features model

For the first machine learning model we tried to build it using hand-crafted features. To do that, we used opinion lexicons with positive and negative words typical for Slovenian language. We used manually translated opinion lexicon used in (Kadunc and Robnik-Šikonja). It contains 62,148 negative word forms and 27,681 positive word forms.

Using SentiCoref dataset and lexicon we implemented different hand-crafted features that we thought they can provide some kind of sentiment indicators for each entity. One sample that is forwarded to machine learning model describes one entity of a particular document with the following features:

- number of positive words up to *n_pos* words before and after (around) current entity occurrence,
- number of negative words up to *n_neg* words around current entity occurrence,
- number of positive words for which the current entity is the closest,

- number of negative words for which the current entity is the closest,
- number of positive words where it is only current entity in a sentence,
- number of negative words where it is only current entity in a sentence,
- number of positive words in the text,
- number of negative words in the text,
- number of different entities in the text,
- number of occurrences of the entity.

Sentence splitting is done using the pretrained Punkt sentence tokenizer (Kiss and Strunk, 2006) for Slovene provided with the `nltk` Python package (Bird et al., 2009).

We tried different numbers of *n_pos* and *n_neg* to represent features. For example, we used [3, 5, 10] for *n_pos* one sample, therefore we generated 3 column features. We also normalized features in some cases. Since the features can have different range of values this can provide better results. In the results section we described what set of features suited us the most.

For a machine learning model we could use a numerous models such as Nearest Neighbours, *SVM*, Naive Bayes. We selected Random Forest (*RF*) model since it performs quite well in most cases. Parameters that we focused for building *RF* are the number of trees in the forest, the function to measure the quality of a split (we tried Gini impurity and information gain) and the maximum depth of a tree. The optimal parameters for the model are chosen using grid-search with 5-fold cross-validation.

3.2.4 Dependency model

The dependency parsing model attempts to improve on the lexicon features model by using word dependency information within sentences.

For dependency parsing, we use the Stanza library (Qi et al., 2020) with the pretrained model for Slovene. The model was trained on the Universal Dependencies Treebank for Slovenian (Dobrovoljc et al., 2017).

The Stanza pipeline takes tokenized sentences as input and outputs sentences annotated with lemmas, part-of-speech tags and dependency information.

Each document is split into sentences, then the sentences are analyzed using the Stanza pipeline to obtain the dependency tree. Using the dependency information, we then calculate distances between each positive/negative word - entity pair. Similar to the lexicon features model, we then construct a feature vector for each entity. We count for how many positive/negative words a specific entity is the closest in the dependency trees, and how many positive/negative words are within N distance in the tree, one feature for positive words and one feature for negative words within each distance.

We expect dependency tree distances to more accurately determine which positive/negative words correspond to which entities compared to regular distances, especially in cases such as nested sentences.

3.2.5 BERT model

The well known BERT (Bidirectional Encoder Representations from Transformers) model (Devlin et al., 2018) can be used for variety of natural language processing tasks, including the sentiment analysis. We decided to utilize the pretrained multilingual BERT embedding layers and combine them with additional convolutional layers and some dense layers at the end for direct classification of the sentiment.

In order to prevent to much variation in the length of inputs (the number of mentions can vary significantly from entity to entity), we presented the idea of treating each mention of an entity as a separate learning case. Therefore we constructed a training set such that each input was represented with a set of words from the neighbourhood of the entity mention, and each output was a sentiment class (1 - 5) of the mentioned entity. In the prediction stage, where the goal is to predict the sentiment for the entity and not for all individual mentions of the entity, we decided to sum predictions over all mentions of the entity, and we select the class with maximum support. The main assumption here was, that all mentions of an entity has similar (hopefully equal) sentiment. Therefore, we may expect some issues if an entity would have extremely negative sentiment in some mentions and extremely positive sentiment in other mentions, but the average sentiment for this entity would be labeled neutral in our dataset. In this case, we would also assign neutral sentiment to all mentions of this entity when constructing the training samples, although some sentiments could be extremely negative or extremely positive.

