TEXT MINING

L11. INDUSTRIAL TM

SUZAN VERBERNE 2021

Universiteit Leiden

ASSIGNMENT 2 – NER

- Grading :
  - 5 criteria; max 2 points per criterion
    1. General: length correct (2-3 pages) and proper writing + formatting
    2. Description of the task and the data
    3. Description of the adapted features
    4. Baseline run with features from tutorial & experimental runs with adapted features (show results in table: Precision, Recall, F-score for the B and I tags)
    5. Sensible conclusions

EXAMPLE REPORT

Text Mining - Assignment 2: Sequence Labelling

Vinutha Venkatesh & Koen Ponse

November 15, 2021

1 Introduction

In this assignment we will train a Named Entity Recognition (NER) classifier for the task "Emerging and Rare entity recognition" from the Workshop on Noisy User-generated Text (W-NUT)[1]. Named Entity Recogniition (NER) is the process of identifying information such as name of persons, organizations, locations, and numeric expressions like time, date, money, and so on, from unstructured data. While, in some cases, NER is described as a solved task with high reporting scores [1], NER is actually still a difficult task when considering new, unseen data. Especially when considering the recall. This is made clear from the submissions of the 2017 challenge which, at best, managed F1 scores of 41.86[1].

2 Data

We have been given access to three datasets (train, dev and test) containing data from Twitter, Reddit, Youtube (comments) and StackExchange. This data is noisy and can contain unidentifiable information which even humans can find hard to interpret, for example, a tweet *"so... kktny in 30 mins?!"*. This highlights the difficulty of NER currently and with the increasing amount of similar internet comments and new abbreviations, NER will only get more tricky.

Our three datasets are labeled with six different classes, namely a `person, location, corporation, product, creative-work` and `group (e.g. Nirvana)`. Our goal is to label our testing data with these classes as correctly as possible. The distribution of the different classes for each of the datasets is listed in Table 1

	Training	Dev	Test
Person	660	470	429
Location	548	74	150
Corporation	221	34	66
Product	142	114	127
Creative work	140	104	142
Group	264	39	165
Total	3160	1250	1740

Table 1: Different class sizes for each of the data sets

3 Experiments

We use the **CRFSuite** package of **sklearn** which is an implementation of Conditional Random Fields (CRF) to label our sequential data. CRFs are especially useful in prediction tasks where the current prediction is impacted by contextual information or state of the neighbours.
The Baseline experiment uses the **lgfbs** training algorithm, as described in the SKlearn-CRFsuite tutorial[2]. We then predict the scores on the test set and the results for each algorithm settings is displayed in Figure 1a. The final best performing configuration scores are documented in Table 5.

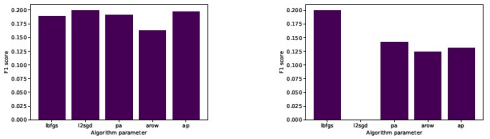
[1] https://noisy-text.github.io/2017/emerging-rare-entities.html
[2] https://sklearn-crfsuite.readthedocs.io/en/latest/tutorial.html

3.1 Hyperparameter tuning

Hyperparameter tuning is performed in order to improve the quality of the model. CRF provides five different training algorithms 'lbfgs' (gradient descent using the L-BFGS method), 'l2sgd' (Stochastic Gradient Descent with L2 regularization term), 'ap' (Averaged Perceptron), 'pa' (Passive Aggressive) and 'arow' (Adaptive Regularization Of Weight Vector). Each of these algorithm has its own set of hyperparamaters to tune. For each of the algorithms we defined ranges for each hyperparameter and we performed a RandomizedSearch with 250 iterations for each of the algorithms (1250 iterations in total). We used 3-fold cross validation for each of the iterations where each iteration was validated on the dev set. For the precise definition of our hyperparamater ranges, we would like to refer to our code. The results for each algorithm parameter setting is displayed in Figure 1a. The final results of the best configuration can be found in Table 5.

3.2 Feature selection

Next, we attempted to enlarge the feature set with extra features, tailored to the data. Due to the high amount of tweets, naturally a boolean, flagging words starting with "@" and "#", should be included. Furthermore, we took inspiration from one of the contestents in the 2017 challange [2], as they included a CRF approach and listed features such as stopwords, first few characters, small_word and containsDigit. The original complete feature list can be found in Table 2 and our additions can be found in Table 3. Before validating our results, we performed the same kind of randomized parameter tuning as in the previous Section. The results can be found in Table 5.



