Domain adaptation for part-of-speech tagging of noisy user-generated text

Luisa März and **Dietrich Trautmann** and **Benjamin Roth**
CIS, University of Munich (LMU) Munich, Germany
{luisa.maerz, dietrich, beroth}@cis.lmu.de

Abstract

The performance of a Part-of-speech (POS) tagger is highly dependent on the domain of the processed text, and for many domains there is no or only very little training data available. This work addresses the problem of POS tagging noisy user-generated text using a neural network. We propose an architecture that trains an out-of-domain model on a large newswire corpus, and transfers those weights by using them as a prior for a model trained on the target domain (a data-set of German Tweets) for which there is very little annotations available. The neural network has two standard bidirectional LSTMs at its core. However, we find it crucial to also encode a set of task-specific features, and to obtain reliable (source-domain and target-domain) word representations. Experiments with different regularization techniques such as early stopping, dropout and fine-tuning the domain adaptation prior weights are conducted. Our best model uses external weights from the out-of-domain model, as well as feature embeddings, pre-trained word and sub-word embeddings and achieves a tagging accuracy of slightly over 90%, improving on the previous state of the art for this task.

1 Introduction

Part-of-speech (POS) tagging is a prerequisite for many applications and necessary for a wide range of tools for computational linguists. The state-of-the art method to implement a tagger is to use neural networks (Ma and Hovy, 2016; Yang et al., 2018). The performance of a POS tagger is highly dependent on the domain of the processed text and the availability of sufficient training data (Schnabel and Schütze, 2014). Existing POS taggers for canonical German text already achieve very good results around 97% accuracy, e.g. (Schmid, 1999; Plank et al., 2016). When applying these trained models to out-of-domain data the performance decreases drastically.

One of the domains where there is not enough data is online conversational text in platforms such as Twitter, where the very informal language exhibits many phenomena that differ significantly from canonical written language.

In this work, we propose a neural network that combines a character-based encoder and embeddings of features from previous non-neural approaches (that can be interpreted as an inductive bias to guide the learning task). We further show that the performance of this already effective tagger can be improved significantly by incorporating external weights using a mechanism of domain-specific L2-regularization during the training on in-domain data. This approach establishes state-of-the-art results of 90.3% accuracy on the German Twitter corpus of Rehbein (2013).

2 Related Work

The first POS tagging approach for German Twitter data was conducted by Rehbein (2013) and reaches an accuracy of 88.8% on the test set using a CRF. They use a feature set with eleven different features and an extended version of the STTS (Schiller et al., 1999) as a tagset. Gimpel et al. (2011) developed a tagset for English Twitter data and report results of 89.37% on their test set using a CRF with different features as well. POS tagging for different languages using a neural architecture was successfully applied by Plank et al. (2016). The data comes from the Universal Dependencies project[1] and mainly contains German newspaper texts and Wikipedia articles.

The work of Barone et al. (2017) investigates different regularization mechanisms in the field of domain adaptation. They use the same L2 regularization mechanism for neural machine translation, as we do for POS tagging.

[1] http://universaldependencies.org

Proceedings of NAACL-HLT 2019, pages 3415–3420
Minneapolis, Minnesota, June 2 - June 7, 2019.

3 Data

3.1 Tagset

The Stuttgart-Tübingen-TagSet (STTS, Schiller et al. (1999)) is widely used as the state-of-the-art tagset for POS tagging of German. Bartz et al. (2013) show that the STTS is not sufficient when working with textual data from online social platforms, as online texts do not have the same characteristics as formal-style texts, nor are identical to spoken language. Online conversational text often contains contracted forms, graphic reproductions of spoken language such as prolongations, interjections and grammatical inconsistencies as well as a high rate of misspellings, omission of words etc.

For POS tagging we use the tagset of Rehbein (2013), where (following Gimpel et al. (2011)) additional tags are provided to capture peculiarities of the Twitter corpus. This tagset provides tags for @-mentions, hashtags and URLs. They also provide a tag for non-verbal comments such as ***Trommelwirbel*** (drum-roll). Additional, complex tags for amalgamated word forms were used (see Gimpel et al. (2011)). Overall the tagset used in our target domain contains 15 tags more than the original STTS.

