Named Entity Recognition on Dutch Parliamentary Items using Frog

Ragger Jonkers
10542604

Bachelor thesis
Credits: 18 EC

Bachelor Opleiding Kunstmatige Intelligentie

University of Amsterdam
Faculty of Science
Science Park 904
1098 XH Amsterdam

Supervisor
dhr. dr. M.J. Marx

Informatics Institute
Faculty of Science
University of Amsterdam
Science Park 904
1098 XH Amsterdam

June 24th, 2016

Contents

Abstract

A dutch natural language processing system, Frog, is used to recognize entities in Dutch parliamentary items, as well is assigning a type to the retrieved entities. Its performance is evaluated by comparing results on the CoNLL-2002 test sets to a parliamentary items test set. To do so a newly annotated corpus has been constructed. The most difficult element seems to type prediction of an entity. Therefore, a type reclassification step is provided, using majority voting and a parliamentary gazetteer that contains entity-type relations. These methods have shown an improvement in type-precision, which, together with high recall, makes Frog perfectly suitable for the task.

1 Introduction

Named Entity Recognition(NER) is a technique used in Information Retrieval which can be described as automatically finding Named Entities(NE's) such as persons, locations and organizations in text documents, as well as disambiguating between said types. Different methods have been tried over the last two decades, most of them language dependent, due to the fact that language features are used as indicators, yet grammar and vocabulary differ greatly across languages. While near-human performing NER-systems have already been developed for English, this does not appear to be the case for Dutch. The reason for this being that most natural language applications are confined to the language in the field of study, for which English is predominant.

Although progress has been made, especially for English, several encountered problems are not paired with unanimously accepted solutions. One might for example reason that a large gazetteer, which is actively updated, could be used in the retrieval of entities. While good results have been found using this approach, it, solely, cannot be carried out into large scale applications, because the completeness of the gazetteer and the application run time are inversely proportional to each other. While small gazetteers will affect recall negatively, comprehensive gazetteers require more time to search through, even with hashing techniques. Secondly, ambiguity introduces itself as the principal difficulty for type assignment: An entity can be of different role, albeit completely identical in appearance. Humans are able to quite easily disambiguate an entity, even though elucidation of choice can be difficult. Therefore, computer systems are set up with an arduous task that automates this decision process.

NER is generally applied to a specific domain for which the task is optimized, such as finding chemical NE's (Rocktäschel, Weidlich, & Leser, 2012). In this paper NER is applied in the domain of Dutch parliamentary items, which are for example resolutions proposed or letters written by De Eerste Kamer (the Senate) and De Tweede Kamer (the House of Representatives) to the government. A suggested goal system that utilizes the acquired NE's, notifies parties of interest of any new, relevant parliamentary items. The answer to the question of how to effectively acquire these NE's with corresponding type for the suggested system, will be sought for. This is an improvement to a baseline string search, because unknown strings are not retrieved. Nor do the retrieved strings contain meaning, as they hold no type information.

A relatively new parser, Frog[1], that has been maintained, can apply a multitude of natural language processing techniques on a text. Frog will be examined thoroughly in section 2. Hitherto no scientific evaluations of the latest iteration of the system have been published. For this reason an evaluation of the NER-task of Frog for the domain of parliamentary items has been performed and results are compared to those achieved on the CoNLL-2002 dataset, which was used in the CoNLL-2002 shared task[2]. Subsequently, a method will be shown for the improvement of entity *type* assignment using majority voting and a parliamentary gazetteer.

I will try to answer the main question by answering the corresponding subquestions:

1. How can the Named Entities in parliamentary items be acquired for the suggested system?
    (a) How does Frog perform in the NER-task?
        i. How does Frog perform on the retrieval of entities?
        ii. How does Frog perform on type assignment of retrieved entities?
    (b) What domain specific improvements can be made?

