Benchmarking Named Entity Recognition Tools for Portuguese

André Pires, José Devezas, and Sérgio Nunes

INESC TEC and
Faculdade de Engenharia, Universidade do Porto,
R. Dr. Roberto Frias, s/n, 4200-465 Porto, Portugal
`andre.r.o.pires@gmail.com`,
`{jld,sergio.nunes}@fe.up.pt`

Abstract. There has been much work in the Information Extraction field, particularly in the Named Entity Recognition task. However, for the Portuguese language, the implementations still perform below the results for other languages, as shown by the HAREM conferences. The goal of this work is to assess the current performance of well established tools, namely Stanford CoreNLP, OpenNLP, spaCy and NLTK, against a Portuguese dataset, specifically the HAREM gold standard collection. These tools were used with an out-of-the-box approach, meaning without any tuning. The results show that the referred tools can match the results of Portuguese state-of-the-art tools, presented in the HAREM conferences, specifically, Stanford CoreNLP with an F-measure of 56.10%, OpenNLP with 53.63%, spaCy with 46.81% and NLTK with 30.97%. Furthermore, a hyperparameter study was performed, improving over the default configurations, for about 2% in each tool, and almost 35% in NLTK's `MaxEnt` classifier.

Keywords: natural language processing, named entity recognition, text mining

1 Introduction

Named Entity Recognition (NER) is a sub-field of Information Extraction. Its main purpose is to identify and classify entities from unstructured text. This extraction is the first part of a typical information extraction pipeline, enabling further information extraction methods, such as semantic analysis or relation extraction. There is a need for having systematic evaluation, so that all NER systems have the same standards when evaluating their performance.

There are multiple techniques proposed to rank NER systems based on their ability to annotate text correctly. These techniques were defined and used in conferences such as Message Understanding Conference (MUC) [1], Conference on Natural Language Learning (CoNLL) [2], Automatic Content Extraction (ACE) [3] and HAREM Avaliação de sistemas de Reconhecimento de Entidades Mencionadas (HAREM) [4]. These conferences not only differ in their evaluation techniques, for example giving score to partial matches or only scoring exact matches, but also on what is considered an entity, so they use different entity class sets. This makes it hard to compare different tools which participated in different conferences.

In this paper, we will present a structured approach for the consistent assessment of the mentioned tools, in order to provide an out-of-the-box comparison of their performance. And afterwards, a hyperparameter study in order to improve the baseline performance.

2 Related work

Initial work in this field was based on pattern matching techniques, for instance Rau [5] extracted company names with rules like the detection of company suffixes, such as "Inc." or "Corp.". Afterwards, dictionary-based approaches began to appear, where the main extraction algorithm was based on matching the words in a text with a gazetteer. An example of this approach was done in Portuguese by Sarmento [6], using the gazetteer REPENTINO [7]. Although this kind of extraction techniques provided good precision, results in terms of recall and F-measure where low and, furthermore, these techniques are difficult to manage because rules and dictionaries need to be constantly updated. This lead to the development of machine learning algorithms, where the most used methods are probabilistic techniques, such as Hidden Markov Models, for example by Bikel et al. [8], Maximum Entropy Models, like the framework by Carvalho [9], and Conditional Random Fields, with a Portuguese example by Amaral and Vieira [10]. Throughout the years there was a shift in focus, where machine learning techniques started to gain more interest. This is due to the scalability of these techniques and the amount of work required, however data in specific domains remains scarce. Machine learning supervised approaches continue to be the most used techniques, but recently many semi-supervised approaches, involving bootstrapping [11], started to appear, as they require less training examples.

The HAREM [4] conference's main objective was to evaluate the current state of the art for NER in the Portuguese language. Apart from the HAREM conference, there have been some experiments in evaluating NER tools with the HAREM dataset, for instance by Amaral et al. [12] who compared NERP-CRF with other publicly available tools for the Portuguese language. Other example is by Rocha et al. [13], who compared PAMPO with other tools. Both of these comparisons used a subset of HAREM's categories. Another comparison using some of the tools of this experiment, for English, was done by Jiang [14], showing that Stanford CoreNLP had the best result, in concordance with our experiments.

