Aspect-based Sentiment Analysis on the SentiCoref 1.0 Corpus

Matevž Fabjančič and **Ela Praznik**
University of Ljubljana, Faculty of Computer and Information Science

Abstract

Identifying the sentiment of texts found on the internet is a common research topic. In this work, we discuss existing methods for sentiment analysis. We propose new classical and neural machine learning models for sentiment classification. We test and evaluate our models on the SentiCoref 1.0 data set. We present results and discuss weaknesses and possible improvements of the proposed models.

1 Introduction

The growing amount of data present on the world wide web, such as news articles, posts on social media and forum threads, calls for methods of information extraction. One type of such information is the sentiment a subject expresses towards an object. Such information can be useful when modelling relationships between users of social media platforms or political bias of news articles.

Sentiment analysis, also referred to as opinion mining and subjectivity analysis, combines information retrieval, natural language processing and artificial intelligence. It is formally expressed as finding a tuple of the form (s, g), where s represents the sentiment and g the target object for which the sentiment is expressed. Aspect-based sentiment analysis is a sub-field focusing on finding and aggregating sentiment of entities or aspects of them. An aspect can be any characteristic or property of the entity. It is used to differentiate sentiment on a more fine grained basis in comparison to document or sentence level sentiment analysis. A single sentence can contain different entities with different sentiments expressed towards them. We can in general differentiate three processing steps: identifying sentiment-target pairs in text, classifying these pairs and aggregating sentiment values for each aspect. The usual concerns of robustness, flexibility and speed of the models are also to be kept in mind (Schouten and Frasincar, 2015).

Traditionally solutions have been divided into lexicon-based and machine learning approaches, either unsupervised or supervised, with the latter sometimes including lexicon-based information as well. However in recent years with the proliferation of word-embedding algorithms there have been several examples of hybrid models. These usually attain a higher degree of accuracy on different sentiment analysis tasks. Bellow we discuss a few recent approaches.

In (Ding et al., 2018), Ding et al. have developed a sentiment analysis tool SentiSW, which classifies texts related to software development. This system was evaluated on a corpus of GitHub repository issues, which they have manually annotated. Their main goal was to predict a binary sentiment (positive, negative) and whether this sentiment is expressed towards another developer (person) or the project. SentiSW uses document vectorization approaches for feature extraction, supervised machine learning algorithms, such as random forest and support vector machines for training and sentiment classification, while using rule based methods for entity recognition.

An example of deep neural network approach is presented in (Tang et al., 2016). They propose neural network architectures for learning word embeddings which capture context of words and sentiment separately. They also combine both embeddings into a hybrid model, which captures both the word context and sentiment. Their evaluation has shown that the hybrid approach yields best results among the traditional embeddings (e.g. Word2Vec) and proposed neural embeddings.

An example of a sentiment lexicon-based technique combined with a word embeddings approach has been implemented by (Sweeney and Padmanabhan, 2017). The goal was to develop a binary classifier of tweet sentiment (positive

or negative) toward each related entity. The dataset used for training and testing contained more than a million classified tweets with sentiment labels. Single entity tweets were processed by the Word2Vec algorithm producing distributed vector representations for tweet words and using them for scoring sentiment using a Random Forest classifier. Tweets containing more than one entity were first processed separately using a Tweet specific parser, TweeboParser, followed by extraction of neighbouring descriptor words (Adverbs, Adjectives and Verbs) and sentiment identification of said descriptor words using SentiWordNet lexicon. The descriptor words were further used to score sentiment relating to that entity, allowing for identification of different polarity contained within one tweet. For comparison, a baseline classifier(which the model outperformed) using only the lexicon-based approach was used on the same dataset.

(Biyani et al., 2015) tackled the problem of entity-specific sentiment classification (positive, negative and neutral) in Yahoo news comments. Besides the challenge of identifying irrelevant entities, sentiments in news comments are additionally difficult to classify as they deal with a variety of different domains. The researchers extracted entities with Stanford Named Entity Recognizer, followed by linking each entity targets with its sentiment context. Extracting the context made use of several heuristics. Classification was composed of two steps: first, the context of an entity was classified into polar vs. neutral (irrelevant), using content, lexical and several non-lexical features, and second, the polar entities were classified into positive or negative based on comment-specific features. Several supervised machine learning algorithms were used to implement the classifiers, with Logistic Regression giving the best results in the first step and Naive Bayes in the second. The methods outperformed several baselines.

