Project Report: Part-Of-Speech Tagging and Named Entity Recognition with Probabilistic Graphical Models

Clemens Biehl (clemens.biehl@stud.tu-darmstadt.de)[1], Daniel Wehner (daniel.wehner@stud.tu-darmstadt.de)[1], and Fabian Otto (fabian.otto@stud.tu-darmstadt.de)[1]

[1]Technische Universität Darmstadt

***Abstract*—Natural language processing is an increasingly important task, in which Part-of-Speech (POS) tagging and Named Entity Recognition (NER) are important steps for complex semantic processing tasks. Therefore, in this project we evaluate different probabilistic graphical models – Naïve Bayes, Hidden Markov Models and Conditional Random Fields – for those two tasks and additionally compared the performance of different handcrafted features as well as GloVe word embeddings. Furthermore, we also evaluated the models using a significantly reduced size of the training data set. The results indicate that the CRF model performs best in various settings almost irrespective of the features chosen. For POS tagging HMMS and CRFs tend to work better since they operate on sentence-level.**

I. Introduction

Natural Language Processing (NLP) is the application of computational techniques for the automatic analysis and representation of human language. This field becomes increasingly interesting as the number of available resources on the web is increasing and millions of websites and documents can be accessed and processed. Making use of this large amount of unstructured data to enable and improve computational natural language understanding is therefore a task worth investigating more in depth [**?**].

The tasks of **part-of-speech (POS) tagging** as well as **named entity recognition (NER)** are required for various downstream tasks such as **question answering** or **information extraction**. However, these tasks are non-trivial, since natural language expressions can be ambiguous in multiple ways. For instance a word can have multiple parts-of-speech and a term like *Florence* might refer to either a named entity `PERSON` or `LOCATION` [**?**, cf. pages 167-169].

The goal of this project is to compare different approaches of manual feature engineering and word embeddings as well as different probabilistic graphical models on the tasks of part-of-speech tagging and named entity recognition. The **Groningen Meaning Bank** of University of Groningen is going to be used as a training corpus [**?**].

II. Literature Review

A. Natural Language Processing Pipeline

Solving a natural language processing (NLP) problem necessitates a pipeline – a sequence of tasks which have to be dealt with in order. POS tagging is one key part in almost every NLP pipeline. It can e. g. be used to facilitate the identification of named entities or constituents and dependencies. Figure 1 depicts such an NLP pipeline.

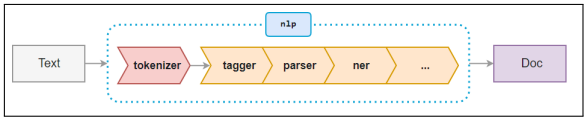

Figure 1. Natural Language Processing consists of several steps. These steps can be arranged in an NLP pipeline.

As can be seen in Figure 1 POS tagging is applied at an early stage of the pipeline which emphasizes its very importance. Almost all tasks in NLP rely on POS tagging making its performance crucial for the performance of the entire pipeline.

B. Part-Of-Speech Tagging

POS tagging refers to the assignment of part-of-speech tags to words in a text, i. e. the task is to decide whether a word is a noun (`NN`), plural-noun (`NNS`), verb (`VB*`), etc. There are several *rule-based* and *stochastic algorithms* to perform this classification of word categories.

While most words in large text corpora are nouns, accurate part-of-speech tagging is still a challenging task, because words can be ambiguous (polysemy), i. e. can have different parts-of-speech. One way to distinguish between the different possible parts-of-speech for a particular word occurrence is to incorporate the context of this word, i. e. the surrounding tokens, as well as morphological features such as affixes, prefixes, etc. [**?**, cf. pages 167-170]

State-of-the-art POS taggers reach a word-level accuracy of 0.97 to 0.98, whereas the sentence-level performance is around 0.56 [**?**] [**?**] [**?**].

Example:

> [**She** — `PRP`] [**sells** — `VBZ`] [**seashells** — `NNS`] [**on** — `IN`] [**the** — `DT`] [**seashore** — `NN`] [**.** — **.**]

C. Named Entity Recognition

Named Entity Recognition (NER) is the task of identifying spans of text that represent proper names and classifying these

as a particular type of entity such as PERSON, LOCATION or DATE, etc. (see example below) A model which seeks to solve this task needs to deal with ambiguities arising from the fact that some words (e. g. *Florence*) could refer to multiple entities (e. g. PERSON, LOCATION).
Again, the context of a named entity can be a useful feature for NER and help to disambiguate between multiple entity types in such cases [**?**, cf. pages 761-765]. Deep learning based methods for NER achieve an F1 measure of about 0.91 on the CoNLL-2003 test set[**?**].

Example:

> *The decision by the independent member of parliament Andrew Wilkie to withdraw his support for the minority Labor government sounded dramatic but it should not further threaten its stability. When, after the 2010 election, Wilkie, Rob Oakeshott, Tony Windsor and the Greens agreed to support Labor, they gave just two guarantees: confidence and supply.*

Colours: Person – Organization – Date

D. Naïve Bayes

The Naïve Bayes classifier is a generative model which assumes that all subsets of features are independent given the class label: $(\boldsymbol{X} \perp\!\!\!\perp \boldsymbol{Y}|C)$ Thus, the maximum aposteriori (MAP) can be computed as follows:

$$C^*_{MAP} = \arg\max_{C \in \mathcal{C}} P(C) \cdot \prod_{i=1}^{m} P(x_i|C)$$

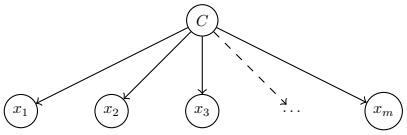

Figure 2. Naïve Bayes as a probabilistic graphical model. The features $\boldsymbol{x}$ are independent given the class label C.

