Improving Implicit Stance Classification in Tweets Using Word and Sentence Embeddings

Robin Schaefer(✉) and Manfred Stede

Applied Computational Linguistics, University of Potsdam, Potsdam, Germany
{robin.schaefer,stede}@uni-potsdam.de

Abstract. Argumentation Mining aims at finding components of arguments, as well as relations between them, in text. One of the largely unsolved problems is implicitness, where the text invites the reader to infer a missing component, such as the claim or a supporting statement. In the work of Wojatzki and Zesch (2016), an interesting implicitness problem is addressed on a Twitter data set. They showed that implicit stances toward a claim can be found with some success using just token and character n-grams. Using the same dataset, we show that results for this task can be improved using word and sentence embeddings, but that not all embedding variants perform alike. Specifically, we compare fastText, GloVe, and Universal Sentence Encoder (USE); and we find that, to our knowledge, USE yields state-of-the-art results for this task.

Keywords: Argumentation mining · Social media · Stance classification

1 Introduction

Argumentation mining can be defined as the task of automatically identifying argument components and their relations in text. While the majority of prior work has been focused on well-structured text genres, e.g. persuasive essays [19] or legal texts [14], a more recent branch of research has concentrated on argumentation mining in social media, like Twitter messages [9,12].

One mainly unsolved issue in argumentation mining is the implicitness of a core component of the argumentation. This is especially the case for social media where claims and their premises tend to be incomplete, which is also due to the often short nature of social media texts. In addition, messages are often conveyed using incorrect language leading to noisy textual data [18]. This leads to increased demands for the argument detection system.

Following [20] (henceforth WZ16), we approach the issue of implicit argument detection by reconsidering a part of argumentation mining as a stance classification problem. This is based on the assumption that a claim can be reformulated as a stance toward the topic at hand, which allows us to infer the

C. Benzmüller and H. Stuckenschmidt (Eds.): KI 2019, LNAI 11793, pp. 299–307, 2019.
https://doi.org/10.1007/978-3-030-30179-8_26

implicit claim from the data. For instance, in example (1) a tweet on atheism is given containing an explicit premise toward the claim (2) to be inferred. (3) shows the claim reformulated as an implicit stance, which could be the output of a stance classifier taking (1) as input.

(1) Bible: that infidels are going to hell! (explicit premise) [WZ16]
(2) *Therefore, atheism is to be rejected. (implicit claim)*
(3) *I am against atheism. (implicit claim reformulated as implicit stance)* [WZ16]

In this paper, we will replicate the classification results of WZ16 with our own implementation. Afterward, we will show that different variants of word and sentence embeddings considerably surpass the n-gram-based results reported in WZ16.[1]

We structure this paper as follows: Sect. 2 gives an overview of the related research including WZ16, which is the basis for our work. Section 3 describes the data set. Section 4 describes the applied method including the embedding variants that we used. Section 5 presents the results before we conclude in Sect. 6 with a discussion and a short outlook.

2 Related Work

[4] present a preliminary pipeline for argumentation mining in Twitter data. This pipeline consists of different sub-steps like separating argumentative from non-argumentative parts and defining argumentative relations between tweet pairs. Their work is based on DART [3], a Twitter data set containing tweets annotated for their argumentative relations. While this work presents an interesting approach to argumentation mining in tweets, it is ignoring implicit components of argumentation which renders it less relevant for our work.

[2] use features based on textual entailment (following [5]) and semantic textual similarity in order to automatically detect arguments in online debates. This task is motivated by the assumption that argument detection improves the classification of stances toward the debate topic. Like WZ16, the authors recognize implicitness as a central issue of argumentation mining. However, unlike WZ16, [2] approach this issue using features that are somewhat more semantic in nature than n-grams. We follow this idea by using pre-trained word embeddings.

WZ16 propose to reformulate an implicit claim as a stance toward the given topic, thereby modeling argumentation mining as a stance classification problem. As the actual implicit claim is hard to extract, classifying a newly formulated implicit stance can help approximating it. In addition, WZ16 created the Atheism Stance Corpus (ASC), which is based on the Twitter data set provided for the first shared task on automatic stance detection (Task 6, SemEval 2016) [15]. They annotated the tweets according to their stances toward a predefined set

[1] The code can be downloaded from https://github.com/RobinSchaefer/tweet-stance-classification.

of topics (see Table 1), one of which is the implicit stance to the overall topic *atheism*. They further implemented a linear Support Vector Machine (SVM) and used different sizes of token and character n-grams as features. This model achieved an F1 score of 0.66. In order to investigate the full potential of implicit stance classification using explicit stances toward subtopics, WZ16 create an oracle condition utilizing the manually annotated explicit stances as features. With respect to (1), this could mean a positive stance toward the topic *Christianity*. This model achieved an F1 score of 0.88. In this work, we will compare our results both to the results elicited by the n-gram model and to the more 'artificial' oracle results.