3.2 Corpora

Two corpora with different domains are used in this work. One of them is the TIGER corpus and the other is a collection of German Twitter data.

The texts in the TIGER corpus (Brants et al., 2004) are taken from the Frankfurter Rundschau newspaper and date from 1995 over a period of two weeks. The annotation of the corpus was created semi automatically. The basis for the annotation of POS tags is the STTS. The TIGER corpus is one of the standard corpora for German in NLP and contains 888.505 tokens.

The Twitter data was collected by Rehbein (2013) within eight months in 2012 and 2013. The complete collection includes 12.782.097 distinct tweets, from which 1.426 tweets were randomly selected for manual annotation with POS tags. The training set is comparably small and holds 420 tweets, whereas the development and test set hold around 500 tweets each (overall 20.877 tokens). Since this is the only available German annotated Twitter corpus, we use it for this work.

3.3 Pretrained word vectors

The usage of pretrained word embeddings can be seen as a standard procedure in NLP to improve the results with neural networks (see Ma and Hovy (2016).

3.4 FastText

FastText[2] provides pretrained sub-word embeddings for 158 different languages and allows to obtain word vectors for out-of-vocabulary words. The pretrained vectors for German are based on Wikipedia articles and data from Common Crawl[3]. We obtain 97.988 different embeddings for the tokens in TIGER and the Twitter corpus of which 75.819 were already contained in Common Crawl and 22.171 were inferred from sub-word units.

3.5 Word2Vec

Spinningbytes[4] is a platform for different applications in NLP and provides several solutions and resources for research. They provide word embeddings for different text types and languages, including Word2Vec (Mikolov et al., 2013) vectors pretrained on 200 million German Tweets. Overall 17.030 word embeddings form the Spinningbytes vectors are used (other words are initialized all-zero).

3.6 Character level encoder

Lample et al. (2016) show that the usage of a character level encoder is expedient when using bidirectional LSTMs. Our implementation of this encoder follows Hiroki Nakayama (2017)[5], where character embeddings are passed to a bidirectional LSTM and the output is concatenated to the word embeddings.

4 Experiments

This section describes the proposed architecture of the neural network and the conditional random field used in the experiments. For comparison of the results we also experiment with jointly training on a merged training set, which contains the Twitter and the TIGER training sets.

[2] https://fasttext.cc
[3] https://commoncrawl.org
[4] https://www.spinningbytes.com
[5] https://github.com/Hironsan/anago

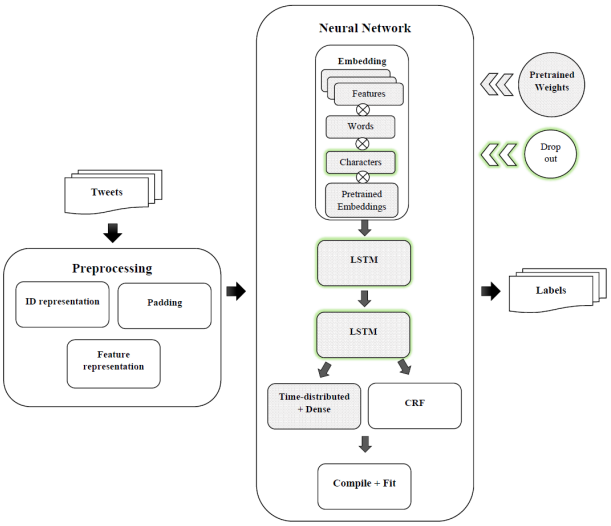

Figure 1: Final architecture of the neural model. Layers that are passed pretrained weights are hatched in gray. Dropout affected layers are highlighted in green.