Naïve Bayes operates on token level, i. e. the context cannot be incorporated into the predictions. In the context of our experiments the Naïve Bayes classifier is going to serve as a baseline for the performance in both tasks POS tagging and NER. The implementation of the *nltk* framework is going to be used.[1]

E. Hidden Markov Models

Hidden Markov models (HMMs) are directed probabilistic graphical models which – in contrast to the Naïve Bayes algorithm (which classifies on token level) – operate on sequence level, i. e. they assign the most probable sequence of labels. This allows it to incorporate the context into the prediction. The latter property is especially useful for natural language processing, where words can have different meanings depending on the context (words around the word of interest.) In POS tagging we want to find the best tag sequence $\boldsymbol{t}^*$ given an input sequence of words $\boldsymbol{w}$:

$$\boldsymbol{t}^* = \arg\max_{\boldsymbol{t}} P(\boldsymbol{t}|\boldsymbol{w}) \propto \arg\max_{\boldsymbol{t}} P(\boldsymbol{w}|\boldsymbol{t})P(\boldsymbol{t})$$

Unfortunately, this equation is hard to compute. This is why HMMs make two simplifying assumptions which facilitate the computation process:

1) The probability of a word appearing depends only on its own POS tag (it is independent of all the other words and tags around it): $P(\boldsymbol{w}|\boldsymbol{t}) = \prod_{i=1}^{n} P(w_i|t_i)$
2) Due to the sparsity of the training data it is in general hard to estimate $P(\boldsymbol{t})$ well enough. Hence the so-called **Markov property** is assumed, i. e. the probability of a particular state depends solely on the previous state and not on earlier ones.[2] [**?**, cf. pages 211-212]: $P(w_i|w_1, \ldots, w_{i-1}) = P(w_i|w_{i-1})$

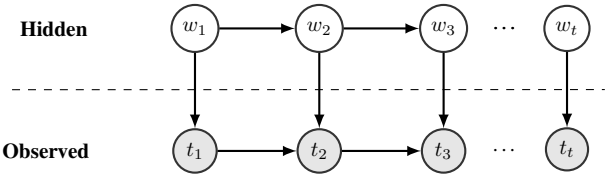

Figure 3. An example of a Hidden Markov model.

Putting all components together the HMM model can be written as:

$$\boldsymbol{t}^* = \arg\max_{\boldsymbol{t}} \prod_{i=1}^{n} P(w_i|t_i) \cdot P(t_i|t_{i-1})$$

In the task of POS tagging the probabilities can be obtained by counting the occurrences in a large training corpus (e. g. the BROWN corpus):

$$P(t_i|t_{i-1}) = \frac{\#\{t_{i-1,t_i}\}}{\#\{t_{i-1}\}}$$

$$P(w_i|t_i) = \frac{\#\{t_i, w_i\}}{\#\{t_i\}}$$

Among others the **Viterbi** algorithm (dynamic programming approach) can be used in order to compute the best matching tag sequence.

Example use cases for HMMs: In [**?**] the authors use Brant's TnT tagger [**?**] for POS tagging which uses Hidden Markov Models (HMM). Based on the error the tagger makes they additionally learn transformation rules which should correct the errors. The authors report an F1-Score of 79.66 % (without error correction) and 80.74 % with error correction.

[1] https://www.nltk.org/

[2] *The future is independent of the past given the present*

F. Conditional Random Fields (CRFs)

In general, the CRF algorithm can be viewed as logistic regression applied to sequence data and belongs to the family of discriminative machine learning approaches. Conditional Random Fields (CRF) are also closely related to hidden markov models. But unlike HMMs they are of undirected nature. [**?**] When making predictions CRFs have to take the previous context into account. This is achieved by using so-called feature functions. In the context of part-of-speech tagging the feature function could be of the following type:

$$f(\boldsymbol{X}, i, l_{i-1}, l_i)$$

Where $\boldsymbol{X}$ is the set of input vectors, i the position of the data point we want to predict and l_{i-1} / l_i the label of the previous word and the current word respectively. The conditional probability $P(y|\boldsymbol{X}, \lambda)$ can be computed as follows:

$$P(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\lambda}) = \frac{1}{Z(\boldsymbol{X})} \exp\left\{\sum_{i=1}^{n}\sum_{j}\lambda_j f_i(\boldsymbol{X}, i, y_{i-1}, y_i)\right\}$$

$$Z(\boldsymbol{X}) = \sum_{y' \in \boldsymbol{Y}}\sum_{i=1}^{n}\sum_{j}\lambda_j f_j(\boldsymbol{X}, i, y'_{i-1}, y'_i)$$

For the training procedure it is required that the training data be i.i.d. (independent and identically distributed): $\mathcal{D} = \left\{(x^{(1)}, y^{(1)}, (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)}\right\}$. The task is to find the optimal parameters $\boldsymbol{\lambda}^*$. For that, the negative log-likelihood is used:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\lambda}, \mathcal{D}) &= -\log\left(\prod_{k=1}^{m} P(\boldsymbol{y}^{(k)}|\boldsymbol{x}^{(k)}, \boldsymbol{\lambda})\right) \\ &= -\sum_{k=1}^{m}\log\left[\frac{1}{Z(\boldsymbol{x}^{(m)})}\exp\left\{\sum_{i=1}^{n}\sum_{j}\lambda_j f_j(\boldsymbol{x}^{(m)}, i, y^k_{i-1}, y^k_i)\right\}\right]\end{aligned}$$

Optimizing the above equation yields the optimal parameter configuration $\boldsymbol{\lambda}^*$. As it turns out this optimization problem is convex and has therefore no local optima. The global maximum of the function can be found by minimizing it:

$$\boldsymbol{\lambda}^* = \arg\min_{\boldsymbol{\lambda}}\left\{\mathcal{L}(\boldsymbol{\lambda}, \mathcal{D}) + \underbrace{C\frac{1}{2}\|\boldsymbol{\lambda}\|^2}_{\text{regularization}}\right\}$$

The parameter C controls the degree of regularization which is a method to prevent overfitting. In the equation above the L_2 regularization is employed.

Example use cases for conditional random fields: The authors of [**?**] make use of a conditional random field ('CRF++, Yet another CRF package') in order to perform POS tagging for the language Hindi. The authors of the paper trained the model on 21.000 words and report an accuracy of 82.67 %.
Named entity recognition represents another NLP problem which can be addressed by CRFs [**?**]. The authors report results on the CoNLL-2003 named entity recognition shared task consisting of tagged news articles. They achieved an F1-score of 84.04 % (for the English language) and 68.11 % (for the German language). The classes were as follows: `PERSON`, `LOCATION`, `ORGANIZATION` and `MISC`.

III. Methodology and Objective

A. General Objective

This project seeks to accurately identify POS and NE tags by using different probabilistic graphical models. Further, this project utilizes two distinct feature sets – manually engineered features (1) and GloVe word embeddings (2) [**?**] – in combination with a Naïve Bayes and a CRF. Additionally a hidden Markov model is evaluated given the token sequence as input. For computing the Naïve Bayes model, the *nltk* framework[3] is going to be used, the hidden Markov model is based on the implementation from nltk[4] as well, since it already provides an interface for conveniently processing natural language. The *python-crfsuite*[5] package provides a wrapper to *CRFSuite* as an implementation of linear chain conditional random fields which is used in this project.
All approaches are evaluated with the micro-averaged F1 measure as harmonic mean between Precision and Recall and compared to up-to-date benchmarks.

