Final Paper for XCS224U Project *gNER*

Matthias Droth
matthias.droth@gmail.com

Martin Nigsch
martin@nigsch.eu

Vasco Ribeiro
vascosousaribeiro@yahoo.com

Abstract

In our project gNER, we assess how to efficiently perform Named Entity Recognition (NER) in German language in a real-life scenario with scraped newspaper data. We assess a small range of model choices and to a minor extent annotation choices with the overall goal of finding an optimum between maximizing extraction accuracy while minimizing annotation and computational costs. The data exposes minimal variations in terms of vocabulary and style, yet is relevant in practical life as it contains property prices as sold in the region Vorarlberg in Austria.

1 Introduction

In this document we implement and apply a variety of neural-network based models to perform named entity recognition on our custom dataset. We begin by experimenting with linear-chain Conditional Random Fields (CRF) stand-alone and after adding custom feature functions. Next we experiment with a one-layer LSTM which we then proceed to associate with a CRF layer.

We hypothesize that CRF algorithms perform on par with deep learning algorithms when trained only on a relatively small named entity recognition (NER) dataset such as the one we are working with. Finding an architecture that produces acceptable results even in the presence of small annotated datasets can be of great value to practitioners looking for high quality out-of-sample inference.

We find that simple CRF models outperform LSTM-based networks. This outperformance is magnified by the application of custom feature functions (such as word capitalization and use of gazetteers).

2 Related work

We seek to extract named entities out of German text at a minimum overall cost. In order to do so, there are several areas to explore: (i) better models

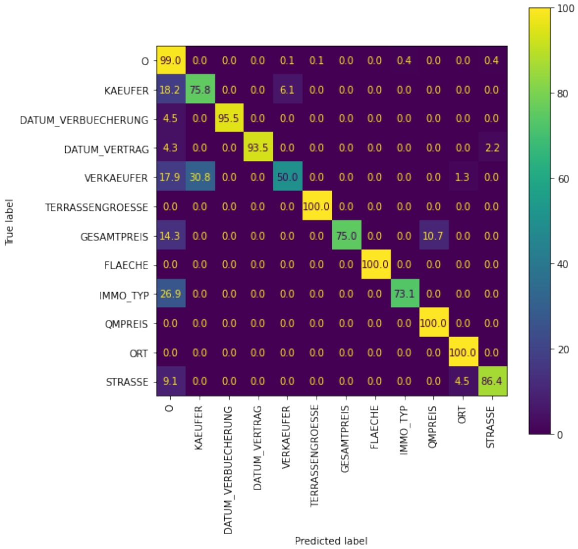

Figure 1: Confusion matrix for the best CRF model found via hyperparameter search. The cells show percen

making better use of training data annotation, (ii) pretraining with colloquial vs. domain specific language, (iii) model choices beyond the obvious hyperparameter optimization (e.g. tokenization), and (iv) advantageous annotation strategies where the marginal benefit of adding more annotations is the greatest.

While we acknowledge the potential of cleverly chosen annotation strategies, we think that this area is the most time consuming and the hardest to implement within the time available for this project. For that reason, we focus on comparing NER efficiency resulting from the choice of algorithms as well as their pretraining.

A large part of our project focusses on CRF and LSTM architectures. We drew heavily on work from (Sutton and McCallum, 2012) for a theoretical understanding of CRF's. We further deepened this knowledge by working through course materials from Michael Collins of Columbia University. For joining LSTM and CRF architectures we followed approaches laid out in Huang et al. (2015) and Lample et al. (2016).

In Yadav and Bethard (2018) a survey of research in Named Entity Recognition (NER) is performed with a particular emphasis on comparing results achieved by more recent approaches is performed.

Using well thought-out paper selection criteria, the survey starts with knowledge-based and unsupervised bootstrapped systems moving on to a description of both classic feature-engineered supervised systems as well as modern feature-inferring neural network models. Drilling in on the latter the paper establishes the taxonomy as word-level, character-level and hybrid (character+word and, finally, char+word+affix). The vast majority of architectures presented rely on bi-directional LSTM layers coupled with Conditional Random Fields for final classification. The paper then proceeds to summarize the results obtained by each type of model on standard NER datasets enabling an overall ranking to be determined (in order of increasing performance) as follows: 1. Feature-engineered systems (least performant even despite using domain-specific rules, knowledge, features and lexicons); 2. Word-based and char-based models; 3. Char+word based hybrid models; 4. Char+word+affix hybrid models. The char+word+affix model showcases the potential of combining useful past insights with modern neural network models which, the paper concludes, represents an exciting avenue for future research.