The architecture of our model is shown in Figure 1. The model is roughly constructed from three parts - first part converts a sequence of words into word embeddings, where the pretrained BERT is used; the second part is convolving with different filters over the sequence of embeddings; and the third part is used for classifying the features, that were extracted with the convolution, into one of the five sentiment classes.

The sequence of words is first extracted from the neighbourhood of the entity mention (parameter *n_words_left_right* defines how many words to the left and right are used). Then before using the BERT embedding layers, we encode the sequence of words and pad the sequence to a fixed length of 128. After obtaining the BERT embeddings, we use three parallel convolutional layers with *conv_filters* number of filters, where the kernels are of sizes 1, 2 and 3 correspondingly. To each filter we apply the max pooling, and then we concatenate the results into a vector of size $3 \cdot$*conv_filters*. The concatenated vector is then put through the dense layer with *dense_units* number of neurons, then we apply the dropout to prevent overfitting, and lastly we use the dense layer with 5 neurons and the softmax activation for the final classification (other neurons are using the relu activation).

The model fitting procedure was done with different dropout rates (parameter *dropout_rate*) in batches of size *batch_size*, and with *epochs* number of training repetitions. During the fitting, the weights from BERT embedding layers were not updated.

3.3 Evaluation

Evaluation was performed on the SentiCoref 1.0 dataset, using a randomized 0.67/0.33 train/test split on the documents. For each model, we measured the F1 score, precision, recall and RMSE metrics. The metrics were calculated using the `sklearn` library with the `average="weighted"` parameter, i.e. calculated for each label and then averaged, weighted by the number of true instances for each label.

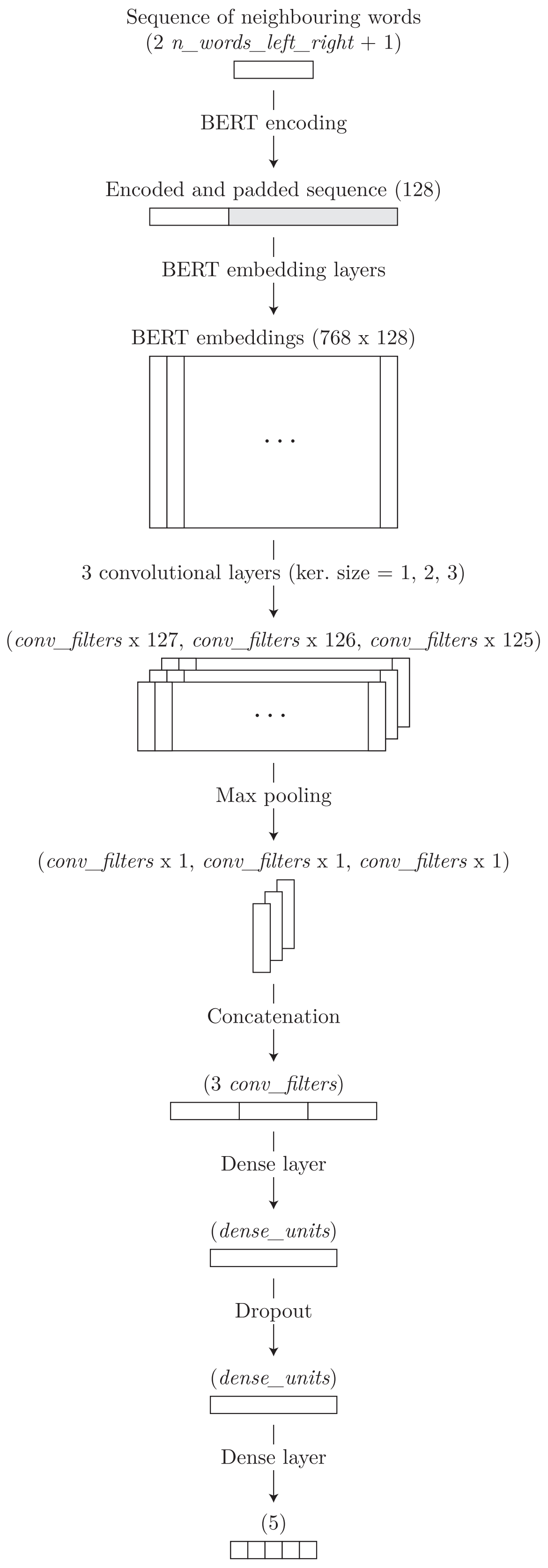
Sequence of neighbouring words
(2 n_words_left_right + 1)
BERT encoding
Encoded and padded sequence (128)
BERT embedding layers
BERT embeddings (768 x 128)
3 convolutional layers (ker. size = 1, 2, 3)
(conv_filters x 127, conv_filters x 126, conv_filters x 125)
Max pooling
(conv_filters x 1, conv_filters x 1, conv_filters x 1)
Concatenation
(3 conv_filters)
Dense layer
(dense_units)
Dropout
(dense_units)
Dense layer
(5)