[1]https://languagemachines.github.io/frog/
[2]http://www.cnts.ua.ac.be/conll2002/ner/

1.1 Overview of thesis

At first research that has been done on this topic, is looked into, which will provide a more in-depth look at Frog and its NER-module. Secondly, the approach of evaluating Frog on parliamentary items is described, as well as the procedure of the error analysis and how improvement of the entity type assignment was attempted. This approach is followed by the results, which will be used to form a conclusion about the suitability of Frog for the main task in addition to possible enhancements. A discussion of approach will conclude this paper.

2 Related work

Named-entity recognition(NER) comprises two tasks. Retrieval of named-entities is regarded as the primary task, and type classification of retrieved entities as the secondary task (Buitinck & Marx, 2012). In the paper of Buitinck and Marx an averaged perceptron is used for both stages, each consisting of its own feature parameters. The benefit of two-staged NER, in which a different algorithm per stage is used, lies in the opportunity of optimizing each algorithm individually.

Recently, the most prevalent implementation of NER applies a supervised machine learning approach. An annotated corpus is required for apprehension of features (for instance syntax and context) that indicate the occurrence of a named-entity, as well as the type. This is state of the art as opposed to hand-written extraction rules. The downside of learning off an annotated corpus is the domain limitation that the corpus entails. Aditionally, a large training corpus is required for supervised learning, a corpus with domain resemblance to the goal domain of the NER-application. Currently, the only Dutch corpus that was made publicly available, is the CoNLL-2002 data set of news items that appeared in the Belgian newspaper *De Morgen*. The data set contains 301,418 tokens with annotated Part-of-Speech(PoS) and IOB-tag[3] hyphenated with entity type. IOB, short for inside-outside-beginning, is used to show if a token is the beginning of an entity consisting of multiple tokens; is second or one thereafter in the tag sequence; or outside, meaning not an entity itself or part of one.

A variety of classifier types can be chosen from, such as a memory-based learning classifier, a support-vector machine or a conditional random field. No significant results are achieved by a combination of different classifiers, subsequently classifying based on weighted votes, as is shown by an in-depth evaluation of an ensemble of classifiers for Dutch (Desmet & Hoste, 2014). Higher precision and recall are found by the optimization of features in a single classifier.

Likewise an *unsupervised* machine learning approach has been suggested, for which no manual construction of gazetteers or annotated corpora is required (Kazama & Torisawa, 2008). While gazetteers are considered the most precise for NER, maintaining large dictionaries can be expensive. Clustering of verb-multi-noun dependencies allows for the automatic formation of gazetteers, avoiding said issue.

Frog is an expansion to TADPOLE (Bosch, Busser, Canisius, & Daelemans, 2007). The system is modular, composed of tokenization, lemmatization, chunking and morphological segmentation of word tokens. In addition dependency relations and NE's are detected in Dutch texts using the memory-based learning software TiMBL (Daelemans, Zavrel, van der Sloot, & Van den Bosch, 2004). Frog is designed for high accuracy, fast processing and low memory usage, qualifying itself to process numerous parliamentary items. Frog is trained on the SoNar corpus consisting of one million annotated and manually verified NER labels. The NER module has not been trained on

[3]`https://en.wikipedia.org/wiki/Inside_Outside_Beginning`

Type	Count
Parliamentary item	131906
News item	95261
Chamber inquiry	38804
Voting	20930
Chamber letter	19828
Agenda item	18950
Resolutions	4792

Figure 1: The distribution of the initial data

PoS, but on the contextual relation between words and IOB-tags directly. The assignment decision for each token is made with a look-up in memory, in which the instance base is stored. Unknown words are classified by a comparison to this instance base. The class of the instance that matches the contextual features of the unknown instance the closest, is chosen.