2.1 Tool descriptions

The main purpose of this experiment was to compare some of the main tools capable of performing NER in multiple languages, with particular focus on the Portuguese language. The main criteria for choosing the tools were:

– The tool has to be completely free.
– The tool has to be language and domain independent.
– The tool has to allow training a custom model for NER, with custom entity classes.

With these criteria in mind, we chose four different, well-established tools: Stanford CoreNLP [15], OpenNLP [16], spaCy [17] and NLTK [18]. These tools are described in Table 1, which shows that all are freely available, use either Java or Python as the main language and, apart from already having multiple default language models, all allow training with other languages.

Table 1. Overview of the assessed tools.

Tool	**License**	**Version**	**Language**	**NER classifiers**	**Default language models**
Stanford CoreNLP	GNU General Public License	3.7.0 (2016)	Java	Conditional Random Fields	Arabic, Chinese, English, French, German, Spanish
OpenNLP	Apache License	1.7.2 (2017)	Java	Maximum Entropy	Dutch, English, Spanish
spaCy	MIT License	1.7.2 (2017)	Python	Thinc linear model	English, German
NLTK	Apache License	3.2.2 (2016)	Python	Naive Bayes, Decision Tree, Maximum Entropy	English

Neither of these tools have a Portuguese model for NER, so all of them require training with a Portuguese corpus, such as HAREM. Apart from spaCy, all tools use well known algorithms to perform NER, for instance CRF and Maximum Entropy. spaCy uses its own implementation of a structured average perceptron, called Thinc [19] linear model.

3 The HAREM gold standard collection

HAREM is an evaluation contest for NER in Portuguese. There were two main HAREM events, in 2005 [4,20] and 2008 [21]. Both provided gold standard collections and, for this experiment, we used only the second one.

3.1 Corpus description

HAREM's gold standard collection [22] is a collection of Portuguese texts from several genres, such as web pages and newspapers, in which named entities have been identified, semantically classified and morphologically tagged in context. This collection is composed mainly of news documents (about 35%) and didactic documents (about 23%). It defines three levels of entity annotations, namely categories, types and subtypes.

There were 10 categories identified, specifically Works of art (*Obra*), Event (*Acontecimento*), Organization (*Organizacao*), Misc (*Outro*), Person (*Pessoa*), Abstraction (*Abstracao*), Time (*Tempo*), Value (*Valor*), Local (*Local*) and Thing (*Coisa*). Apart from the categories, it has a total of 43 types and 21 subtypes.[1]

Table 2 shows the number of named entities in each category in the second HAREM gold standard collection. *Pessoa* is the most common category with 2,035 words, and *Outro* is the least common, with only 148 words. This gold standard collection has a total of 129 documents, divided into two Portuguese variants, for Portugal and Brazil, as shown in Table 3, being Portuguese from Portugal the main variant.

Table 2. Number of named entities in each category. Values from do Amaral et al. [12].

Categories	**Number of entities**	**%**
Pessoa	2,035	28%
Local	1,250	17%
Tempo	1,189	16%
Organizacao	960	13%
Obra	437	6%
Valor	352	5%
Coisa	304	4%
Acontecimento	302	4%
Abstracao	278	4%
Outro	148	2%
Total	7,255	100%

Table 3. Document distribution. Table from Mota et al. [23].

Portuguese variant	**Number of documents**	**%**
pt_PT	93	72.09%
pt_BR	36	27.91%
Total	129	100%

3.2 Corpus transformation

The gold standard collection is available in XML format, and contains information that was not used in this experiment. The collection cannot be directly used as input to the selected tools, so it had to be transformed for each tool.

Initial transformation This first transformation was applied for all tools.