(a) Hyperparamter tuning (b) Hyperparamter tuning with new feautures

Figure 1: F1 scores for different algorithms settings with their individual hyperparameters optimized

Feature name	Description
word.lower()*	Word in lowercase format
word[-3:]	Last three characters of the word
word[-2:]	Last two characters of the word
word.isupper()*	Boolean value to check if word is in Uppercase
word.istitle()*	Boolean value to check if word is in Titlecased
word.isdigit()	Boolean value to check if word is a digit
postag*	Part-of-speech tag of the word
postag[:2]*	First two characters of POS tag of word
BOS	Checks if word is at the beginning of sentence
EOS	Checks if word is at the end of sentence

*Each word also carried the information marked with asterisks of the previous and next word (if possible)

Table 2: The Features used in the **sklearn-crfsuite** tutorial.

4 Conclusion

We have implemented a CRF with CRFSuite in python to perform Named Entity Recognition (NER) on noisy User-generated text data. We concluded that the learning task of this particular data set is quite difficult as high F1 scores are hard to obtain. We performed hyper parameter tuning with in total 1250 random iteration on some basic features. This increased performance over the baseline implementation. We then added more features and performed the same hyper parameter tuning, unfortunately little to no extra performance was gained from the additional features. Possible the curse of dimensionality kicked in when adding more features. Feature reduction techniques may aid in this and may improve performance further then we analysed.

Feature name	Description
word[:2]	First two characters of the word
word[:3]	First three characters of the word
wordFreq	Lists the word frequency
word_small*	Checks whether the word is less than 5 characters
stopword*	Checks whether the word is a stop word
containsdigit*	Checks whether the word contains a digit
word.@*	Checks whether word starts with '@'
word.#*	Checks whether word starts with '#'
word.url*	Checks whether the word is a url

*Each word also carried the information marked with asterisks of the previous and next word (if possible)

Table 3: The list of custom features used for this task.

Table 4: The Precision, Recall and F1 scores for Baseline results and after Hyperparameter Optimization(Part1).

	Baseline			Hyperparameter			Hyperparameter with custom features		
	P	R	F1	P	R	F1	P	R	F1
B-corporation	0.000	0.000	0.000	0.333	0.015	0.029	1.000	0.015	0.030
I-corporation	0.000	0.000	0.000	0.000	0.000	0.000	0.00	0.000	0.000
B-creative-work	0.333	0.035	0.064	0.193	0.113	0.142	0.183	0.092	0.122
I-creative-work	0.296	0.037	0.065	0.185	0.225	0.203	0.183	0.202	0.192
B-group	0.300	0.036	0.065	0.500	0.012	0.024	0.000	0.000	0.000
I-group	0.357	0.071	0.119	0.600	0.043	0.080	0.000	0.000	0.000
B-location	0.385	0.233	0.290	0.387	0.193	0.258	0.435	0.180	0.255
I-location	0.231	0.064	0.100	0.286	0.064	0.104	0.294	0.053	0.090
B-person	0.551	0.138	0.220	0.442	0.364	0.399	0.464	0.350	0.399
I-person	0.547	0.221	0.315	0.442	0.328	0.369	0.458	0.336	0.388
B-product	0.600	0.024	0.045	0.123	0.071	0.090	0.174	0.063	0.092
I-product	0.375	0.048	0.085	0.070	0.040	0.051	0.078	0.032	0.045
micro avg	0.430	0.093	0.153	0.302	0.183	0.228	0.324	0.170	0.223
macro avg	0.331	0.076	0.114	0.295	0.122	0.146	0.273	0.110	0.134
weighted avg	0.401	0.093	0.142	0.327	0.183	0.208	0.297	0.170	0.200

Table 5: The Precision (P), Recall (R) and F1 scores (F1) for Baseline results, after Hyperparameter Optimization(based on tutorial) and after Hyperparameter Optimization using Custom Features

References

[1] L. Derczynski, E. Nichols, M. van Erp, and N. Limsopatham, "Results of the wnut2017 shared task on novel and emerging entity recognition," in *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pp. 140–147, 2017.

