DS at SemEval-2019 Task 9: From Suggestion Mining with neural networks to adversarial cross-domain classification

Tobias Cabanski
t.cabanski@posteo.de

Abstract

Suggestion Mining is the task of classifying sentences into suggestions or non-suggestions. SemEval-2019 Task 9 sets the task to mine suggestions from online texts. For each of the two subtasks, the classification has to be applied on a different domain. Subtask A addresses the domain of posts in suggestion online forums and comes with a set of training examples, that is used for supervised training. A combination of LSTM and CNN networks is constructed to create a model which uses BERT word embeddings as input features. For subtask B, the domain of hotel reviews is regarded. In contrast to subtask A, no labeled data for supervised training is provided, so that additional unlabeled data is taken to apply a cross-domain classification. This is done by using adversarial training of the three model parts label classifier, domain classifier and the shared feature representation. For subtask A, the developed model archives a F1-score of 0.7273, which is in the top ten of the leader board. The F1-score for subtask B is 0.8187 and is ranked in the top five of the submissions for that task.

1 Introduction

For getting feedback from costumers or users, an organization often uses forums and social media channels. Also ratings of products on rating platforms can be an useful feedback to make a product better. The feedback from a customer can be in the form of a suggestion which appears in a rating text or is directly asked from the customer. The task of suggestion mining can be defined as the extraction of sentences that contain suggestions from unstructured text (Negi et al., 2018). SemEval-2019 Task 9 Subtask A provides the challenge to do suggestion mining on data from an online suggestion forum. For that subtask, a train and validation set is provided so that it is possible to apply supervised training. For subtask B, suggestions in hotel reviews should be identified. An additional difficulty for that subtask is that no labeled data is given except a small validation set, which is not allowed to be used for supervised training. For both tasks, silver standard datasets are allowed to use, which means that data that is likely to belong to a certain class can be taken as long as it is not manually labeled. A more detailed task description can be found in (Negi et al., 2019).

2 Data

The dataset for subtask A provides an overall count of 8500 examples, where 6415 examples are labeled as non-suggestion and 2085 as suggestion. Also a trial dataset is provided that contains 592 examples, divided in 296 examples for each class. Every example contains only one sentence, which could be part of a whole post in the forum where it was extracted.

The domain of software suggestion forum posts in general provides more balanced data than other domains, for example hotel reviews. Also the domain contains very specific vocabulary which is frequently used in software development, so that it can be difficult to use a trained model of this domain for other domains (Negi et al., 2018).

For subtask B, only a validation dataset is provided. The set contains an overall count of 808 examples with 404 examples for each class. As mentioned in the introduction, it is not allowed to use the validation data for supervised learning for this subtask. The data is only allowed to be used for model evaluation and error analysis and also for automatic hyperparameter tuning. The presented solution in this paper uses the validation data for early stopping at a fixed count of train steps after the best score is reached. The model state at the best score is then returned and used for the predic-

Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019), pages 1192–1198
Minneapolis, Minnesota, USA, June 6–7, 2019. ©2019 Association for Computational Linguistics

tion of the test data.

For both subtasks, additional data is allowed that is not manually labeled. In this work, the hotel review dataset, which is presented in (Wachsmuth et al., 2014), is used to apply cross-domain classification for subtask B. The dataset comes with nearly 200k examples of hotel reviews without labels.

3 Related Work

The task of text classification improved a lot during the last years. In the past, machine learning techniques like support vector machines were used to assign a class to a text. In (Joachims, 1998), a text classification with support vector machines is presented. For that, a document vector is extracted for the whole text and used as the feature vector for the classification.

Since such methods can provide a good base line today, the increasing popularity of neural network approaches provides new methods that can classify texts more exactly. Especially the introduction of word embeddings in (Mikolov et al., 2013) was a big step forward in the field of text processing and opened new opportunities for many natural language processing tasks. Also for the task of text classification, word embeddings are useful features and can lead to good results. Since the release of these word embeddings, many other word embedding approaches have been introduced. A very recent one is shown in (Devlin et al., 2018) and is called BERT: Bidirectional Encoder Representations from Transformers. These embeddings are the result of the training process of transformer, which is described in (Vaswani et al., 2017) and delivers a state-of-the-art method for different natural language generation tasks, especially for translation.