3 Data

In this work, we use the ASC from WZ16, which contains 715 tweets from the *atheism* sub-part of the SemEval data set. It has been annotated for *the implicit stance toward the debate topic*, i.e. the tweet being pro/contra atheism, and for *the explicit stances toward a set of derived topics*. These topics include the subtopics *Christianity* and *supernatural power*. A full list including their annotations in percentages is given in Table 1.

Table 1. The topics and their annotations in percentages

Topic	Pro	Contra	None
Atheism (debate stance)	**20**	**49**	**31**
Christianity	27	04	69
Conservatism	01	01	98
Freethinking	04	–	96
Islam	04	02	94
Life after death	01	01	98
No evidence	01	–	99
Religious freedom	01	–	99
Same-sex marriage	02	–	98
Secularism	03	–	97
Supernatural power	39	08	53
USA	04	–	96

4 Method

In this work we investigate the applicability of different feature types to solve the task of implicit stance classification. Specifically, we make use of n-grams

and different variants of word and sentence embeddings (fastText, GloVe and USE).

To begin with, we present our implementation of the processing pipeline reported in WZ16 using the scikit-learn library [16][2] in Python. For preprocessing we apply the Tweet NLP tokenizer [10]. We further implement certain text cleaning and normalization steps: tokens are turned to lowercase; stopwords and punctuation[3] are removed; the Snowball Stemmer[4] is applied. We report results for different processing pipelines which will give insights regarding the most promising ways to approach this task.

We classify our data using a linear SVM with the C hyperparameter set to 1. We further apply 10-fold cross-validation and report F1 scores for all models.

4.1 N-Grams

Following WZ16, we derive n-grams from the data, which we treat as a baseline for the word and sentence embedding models. We use a binary bag-of-n-gram representation (existent/nonexistent). Apart from stopword and punctuation removal, we do no restrict our set of tokens. This contrasts with WZ16 who only use the 1000 most frequent tokens. However, this leads to an arbitrary set of tokens due to the majority of tokens occurring only once. We further experiment with different token n-gram sizes and their combinations.

4.2 fastText Embeddings

We use pre-trained 300-dimensional fastText [11] word vectors that have been trained on Wikipedia and Common Crawl data. For training, an extension of the CBOW model has been used. Thus, a word vector is created based on a prediction of the respective word given the context it appears in. In addition, the model makes use of subword information, i.e. each word vector is the result of the summation of the underlying subword vectors. Also, position weights are incorporated in order to weigh the impact of the context words.

In order to obtain a vector for a whole tweet we create document vectors by averaging the respective word vectors for the words used in the tweet. If a word is not included in the set of word vectors it is discarded.

4.3 GloVe Embeddings

In addition, we apply pre-trained 200-dimensional GloVe [17] word vectors. These vectors have been trained on tweets which may have a positive impact on the results. However, in order to directly compare results based on differences in

[2] Note that WZ16 apply the DKPro Core [6] and DKPro TC frameworks [8].
[3] Note that during punctuation removal #'s are ignored in order to maintain hashtags, which we assume to be meaningful for our task.
[4] The Snowball Stemmer is implemented using NLTK [13].

algorithms and not on differences in dimensions we also apply pre-trained 300-dimensional GloVe word vectors trained on Common Crawl.

In contrast to fastText, GloVe incorporates global statistics using matrix factorization. Also, the model has been trained on words instead of subwords, meaning that it has issues processing unseen words.

To receive the final tweet vectors we follow the same steps as for the fastText vectors.

4.4 USE Embeddings

Finally, we apply the pre-trained language-agnostic USE model based on English as the source language and German as the target language [7]. The model has been trained via a dual-encoder architecture that consisted of different monolingual and cross-lingual tasks. The central difference to our other embedding variants lies in the direct training on sentence data.