Our project focuses on evaluation of the SentiCoref 1.0 corpus (Žitnik, 2019). This corpus contains a subset of articles from SentiNews 1.0 corpus (Bučar, 2017a). SentiCoref corpus contains annotations of named entity tags, coreference chains, and a 5-level sentiment for each coreference chain. We develop and evaluate a pipeline for predicting sentiment values for each coreference chain.

2 Methods

2.1 Data preprocessing

We preprocess the data from SentiCoref using Stanza (Qi et al., 2020). We apply built in model for POS tagging on the already tokenized text. In order to be able to use sentiment lexicons, we lemmatise words using the lemmatisation model provided in (Ljubešić, 2020).

In addition to lemmas and POS tags we augment the SentiCoref dataset with word sentiments and sentence sentiments (added to each word). The sentence sentiments are taken from the SentiNews data set. Linking the two data sets has proven to be a difficult task, since the two data sets are not organised in the same way. We do sentence alignment by simply observing punctuation. In some cases this leads to misalignment of the sentences, and should be improved on.

2.2 Feature extraction

After the preprocessing step is completed, we extract features from the data. These features are extracted for each occurrence of an entity in a coreference chain. We construct several different features using the sentiment lexicon.

- Entity type. This feature represents the type of some named entity in a coreference chain (person, organisation, location).
- Sentence sentiment. The sentence sentiment provided in SentiNews data set for some word.
- Sum of sentiments of context words given a context size N, the sentiments of N left and N right words are summed together. This feature is used by only taking nouns, adjectives and verbs into consideration. As the lexicon JOB 1.0 (Bučar, 2017b) was used.
- POS tags of context words
- Sentiments of context words
- Sentiments of other entity references in that sentence. The number of positive (> 3), negative (< 3) and the ratio positive/negative entity reference's sentiments as evaluated in the SentiCoref dataset.
- Sum of sentiments (positive, negative, positive/negative) of context words in a sentence

Target sentiment	Majority RMSE	PerWordModel RMSE
1	2.00	1.83
2	1.00	0.86
3	0.00	0.20
4	1.00	0.98
5	2.00	1.74
All	0.51	0.62

Table 1: Root mean squared errors for the PerWordModel regression model, compared with a majority baseline.

scored based on the lexicon JOB 1.0 (Bučar, 2017b).

- Ratio of sum of positive to the sum of negative sentiment of all the words in a coreference chain scored based on the lexicon JOB 1.0 (Bučar, 2017b).

Non-numeric features were binarized for use in regression machine learning algorithms. Furthermore, for regression algorithms, some features were combined by multiplication (POS tag and word sentiment, for example) in order to enrich the feature space.

2.3 Per-word sentiment prediction using linear regression (PerWordLinear)

This algorithm uses linear regression on features presented in section 2.2.

Training was done on 80% of the coreference chains using the linear regression provided by the Scikit-learn (Pedregosa et al., 2011) library for Python. The remainder was used for model evaluation. Results are shown in table 1. In comparison with the baseline model, which always predicts sentiment *3 - Neutral*, it yields overall worse results, as indicated by the RMSE measure. However, the data set is significantly unbalanced – 75% of the coreference chains are annotated as *3 - Neutral*. By observing RMSE measures within each group of coreference chains with the same sentiment labels, we can see that our algorithm yields marginally better results than the baseline model. It seems that the vast majority of neutral entities prevents linear regression from adapting to entities with different class labels.

Label	Count
1 - Very negative	29
2 - Negative	1856
3 - Neutral	10634
4 - Positive	1785
5 - Very positive	27

Table 2: Distribution of entities with respect to their sentiments.

2.4 Data set balance

Significant lack of balance in the data set, as shown in table 2, proved to be an issue for model training. To improve trained model quality some sort of data selection was needed. In our work we use the following two approaches.

Label joining. In all of the following models, we joined labels *1 – Very negative* and *2 – Negative* into one, and labels *4 - Positive* and *5 - Very positive* into one. Reason for this is the lack of entities with labels *1 – Very negative* and *5 - Very positive*.