B. Research questions

- **Q1** How do CRF and HMM compare to the baseline of Naïve Bayes?
- **Q2** Which handcrafted features are suited best for the different models in order to reach a high accuracy for POS tagging and NER?
- **Q3** How do handcrafted features compare to word embeddings?
- **Q4** Can graphical models with well-engineered features beat deep learning benchmarks?

IV. Results and Evaluation

For our evaluation we assessed different feature combinations. These included:

- word features:
  - **w:** the word itself
  - **l:** the lowercased word
  - **s:** the stem of the word
- unknown word features (**uwf**):
  - word shapes[6] (e. g. Hello $\longrightarrow$ Xxxxx)
  - capitalization
  - affixes
- bigram features (**bf**)
- a combination of all of the above

For named entity recognition we also included the gold POS tags (of the current token in a sequence as well as the previous token) into our feature set (**pos**).
All results are evaluated on word-level using the micro averaged F1 score and by sentence-level accuracy.

[3] `https://www.nltk.org/`
[4] `https://www.nltk.org/_modules/nltk/tag/hmm.html`
[5] `https://python-crfsuite.readthedocs.io/en/latest/`
[6] The term *word shape* refers to an abstract representation of a character sequence in which alphanumerical characters are replaced by *x* or capital *X* and digits are replaced by *d*

All models are trained using a **maximum likelihood estimation** approach. For the Naïve Bayes model the number of feature occurrences per label is counted, the sufficient statistics for the HMM model are obtained by collecting the frequencies of transitions between hidden states and token observations within each state. The CRF model is trained using the **orthant-wise limited-memory quasi-Newton** optimization algorithm, a variant of the L-BFGS optimization algorithm, which optimizes the L_1-regularized log-likelihood [**?**]. Using L-BFGS optimization without L_1 regularization led to a notable decrease of 0.017 in sentence-level accuracy for the best CRF model for NER.

A. Part-of-Speech Tagging

Compared to the Bi-LSTM baseline used by the creators of the data set, Bjerva et al. [**?**], we found that our CRF model achieves a higher F1 score. However, the authors used an older version of the Groningen Meaning Bank comprising less data. Further, we did not find any additional work utilizing the Groningen Meaning Bank to compare our results with.

Naïve Bayes

Table I shows the results of a Naïve Bayes classifier using only the word features. This setting already achieves an F1 score of approximately 0.92. This can be explained by the unbalanced data set which mainly contains nouns (`NN`). Adding bigram features significantly increased the F1 score performance to 0.957 and almost doubled the sentence-level accuracy. In general we found that Naïve Bayes performs poorly on sentence-level accuracy. This is due to the word level classification which also explains the increase by using bigram features. In contrast, the unknown word features led to a decrease in performance. By analyzing the probabilities and features importance we did not find any clear evidence for the performance decrease caused by the unknown word features.

Table I
POS PERFORMANCE EVALUATION OF NAÏVE BAYES

Naïve Bayes [POS]		
Features	**F1 score**	**Acc (word / sent)**
w, l, s	0.927	0.927 / 0.240
w, l, s, uwf	0.928	0.928 / 0.241
w, l, s, bf	0.957	0.957 / 0.467
w, l, s, uwf, bf	0.949	0.949 / 0.369

The Naïve Bayes model is able to identify basic rules to identify the POS tags correctly, as can be seen by observing the most important features when ranked by their conditional probability *(e. g. the suffix st is a strong indicator of an adjective in superlative, while the suffix ll indicates a modal verb)*. There are a lot of rather trivial yet high-probability features such as that *'s* is a possessive ending or that *a* is a determiner.

The Naïve Bayes model frequently confuses the tags `VBG` (e. g. the verb *be* as gerund) and `VBP` (e. g. the verb *be* in singular present form, *am*). This is the case for 22 % of all `VBG` tags in the test set. An explanation for this is that these words occur in very similar contexts. However, even with the bigram features included the model still is not able to discriminate them clearly as the prediction of one POS tag is independent of the prediction of the next tag in the sequence.

Hidden Markov Models

As HMMs do not support any features we only evaluate sentence-level and word-level accuracy for the word sequence itself (see table II). We found that the HMM could not generalize well and often incorrectly predicted very frequent POS tags, especially plural nouns (`NNS`). On sentence-level accuracy the HMM shows a slight improvement compared to Naïve Bayes. This can be attributed to the structure of the HMM which enables it to find the **most probable tag sequence** for the whole sentence. Hence, it does not only operate on the word-level like Naïve Bayes does.

The most common misclassifications included the incorrect prediction of plural nouns (`NNS`). Additionally, adjectives and adverbs in comparative form were frequently mixed up by the classifier (23 % of all comparative adverbs in a test set of 20 %).

Table II
POS PERFORMANCE EVALUATION OF HMM

HMM [POS]		
Features	**F1 score**	**Acc (word / sent)**
word sequence	0.837	0.837 / 0.496

Conditional Random Fields

Like the HMM the CRF operates on sentence-level, but unlike the HMM they also allow for the usage of features, which results in an additional increase in performance. Furthermore, arbitrary model parameters/weights can be learned and are not only inferred from the data as conditional probabilities. This allows for a richer structure. Table III proves that CRFs performed **uniformly well, independent of the specific choice of features**, in comparison to the other two models. This also applies to the sentence-level accuracy. However, we found that unknown word features were more important than bigram features when used with a CRF model. Combining all features leads to the best result.

We analyzed the most informative features, i. e. the features with the largest positive weights. These features are mostly encoding common linguistic rules, such as:

- capitalized $\longrightarrow$ noun
- previous word *have* or *be* $\longrightarrow$ verb
- suffix *ly* $\longrightarrow$ adverb
- suffix *ed* $\longrightarrow$ verb in past tense
- suffix *ing* $\longrightarrow$ verb as gerund
- suffix *ous* $\longrightarrow$ adjective

The model also incorporated frequent mappings of words to POS tags, e. g. that *toward* is a usually a preposition or that *dead* is usually an adjective.

