Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Natural Language Processing and the Web
Practice Class 4

Prof. Chris Biemann and Seid Muhie Yimam

21 November 2017

The goal of the machine learning project is to develop and evaluate a component that solves essential natural language preprocessing steps. You have the choice between writing a **Named Entity Recognition (NER)** or a **Text Chunking** component. Both components are very similar in terms of the implementation effort. But they will slightly/heavily vary on the evaluation, as the features you have to select to achieve good classification results, will differ.

1 Problems

For both problems you can extend the POS-tagging project and change the classes to perform the new task. The training and classification can be done with the Conditional Random Field(CRF) algorithm which was applied for POS-Tagging. For this task the ClearTk framework [1] has to be used.

To get you started, skeleton of the NER project is also included. `NERAnnotator` should be extended to have more features. `ExecuteNER` should be extended so that training, development and test data can be loaded. Also, you should include the consumer in the pipeline. `PersonNameExtractor` is added to show you how to add features from a *list* or *gazetteers*.

1.1 Named Entity Recognition (NER)

Named entities are phrases/words that contain the name of companies, persons, locations, percentages, etc. [1]. Named Entities can e.g. be used for machine translation and in the field of information retrieval. As an example, the following sentence can be annotated like this:

The U.N.$_{\text{Organization}}$ special representative Anthony Nyakyi$_{\text{Person}}$ said :

The untagged words can be annotated as *other*.

1.2 Text Chunking

"Text chunking consists of dividing a text in syntactically correlated parts of words." [2] Text chunking is useful as preprocessing step for parsing, information extraction and information retrieval. An example can be seen below.

He$_{\text{NP}}$ reckons$_{\text{VP}}$ the current account deficit$_{\text{NP}}$

will narrow$_{\text{VP}}$ to$_{\text{PP}}$ only # 1.8 billion$_{\text{NP}}$ in$_{\text{PP}}$ September$_{\text{NP}}$.$_{\text{O}}$

[1] http://www.cnts.ua.ac.be/conll2003/ner/
[2] Conll2000,http://www.cnts.ua.ac.be/conll2000/chunking/

word	IOB1	IOB2
The	O	O
U.N.	I-ORG	B-ORG
special	O	O
representatives	O	O
Anthony	I-PER	B-PER
Nyakyi	I-PER	I-PER

Table 1: IOB Tags for the NER-tagged sentence from section 1.1

	IOB1	IOB2
He	I-NP	B-NP
reckons	I-VP	B-VP
the	I-NP	B-NP
current	I-NP	I-NP
account	I-NP	I-NP
deficit	I-NP	I-NP
will	I-VP	B-VP
narrow	I-VP	I-VP
to	I-PP	B-PP
only	I-NP	B-NP
#	I-NP	I-NP
1.8	I-NP	I-NP
billion	I-NP	I-NP
in	I-PP	B-PP
September	I-NP	B-NP
.	O	O

Table 2: IOB Tags for the Chunk-tagged sentence from section 1.2

2 Datasets

The datasets for both tasks are in the CoNLL-Format. This format specifies, that each word is written at the beginning of a new line followed by a set of features. In most cases, the last feature is the one which will be predicted. An empty line specifies the begin of a new sentence.

The chunking and the named entity feature within the dataset follow the IOB1/IOB2 format. Within the NER dataset (see section 2.1) the IOB1 and in the Chunking dataset (section 2.2) the IOB2 format is used.

In the IOB1 format, annotations begin with an `I`. Successive words with equal NE annotation or chunking annotation form a phrase. The annotation starts only with a `B`, if two different phrases of the same type follow each other immediately.

The IOB2 format specifies, that each new phrase has to begin with a `B`. An equal annotation starting with `I` only follows if it belongs to the same phrase.

As an example, the two sentences from section 1.1 and 1.2 are shown with both annotation formats in table 2 and 1.

Further information about the IOB formats can be found in [2].

Each data set has two files: A training set for learning a model from selected features and a development set to test and tune the performance of the model. You should avoid combining these files while you are evaluating the feature selection of your algorithm: Train only on the training set or a subset of the training set, and test only on the development/test set.

All data sets are located on the Moodle platform.

2.1 Data Set for NER

You can decide whether to use files in German or English language. The English files have four columns containing the word itself, followed by the POS-tag, a chunking annotation and the named entity. The German files have an additional feature: The stem of the word. The different types of named entities used in these files are listed in table 3.

Abbrev.	Long form	example
LOC	Location	Brussel, European, Germany, ...
MISC	Miscellaneous	BSE, German, Gulf War, ...
ORG	Organization	TU Darmstadt, U.N., ...
PER	Person	Werner Mustermann, Biemann, ...

Table 3: Named Entity types

2.2 Dataset for Text Chunking

For text chunking, only an English dataset is available. It has three columns, starting with the word, the POS-tag and a chunking feature. This dataset is also available at the Moodle page.

3 Additional Resources

There are some additional datasets you can use to generate new features. You can think about using only parts of these files to develop efficient algorithms that can handle the data in acceptable time, as you should evaluate different features. You can also keep them in a database. Please be aware that not all additional data might be useful for both problems. All files can be downloaded from the Additional Resources section via Moodle. For the task at least one external resource has to be used.