4.1 Methods

4.1.1 Conditional random field baseline

The baseline CRF of Rehbein (2013) achieves an accuracy of 82.49%. To be comparable with their work we implement a CRF equivalent to their baseline model. Each word in the data is represented by a feature dictionary. We use the same features as Rehbein proposed for the classification of each word. These are the lowercased word form, word length, number of uppercase letters, number of digits and occurrence of a hashtag, URL, @-mention or symbol.

4.1.2 Neural network baseline

The first layer in the model is an embedding layer. The next layers are two bidirectional LSTMs. The baseline model uses softmax for each position in the final layer and is optimized using Adam core with a learning rate of 0.001 and the categorical crossentropy as the loss function.

4.1.3 Extensions of the neural network

The non neural CRF model benefits from different features extracted from the data. Those features are not explicitely modeled in the neural baseline model, and we apply a feature function for the extended neural network. We include the features used in the non-neural CRF for hashtags and @-mentions. In addition, we capture orthographic features, e.g., whether a word starts with a digit or an upper case letter. Typically, manually defined features like these are not used in neural networks, as a neural network should take over feature engineering completely. Since this does not work optimally, especially for smaller data sets, we have decided to give the neural network this type of information as well. Thus we combine the advantages of classical feature engineering and neural networks. This also goes along with the observations of Plank et al. (2018) and Sagot and Martínez Alonso (2017), who both show that adding conventional lexical information improves the performance of a neural POS tagger. All words are represented by their features and for each feature type an embedding layer is set up within the neural network in order to learn vectors for the different feature expressions. Afterwards all the feature embeddings are added together. As the next step we use the character level layer mentioned in section 3.6 (Lample et al., 2016). The following vector sequences are concatenated at each position and form the input to the bidirectional LSTMs:

- Feature embedding vector
- character-level encoder
- FastText vectors
- Word2Vec vectors

4.1.4 Domain Adaptation and regularization

We train the model with the optimal setting on the TIGER corpus, i.e., we prepare the TIGER data just like the Twitter data and extract features, include a character level layer and use pretrained embeddings. We extract the weights $\hat{W}$ that were optimized with TIGER. The prior weights $\hat{W}$ are used during optimization as a regularizer for the weights W used in the final model (trained on the Twitter data). This is achieved by adding the penalty term R_W, as shown in Equation 1, to the objective function (cross-entropy loss).

$$R_W = \lambda ||W - \hat{W}||_2^2 \quad (1)$$

The regularization is applied to the weights of the two LSTMs, the character LSTM, to all of the embedding layers and to the output layer.

As a second regularization mechanism we include dropout for the forward and the backward LSTM layers. We also add 1 to the bias of the forget gate at initialization, since this is recommended in Jozefowicz et al. (2015). Additionally, we use early stopping. Since the usage of different regularization techniques worked well in the experiments of Barone et al. (2017), we also tried the

combination of different regularizers in this work. Figure 1 shows the final architecture of our model.

4.2 NCRF++

We also report results obtained by training the sequence labelling tagger of Yang and Zhang (2018), NCRF++. They showed that their architecture produces state-of-the-art models across a wide range of data sets (Yang et al., 2018) so we used this standardized framework to compare it with our model.

5 Results

5.1 Experimental Results

Table 1 shows the results on the Twitter test set. The feature-based baseline CRF outperforms the baseline of the neural net with more than 20 percentage points. After adding the feature information, the performance of the neural baseline is improved by 13 percentage points, which is understandable, because many German POS tags are case sensitive.

experiment	**accuracy**
baseline crf	0.831
baseline neural model	0.634
neural model	
+features	0.768
+character embeddings	0.796
+pretrained word vectors	0.845
+l2 domain adaptation	0.896
+dropout	**0.903**
neural model joint training	0.894
final CRF of Rehbein 2013	0.888
NCRF++ system	0.887

Table 1: Results on the test set using the time-distributed layer.

The model's performance increases by another 3 percentage points if the character level layer is used. Including the pretrained embeddings, FastText and Word2Vec vectors, the accuracy is 84.5%, which outperforms the CRF baseline.