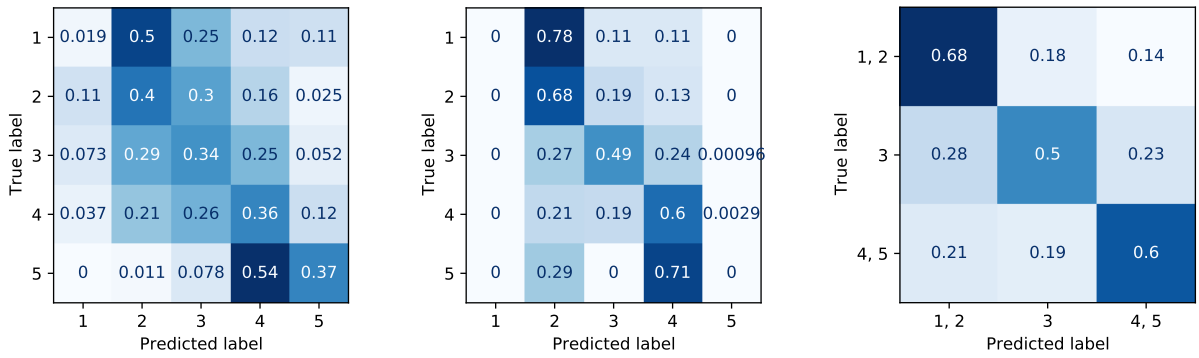
Data balancing. In some models (indicated for each model), we also balanced the data set with respect to aforementioned joined labels. This was done by sub-sampling an amount of coreference chains equal to the amount of entities with the least-represented sentiment.

2.5 Per-word sentiment prediction using a random forest classifier (PerWordRF)

In this approach, we tried to minimize the issue of balance in the data set by sub-sampling coreference chains with sentiment labels 2, 3 and 4 in a way that yielded samples of equal size. This could not be done for labels 1 and 5 as they are scarce.

This algorithm uses a random forest classifier with default parameters implemented in the Scikit-learn library for Python. Training was done on 80% of the coreference chains, the remainder was used for evaluation.

Figure 1a shows the confusion matrix of predictions yielded by this algorithm. We can observe that the model produces somewhat accurate predictions. However, most of the entities in coreference chains labelled as *1 - Very negative* and *5 - Very positive* are mistaken for *2 - Negative* and *4 - Positive* respectively. Most other mistakes are made between neighbouring classes, which further confirms that our features are informative.



(a) Confusion matrix using the **per-word** RF model. Rows of the matrix sum to 1.

(b) Confusion matrix using the **per-chain** RF model. Rows of the matrix sum to 1.

(c) Confusion matrix using the **per-chain** RF model using label joining

Figure 1: Confusion matrices for per-word and per-chain RF classifiers

2.6 Per-chain sentiment prediction using a random forest classifier (PerChainRF)

This model is exactly the same as the per-word RF model. The only difference is how the data is handled before training. In this case the features extracted in neighbourhoods of each occurrence are aggregated. This process yields a single data point for each coreference chain, as opposed to multiple in the per-word RF model.

Confusion matrix for this approach is shown in figure 1b. We can observe similar performance. The main differences come from the fact that there are fewer data points overall, which makes results less smooth. This phenomenon is most apparent for labels 1 and 5.

In figure 1b it is clear that the model is unable to discern between labels *1, 2* and *4, 5*. Therefore, we applied the label joining technique described in section 2.4. This model was used to obtain final results reported in table 3. Its confusion matrix is shown on figure 1c.

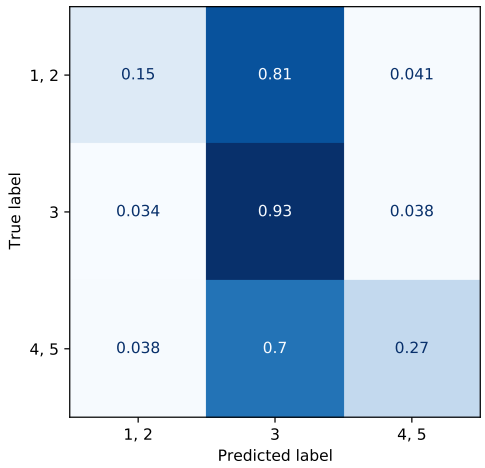
2.7 Neural approach using BERT for tokenization (CustomSentiCorefModel)

In order to capture more complex relationships within sentences a neural model was constructed. We used a pretrained BERT tokenizer, trained on Slovene, Croatian and English, to transform data into vectors suitable as inputs to a neural network. Texts for BERT tokenization were constructed by identifying sentences containing some entity. These sentences were concatenated. The whole sequence was then fed to the BERT tokenizer. This gave us vectors of size 512, which served as inputs to the neural network.

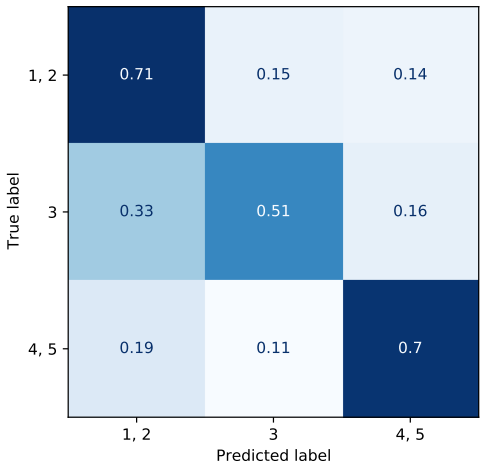
The neural network consists of 2 convolutional layers, with filter sizes 2, 3. Each uses 128 filters. A pooling layer is attached to each of the convolutional layers. Following the convolutional layers is a fully connected layer of length 128. We experimented with different dropout rates. The dropout rate had negligible impact on performance on test data set. Results are accumulated using *softmax*.