3.1 Named Entity List

The files [deu—eng].list contains entries with the named entities type and the named entity itself. An excerpt from the English list can be seen below:

```
...
LOC Bordeaux
LOC Born
...
MISC International Petroleum Encyclopedia
MISC Internet
MISC Iranian
...
ORG The Wall Street Journal
ORG The Walt Disney Co
ORG The Washington Post
...
PER Matthias Sammer
PER Matt Lawton
...
```

3.2 JoBimText

JoBimText provides distributional thesaurus access which is computed based on distributional similarity[3]. You can also use either the JoBimText API [4] or JoBimText calculated models if you like.

3.3 Think about other resources

Feel free to use different resources to improve the classification performance. These additional resources should be referenced and described in the documentation.

4 What has to be done

The number of achieved points depends on the implementation, your evaluation, the documentation and the presentation and slightly on the results. The project can be done in groups of two or three. All files should be uploaded via Moodle.

[3] http://maggie.lt.informatik.tu-darmstadt.de/jobimtext/
[4] http://maggie.lt.informatik.tu-darmstadt.de/jobimtext/web-demo/api-and-demo-documentation/

4.1 Implementation

Submit the source code with all resources you have been using. The annotator for the feature extraction should contain the code and configuration that achieves the best results for the given development set.

4.2 Evaluation

For the evaluation, use the Perl scripts of the CoNLL task from 2000/2003. They are available at the Moddle platform named *conlleval_[ner—chunking].pl* within the section *Evaluation Scripts*. To start the scripts, write your classification results to a text file, which is equal to the input file plus an additional column containing the classified result. The delimiter between the features is a single whitespace. An example for the English NER data is printed below:

```
CRICKET NNP I-NP O O
- : O O O
LEICESTERSHIRE NNP I-NP I-ORG I-PER
TAKE NNP I-NP O O
OVER IN I-PP O I-ORG
AT NNP I-NP O O
TOP NNP I-NP O O
AFTER NNP I-NP O O
INNINGS NNP I-NP O O
VICTORY NN I-NP O O
. . O O O
```

The Perl scripts can be executed for the NER classified file via[5]:

```
perl conlleval_ner.pl < your_classified_file
```

and vice versa for the chunking file via:

```
perl conlleval_chunking.pl < your_classified_file
```

The result for the classified example NER data above can be seen in the following. For the evaluation, have a look at the summarized values (accuracy, precision, recall and F-measure) listed in the second line. To improve single classes, have a look at the values of the single classes, listed from the third row.

```
processed 12 tokens with 1 phrases; found: 2 phrases; correct: 0.
accuracy:  83.33%; precision:   0.00%; recall:   0.00%; FB1:   0.00
              ORG: precision:   0.00%; recall:   0.00%; FB1:   0.00  1
              PER: precision:   0.00%; recall:   0.00%; FB1:   0.00  1
```

Serialize the features you used for each evaluation in the XML file format. For simple handling, there is a class called `Feature2Xml` which gives you the possibility to serialize your features into XML files, using simplified XML tags. This gives you the possibility to store your configurations for the evaluation part. Submit these XML files into your source code and explain in your documentation, which configuration contains which features.

The `PosTaggerAnnotator` from the previous tutorial has one parameter to specify the name of the XML file for the feature extractor. An example to use these XML files is listed in the following source code snippet:

```
createEngine(
   PosTaggerAnnotator.class,
   PosTaggerAnnotator.PARAM_FEATURE_EXTRACTION_FILE,"features.xml",
   ...);
```

The bold lines are the ones you have to add and to modify. In this example, the features and context features are loaded from the file *features.xml*.

Hint: As the number of features increases, also the runtime for building the model and for classification increases. For the evaluation task, it might be helpful to use only a subset of the training data. **But**, if you are using only a subset, document it!

[5]For additional functionality of the scripts, read the comments written in the first lines within these scripts.

4.3 Documentation

Write a short documentation about your component. Write about the classes you have developed and document the evaluation for the classification based on the development set with different features and their combination. Please also write about features you used, which did not improve the result. Describe the external resources you have used and add a references where it can be found.

4.4 Presentation

There will be a presentation on 05.12.2017 and 06.12.2012. Please prepare some slides to present your results within 5 minutes. Report about your evaluation, your chosen features and the best performance of your classification component.

4.5 Test the performance of your component

On 04.12.2017, you will get a test set which will be used to compare the performance of your selected features against other components. This result and the used features have to be written up into your documentation! For this task you are allowed to merge your training and development set for the model learning step.

5 Summary

- **write a (Text Chunking —— Named Entity) component**
- **Use at least one lexical resource [when doing the Named Entity task]**
- **write a documentation**
- **prepare a presentation lasting 5 minutes**
- **Solve the task in groups of two or three**
- **Overall score: 60 points.**
- **Test-set available: 04.12.2017 based on your progress**
- **Submit your files on the Moodle page.**
- **Presentation of your results: 05.12.2017 and 06.12.2012**

References

[1] Philip V. Ogren, Philipp G. Wetzler, and Steven Bethard. ClearTK: A UIMA toolkit for statistical natural language processing. In *Towards Enhanced Interoperability for Large HLT Systems: UIMA for NLP workshop at Language Resources and Evaluation Conference (LREC)*, 2008.

[2] Hong Shen and Anoop Sarkar. Voting between multiple data representations for text chunking. In *In Advances in Artificial Intelligence: 18th Conference of the Canadian Society for Computational Studies of Intelligence*, 2005.