[2] U. K. Sikdar and B. Gambäck, "A feature-based ensemble approach to recognition of emerging and rare named entities," in *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pp. 177–181, 2017.

Universiteit Leiden

EXAMPLE RESULTS TABLES

featureset	precision	recall	F1-score
Default	**0.330**	0.106	0.153
All features	0.293	**0.119**	**0.163**
Ablation of POS tag	0.324	0.103	0.150
Ablation of prefixes	0.264	0.119	0.158
Ablation of suffixes	0.281	0.101	0.141
Ablation of context	0.277	0.083	0.122
Ablation of shape	0.270	0.109	0.149

Feature set	Precision	Recall	F1
Baseline	0.399	0.098	0.145
R2	0.363	0.121	0.169
R3	0.376	0.097	0.140
R4	0.397	0.129	0.181
Baseline + optimised parameters	0.258	0.193	0.205
R2 + optimised parameters	0.293	0.199	0.221
R3 + optimised parameters	0.249	0.207	0.215
R4 + optimised parameters	0.282	0.214	0.228

	Model With Prefixes And Word Length			Model With Gazetteers		
BIO-tag	Precision	Recall	F1-score	Precision	Recall	F1-score
B-corporation	0.400	0.030	0.056	0.200	0.045	0.074
I-corporation	0.000	0.000	0.000	0.667	0.091	0.160
B-creative-work	0.250	0.056	0.092	0.294	0.070	0.114
I-creative-work	0.276	0.073	0.116	0.244	0.087	0.128
B-group	0.250	0.048	0.081	0.185	0.030	0.052
I-group	0.231	0.086	0.125	0.222	0.057	0.091
B-location	0.381	0.267	0.314	0.402	0.300	0.344
I-location	0.310	0.138	0.191	0.457	0.223	0.300
B-person	0.608	0.322	0.421	0.611	0.322	0.421
I-person	0.543	0.336	0.415	0.579	0.336	0.425
B-product	0.136	0.024	0.040	0.174	0.031	0.053
I-product	0.158	0.048	0.073	0.179	0.056	0.085
weighted avg	0.365	0.163	0.217	0.376	0.174	0.231

ADVICE FOR FINAL ASSIGNMENT

- Formulate your own RQ
  - But don't feel the urge to be extremely original; comparing methods or representation (embedding/feature) types is perfectly fine
- Start experimentation early
  - So that you can change plans if it doesn't work as expected
- Feel free to experiment with methods of others
  - Applying an existing method to a different problem can be interesting
- If you want to use BERT models, have a look at
  - https://huggingface.co/
  - https://www.sbert.net/ (sentence BERT models)
- Make sure you understand what you do
- Negative results are fine! But always report what you did and what came out

ADVICE FOR FINAL ASSIGNMENT REPORT

- Describe the problem and define the task in your introduction
- Describe the data qualitatively and quantitatively
- Report what settings you compared and what the results were
- Summarize the most important results in neatly formatted tables
- *Never* copy text from (web) sources or other students. This is considered plagiarism and will be reported to the Board of Examiners.
- Observe the page limit! Don’t use appendices that go beyond the page limit

HOMEWORK

- In preparation of the final assignment, have a look at (see Brightspace):

1. Verberne, S., D'hondt, E., van den Bosch, A., & Marx, M. (2014). Automatic thematic classification of election manifestos. *Information Processing & Management*, *50*(4), 554-567.
2. Karimi, S., Metke-Jimenez, A., Kemp, M., & Wang, C. (2015). Cadec: A corpus of adverse drug event annotations. *Journal of biomedical informatics*, *55*, 73-81.
3. Rosenthal, S., Farra, N., & Nakov, P. (2017, August). SemEval-2017 task 4: Sentiment analysis in Twitter. In *Proceedings of the 11th international workshop on semantic evaluation (SemEval-2017)* (pp. 502-518).
4. Stab, C., & Gurevych, I. (2017). Parsing argumentation structures in persuasive essays. *Computational Linguistics*, *43*(3), 619-659.

And choose one of the topics before December 6.

GUEST LECTURE

KASPER KOK, TEXTKERNEL