First of all, the tag `OMITIDO` was ignored in the HAREM evaluation, so it was stripped from the gold standard collection, keeping the text without any annotation. Then, since HAREM provides alternatives to the annotation of entity tags, regarding the entity boundary, using the `ALT` tag — in other words, allows multiple annotations to the same span of text with different boundaries —, we stripped these tags, keeping only the alternative which had the highest amount of entity tags, and choosing the first alternative when there were no tags inside the `ALT` tag. This was required because the selected tools cannot handle

[1] See full hierarchy table at http://www.linguateca.pt/aval_conjunta/HAREM/-tabela.html - Accessed on: 2017-06-01.

alternative annotations. Apart from the alternatives using the `ALT` tag, there were alternatives in the annotation, regarding the category, type and subtype of an entity. In this case, we selected the first alternative for each case. Also, there were tags which had no categories, and were identified only as an entity. In this case, the tag was stripped.

Since the tools only allowed one level of entities, we flattened the levels, producing three different outputs: one with only the categories, one with only the types and another with only the subtypes. For scripting purposes, the types and subtypes were concatenated to keep the semantics and context inside the parent category (e.g. "LOCAL_HUMANO_PAIS").

Finally, apart from the category attribute, all other attributes were removed. For evaluation purposes, this dataset was divided into folds for repeated cross validation (see Section 5).

4 Main steps to run each tool

Each tool has a particular set of requirements in order to train a NER model and then perform NER. The steps for each tool are presented next, namely the required input format, the steps for converting HAREM into that format, how to train the NER model, how to perform NER, and finally how to convert it to the CoNLL evaluation format.

4.1 Stanford CoreNLP

CoreNLP requires a tokenized file, where each line contains a token and its entity class separated by a tab character. This entity class is the class of the entity for entity tokens, and an "O" class for other tokens. We used the Stanford CoreNLP tokenizer `edu.stanford.nlp.process.PTBTokenizer` to tokenize the text. The entity classes in XML were also tokenized in the process, so afterwards we "de-tokenized" them and we looped through the file adding the entity classes from the first match in an entity tag to a match in the closing tag. Every token outside this, was tagged "O". The classifier was trained using the command:

```
java -cp stanford-corenlp.jar edu.stanford.nlp.ie.crf.\
    CRFClassifier -prop <file.prop>
```

where `file.prop` sets the hyperparameters and features to use and the path for the training file and output model.

To classify text documents, we used the following command:

```
java -cp stanford-corenlp.jar edu.stanford.nlp.ie.crf.\
    CRFClassifier -loadClassifier <ner-model.ser.gz> --\
    testFile <file_test.txt>
```

where `file_test.txt` represents the input unannotated text to be classified. Then, after performing the Named Entity Recognition (NER), we added IOB tags to the output for it to be comparable in the evaluation stage.

4.2 OpenNLP

This tool requires a sentence per line as input, where entities are annotated with a starting tag (`<START:tag-name>`) and an end tag (`<END>`). First, we converted the HAREM XML entity tags to the OpenNLP format. Then, using NLTK's sentence segmentation module, the dataset was segmented by sentences. Since this segmentation was not perfect, we had to join faulty segmentations by checking if every open entity tag had a corresponding closing tag and vice versa, in each sentence. When they had no corresponding tag, we joined the current sentence with the previous sentence. Furthermore, we had to make sure that there was a space character before and after each tag, or else OpenNLP's interpreter would not work. To train the model, we had to run the command:

```
opennlp TokenNameFinderTrainer -model <model.bin> -lang <pt>\
    -data <training_data.txt> -encoding <UTF-8>
```

and to classify the text the command was:

```
opennlp TokenNameFinder <model.bin> < <corpus_test.txt> > \ <
   output file>
```

Finally, after performing NER, we converted the output from the OpenNLP format to the CoNLL format, adding IOB tags.