3 Data

The base data consists of articles from a regional Austrian Newspaper called Vorarlberger Nachrichten. These articles summarize recent changes in ownership in real estate and cover about a quarter of all property transactions.

3.1 Procurement

In order to be as complete as possible also for future use of the data, we have developed a scraper which logs into the newspaper and fetches a full list of urls which are of interest. In order to avoid fetching an existing url multiple times when refreshing the scraping, each url is assigned a unique ID with the blake2b algorithm.

In a further step, for every url possible images are extracted and stored, as well as a JSON file containing the HTML as well as an extraction of just the textual portion of the HTML.

The final extract contained then 3701 individual scraped newspaper articles published between 2019 and March 2022.

3.2 Annotation

In order to train a NER model, the tool Prodigy has been used to manually annotate. The following eleven entities were annotated for 160 examples, i.e. a fraction of 4% of the full dataset.

- ORT (location)
- VERKAEUFER (seller)
- KAEUFER (buyer)
- GESAMTPREIS (total price)
- FLAECHE (surface)
- STRASSE (street)
- DATUM_VERTRAG (date of contract)
- DATUM_VERBUECHERUNG (date of entry in cadastral register)
- IMMO_TYP (type of property)
- QMPREIS (price per square meter)
- TERRASSENGROESSE (size of terrace)

The annotation process itself was sped up by the use of a gazeteer for the towns known in the region of Vorarlberg. Further, a regex was used to identify total price.

When annotating, it became apparent that the premeditated annotation scheme isn't necessarily unique: sometimes in real languate, the same two words *"private persons"* were used to designate non identified individuals as buyers and sellers, which made annotation of overlapping entities necessary. As well, in some instances for a house, the square meters of the building were given as well as the square meters of the plot of land. This isn't necessarily a problem for the annotation itself, but it is expected to render making sense of the extracted data much more difficult.

A need for an annotation strategy was identified, as there are choices to be made e.g. when annotating a price: whether "Euro" is part of the price or not shouldn't be decided by the individual annotator, but rather consistently across the boards.

Similarly, technical difficulties were encountered during annotation as depending on the length of the entity to be annotated, a trailing "." was sometimes elected in the user interface.

3.3 Postprocessing

All of the data was extracted out of the prodigy database into a single JSON.

Given the way user has to select entirety of token success to be assigned to a particular label it is very easy for a trailing "." to be wrongly selected. As such we removed last trailing non-alphanumeric character but only for those tokens that are the last in a chain of (possibly many) tokens classified in same class. In other words we leave unchanged non-alphanumeric characters that aren't either the last character of a token or that are the last character of a token but that token is itself part of a group of successive tokens all classified in the same label (named entity). For e.g. we want to change labelling of following tokens related to a corporation "ABC GmbH." so as to only label "ABC GmbH" (i.e. remove the trailing ".").

3.4 Summary Statistics

As is to be expected for a usual NER dataset the unlabelled class is much more prominent than labelled ones. Indeed approximately 75pct of all distinct tokens are unlabelled. The lexicon is quite small at 1,152 distinct tokens, with 37pct of these being numbers. The table below contains some further summary statistics of the dataset.

Total Sample Texts	140
Total Tokens	7,717
Total Distinct Labels	1,283
Total Distinct Labelled Tokens	1,822
Avg. No. of Tokens per Label	1.4
Avg. No. of Tokens per Sample Text	55
Std. Dev. No. of Tokens per Sample Text	23
Avg. No. of Labels per Sample Text	9

4 Models

We relied very substantially on linear-chain CRF's (Lafferty et al., 2001; ?; Sutton and McCallum, 2012). By explicitly modelling label-word emissions and label-label transitions these models introduce dependency between successive classes that are actual features of the NER process. Plain-vanilla LSTM-based sequence classification models, on the other hand, treats the classification of successive tokens as effectively independent.
For the pytorch-based CRF implementation we computed starting emissions from conditional probabilities of emission to each word from each one of the 11 class labels.
Besides the plain-vanilla CRF model architecture we then tried to apply it on top of a bi-directional one-layer LSTM network. We essentially took the outputs from the forward-pass of the LSTM layer as the model (label-word) emission probabilities for the CRF layer sitting above it. We essentially drove the neural-network optimization process on the loss expressed as the negative log likelihood coming from the optimal path as per the Viterbi algorithm.