Figure 1: Architecture of the BERT model.

4 Results

In this section we are presenting the results of evaluation of all tested models. In Table 2, the results of first two simple models are given as a reference for comparing the results of other models, described in the following subsections.

	F1 score	Precision	Recall	RMSE
Rnd	0.2764	0.5976	0.2011	1.4912
Maj	0.6461	0.5661	0.7524	0.5058

Table 2: Evaluation results of the Random model and the Majority model.

4.1 Lexicon Features model

The results of the Lexicon Features model are shown in Table 3 and 4. We can see that results do not vary a lot. If we look closely enough we see that models without normalization performed slightly worse. We could say that the best model is the model with *key* 2. That is not quite surprising since it contains 6 features that indicate positive and negative sentiments around entity and here we also normalize features. All the other models have less number of features (*n_pos* and *n_neg*).

Compared to Random and Majority model we can conclude that our hand-crafted features provide some information about entity sentiment. Lexicon Features model performed better on all the metrics compared to Random model. The Majority model performed slightly better when we talk about Recall and RMSE most probably because we have very imbalanced dataset. Despite the fact that the dataset is imbalanced, all the variations of Lexicon Features have better F1 score and Precision so we would still rather use Lexicon Features model in real world.

key	F1 score	Precision	Recall	RMSE
1	0.6742	0.6676	0.7477	0.5312
2	0.6768	0.6827	0.7522	0.5225
3	0.6764	0.6730	0.7498	0.5272
4	0.6768	0.6719	0.7479	0.5261
5	0.6730	0.6673	0.7467	0.5296
6	0.6689	0.6608	0.7447	0.5298

Table 3: Evaluation results of the Lexicon Features model. Because the table would be too wide we split the table in two parts. The other part is in Table 4 and describes parameters for this results.

4.2 Dependency model

The results for the dependency model are shown in Table 5.

The model performs similarly to the Lexicon Features model. With the best set of parameters (model 6) we achieve only slightly better values compared to the Lexicon Features model, which shows that in certain cases using the dependency tree neighborhood helps assign the sentiment to the correct entity. Despite that, we still fail to achieve better values than the Majority model for Recall and RMSE, and calculating the dependencies for each sentence requires additional computation.

key	*n_pos*	*n_neg*	Norm
1	(5)	(5)	F
2	(3,5,10)	(3,5,10)	T
3	(3,5,10)	(3,5,10)	F
4	(3,5,10)	(3)	T
5	(5)	(3,5,10)	T
6	(3,15)	(10,15)	F

Table 4: Parameters for each model shown in Table 3. The parameter norm represents weather we normalized features before we put to the model (T) or not (F). The parameters *n_pos* and *n_neg* are described in Models section. In summary, if we have 3 numbers in *n_pos* column, that means we used 3 features (columns in data set) that count positive words around entity.

key	F1 score	Precision	Recall	RMSE
1	0.6733	0.6612	0.7396	0.5359
2	0.6765	0.6676	0.7441	0.5328
3	0.6767	0.6653	0.7435	0.5369
4	0.6739	0.6679	0.7439	0.5277
5	0.6767	0.6721	0.7476	0.5302
6	0.6811	0.6784	0.7493	0.5245

Table 5: Evaluation results of the dependency model. The parameters used for each model are shown in Table 6

4.3 BERT model

The model was developed with variety of free parameters. We tested different configurations in order to see the impact of each parameter on the model performance. For the starting point we set the parameters as follows: *n_words_left_right*: 6, *conv_filters*: 100, *dense_units*: 256, *dropout_rate*: 0.2, *batch_size*: 128, *epochs*: 5. Then we evaluated several experiments were we were changing one parameter at a time and left all others fixed. All the results are given in Tables 7, 8, 9, 10, 11, 12 and in Figure 2.