Table 2. F1 scores per feature set and preprocessing steps (l = lowercase, s = stopwords, p = punctuation, st = stemmer)

Feature set	Preprocessing	F1 (micro)	F1 (macro)
WZ16 baseline	Not available	0.66	Not available
WZ16 oracle condition	Not available	**0.88**	Not available
Unigrams	-	0.67	0.62
	l	**0.70**	0.65
	l, s, p	0.68	0.63
	l, s, p, st	0.68	0.64
	st	0.69	0.65
	l, st	0.69	0.65
Bigrams	l	0.59	0.51
Trigrams	l	0.49	0.25
Uni- and bigrams	l	0.69	0.63
Bi- and trigrams	l	0.58	0.48
Uni-, bi- and trigrams	l	0.68	0.61
fastText embeddings (300d)	-	0.71	0.67
	p	0.73	0.69
GloVe embeddings (200d)	-	0.60	0.55
	p	0.60	0.55
GloVe embeddings (300d)	-	0.61	0.56
	p	0.59	0.54
USE embeddings (512d)	-	0.77	0.73
	p	**0.78**	0.75

We implement the USE using its TensorFlow [1] module. Text data simply gets loaded into the module and it outputs an embedding with 512 dimensions.[5] As the encoder also can handle texts longer than single sentences it can be fruitfully utilized for our task.

5 Results

Results, i.e. micro and macro averaged F1 scores, are reported in Table 2. Consistent with WZ16, we only discuss micro averaged F1 scores. With our n-gram model we achieve an F1 score of 0.70, thereby beating the baseline of 0.66 that has been reported in WZ16. Our simplest model only including unigrams yields the best results. Furthermore, the choice of preprocessing steps has a large effect on the performance. While lower casing improves the results substantially, additionally removing stopwords and punctuation in fact decreases the evaluation scores. Stemming can have a positive effect if used without token removal but does not achieve best results.

Using more complex n-grams like bigrams, trigrams or their concatenated combinations does not outperform the simple unigram model. In fact, applying bigrams and trigrams considerably decreases F1 scores to 0.59 and 0.49, respectively. Only the combinations of uni- and bigrams and of uni-, bi- and trigrams yield results comparable to the unigram model, albeit with higher computational costs.

The word and sentence embeddings yield mixed results. While the highest F1 score of the fastText embeddings (0.73) is above the n-gram results, GloVe embeddings reach only low scores. Importantly, both 200- and 300-dimensional GloVe vectors yield similar results. The highest result of 0.78, however, is achieved using USE embeddings. For preprocessing no cleaning or normalization step is applied apart from punctuation removal.

6 Discussion and Outlook

The benefit of our work is twofold. First, we replicate the results of WZ16 by implementing our own processing pipeline using the Python language. The replication includes different preprocessing steps, n-gram feature extraction and supervised classification using an SVM. By applying basic preprocessing (lower casing) and using simple unigrams we are able to enhance the original results.

Second, we make use of different pre-trained word and sentence embeddings to enhance the performance even further. Importantly, the different embedding variants do not yield similar results. While fastText embeddings achieve comparable but slightly better results than the n-gram model, our GloVe vectors cannot compete. This is surprising as the 200-dimensional GloVe vectors have

[5] As the USE model has been trained exclusively for 512-dimensional vectors [7], we are unable to create 300-dimensional vectors that would have been more directly comparable to the fastText and GloVe vectors.

been pre-trained on Twitter data. The best results are achieved via the language-agnostic USE. With an F1 score of 0.78 we improve the original results of WZ16 by more than 0.1. Hence, by using sentence embeddings instead of n-grams we take a large step toward the potential result given by WZ16's oracle condition. Recall that this is based on manual explicit stance annotations and yields an F1 score of 0.88.

The question remains why different embedding variants lead to substantially different results. One potential cause of performance differences could lie in the diverging complexities of the vectors. While our 512-dimensional USE vectors yield best results, F1 scores seem to decrease with a reduction of vector dimensions. Lowest scores were achieved with the 200-dimensional GloVe vectors. This is not surprising if we assume that more complex embeddings can capture semantic content to a more fine-grained degree. Recall, however, that the usage of 300-dimensional GloVe vectors does lead to similar results as the 200-dimensional GloVe vectors, thus providing evidence that vector dimensions alone are unlikely to explain the result pattern that we report.

Another reason for the better results of USE embeddings could be the fact that they have been trained on sentences instead of words. This could lead to a better encoding of syntactic information. In contrast, one possible constraint of word embeddings is that only word information, or subword information, is encoded. One may also conclude that the averaging of word vectors in order to create a document vector cannot compete with direct training on segments of text that are larger than words.

Finally, we want to point out that we can only make limited assumptions regarding the generalization of our model. This is due to the focus of the corpus on one topic. While the large improvement of results driven by USE vectors may give confidence that they can be applied for other corpora as well, further research is needed to prove this assumption.

To conclude, while the results of our USE embeddings are promising, clearly more work is needed on the causes of performance differences. Our next step will be to move to German as our language of interest. Having used language-agnostic USE embeddings and fastText embeddings that have been trained in parallel with a huge amount of other languages enables us to directly compare both NLP pipelines and their performances.