The most likely label-to-label transitions according to the highest positive weights of the model are also linguistically plausible:

- modal verb ⟶ verb
- adjective ⟶ noun or adjective
- adverb ⟶ verb
- personal pronoun ⟶ verb
- to ⟶ verb

Table III summarizes the results of our experiments.

Table III
POS PERFORMANCE EVALUATION OF CRF

CRF [POS]		
Features	**F1 score**	**Accuracy (word / sent)**
w, l, s	0.974	0.974 / 0.607
w, l, s, uwf	0.980	0.980 / 0.749
w, l, s, bf	0.978	0.978 / 0.669
w, l, s, uwf, bf	0.985	0.985 / 0.740

B. Named Entity Recognition

Similar to the POS tagging task the CRF model performed best on NER, while the HMM outperformed the Naïve Bayes baseline on sentence-level as well as on word-level accuracy. Similar to the POS tagging task, the HMM exhibits a considerably higher sentence-level accuracy. Again, the Naïve Bayes model yields a fairly high F1 score due to the class label imbalance as there are mostly outside (`O`) tags/non-entity tokens in the data set (see table IV). Tables IV, V and VI contain the evaluation of Naïve Bayes, HMM and CRF for the NER task using a training set of 80 % of the Groningen Meaning Bank.

Table IV
NER PERFORMANCE EVALUATION OF NAÏVE BAYES

Naïve Bayes [NER]		
Features	**F1 score**	**Acc (word / sent)**
w, l, s	0.921	0.921 / 0.262
w, l, s, uwf	0.922	0.922 / 0.237
w, l, s, uwf, pos	0.921	0.921 / 0.240
w, l, s, uwf, pos, bf	0.929	0.929 / 0.288

Interestingly, the performance of Naïve Bayes does **not vary much when changing the feature sets**, the basic word features (**word**, **lowercased word**, **stem**) seem to provide most of the meaningful information, while the **unknown word features** and **pos** tags do not seem to add any valuable information for discriminating the NER tags. The bigram features slightly improved the performance, which indicates the importance of incorporating the context of a token in such a sequence labelling task.

The CRF model seems to benefit more from incorporating additional features, concretely the **unknown word features**, **pos** tags and **bigrams**. The most informative features for the Naïve Bayes model include many idiosyncrasies, i. e. highly specialized trivial feature-to-label mappings based on the training corpus, such as that the stem *russian* indicates the NER tag `gpe-nam` (a nation as geopolitical entity). But there are also some meaningful abstractions, e. g. that the word shape *Xx.* indicates a person title (NER tag `per-tit`) and often preceeds a family name (NER tag `per-fam`) or that a token consisting only of digits is likely to be a year (NER tag `tim-yoc`).

Table V
NER PERFORMANCE EVALUATION OF HMM

HMM [NER]		
Features	**F1 score**	**Acc (word / sent)**
word sequence	0.946	0.946 / 0.523

Table VI
NER PERFORMANCE EVALUATION OF CRF

CRF [NER]		
Features	**F1 score**	**Accuracy (word / sent)**
w, l, s	0.965	0.965 / 0.604
w, l, s, uwf	0.969	0.969 / 0.639
w, l, s, uwf, pos	0.969	0.969 / 0.645
w, l, s, uwf, pos, bf	0.974	0.974 / 0.693

The list of features with the largest positive weights in the CRF model contains more context features in comparison the Naïve Bayes model. Examples of these most weighted features are:

- next lowercased token is *korea* ⟶ `gpe-nam`
  (i. e. the words *south* and *north* belong to a geopolitical entity when succeeded by *korea*)
- suffix *ber* ⟶ `tim-moy` (month of year)
- suffix *tty* ⟶ `per-nam` (person name)
- next lowercased token is *mayor* → `gpe-nam`
  (i. e. the word *mayor* is usually preceeded by a city/geopolitical entity)

The most likely label-to-label transitions as indicated by the largest positive weights of the CRF model show that the model abstracts basic rules from the training corpus:

- `per-giv` ⟶ `per-fam`
  (i. e. a person's given name is likely to be followed by their family name)
- `per-tit` ⟶ `per-giv`
  (i. e. a person title is likely to be followed by a person's given name)
- `org-nam` ⟶ `org-leg`
  (i. e. an organization name is likely to be followed by the organization's legal form)

The CRF model also learns that some transitions are rather unlikely, such that a geopolitical entity name is followed by a person title or that a person's given name is directly followed by the outside tag (`O`). These transitions are all very plausible in the context of the Groningen Meaning Bank. However, they can be interpreted as a sign of overfitting to this corpus. It is for instance very domain-dependent whether the notion that given names and family names most likely occur together holds true. A counterexample is the domain of chats or social media posts, where last names are frequently omitted. Therefore the generalization across multiple data sets will be limited.

C. Comparison

The experiments demonstrate that the **CRF model outperforms** HMMs and Naïve Bayes on both POS tagging as well as NER. HMMs showed **increased performance on sentence-level** accuracy compared to the baseline of Naïve Bayes, but performed equally well regarding the word-level accuracy. When learning with only 2 % of the available training data (approximately 1000 example sentences) and all features except word embeddings, a **CRF model still provided a comparable performance** of 0.958 word-level F1 score and a lower sentence-level accuracy of 0.548. For Naïve Bayes the word-level F1 score decreased to 0.910, but the sentence-level accuracy dropped to 0.218. Utilizing only 0.1 % of the training data reduced the performance of the CRF model below the baseline trained on 80 % of all available data. The CRF model increasingly confuses person, organization and event names as the amount of training data is reduced. The Naïve Bayes model also confuses these entities, but more frequently, and additionally predicts the label `year` for tokens which are actually assigned to the entity `day-of-month`.

Using only the 100-dimensional GloVe word embeddings[7] (trained on Wikipedia 2015 and the Gigaword 5 corpus) for the CRF model led to a **strong drop in performance** (to 0.882 F1 score), almost down to the majority class baseline (outside tag `O`) of 84.4 %. A combination of hand-crafted features and the word embeddings for the CRF model yielded better results (0.957 word-level F1 score, 0.546 sentence-level accuracy), but did not achieve a better performance than the hand-crafted features themselves. While it is generally possible to use continuous features such as the GloVe word embeddings with CRF models, the bad performance in this case might be caused by the implementation.