Figures 2a shows the confusion matrix of the neural model. We used label joining described in section 2.4 on the training data. We can see that most of the entities are predicted as *3 - Neutral*. However, for labels *(1,2)* and *(4,5)* there is an increase of correct predictions. This indicates that the model captures some information regarding the sentiment of entities.

To get a better idea of how this model performs, we used data balancing on described in section 2.4 on training data in addition to label joining. This yielded in a perfectly balanced but smaller training data set. Figure 2b shows the confusion matrix for this model. We can observe that the model can in fact recognize negative and positive sentiments for entities. However, it seems that neutral sentiment is harder to distinguish. From confusion matrix 2b we can see that predictions for neutral entities lean towards the negative ones. This might indicate that the variety of coreference chains marked as *3 - Neutral* is much greater than the one of negative and positive coreference chains. Furthermore, it might indicate that the difference between negative and neutral sentiments is smaller than the difference between neutral and positive sentiments.



(a) Confusion matrix using the neural model **without** data balancing. Rows of the matrix sum to 1.



(b) Confusion matrix using the neural model **with** data balancing. Rows of the matrix sum to 1.

Figure 2: Confusion matrices for CustomSentiCorefModel.

2.8 Neural model using BERT embeddings (BertEmbeddingsSentiCoref)

For our other neural approach we decided to replace the embedding layer with BERT embeddings constructed from our dataset. We used the same pretrained BERT model trained on Slovene, Croatian and English, as above. The inputs were prepared as stated above yielding vectors of size 512.

We used the last layer of weights from BERT as the embedding layer. It was not trained during the training phase as this would have given worse results because of the relatively small dataset. Next the neural network consisted of 3 convolutional layers with a pooling layer attached to each, followed by a fully connected layer. Results were accumulated using softmax. The model was trained on Google Colab GPU. The same label joining and balancing of training data as described above applied. We started with balancing also the test set, however later decided to keep the test data as intact as possible.

For the first run with the balanced test set we used 100 filters for each of the convolutional layers, a dense layer of the length 256 and dropout rate 0.2. We ran 10 epochs. The confusion matrix can be seen in Figure 3a. The model predicts the labels (1,2) and (4,5) with better accuracy than neutral. Training for the second time we lowered the number of epochs to 6 due to the rather long training time. The confusion matrix can be seen in Figure 3a.

3 Results

In table 3 we present an overview of our models. We report on precision, recall and F1 measures for a baseline model and 3 implemented models. We report on two different methods of aggregating per-class measures.

4 Discussion

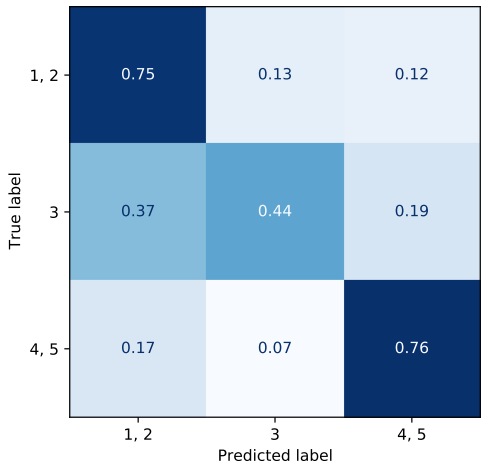
4.1 The data

While analysing the SentiCoref 1.0 data set, we encountered some inconsistencies in the data. For example, some coreference chains do not have sentiment annotations. We could annotate these chains as *3 - Neutral*. However, since a vast majority of the entities are neutral, we decided against assigning the neutral sentiment to them.