Acknowledgements. We would like to thank Michael Wojatzki for sharing further details about their implementation with us. We would further like to thank the anonymous reviewers for their helpful comments.

References

1. Abadi, M., et al.: TensorFlow: large-scale machine learning on heterogeneous systems (2015). http://tensorflow.org/
2. Boltužić, F., Šnajder, J.: Back up your stance: recognizing arguments in online discussions. In: Proceedings of the First Workshop on Argumentation Mining, Baltimore, Maryland, pp. 49–58. Association for Computational Linguistics, June 2014. https://doi.org/10.3115/v1/W14-2107

3. Bosc, T., Cabrio, E., Villata, S.: DART: a dataset of arguments and their relations on twitter. In: Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016), Portorož, Slovenia, pp. 1258–1263. European Language Resources Association (ELRA), May 2016
4. Bosc, T., Cabrio, E., Villata, S.: Tweeties squabbling: positive and negative results in applying argument mining on social media. In: Proceedings of the 6th International Conference on Computational Models of Argument, Potsdam, Germany, September 2016
5. Cabrio, E., Villata, S.: Combining textual entailment and argumentation theory for supporting online debates interactions. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Jeju Island, Korea, pp. 208–212. Association for Computational Linguistics, July 2012
6. de Castilho, R.E., Gurevych, I.: A broad-coverage collection of portable NLP components for building shareable analysis pipelines. In: Proceedings of the Workshop on Open Infrastructures and Analysis Frameworks for HLT, Dublin, Ireland, pp. 1–11. Association for Computational Linguistics and Dublin City University, August 2014. https://doi.org/10.3115/v1/W14-5201
7. Chidambaram, M., et al.: Learning cross-lingual sentence representations via a multi-task dual-encoder model. CoRR abs/1810.12836 (2018)
8. Daxenberger, J., Ferschke, O., Gurevych, I., Zesch, T.: DKPro TC: a Java-based framework for supervised learning experiments on textual data. In: Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, Baltimore, Maryland, pp. 61–66. Association for Computational Linguistics, June 2014. https://doi.org/10.3115/v1/P14-5011
9. Dusmanu, M., Cabrio, E., Villata, S.: Argument mining on Twitter: arguments, facts and sources. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Copenhagen, Denmark, pp. 2317–2322. Association for Computational Linguistics, September 2017. https://doi.org/10.18653/v1/D17-1245
10. Gimpel, K., et al.: Part-of-speech tagging for Twitter: annotation, features, and experiments. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Portland, Oregon, USA, pp. 42–47. Association for Computational Linguistics, June 2011
11. Grave, E., Bojanowski, P., Gupta, P., Joulin, A., Mikolov, T.: Learning word vectors for 157 languages. In: Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018) (2018)
12. Grosse, K., González, M.P., Chesñevar, C.I., Maguitman, A.G.: Integrating argumentation and sentiment analysis for mining opinions from twitter. AI Commun. **28**(3), 387–401 (2015)
13. Loper, E., Bird, S.: NLTK: the natural language toolkit. In: Proceedings of the ACL 2002 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics, ETMTNLP 2002, Stroudsburg, PA, USA, vol. 1, pp. 63–70. Association for Computational Linguistics (2002). https://doi.org/10.3115/1118108.1118117
14. Moens, M.F., Boiy, E., Palau, R.M., Reed, C.: Automatic detection of arguments in legal texts. In: Proceedings of the 11th International Conference on Artificial Intelligence and Law, ICAIL 2007, pp. 225–230. ACM, New York (2007). https://doi.org/10.1145/1276318.1276362

15. Mohammad, S., Kiritchenko, S., Sobhani, P., Zhu, X., Cherry, C.: SemEval-2016 task 6: detecting stance in tweets. In: Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), San Diego, California, pp. 31–41. Association for Computational Linguistics, June 2016. https://doi.org/10.18653/v1/S16-1003
16. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
17. Pennington, J., Socher, R., Manning, C.D.: GloVe: global vectors for word representation. In: Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543 (2014)
18. Snajder, J.: Social media argumentation mining: the quest for deliberateness in raucousness. CoRR abs/1701.00168 (2017)
19. Stab, C., Gurevych, I.: Identifying argumentative discourse structures in persuasive essays. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, pp. 46–56. Association for Computational Linguistics, October 2014. https://doi.org/10.3115/v1/D14-1006
20. Wojatzki, M., Zesch, T.: Stance-based argument mining - modeling implicit argumentation using stance. In: Proceedings of the KONVENS, pp. 313–322 (2016)