To use the word embeddings as features for text classification, a commonly used approach is Long-short term memory (LSTM), which is described in (Hochreiter and Schmidhuber, 1997). The advantage of using LSTM cells over support vector machines as classifier is the processing of features in time steps. By passing a single word embedding into a single time step of the LSTM, every feature is processed separately. Since the features are processed one after the other, also the order of the features has influence on the classification process. In addition to that, LSTM cells have a state that enables them to save information for many previous time steps. For text classification, this can be useful when there are connections between words in a text that are far apart.

Another method to process word embeddings are convolutional neural networks (CNN), which are introduced in (LeCun and Bengio, 1998). With the ability to extract features of two-dimensional input data by defining a sliding window of variables, the method is often used for image processing. But also good results for text classification are reported, for example in (Kim, 2014). The results show that even a simple CNN with one layer performs very well and a tuning of hyperparameters brings an improvement of the performance.

For subtask B, the focus gets in the direction of methods for cross-domain classification. By the introduction of Generative Adversarial Nets (GAN) in (Goodfellow et al., 2014), a new way for training neural networks was provided that leads to new opportunities for different task, especially image generation. In (Chen and Cardie, 2018) is shown that this training technique could also be used for cross-domain classification of texts of different domains. As the main four components, a shared feature extractor, a domain feature extractor, a text classifier and a domain discriminator are introduced. The main goal of that system is to learn a domain invariant feature representation by training the shared feature extractor with the discriminator and the text classifier. The discriminator learns to separate the domains and the training goal for the shared feature extractor is to increase the loss of the discriminator. The extracted features become invariant for the domains, so that the text classifier results improve for the domain where no labels are given.

4 Models

In this section, the models for subtask A and B are presented. The overall idea of the model for subtask A is using an ensemble of LSTM and CNN networks. As input features, pre-trained BERT embeddings for the texts are used.

For subtask B, the idea is to extend the model from subtask A with a domain discriminator and shared features. Since that adds a lot of parameters to the model, the text classifier has a simpler structure than in subtask A and uses only CNNs for classification. The full TensorFlow implementation of the models can be found at GitHub.[1]

[1] https://github.com/tocab/semeval2019Task9

4.1 BERT embeddings

For both subtasks, BERT embeddings are used to create a representation of the text. The TensorFlow implementation, which is openly available and comes with pre-trained multi-language embeddings, is taken for that.[2] The model for creating the embeddings is the small uncased model, which has been trained on lower cased wikipedia texts. It has a total count of twelve layers and a layer size of 768 in each hidden layer. The whole model has a 110 million parameters in total.

To extract the embeddings out of the model, the text gets tokenized and mapped into a sequence of integers by using the vocabulary of the pre-trained model. This representation is then given into the network, where it passes the different transformer layers. The embeddings are delivered by the hidden layers of the model. In this project, the output from the last four layers before the output layer is taken as the representation of a word, so that every word is represented by a vector of the shape $(4, 768)$.

4.2 Subtask A

For subtask A, a text classification ensemble of LSTM and CNN is built. To bring all sentences to the same length, a maximum sequence length of 40 is defined. With that sequence length, for around 95% of all train data sentences all words are taken as input. Only for the remaining 5% which have more than 40 words, the texts are cut to the maximum sequence length. If a text is shorter than 40 words, it is filled with zeros. Using the batch size of 64, the input shape for a batch for the training process is $(64, 40, 768)$ for every of the four extracted input features from BERT.

One problem of the data is the imbalance of the classes. When taking random batches out of the whole dataset, it is likely that the count of one class is always higher than of the other. The algorithm learns to predict the class with the higher example count with a higher probability. To avoid that, the technique of oversampling is applied to the training process. The data is separated into two sets, each for every of the two classes and then fed into the network alternately. When all examples of the class with the lower count were used as training input, the set gets repeated so that these examples occur more often as training input.

[2] https://github.com/google-research/bert

The model structure for subtask A can be divided into the three following main parts:

- Processing of single words with dense layers.
- Processing of the whole text with LSTM cells.
- Processing of sliding windows through the text with CNN.

For the processing of the dense layer and the LSTM, separated graphs are created for each of the input features from BERT. As mentioned before, four embeddings for a word are gathered, each of a different hidden layer of the transformer. Thus four graphs of dense layers and LSTM layers are created, each for processing a different embedding type of the input text.