4.3 spaCy

spaCy requires that the input dataset is in the standoff format, that is to say, there have to be two files: one with the plain text, and another with the entity annotations, containing a tab separated entry with the beginning and ending positions of the entity along with its class. The text had to be separated in sentences. Since we had already converted HAREM to the OpenNLP format, we used it to transform it to standoff. The transformation was done using the following steps:

1. Search until `<START:tag>` tag
2. Save starting position
3. Delete matched tag
4. Search for `<END>` tag
5. Save end position
6. Delete matched tag
7. Save `standoff = (beginPos, endPos, tag)`
8. Repeat from step 1 until no matches occur
9. Output to a tab separated file

The resulting files were then used to train a NER model in spaCy. The training script was based on an example script in spaCy's repository, with the additional preprocessing task of converting to the standoff format. The main NER classifier was changed from `EntityRecognizer` to `BeamEntityRecognizer`. After the NER process, the result was converted to the CoNLL format with IOB tags for evaluation purposes.

4.4 NLTK

This toolkit not only requires that the input dataset is in the CoNLL format with IOB tags, but also that it has the associated POS tag. In other words, the input file must be a tab separated file, where each line has the token, the POS tag and the entity tag in IOB format. This aspect required three major steps, namely tokenizing and performing POS tagging, tokenizing while keeping the entity tags, and joining both files. The steps for this were the following:

1. Tokenize and POS tag
   - Remove all tags from the HAREM dataset, in order to tokenize the text
   - Tokenize the dataset using `nltk.tokenize.word_tokenizer`
   - Using the resulting tokenized text, perform POS tagging
     - Train POS model using *floresta* corpus from `nltk.corpus`
     - POS tag resulting file from tokenizer step
2. Tokenize while keeping the entities
   - Tokenize the dataset using `nltk.tokenize.word_tokenizer`
   - Join consecutive tokenized entity tags
   - Transform dataset, matching the entity tags using regular expressions, assigning tags to each token
     - For each token, after an entity tag, assign a *B-tag*
     - For each token after the first entity token (previous step), or after an Inside (I) token, assign an *I-tag*
     - Assign an *O* tag for other tokens
3. Join POS tagged file with entity tagged file
   - Iterate through both files simultaneously
   - For each line, set *token POS-tag IOB-entity-tag*

To train the NLTK's classifiers, we used NLTK trainer[2]. This allowed us to run the following command:

```
python train_chunker.py <path-to-training-file> [--fileids\ <
    fileids>] [--reader <reader>] [--classifier <classifier\
    >]
```

We had to choose a different document reader, `nltk.corpus.reader.conll-.ConllChunkCorpusReader`, since the training files were in the CoNLL format. For this reader, we had to specify the entity categories in the NLTK trainer *init* file. Running the command resulted in a serialized model saved in the pickle format. In order to classify text, the model had to be loaded, the dataset to be classified was required to contain POS tag annotations and then classified with `chunker.parse(tagged)`. The parser returned the result in a tree format, which was converted to the CoNLL format using `nltk.chunk.util.tree2conlltags(ner_result)`.

[2] `https://github.com/japerk/nltk-trainer`

5 Evaluation

The method we used to assess the performance of each tool is described next. Each tool has its own evaluation scheme, however they are not directly comparable. To make it comparable we used a common evaluation scheme, the one used in the CoNLL [2] conference, where credit was only given to exact-matches or, in other words, both entity tags and boundaries had to be correct for it to count as a correct match.

Evaluation was done using the *conlleval*[3] script, taken directly from the website for the conference's occurrence in 2000. The script requires a file in the CoNLL format, with both the output of the NER tool and the gold standard. To be more precise, it is a space separated file, where each line contains a token, the gold standard tag and the predicted tag. It accepts files with or without IOB (Inside, Outside, Beginning) tags, being that, for the latter, each identification is treated as a single token entity. For this experiment, we evaluated the results with IOB tags.

Since not all the tools outputted the results with IOB tags, we added them whenever they were missing. After each run, we had to merge the outputs with the gold standard. For this merge to be correct, we had to use each tool's tokenizer for the testing set, or else the merge would not be successful as the tokens would not match correctly to the gold standard set.