Figure 2 shows the impact of domain adaptation and fine-tuning the prior weight. The value of the λ parameter in the regularization formula 1 can control the degree of impact of the weights on the training. Excluding the pretrained weights means that λ is 0. We observe an optimal benefit from the out-of-domain weights by using a λ value

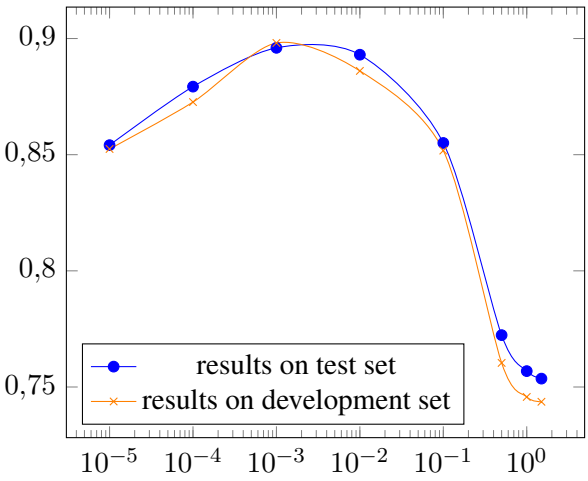

Figure 2: Influence of fine-tuning on the results on dev and test set in accuracy (y-axis). The x-axis corresponds to the different λ values.

of 0.001. This is in line with the observations of Barone et al. (2017) for transfer-learning for machine translation.

Overall the addition of the L2 fine-tuning can improve the tagging outcome by 5 percentage points, compared to not doing domain adaptation. A binomial test shows that this improvement is significant. This result confirms the intuition that the tagger can benefit from the pretrained weights. On top of fine-tuning different dropout rates were added to both directions of the LSTMs for the character level layer and the joint embeddings. A dropout rate of 75% is optimal in our scenario, and it increases the accuracy by 0.7 percentage points.

The final 90.3% on the test set outperform the results of Rehbein (2013) by 1.5 percentage points.Our best score also outperforms the accuracy obtained with the NCRF++ model. This shows that for classifying noisy user-generated text, explicit feature engineering is beneficial, and that the usage of domain adaptation is expedient in this context. Joint training, using all data (out-of-domain and target domain), can obtain an accuracy score of 89.4%, which is about 1 percentage point worse than using the same data with domain adaptation. The training setup for the joint training is the same as for the other experiments and includes all extensions except for the domain adaptation.

5.2 Error Analysis

The most frequent error types in all our systems were nouns, proper nouns, articles, verbs, adjec-

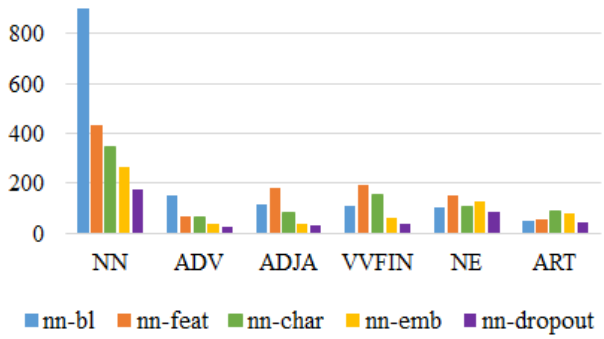

Figure 3: Total number of errors for the six most frequent POS-tags and different experimental settings

tives and adverbs as pictured in figure 3. By including the features the number of errors can be reduced drastically for nouns. Since we included a feature that captures upper and lower case, and nouns as well as proper nouns are written upper case in German, the model can benefit from that information. The pretrained word embeddings also help classifying nouns, articles, verbs, adjectives and adverbs. Only the errors with proper nouns increase slightly. Compared to only including the features, the model can benefit from adding both, the character level layer and the pretrained word vectors, while the results for tagging proper nouns and articles are still slightly worse than the baseline. In contrast the final experimental setup can optimize the results for every POS tag compared to the baseline, see figure 3. Slightly in case of articles and proper nouns, but markedly for the other tags. A comparison of the baseline errors and the errors of the final system shows that Twitter specific errors, e.g. with @-mentions or URLs, can be reduced drastically. Only hashtags still pose a challenge for the tagger. In the gold standard words with hashtags are not always tagged as such, but sometimes are classified as proper nouns. This is due to the fact that the function of the token in the sentence is the one of a proper noun. Thus the tagger has decision problems with these hashtags. Other types of errors, such as confusion of articles or nouns, are not Twitter-specific issues, but are often a problem with POS tagging and can only be fixed by general improvement of the tagger.