The first step is a transformation of every single input word with a dense layer. This approach is applied to focus on single signal words that can occur in the text. For example, the occurrence of the word *recommend* could be a hint for a suggestion, without regarding other words. The outputs of the single word processings are concatenated and forwarded to a dense layer to reduce the dimension.

The LSTM is represented by two cells to realize a bidirectional approach. The output of the two cells is concatenated and followed by a `GlobalMaxPool`-Layer, which takes the maximum of the output's timestep axis to bring it into a one-dimensional representation. To do a further feature transformation, a dense layer is applied to the output. The result is concatenated to the output of the previous described single word approach.

Unlike the single word processing dense layer and the LSTM, the CNN approach processes all four BERT features for the words in the same network. When using CNNs for image processing, the colors of an image are arranged as additional channels that the CNN can process. For feeding the CNN with all BERT features for the words, they are shaped similarly to an image and can be seen as the channels of the word. By using that approach, the words are given with four channels into the CNN. The output of the CNN is then processed by a dense layer. The CNN approach can be seen as an bag of words approach, which takes the words within a sliding window until the end of text is reached. The amount of words is fixed, the approach is build for each of 2-5 words.

At the end of the processing, the dense- and LSTM-features and the CNN-features are concatenated and given to the classification layer, which is composed of two dense layers. For more robust predictions, three graphs are build to get three predictions, the final result is formed by the mean. For training the model, the cross-entropy-loss is used. The model gets optimized with `AdamOptimizer`. On every training step, the model is validated with the provided validation dataset, and the model weights on the best F1-score are taken to predict the test examples.

4.3 Subtask B

For subtask B, a similar model as in (Chen and Cardie, 2018) is built to do cross-domain classification. The model in this work is composed of three major parts:

- **Label classifier**: Model that predicts if an example is a suggestion.
- **Domain classifier**: Model for the prediction of the domain of an example.
- **Shared features**: Model that applies a transformation on the input features.

The training of the model can be split into two phases: The pre-training phase of the supervised label classifier and the adversarial training of the domain classifier and the shared features.

In the pre-training phase, the model uses supervised training like for subtask A. In this phase, the label classifier and the shared features are trained to get the best score on the suggestion data of the domain of online suggestion forums. The shared features get the word embeddings as input and apply a CNN on each BERT embedding. The size of the CNNs is the same as the length of the embedding, so that the projected word features are of the same shape as the input features.

In the next step, the projected features are classified with the label classifier. Unlike to subtask A, only the CNN part is used for the classification because of the amount of additional parameters of the shared features and the domain classifier. The CNN works like in subtask A and processes the four BERT features as single channels. In this task, a sliding window of the word counts from 2-6 words is taken. The optimization is applied with `AdamOptimizer` and stops when the best score is reached on the validation data of subtask A.

	train	**val**	**test**	**pred**
#examples	8500	592	833	833
#suggestions	2085	296	87	133
#non-suggestions	6415	296	746	700

Table 1: Count of the examples for the different datasets and the prediction on the test set for subtask A.

In phase two, the domain classifier starts with the training and learns to choose the right domain for the examples. To do that, also examples of the unlabeled hotel review dataset are given into the net. The domain classifier has the same structure as the text classifier and uses CNNs to find the right label for an example.

After one train step of the domain classifier, the shared features are retrained in an adversarial way to maximize the loss of the domain classifier. To realize this, the parameters are trained with the switched labels for the hotel review examples which are marked as suggestions for this training step.

After the training step of the domain classifier and the shared features, the validation examples of subtask B are used to predict a score. To do that, the examples are predicted with the text classifier, which uses the updated shared features to make a prediction if the example is a suggestion. Early stopping is used to find the best model with the validation data, so the model stops at the maximum F1-score for the validation set for subtask B.

5 Results

In this section, the results for subtask A and B are discussed. Also the test data, which has been provided to the participants after the evaluation phase, is used for the analysis.