For robustness, we used repeated 10-fold cross validation, with 4 repeats, and calculated the average of the precision and recall, and the macro-average for the F-measure, from every run. In other words, in each repeat, we split the dataset into 10 equal sized folds (in terms of documents), with different training and testing sets each. Then we ran each tool for each fold and for each repeat, and also for each level (categories, types and subtypes). Resulting in a total of $4\ repeats \times 10\ folds \times 3\ levels \times 4\ tools = 480\ runs$.

Finally, after running all the folds and repeats for each tool, that is to say training the models, and performing NER on the testing set, we evaluated the outputs using the CoNLL script and then computed the average for each level. This resulted in an average for each tool, each level, and each entity tag. The results are presented in section 6.

6 Results

The results from the repeated 10-fold cross validation are presented next. Table 4 represents the comparison among the tools for the highest entity class level (categories). It shows the average of three metrics (precision, recall and F-measure) for a single level of entity annotation, namely categories. Stanford CoreNLP takes the lead with an F-measure of 56.10%, followed by OpenNLP with 53.63% and then by spaCy (46.81%) and NLTK (30.97%). The results for NLTK are the ones obtained using the Naive Bayes classifier. It was NLTK's

[3] See `http://www.cnts.ua.ac.be/conll2000/chunking/conlleval.txt` - version: 2004-01-26

best result but it was the lowest score among all tools, below the other tools performance.

Table 4. Results for categories only.

Tool	Precision	Recall	F-measure
Stanford CoreNLP	58.84%	53.60%	56.10%
OpenNLP	55.43%	51.94%	53.63%
SpaCy	51.21%	43.10%	46.81%
NLTK	30.58%	31.38%	30.97%

Table 5 shows the comparison between the tools but with the average F-measure for all the levels. The ranking was similar to the categories results. In other words, Stanford CoreNLP remains on top in the category level, followed by OpenNLP, then spaCy and finally NLTK, in all levels. Again, NLTK's results remain far below in comparison to the other tools. Naive Bayes remained the best scoring algorithm amongst NLTK's algorithms. It is important to note that Stanford CoreNLP is highly computationally demanding, so we were not able to run it with the default configuration for the remaining levels, i.e. types and subtypes.

Table 5. F-measure for all levels.

Tool	Categories	Types	Subtypes
Stanford CoreNLP	56.10%	-	-
OpenNLP	53.63%	48.53%	50.74%
SpaCy	46.81%	44.04%	37.86%
NLTK	30.97%	28.82%	21.91%

Since NLTK provides different classifiers to perform NER, we present in Table 6 the average results for each classifier, for the highest level (categories). The `NaiveBayes` and `DecisionTree` classifiers have similar results, although `NaiveBayes` has slightly better results. `Maxent` (Maximum Entropy), however, performed worse than the other classifiers, with almost zero recall and F-measure. One possible explanation for these results is the default number of iterations (ten) that the algorithm goes through, or even the features used for training. OpenNLP's classifier also uses Maximum Entropy and it performed better, but it was configured to use 100 iterations in the default configuration.

6.1 Hyperparameter study

In order to improve the results obtained in the default configuration, we performed a hyperparameter study, where we checked the effects of some individual hyperparameters for each tool. Since the tools require lots of time to train NER models, it was not possible to perform repeated 10-fold cross validation as before,

Table 6. Results for the category level in NLTK, for all classifiers.

Classifier	Precision	Recall	F-measure
NaiveBayes	30.58%	31.38%	30.97%
Maxent	18.19%	0.58%	1.13%
DecisionTree	21.84%	25.72%	23.62%

so we only perform repeated holdout, again with 4 repeats, and a split of 70% training and 30% testing. This lead to different values for default configurations, in comparison with the baseline study. Table 7 presents a summary of the best obtained results, together with the default results.

Table 7. Summary results for all tools, category level.