The appendix contains the confusion matrices for all models on the NER task using all hand-crafted features as well as two figures showing

D. Conclusion

This project examined the performance of Naïve Bayes, Hidden Markov Models and Conditional Random Fields on the Groningen Meaning Bank data set using different feature sets. We can conclude that a **CRF model provided the best performance** and was – in contrast to the Naïve Bayes baseline – tolerant towards less informative features, such as word shape features in the POS tagging task. We also tried to use word embeddings as features for the CRF model, but with limited success, possibly due to the handling of continuous data by the concrete implementation. Our CRF model achieved a sentence-level accuracy of 0.740 on the POS tagging task, which is remarkably higher than the baseline of 0.560 reported by Manning [**?**]. This indicates that we should verify our results thoroughly on other data sets as well as on other POS tag sets (i. e. labels).

Besides learning how HMM and CRF models work, we have learned that sequence labelling tasks are challenging when only small data sets are used and that graphical models such as Conditional Random Fields can achieve a competitive accuracy in this situation, since very large data sets for deep learning may not always be available in practice [**?**] [**?**]. Furthermore, the results we obtained with the graphical models could be interpreted without much overhead, since we could easily obtain the conditional probabilities or weights for single feature observations and use them to explain the model behaviour. Challenging tasks were the performance of the implementation and the development of the data preprocessing pipeline. We would also like to conduct further testing on other data sets to enable better comparisons to the results of others and to test for the generalization across different data sets. Other possible extensions include the usage of external lexical resources to generate additional features, an in-depth comparison with a Bi-LSTM with little data and an adaptation of the CRF implementation to handle the word embeddings as continuous features differently.

The implementation is available on GitHub:
`https://github.com/DaWe1992/PGM_Project`

[7] `https://nlp.stanford.edu/projects/glove/`

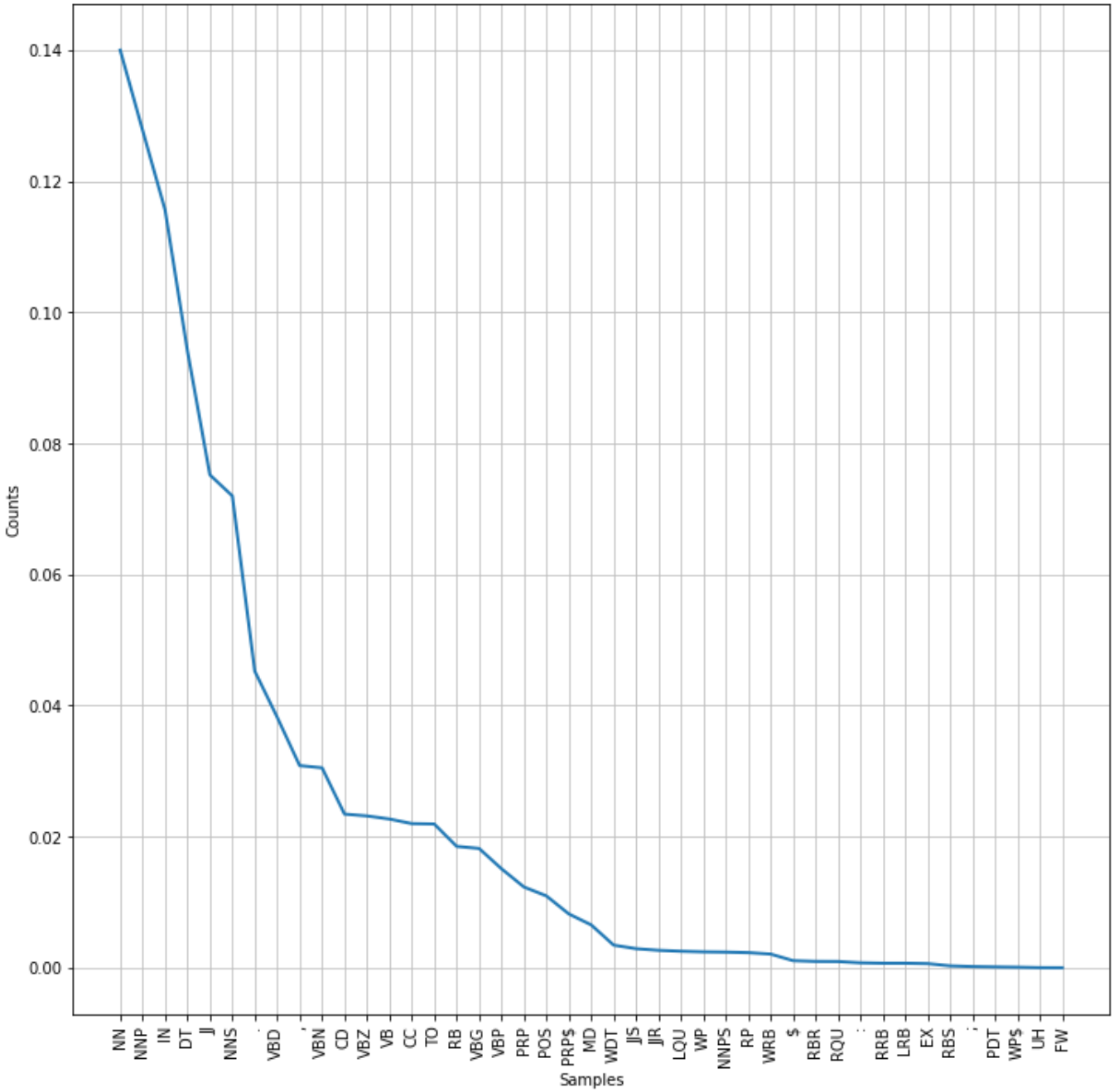

Figure 4. Part-of-Speech tag distribution in the Groningen Meaning Bank.