5.1 Subtask A

For subtask A, the best model reached a F1-score of 0.7273 for the test data. This model archived a validation F1-score of 0.875 which is a noticeable difference to the test data score. To find the difference in the datasets, the example counts are compared in Table 1. It can be seen that there are differences in the ratio of the data. While the train data contains about 25% of examples for suggestions, for the test data there are only about 15%. Since oversampling is used to tackle the problem of class imbalance, this difference of train and test

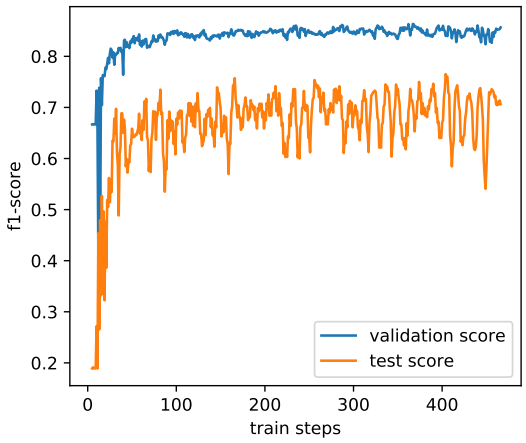

Figure 1: Comparison of the F1-scores for validation and test set during training for subtask A.

data should not have much influence on the result. Also the counts of predicted classes for the test set can be seen in the table, which show a similar amount like the test classes, but has some more predictions for the class of suggestions.

Another factor for the gap between validation and test score could be explained by very different examples in the two sets. In addition to that, the validation set may be better represented by the train set. That could lead to bad results for the test data when stopping at a good F1-score for the validation data. For analyzing this, another train run is started and the curve for the F1-score for the validation set and the test set plotted. The outcome can be seen in Figure 1.

It can be seen that the test F1-score is constantly lower than the validation score. Also there are much variations in the test score, even in a late train phase. Overall it can be determined that the train data describes the validation data better than the test data with the given model for subtask A.

5.2 Subtask B

The model for subtask B reached a final F1-score on the test data of 0.8187. In comparison to subtask A, it can be seen that the final score on the test data is higher, although no labeled data is given for that task except the validation data. The reason for this could be the use of the external hotel review dataset that inputs many new examples into the model. The overall count of examples for hotel reviews is much higher than the labeled data in subtask A, so that the higher score can be explained with the presence of more data.

	val	**test**	**pred**
#examples	808	824	824
#suggestions	404	348	402
#non-suggestions	404	476	422

Table 2: Subtask B dataset and prediction counts.

Also the validation score of 0.884 doesn't differ to the test score as much as in subtask A. This also shows that the use of external data improves the overall result of the model. The validation data for subtask B was used to apply early stopping and saving the weights at the best validation F1-score. Like for subtask A, the class counts for the different datasets are shown in Table 2. It can be seen that the distribution of the classes in the test and validation set is more equal than in subtask A, what could be a reason why the model archived better results. To verify this, another training run is started to compare the test and validation F1-score over the training epochs. The results can be seen in Figure 2. The validation data score for subtask A and B and the test score for subtask B is plotted for every training step. Since the training is separated into two phases, the left graph shows the scores for the pre-training phase and the right graph for the adversarial training.

In the pre-training phase of the model, the score of the validation data of subtask A improves as expected since supervised training is performed. Also it can be seen that the subtask B data score shows a high variance over the epochs, but decreases slightly. Over all epochs, the test and validation score of subtask B is nearly the same what could be a hint that the difference between the datasets is very small. This is confirmed in phase two of the training, where the validation and test score increase in the first epochs. Although the validation score reaches a higher peak for the F1-score, the peak of the test score is found in around the same region of train steps. The validation score for subtask A decreases in the adversarial train phase, what could be caused by a overfitting to the hotel review domain. Also the validation score of subtask B decreases after about 50 epochs. The reason for that could be that the amount of non-suggestions in very high in the hotel reviews data, so that the model unlearns to distinguish between the two classes.