5 Experiments

As an initial baseline, we use a custom model that always predicts the most common label in the training data. We then experimented with a standard CRF (using the sklearn-crf package). By adding some feature functions such as capitalization and a gazetteer for the location field we were able to further improve macro average F1 scores as per the below table.
In terms of metrics we feel macro average F1 score is the most appropriate for an NER use case such as ours that typically has a very large dominant class (the unlabelled or "other" class). Weighted average F1 scores would not be appropriate as they would be focusing our attention precisely on the one (unlabelled or "other") class that we care less about. By weighting each class equally the macro average F1 score attributes importance to correct labelling of all classes.
Regarding metrics a meaningful discussion to be had is whether for two equally-size texts we are indifferent between classifying one 100pct correctly (with the other at zero) versus classifying both at 50pct accuracy. In case we decide only 100pct correctness is appropriate then we can rely on a strict F1 measure.
Finally we should also mention here the critical importance of loss metrics used in the optimization process. As discussed in Li et al. (2020) the typical optimization process relies on loss functions (e.g. cross-entropy loss) that weigh each observation equally. This is not the ideal approach in particular in the presence of the dataset skew we mentioned earlier.
Turning back to the different model architectures, we tried to replicate the same CRF model under the pytorch paradigm. The intent here was not only to explore a different implementation but also to lay the groundwork to add the CRF model to the pytorch-based Stanford XCS224U class code-

base. As can be seen below the results were not entirely encouraging with macro average F1 scores dropping from 66.1pct to 25.2pct for what should essentially be the same CRF model with no feature functions. Further work needs to be done here to account for these differences.

Next we applied the XCS224U class code RNN Sequence Labeling section in Tutorial Pytorch Models. Results here were also considerably underwhelming at 9.0 macro average F1 score (just above always predicting the most common "other" class).

Next we used an explicit implementation of the CRF model (including the Viterbi algorithm) and connected it with features coming from an LSTM network. This model produced good results at average F1 of 74.7, improving over the base CRF with no feature functions (66.1).

Finally we attempted to implement a (pytorch-based) CRF layer on top of the XCS224U LSTM Sequence Labeler (RNNSequenceLabeler class). Judging from the results (at the level of the simple majority baseline and very different from the LSTM-CRF network mentioned just before) it appears there is still an implementation error here.

Model Architecture	M. Avg F1	Strict F1
1. Most Common	7.1	0.0
2. CRF	66.1	11.1
3. CRF (feature f.)	88.6	30.6
4. CRF (pytorch)	25.2	0.0
5. LSTM (XCS224U)	9.0	0.0
6. LSTM+CRF	74.7	26.5
7. LSTM+CRF (XCS224U)	6.9	0.0

6 Analysis

In summary we've concluded for the good performance of feature-function based CRF's at an unbalanced NER task such as the one we attempted. From the high discrepancy between macro-average F1 scores of similarly-situated models we are not entirely confident that either our pytorch-based CRF or our pytorch LSTM-CRF models have been correctly implemented. Further work needs to be done to get these implementations to a state where they can be used for effective research conclusions.

7 Conclusion

Based on this project work, we have two major conclusions. The data work (scraping and annotating) is – contrary to expectations – not as hard as anticipated and can be highly recommended to future cohorts of this course as we had a lot of fun grounding our work in this real life use case.

The second conclusion is that it is harder than expected to reproduce results based on published models: the variance observed in the models we ran is so large that we suspect that there are residual bugs in our implementations.

In this sense, we cannot conclude really with serious comparisons of the model effectiveness.

Aside from ensuring the correctness of our pytorch-based CRF and LSTM-CRF implementations we would like to investigate driving the optimization process on loss functions that don't weigh all observations equally. Indeed we believe the current process is being overly driven by unlabelled tokens (the "other" class) when those are the least interesting to our use case.

Another stream of work we are very curious to undertake is to research optimal number of texts to annotate such that our out-of-sample inference adheres to a preset level of confidence. For this we can draw on the very large unannotated data set that we have held out from the current analysis.

Authorship statement

The three authors contributed throughout to the conclusion of this project. No external assistance outside of the XCS224U course instructors was sought or obtained by any of the authors.

Acknowledgements

Thanks @cgpotts and all course facilitators. We tremendously enjoyed this course and also your generous and candid availability.

References

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

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, San Diego, California. Association for Computational Linguistics.

Xiaoya Li, Xiaofei Sun, Yuxian Meng, Junjun Liang, Fei Wu, and Jiwei Li. 2020. Dice loss for data-imbalanced NLP tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 465–476, Online. Association for Computational Linguistics.

Charles Sutton and Andrew McCallum. 2012. An introduction to conditional random fields. *Foundations and Trends® in Machine Learning*, 4(4):267–373.

Vikas Yadav and Steven Bethard. 2018. A survey on recent advances in named entity recognition from deep learning models. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2145–2158, Santa Fe, New Mexico, USA. Association for Computational Linguistics.