Tool	Default F-measure	Best Configurations	Best F-measure
OpenNLP	50.90%	cutoff=4	52.38%
		iterations = 170	51.52%
SpaCy	45.70%	iterations=110	46.60%
Stanford CoreNLP	54.14%	tolerance=1e-3	54.31%
NLTK DT	26.14%	entropy_cutoff=0.08	26.36%
		support_cutoff=16	26.18%
NLTK ME	1.11%	min_lldelta=0, iterations=100	35.24%

Apart from NLTK's Naive Bayes classifier, all tools allowed tuning some hyperparameters. As we can see, for all tools we managed to improve the default performance, in particular, for the NLTK's Maximum Entropy classifier, there was an improvement of almost 35% (F-measure). This leads us to conclude that the default configurations for this classifier are not good, at all. The best performing models are published in INESC TEC's CKAN research data repository [24]. These models were trained with the HAREM dataset, producing three different models, one per entity level, for each tool, and can be directly used for Portuguese NER.

7 Conclusions and future work

With this experiment, it became clear that it is possible to take well established tools and use them to perform NER in Portuguese corpora. HAREM proved to be a good resource to be used as training and test sets for the assessed tools. However, since it is not updated since 2010, and given that the Portuguese language suffered alterations, with the orthographic agreement, future work on this area would be to create an updated gold standard collection.

This experiment provided a comparison for the baseline configuration of each tool, and showed the best individual configuration. In order to further improve the performance, one has to experiment with feature engineering for each algorithm implementation. While it is important to note that these results are not directly comparable to the ones in HAREM since different methods of evaluation were used, the results showed that these tools perform similar to the results achieved in the HAREM conferences, with the best F-measure of 56.10% by Stanford CoreNLP, with a difference of only 1% in the highest scoring participant in HAREM. Also, the hyperparameter study proved to be beneficial, as we managed to improve on the baseline results.

Acknowledgements This work is partially supported by FourEyes, a research line within the project "TEC4Growth – Pervasive Intelligence, Enhancers and Proofs of Concept with Industrial Impact/NORTE-01-0145-FEDER-000020", financed by the North Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, and through the European Regional Development Fund (ERDF). José Devezas is supported by research grant PD/BD/128160/2016, provided by the Portuguese funding agency, Fundação para a Ciência e a Tecnologia (FCT).

References

1. Grishman, R., Sundheim, B.: Message Understanding Conference-6. In: Proceedings of the 16th conference on Computational linguistics. Volume 1., Morristown, NJ, USA, Association for Computational Linguistics (1996) 466
2. Tjong Kim Sang, E.F., De Meulder, F.: Introduction to the CoNLL-2003 shared task. In: Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003. Volume 4., Morristown, NJ, USA, Association for Computational Linguistics (2003) 142–147
3. Doddington, G., Mitchell, A., Przybocki, M., Ramshaw, L., Strassel, S., Weischedel, R.: The Automatic Content Extraction (ACE) Program Tasks, Data, and Evaluation. In: 4th International Conference on Language Resources and Evaluation, Lisbon, Portugal (2004) 24–30
4. Cardoso, N.F.P.F.: Avaliação de Sistemas de Reconhecimento de Entidades Mencionadas. Master's thesis, University of Porto (2006)
5. Rau, L.F.: Extracting company names from text. In: [1991] Proceedings. The Seventh IEEE Conference on Artificial Intelligence Application. Volume i., IEEE Comput. Soc. Press (1991) 29–32
6. Sarmento, L.: SIEMÊS - A named-entity recognizer for Portuguese relying on similarity rules. In Vieira, R., Quaresma, P., Nunes, M.d.G.V., Mamede, N.J., Oliveira, C., Dias, M.C., eds.: Computational Processing of the Portuguese Language: 7th International Workshop, PROPOR 2006, Itatiaia, Brazil, May 13-17, 2006. Proceedings. Volume 3960 LNAI. Springer Berlin Heidelberg (2006) 90–99
7. Sarmento, L., Pinto, A.S., Cabral, L.: REPENTINO A Wide-Scope Gazetteer for Entity Recognition in Portuguese. LNAI **3960** (2006) 31–40
8. Bikel, D.M., Miller, S., Schwartz, R., Weischedel, R.: Nymble: a High-Performance Learning Name-finder. In: 5th International Conference on Applied Natural Language Processing, Washington, DC, Association for Computational Linguistics (1997) 194–201