Similar trade-off between precision and recall is seen in all experiments. Parameter configurations, evaluated with higher recall (bot lower precision) are performing similarly as the Majority model - most predictions are targeting the neutral sentiment and very few predictions are made

key	*N*	Norm
1	(3)	T
2	(3)	F
3	(1,2,3)	T
4	(1,2,3)	F
5	(1,2,3,4,5)	T
6	(1,2,3,4,5)	F

Table 6: Parameters for each model shown in Table 5. The parameter norm represents weather we normalized features before we put to the model (T) or not (F). The parameter *N* represents the list of distances used to calculate sentiment close in the dependency tree.

	F1 score	Precision	Recall	RMSE
1	0.6459	0.5844	0.7503	0.5072
2	0.6541	0.6131	0.7495	0.5124
3	0.6470	0.6103	0.7474	0.5126
4	0.6628	0.6348	0.7250	0.5544
5	0.6631	0.6383	0.7233	0.5597
6	0.6734	0.6519	0.7114	0.5812
7	0.6692	0.6465	0.7083	0.5775

Table 7: Evaluation results of the BERT model with respect to parameter *n_words_left_right*.

for the other four sentiment classes. The other extreme with higher precision (but lower recall) more likely predicts the non-neutral classes, and in most cases this also results in an overall higher F1 score. In this kind of trade-off situation we may want to simply trust the F1 score for determining the best performing parameter configuration, as it takes both precision and recall into account, but we noticed an interesting correlation between the F1 score and the RMSE. Higher F1 score in most cases also resulted in a higher RMSE, which means more error according to the distance between the predicted and the actual class. This means that one extreme (both F1 score and RMSE high) will more likely predict the edge sentiment classes, but in this case we will more likely fail in some situations when we are dealing with the neutral sentiment. Another extreme (both F1 score and RMSE low) will offer a more stable model with more predictions quite close to an actual class, but this model won't be so informative as it may never predict the edge sentiment classes.

5 Discussion

In this paper we studied the problem of selecting sentiment for different entities in documents provided from dataset SentiCoref 1.0. We found several challenges to this problem and proposed different solutions to address them. In particular, we implemented 3 different approaches of classifying the aspect-based sentiments, one with hand-crafted features, the model with word dependency information and the model built upon the state-of-the-art BERT model. All the models re-

	F1 score	Precision	Recall	RMSE
50	0.6596	0.6455	0.7330	0.5527
100	0.6734	0.6519	0.7114	0.5812
150	0.6625	0.6364	0.7068	0.5835

Table 8: Evaluation results of the BERT model with respect to parameter *conv_filters*.

	F1 score	Precision	Recall	RMSE
64	0.6461	0.5661	0.7524	0.5058
128	0.6585	0.6305	0.7262	0.5527
256	0.6734	0.6519	0.7114	0.5812

Table 9: Evaluation results of the BERT model with respect to parameter *dense_units*.