3 Methodology

3.1 Description of the data

The dataset is a set of lobby documents that has been scraped from the web. Lobbying can be defined as the act of attempting to influence decisions made by officials in a government, most often legislators or members of regulatory agencies. Lobby documents are for example: resolutions, chamber inquiries, letters to the government and more. A distribution of different types can be seen in figure 3.1. The result of scraping these documents is that certain figures or itemization structures are textualized into words concatenated with itemization numbers, table entry titles or something similar. However, Frog's tokenization module will most of the time find the correct tokens. These documents are exported as consistent JSON dicts, with for each doc corresponding meta information about the document such as the document ID, source, and type. The format can easily be read with Python. For domain reduction purposes, only parliamentary items are included in the data set. These items form the majority of the items and are in itself a mix of all types, except for the news items, in which the named entities vary greatly in domain compared to the parliamentary items.

3.2 Methods

The following sections will describe the stages required to examine the suitability of Frog for doing NER on parliamentary items. I will first describe how Frog has been used. Then I will go over the preparation needed to evaluate Frog on the CoNLL-2002 data set and the set of Parliamentary Items. Lastly I will explain how the reclassification of entity types was performed.

3.2.1 Employment of Frog

Frog is open-source software that can be modified or redistributed under the GNU General Public License[4]. The results in this paper are acquired by running Frog through a virtual machine on

[4]`http://www.gnu.org/copyleft/gpl.html`

Windows. All necessary dependencies of Frog, including Frog itself, are provided by the LaMachine software distribution[5]. Frog can be run through Python using the python-frog binding which is also inlcuded in the virtual machine.

3.2.2 Evaluation method

The CoNLL-2002 data set uses the following format.

$$TOKEN - whiteline - POS - whiteline - IOB$$

Every line contains three things, each seperated by space: the token, which is an unprocessed word from a sentence, the token its corresponding part-of-speech, and the token its IOB-tag. Frog its Folia output looks similar with tab-delimited column format , but with additional information per token. To release Frog on CoNLL, the test set first has to be reformatted to its original text. This is done by removing the PoS and IOB-tags, concatenating the remaining words into their original sentences thereafter. Frog chunks together multi-word entities and other words that are closely related on one line in the output. This will make it differ from the CoNLL output, as such the output is splitted with python using regular expressions. Now the processed text has the same line mapping per token as CoNLL.

It needs to be taken into account that the annotation guidelines for the SoNar corpus, on which Frog is trained, differ from CoNLL. SoNar has a wider range of entity types, with addition of EVE for event and PRO for product. These types are annotated in CoNLL as MISC. Therefore the additional types that Frog outputs are mapped to MISC. There is also a guideline difference regarding locational adjectives, such as 'Dutch' or 'European'. CoNLL guidelines[6] annotate such an adjective as MISC, while Frog is trained to assign LOC as type. These LOC outputs of Frog have to be reassigned to MISC as well.

3.2.3 Reclassification of entity types

Firstly, all found entities are re-evaluated by **search in a gazetteer** in the domain of parliamentary items. This gazetteer is constructed with a combination of automatic extraction of entities with Frog and manual annotation of the correct type. The automatic extraction is performed on the training set of thousand parliamentary items. The extracted entities are sorted on popularity, giving annotation priority to the most common entities. The effectiveness of this step is dependent on the 1) size of the gazetteer 2) relevance of the training set 3) recall performance of Frog. Entities that do not appear in the gazetteer cannot be reclassified.

In the case of unknown words the contextual word relations may indicate a type correctly, however, with multiple occurrences of the same entity throughout a text, context differs from time to time. As a result an unambiguous entity can be tagged correctly as a PER 90% of the time, but have an incorrect type assigned for the remaining 10%. This is expected to be resolved using **majority voting**. In the case when a certain type is dominant over a minority, this minority is reclassified as the dominant type. To retrieve information regarding the dominant types per entity, a train set of thousand parliamentary items has been processed by Frog. The total occurrences of the entity types have been counted over all documents. For each token in the test set that has been assigned to be an entity, we perform majority voting. Whether an entity type is dominant for that

[5]`https://proycon.github.io/LaMachine/`
[6]`http://www.cnts.ua.ac.be/conll2003/ner/annotation.txt`

	CoNLL	**Parliamentary items**	**Parliamentary items + reclassification**
Accuracy	0.950	0.973	0.974
Recall	0.845	0.906	0.908
IOB-precision	0.915	0.890	0.890
Type-precision	0.672	0.616	0.662
Precision	0.629	0.563	0.603
F-measure	0.720	0.694	0.725

Figure 2: Performance overview of both sets

entity is decided based on a threshold value. When the fraction of an entity type over the total amount of entity type counts is larger than the threshold value, the type is considered dominant.