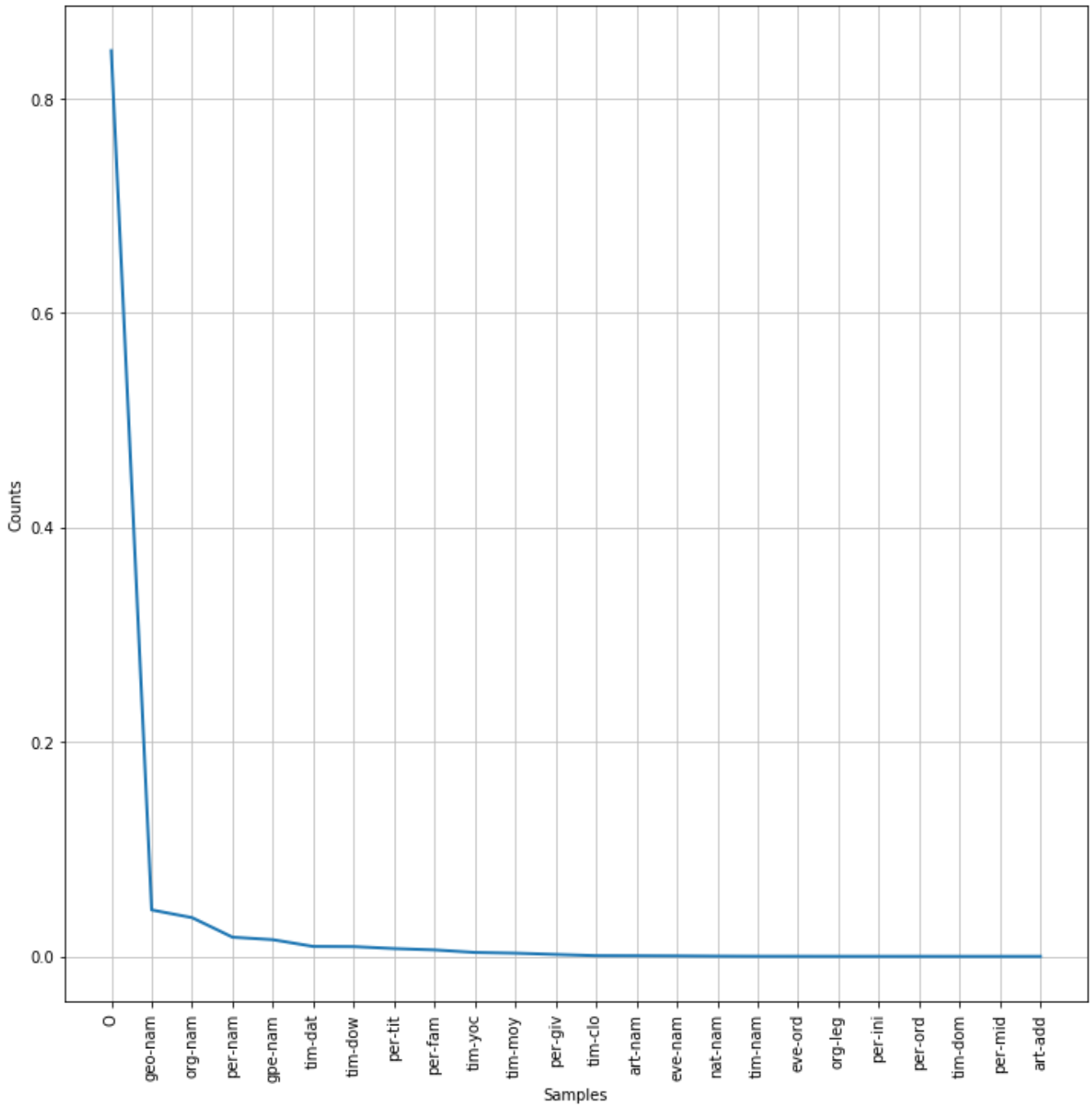

Figure 5. Named Entity tag distribution in the Groningen Meaning Bank.

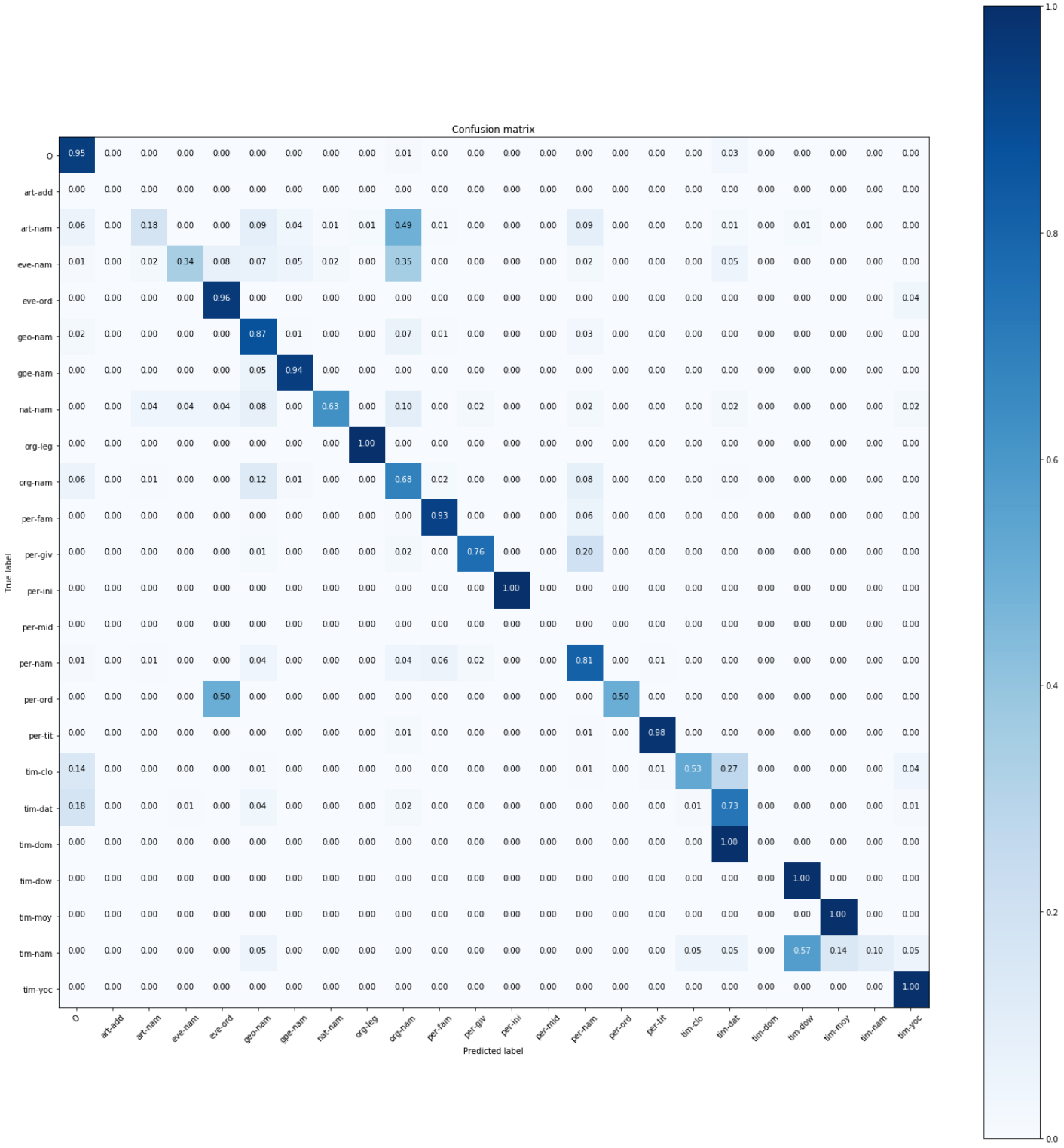

Figure 6. Confusion matrix for Naïve Bayes model on Named Entity Recognition task using 80 % training set and all hand-crafted features.