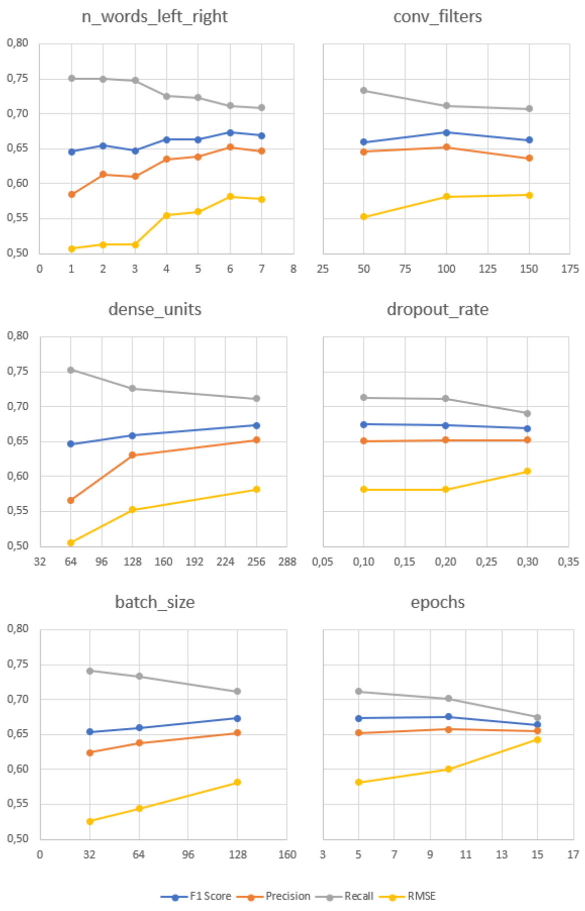

Figure 2: Evaluation results of the BERT model.

	F1 score	Precision	Recall	RMSE
0.1	0.6745	0.6510	0.7131	0.5815
0.2	0.6734	0.6519	0.7114	0.5812
0.3	0.6686	0.6525	0.6905	0.6077

Table 10: Evaluation results of the BERT model with respect to parameter *dropout_rate*.

	F1 score	Precision	Recall	RMSE
32	0.6533	0.6237	0.7410	0.5260
64	0.6592	0.6380	0.7334	0.5440
128	0.6734	0.6519	0.7114	0.5812

Table 11: Evaluation results of the BERT model with respect to parameter *batch_size*.

sulted to quite similar values (F1 score varies from around 0.64 to 0.68) so we can not select one best model. Our conclusion is that we can use a model based on our preferences. If we wanted to pick model based on F1 score we could use Dependency model but if we wanted less error according to the sentiment distance we would check on RMSE measure and pick BERT model.

Because of the distribution of the data, we observe that our models tend to learn to overpredict the neutral sentiment class, which minimizes the amount of significant mistakes, however it also means that the accuracy does not significantly improve compared to the majority model.

In further experiments, we could attempt to combine the hand-crafted features and the BERT embeddings. In the hand-crafted features we could add more different features that hopefully contain additional information of sentiment for a particular entity. Additionally, undersampling could be performed on the neutral class to improve the training of the models. We could also expand our dataset with different languages and analyse how models behave in multi-language dataset.

6 References

References

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.".

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

	F1 score	Precision	Recall	RMSE
5	0.6734	0.6519	0.7114	0.5812
10	0.6750	0.6570	0.7015	0.6005
15	0.6636	0.6548	0.6743	0.6430

Table 12: Evaluation results of the BERT model with respect to parameter *epochs*.

Jin Ding, Hailong Sun, Xu Wang, and Xudong Liu. 2018. Entity-level sentiment analysis of issue comments. In *Proceedings of the 3rd International Workshop on Emotion Awareness in Software Engineering*, pages 7–13.

Kaja Dobrovoljc, Tomaž Erjavec, and Simon Krek. 2017. The universal dependencies treebank for slovenian. In *Proceedings of the 6th Workshop on Balto-Slavic Natural Language Processing*, pages 33–38.

Klemen Kadunc and Marko Robnik-Šikonja. Analiza mnenj s pomocjo strojnega ucenja in slovenskega leksikona sentimenta.

Tibor Kiss and Jan Strunk. 2006. Unsupervised multilingual sentence boundary detection. *Computational linguistics*, 32(4):485–525.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Youwei Song, Jiahai Wang, Tao Jiang, Zhiyue Liu, and Yanghui Rao. 2019. Attentional encoder network for targeted sentiment classification.

Chi Sun, Luyao Huang, and Xipeng Qiu. 2019. Utilizing BERT for aspect-based sentiment analysis via constructing auxiliary sentence. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 380–385, Minneapolis, Minnesota. Association for Computational Linguistics.

Colm Sweeney and Deepak Padmanabhan. 2017. Multi-entity sentiment analysis using entity-level feature extraction and word embeddings approach. In *RANLP*, pages 733–740.

Slavko Žitnik. 2019. Slovene corpus for aspect-based sentiment analysis - SentiCoref 1.0. Slovenian language resource repository CLARIN.SI.