A combination of these reclassification steps can be seen in algorithm 1, which shows the procedure in pseudocode.

4 Evaluation results

The performance evaluation for the different test sets can be seen in figure 2. The CoNLL performance is an average of both the ned.testa -and ned.testb set. The parliamentary item set is of the same size as ned.testa. The last column in the figure shows performance after reclassification of entity types in the parliamentary items set. Accuracy is obtained by binarily evaluating all tokens: if a token is predicted as an entity (regardless of type), but it turns out not to be, this token prediction is marked as incorrect. Tokens not seen as an entity while they are, idem. Low accuracy would be the results of Frog incorrectly identifying non-entities as entities, since the other way around would not hit accuracy hard due to the sparcity of entities in the test sets. Recall is measured as the proportion of entities retrieved that were annotated in the test set, again not looking at entity type. IOB-precision is the precision of the first part of the IOB+type tag. This will measure how well a beginning of an entity consisting of multiple tokens is found, or how different entities following each other are seperated. Type precision also looks at the IOB+type tag, but with exclusion of the IOB part. Regular precision is the combination of type-precision en IOB-precision. F-measure is the harmonic mean of the regular precision and the recall.

Since type-precision has appeared to be relatively low, it would be interesting to analyze errors made per type. Figure 3 shows a prediction distribution for each type. It can be seen that location as a type is predicted very accurately, while prediction of the type person seems to be significantly more difficult.

5 Conclusions

The high recall in the evaluation results indicates that Frog is very capable in retrieving all relevant entities in a text. Type assignment, however, has appeared to be substantually more difficult. As a solution, reclassification of entity types has been proposed, and has proven to increase overall type-precision. The semi-automatic construction of a parliamentary gazetteer as auxilliary, makes Frog well suitable for the NER-task required for the suggested system. Since a semi-automatic gazetteer can be constructed for any given domain, in addition to the fact that majority voting is

Algorithm 1: Reclassification of entity types in Frog output using majority voting and a type gazetteer

```
reclassify (frogged_file, vote_file, gazetteer_file);
Input  : Three files (path) each containing: the output of the Frogged text in folia-XML
         format, a dict of the counts for each type per entity, a dict of entities and their
         correct type
Output: CoNLL format with reclassified types
treshold ← 0.7
type_counts ← load(vote_file)
gazetteer ← load(gazetteer_file)
foreach line in frogged_file do
    token, lemma, pos, chunk, iob+type etc. ← split(line)
    iob ← split(iob+type-tag)
    if is_entity(token) then
        new_type ← lookup(token, gazetteer)
        // if token is in gazetteer, no majority vote needed
        if new_type then
            line ← token, pos, iob+new_type
        else
            new_type ← vote_type(token, type_counts, treshold)
            if new_type then
                line ← token, pos, iob+new_type
            end
        end
    end
end
vote_type (token, type_counts, treshold);
Input  : token of which to decide the type, the type_counts dictionary, and the treshold to
         enact on revoting
Output: new type for the token, none returned otherwise
new_type ← max(type_counts[token])
confidence_score ← max(type_counts[token]) / total(type_counts[token])
if confidence_score > treshold then
    return new_type
end
```