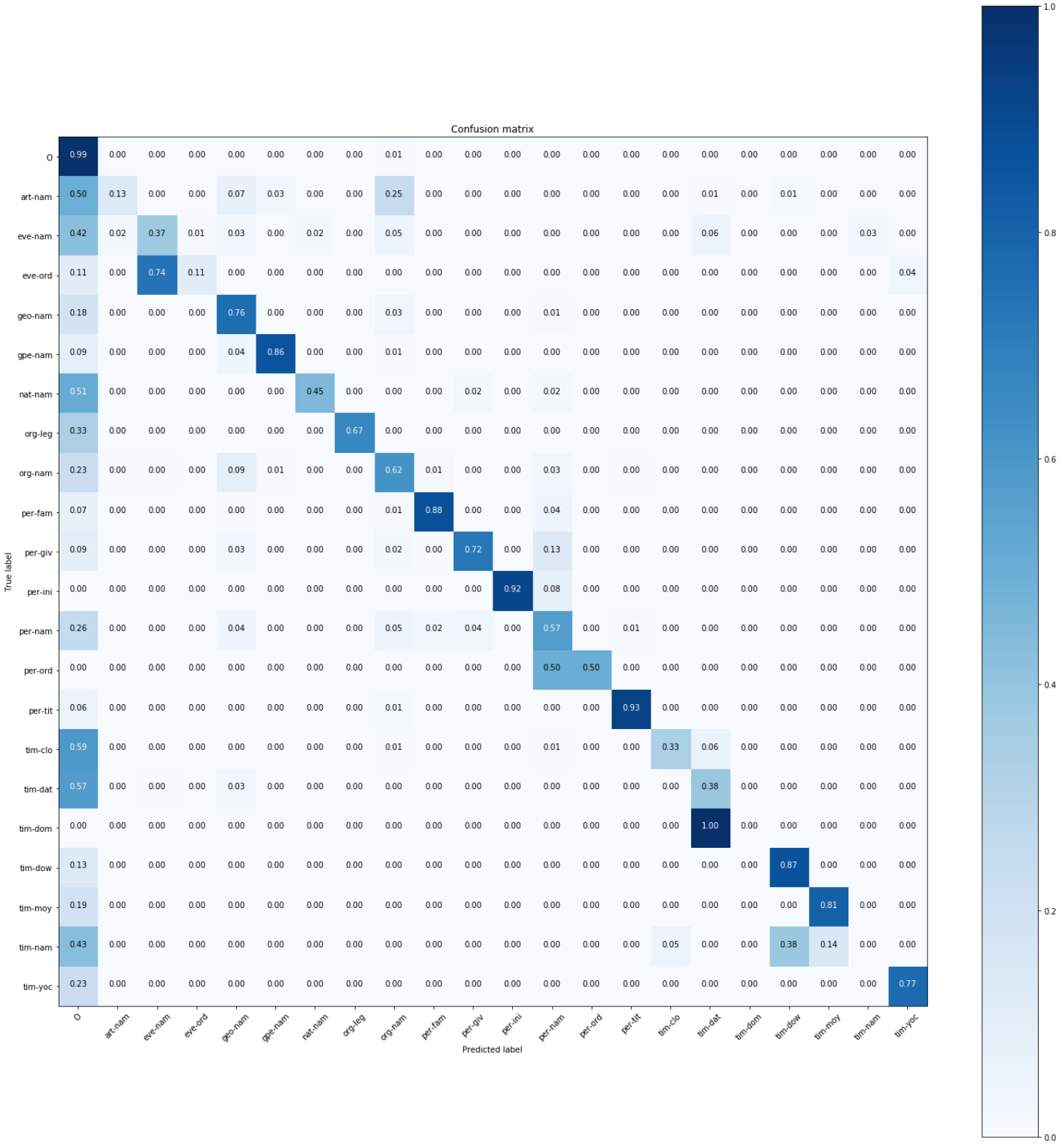

Figure 7. Confusion matrix for Hidden Markov Model on Named Entity Recognition task using 80 % training set and all hand-crafted features.