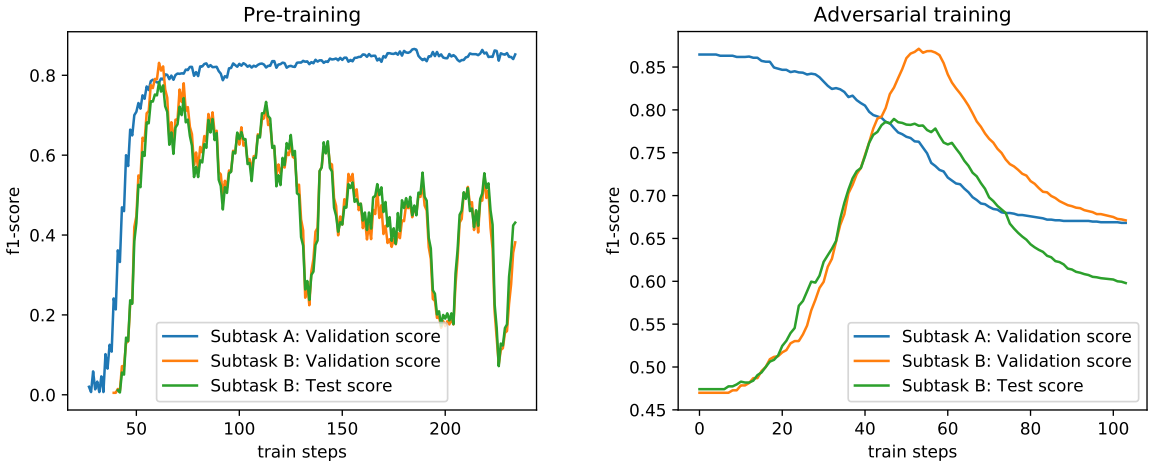

Figure 2: Two training phases for subtask B: First the pre-training with the subtask A data, then the adversarial training with the hotel review dataset.

6 Conclusion and Future Work

In this work, for each subtask of SemEval-2019 Task 9 a solution is presented. For subtask A, a supervised model is built on the neural network techniques CNN and LSTM. As input features, BERT word embeddings are taken, which are pre-trained on huge datasets. One problem in subtask A is the class imbalance of the data, which is tackled with oversampling. Another problem that occurred during the evaluation of the training phase is the difference of examples in the validation and the test set, what could be one of the reasons why the validation score is much higher than the test score. In future works, it can be tried to extend the labeled data with additional unlabeled data to tackle the problem of class imbalance and too few examples. Since extending the data works for subtask B, it could also work for subtask A and the domain of suggestion forum posts.

Subtask B sets the task to classify sentence as suggestion or non-suggestion for the domain of hotel reviews. Unlike to subtask A, only a small validation set is given as labeled data which is not allowed to be used for supervised training. To solve the problem of not having labeled data, the technique of cross-domain classification has been used. This is done by building a neural network model, which is trained in an adversarial way. Like in subtask A, a classification model for the suggestions is given. In addition to that, a shared feature representation and a domain classifier are added. The domain classifier is trained to assign the right domain label to a sentence. The shared feature representation is trained adversarial to the domain classifier, so that it learns to generate a global representation for both domains. For that, an unlabeled external dataset is taken which contains examples for the domain of hotel reviews.

The results for subtask B show that the adversarial training can improve the F1-score for the domain with no labeled data. This happens with the cost of lowering the score for the labeled data, on which the model was pre-trained. Also the score for the hotel review data falls down after the peak is reached. This makes it necessary to have at least a small dataset which contains labeled data for subtask B to measure the score during the training and stop when best score has been reached. For future work, it could be tried to develop a better shared features method where a good feature representation for both domains is formed. That would give a classifier that could be used to predict sentences of both domains. Another improvement could be archived by developing a model where the curve for the unlabeled data doesn't fall down that sharply in the late train phase. This could lead to a method were no labeled data is needed to stop the training at a good point.

References

Xilun Chen and Claire Cardie. 2018. Multinomial adversarial networks for multi-domain text classification. *CoRR*, abs/1802.05694.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of

deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc.

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

Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, ECML'98, pages 137–142. Springer-Verlag.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

Yann LeCun and Yoshua Bengio. 1998. The handbook of brain theory and neural networks. chapter Convolutional Networks for Images, Speech, and Time Series, pages 255–258. MIT Press.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546.

Sapna Negi, Tobias Daudert, and Paul Buitelaar. 2019. Semeval-2019 task 9: Suggestion mining from online reviews and forums. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*.

Sapna Negi, Maarten de Rijke, and Paul Buitelaar. 2018. Open domain suggestion mining: Problem definition and datasets. *CoRR*, abs/1806.02179.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Henning Wachsmuth, Martin Trenkmann, Benno Stein, Gregor Engels, and Tsvetomira Palakarska. 2014. A review corpus for argumentation analysis. In *Proceedings of the 15th International Conference on Intelligent Text Processing and Computational Linguistics*, pages 115–127, Berlin Heidelberg New York. Springer.