6 Conclusion

We present a deep learning based fine-grained POS tagger for German Twitter data using both domain adaptation and regularization techniques. On top of an efficient POS tagger we implemented domain adaptation by using a L2-norm regularization mechanism, which improved the model's performance by 5 percentage points. Since this performance is significant we conclude that fine-tuning and domain adaptation techniques can successfully be used to improve the performance when training on a small target-domain corpus.

Our experiments show that the combination of different regularization techniques is recommendable and can further optimize already efficient systems.

The advantage of our approach is that we do not need a large annotated target-domain corpus, but only pretrained weights. Using a pretrained model as a prior for training on a small amount of data is done within minutes and therefore very practicable in real world scenarios.

References

Antonio Valerio Miceli Barone, Barry Haddow, Ulrich Germann, and Rico Sennrich. 2017. Regularization techniques for fine-tuning in neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1489–1494. Association for Computational Linguistics.

Thomas Bartz, Michael Beisswenger, and Angelika Storrer. 2013. Optimierung des stuttgart-tübingen-tagset für die linguistische annotation von korpora zur internetbasierten kommunikation: Phänomene, herausforderungen, erweiterungsvorschläge. *JLCL*, 28:157–198.

Sabine Brants, Stefanie Dipper, Peter Eisenberg, Silvia Hansen-Schirra, Esther König, Wolfgang Lezius, Christian Rohrer, George Smith, and Hans Uszkoreit. 2004. Tiger: Linguistic interpretation of a german corpus. *Research on Language and Computation*, 2(4):597–620.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. 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: Short Papers - Volume 2*, HLT '11, pages 42–47, Stroudsburg, PA, USA. Association for Computational Linguistics.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proceedings of the 32Nd International Conference on International*

Conference on Machine Learning - Volume 37, ICML'15, pages 2342–2350. JMLR.org.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270. Association for Computational Linguistics.

Xuezhe Ma and Eduard H. Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *CoRR*, abs/1603.01354.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Barbara Plank, Sigrid Klerke, and Zeljko Agic. 2018. The best of both worlds: Lexical resources to improve low-resource part-of-speech tagging. *CoRR*, abs/1811.08757.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. *CoRR*, abs/1604.05529.

Ines Rehbein. 2013. Fine-grained pos tagging of german tweets. In *Language Processing and Knowledge in the Web*, pages 162–175, Berlin, Heidelberg. Springer Berlin Heidelberg.

Benoît Sagot and Héctor Martínez Alonso. 2017. Improving neural tagging with lexical information. In *Proceedings of the 15th International Conference on Parsing Technologies*, pages 25–31. Association for Computational Linguistics.

Anne Schiller, Simone Teufel, Christine Stckert, and Christine Thielen. 1999. Guidelines fr das tagging deutscher textcorpora mit stts (kleines und groes tagset). Seminararbeit, University of Stuttgart, University of Tbingen.

H. Schmid. 1999. *Improvements in Part-of-Speech Tagging with an Application to German*, pages 13–25. Springer Netherlands, Dordrecht.

Tobias Schnabel and Hinrich Schütze. 2014. Flors: Fast and simple domain adaptation for part-of-speech tagging. *Transactions of the Association for Computational Linguistics*, 2:15–26.

Jie Yang, Shuailong Liang, and Yue Zhang. 2018. Design challenges and misconceptions in neural sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING)*.

Jie Yang and Yue Zhang. 2018. Ncrf++: An open-source neural sequence labeling toolkit. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.