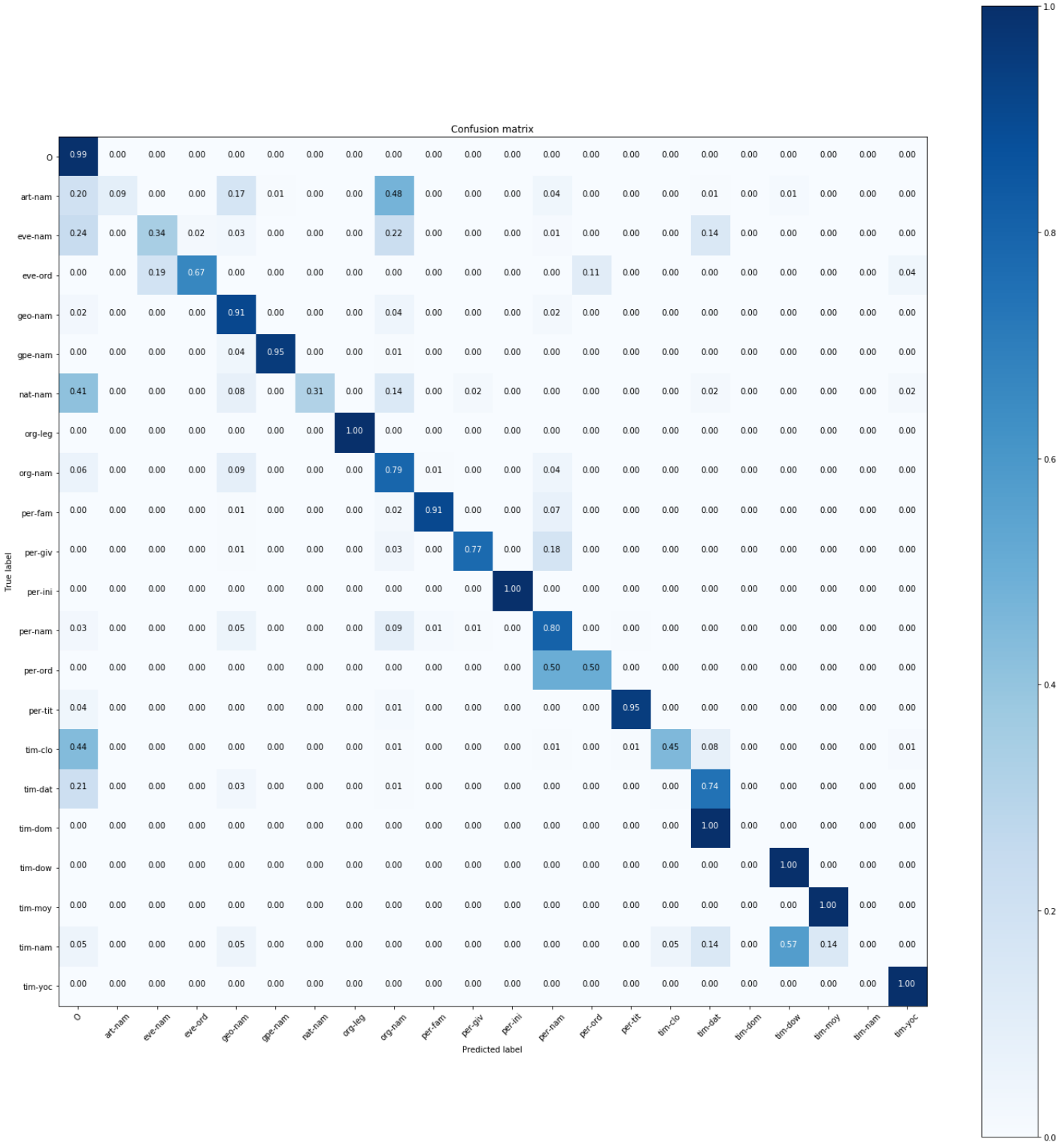

Figure 8. Confusion matrix for Conditional Random Field Model on Named Entity Recognition task using 80 % training set and all hand-crafted features.

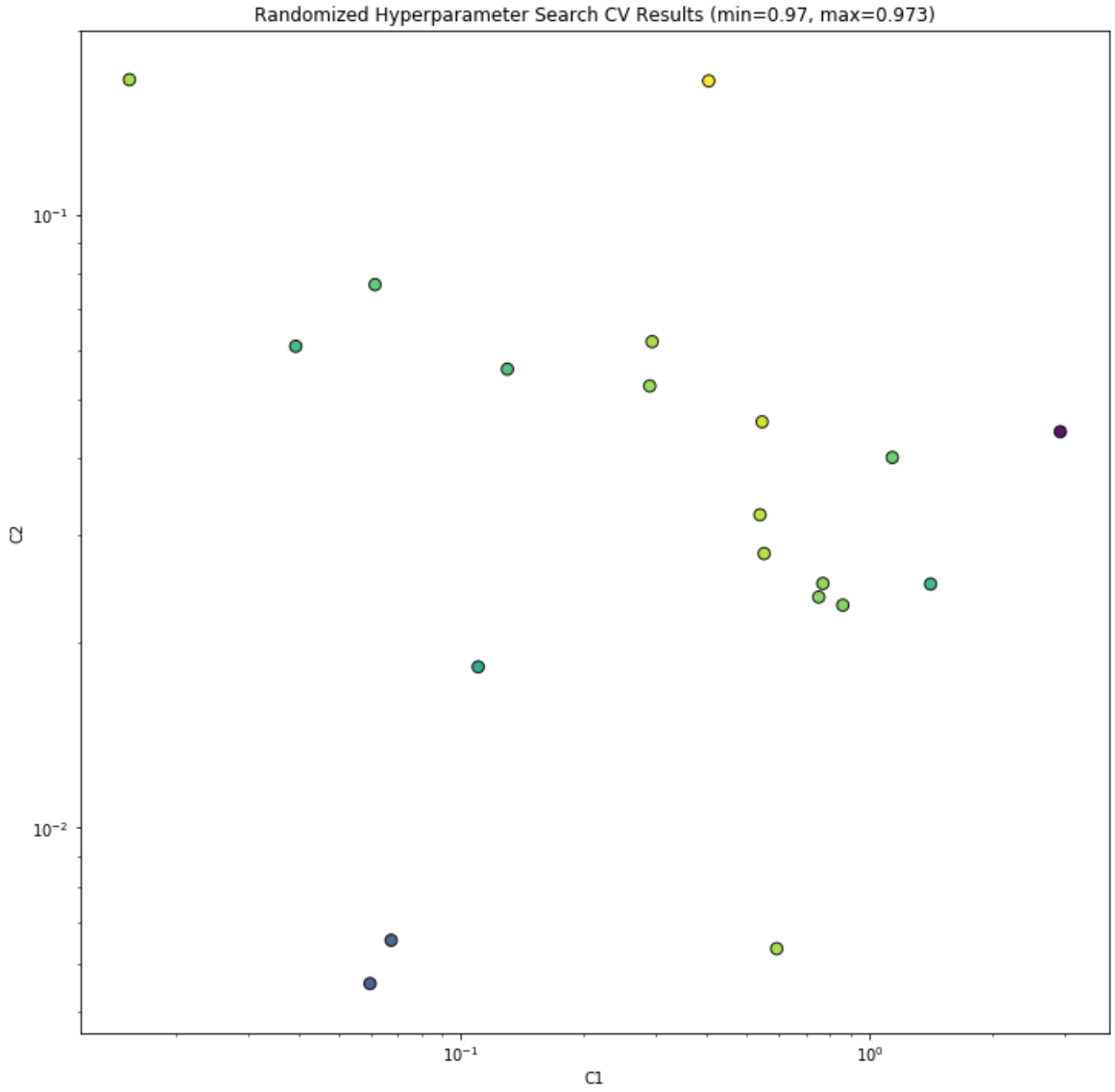

Figure 9. Parameter search results for L_1 (C1) and L_2 (C2) regularization parameters - the influence of the parameter choice is minor, but using L_1 regularization did improve the test error.