9. Carvalho, W.S.: Reconhecimento de entidades mencionadas em português. Master's thesis, Universidade de São Paulo, São Paulo (feb 2007)
10. do Amaral, D.O.F., Vieira, R.: O Reconhecimento de Entidades Nomeadas por meio de Conditional Random Fields para a Língua Portuguesa. In: Proceedings of the 9th Brazilian Symposium in Information and Human Language Technology. (2013) 59–68
11. Teixeira, J., Sarmento, L., Oliveira, E.: A Bootstrapping Approach for Training a NER with Conditional Random Fields. In: Progress in Artificial Intelligence. Springer Berlin Heidelberg (2011) 664–678
12. do Amaral, D.O.F., Fonseca, E., Lopes, L., Vieira, R.: Comparative Analysis of Portuguese Named Entities Recognition Tools. In Calzolari, N., Choukri, K., Declerck, T., Loftsson, H., Maegaard, B., Mariani, J., Moreno, A., Odijk, J., Piperidis, S., eds.: Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14). European Language Resources Association (ELRA) (2014) 244–249
13. Rocha, C., Jorge, A., Sionara, R., Brito, P., Pimenta, C., Rezende, S.O.: PAMPO: using pattern matching and pos-tagging for effective named entities recognition in portuguese. CoRR **abs/1612.09535** (2016)
14. Jiang, R., Banchs, R.E., Li, H.: Evaluating and Combining Named Entity Recognition Systems. In: Proceedings of the Sixth Named Entity Workshop, joint with 54th ACL, Berlin, Germany, Association for Computational Linguistics (2016) 21–27
15. Finkel, J.R., Grenager, T., Manning, C.: Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In: Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL 2005). (2005) 363–370
16. Foundation, T.A.S.: Apache OpenNLP. https://opennlp.apache.org/ Accessed: 2017-04-10.
17. Honnibal, M.: spaCy. https://spacy.io/ Accessed: 2017-06-23.
18. Bird, S., Klein, E., Loper, E.: Natural Language Processing with Python. O'Reilly Media Inc (2009)
19. Honnibal, M.: Thinc: Practical Machine Learning for NLP in Python. https://github.com/explosion/thinc Accessed: 2017-06-23.
20. Santos, D., Seco, N., Cardoso, N., Vilela, R.: HAREM: An Advanced NER Evaluation Contest for Portuguese. In: Proceedings of the 5th International Conference on Language Resources and Evaluation, LREC'2006. (2006) 1986–1991
21. Freitas, C., Mota, C., Santos, D., Oliveira, H.G., Carvalho, P.: Second HAREM : Advancing the State of the Art of Named Entity Recognition in Portuguese. In: Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10). Number 3 (2010) 3630–3637
22. Santos, D., Cardoso, N.: A Golden Resource for Named Entity Recognition in Portuguese. In: Proceedings of the 7th International Workshop, PROPOR 2006, Springer Berlin Heidelberg (2006) 69–79
23. Mota, C., Santos, D.: Apresentação detalhada das colecções do Segundo HAREM. In Mota, C., Santos, D., eds.: Desafios na avaliação conjunta do reconhecimento de entidades mencionadas: O Segundo HAREM. Linguateca (2008) 355–377
24. Pires, A.: HAREM NER Models for OpenNLP, Stanford CoreNLP, spaCy, NLTK. https://rdm.inesctec.pt/dataset/cs-2017-005 (jun 2017) Accessed: 2017-06-23.