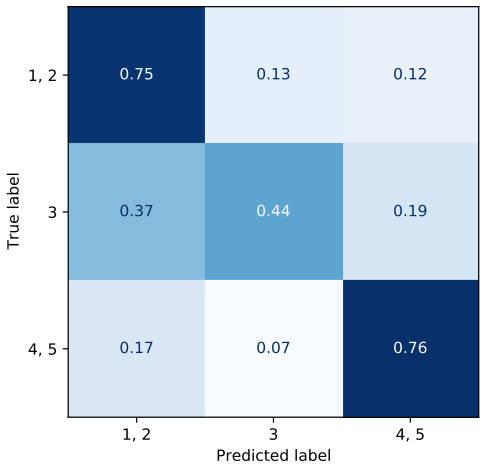
Data inbalance is antoher problem that required attention. In section 2.4 we describe the balancing process. In results table 3 we can observe that the result metrics are highly dependent on the method of averaging per-class scores. It would be worth exploring more robust data set balancing techniques.

4.2 Syntactic dependencies within sentences

Approaches presented in sections 2.3 and 2.6 are lacking in numerous aspects. For example, they do not take into account any syntactic dependencies between words of sentences. This can hinder the performance especially in sentences which contain multiple entities and therefore apply to multiple coreference chains (Sweeney and Padmanabhan,



(a) Confusion matrix using BertEmbeddingsSentiCoref with test set balanced. Rows of the matrix sum to 1.



(b) Confusion matrix using BertEmbeddingsSentiCoref with test set unbalanced. Rows of the matrix sum to 1.

Figure 3: Confusion matrices for the BertEmbeddingsSentiCoref model tested on balanced and unbalanced test set.

Model		Macro Average Precision	Macro Average Recall	Macro Average F1	Weighted Average Precision	Weighted Average Recall	Weighted Average F1
PerChainDummy		0.25	0.33	0.29	0.56	**0.75**	0.64
PerChainRF	2.6	0.47	0.59	0.46	**0.74**	0.53	0.57
CustomSentiCorefModel	2.7	0.48	0.59	0.49	**0.74**	0.58	0.62
BertEmbeddingsSentiCoref	2.8	**0.66**	**0.65**	**0.64**	0.66	0.65	**0.65**

Table 3: Comparison of performance for different models on the SentiCoref 1.0 data set. All of the models use label joining presented in section 2.4. Best scores are highlighted in bold.

2017). In future methods improving on this by augmenting our the data set with such information would be required.

4.3 Polarity shifters

For our feature based models the inclusion of so called polarity shifters ((Xia and Cambria, 2016)) such as negation marking. Such data augmentation would help us to gain more information about the correct polarity of an entity as we can suspect that in our current approach entities with negated positive words were labeled as positive instead of negative. Indeed there exists several sources of negation words in Slovene that could be used in the future to enrich our data (e.g. (Ilc, 2008)).

4.4 Neural networks, BERT

In the future we could try more hyper-parameters and longer training time for our neural models. Another possibility would be to fine-tune the pre-trained Slovene BERT on SentiNews 1.0 corpus from where SentiCoref is derived. This could capture the representation of BERT embeddings better and therefore potentially aid classification.

References

Prakhar Biyani, Cornelia Caragea, and Narayan L. Bhamidipati. 2015. Entity-specific sentiment classification of yahoo news comments. *CoRR*, abs/1506.03775.

Jože Bučar. 2017a. Manually sentiment annotated slovenian news corpus SentiNews 1.0. Slovenian language resource repository CLARIN.SI.

Jože Bučar. 2017b. Slovene sentiment lexicon JOB 1.0. Slovenian language resource repository CLARIN.SI.

Jin Ding, Hailong Sun, Xu Wang, and Xudong Liu. 2018. Entity-level sentiment analysis of issue comments. pages 7–13.

Gašper Ilc. 2008. O zanikanju in nikalnici v slovenščini. *Jezik in slovstvo*, 53(2):65–79.

Nikola Ljubešić. 2020. The CLASSLA-StanfordNLP model for lemmatisation of standard slovenian 1.1. Slovenian language resource repository CLARIN.SI.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A python natural language processing toolkit for many human languages.

Kim Schouten and Flavius Frasincar. 2015. Survey on aspect-level sentiment analysis. *IEEE Transactions on Knowledge and Data Engineering*, 28(3):813–830.

Colm Sweeney and Deepak Padmanabhan. 2017. Multi-entity sentiment analysis using entity-level feature extraction and word embeddings approach. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 733–740, Varna, Bulgaria. INCOMA Ltd.

D. Tang, F. Wei, B. Qin, N. Yang, T. Liu, and M. Zhou. 2016. Sentiment Embeddings with Applications to Sentiment Analysis. *IEEE Transactions on Knowledge and Data Engineering*, 28(2):496–509.

Feng Xu Jianfei Yu Yong Qi Xia, Rui and Erik Cambria. 2016. Polarity shift detection, elimination and ensemble: A three-stage model for document-level sentiment analysis. *Information Processing Management*, 52(1):36–45.

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