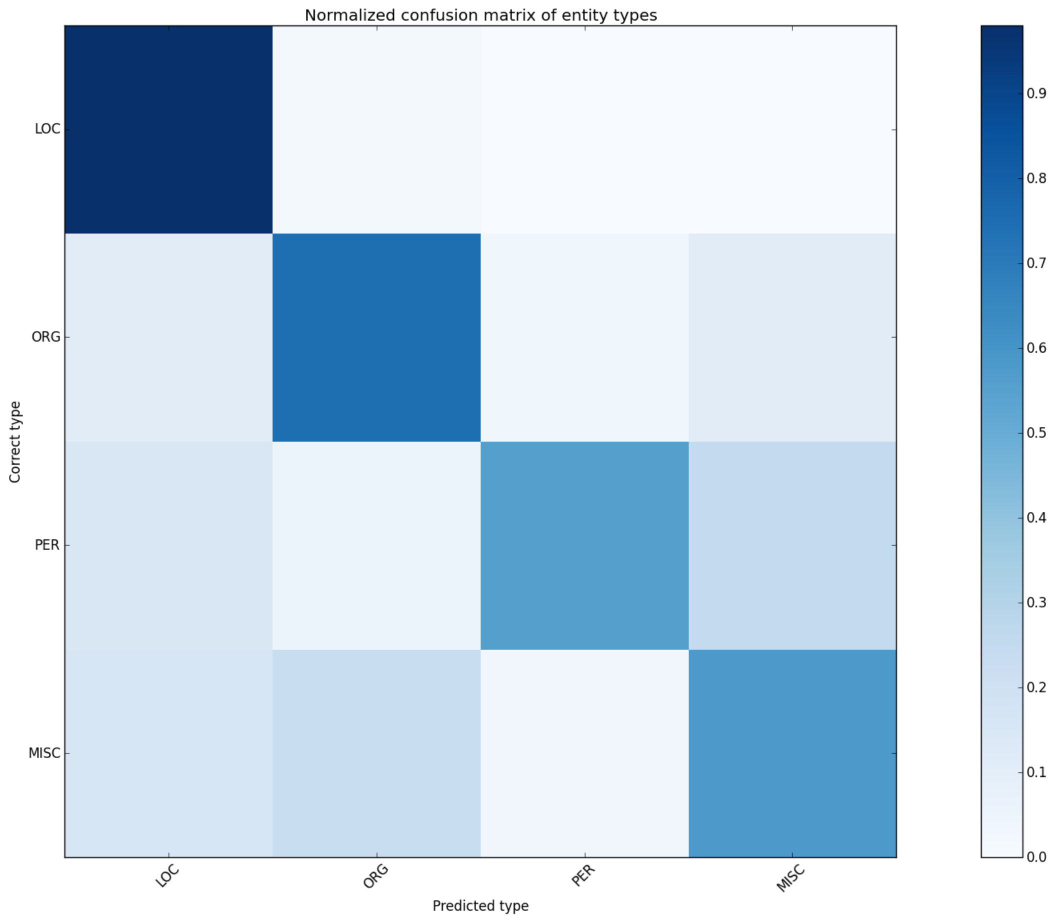

Figure 3: Prediction distribution per type

domain-independent, Frog can be exported to similar Dutch NER-tasks.

6 Discussion of approach

7 Acknowledgements

References

Bosch, A. v. d., Busser, B., Canisius, S., & Daelemans, W. (2007). An efficient memory-based morphosyntactic tagger and parser for dutch. *LOT Occasional Series*, *7*, 191–206.

Buitinck, L., & Marx, M. (2012). Two-stage named-entity recognition using averaged perceptrons. In *Natural language processing and information systems* (pp. 171–176). Springer.

Daelemans, W., Zavrel, J., van der Sloot, K., & Van den Bosch, A. (2004). Timbl: Tilburg memory-based learner. *Tilburg University*.

Desmet, B., & Hoste, V. (2014). Fine-grained dutch named entity recognition. *Language resources and evaluation*, *48*(2), 307–343.

Kazama, J., & Torisawa, K. (2008). Inducing gazetteers for named entity recognition by large-scale clustering of dependency relations. *Proceedings of ACL-08: HLT*, 407–415.

Rocktäschel, T., Weidlich, M., & Leser, U. (2012). Chemspot: a hybrid system for chemical named entity recognition. *Bioinformatics*, *28*(12), 1633–1640.