NLTK - Training Material | WP3

Author: Bob Boelhouwer. INL Internal Review: Katrien Depuydt, Jesse de Does – November 2013

Tool Description

The Natural Language Toolkit (NLTK) is an extensive set of tools for text processing. It also comes with a large data set.
Named entity recognition in NLTK uses a statistical approach. That means that training material in the form of a manually verified corpus tagged with named entity markup is needed to produce models for the classification of named entities. The description on how to produce training material is beyond this guide.
Although the website of NLTK http://nltk.org/ contains documentation on most modules, the documentation for the named entity tools is sparse. Below I will sketch a method to produce a language model for named entity recognition and a method to tag texts.

Install software

NLTK consist of a collection of Python scripts. So, in principle you can use it in any environment for which a Python interpreter is available.
First you need to install the software. The several packages can be obtained from these locations:

- NLTK Toolset: http://nltk.org/install.html
- NLTK Data: http://nltk.org/data.html
- nltk-trainer from github: https://github.com/japerk/nltk-trainer
- copy the trainer scripts 'train_chunker.py' and 'train_tagger.py' from the git folder to the project folder
- for some reason the modules, even when installed in dist-packages, will not be found by the script. solution: copy also the library 'nltk_trainer' to your project folder.

Documentation

This guidance is very limited. More documentation can be found at these locations:

- NLTK named entity chunking: http://nltk.org/api/nltk.chunk.html#module-nltk.chunk.-named_entity
- NLTK-trainer: http://nltk-trainer.readthedocs.org/en/latest/

formats

Input format for training

Succeed is supported by the European Union under FP7-ICT and coordinated by Universidad de Alicante.

The training scripts can use the standard corpora supplied with nltk_data. It is also possible to produce new corpora. In that case the files should have one of the prescribed formats, e.g. CONLL2002. (See https://github.com/danielfrg/kaggle-yelp-recruiting-competition/blob/master/src/nltk-trainer/docs/train_tagger.rst for more details.)
The conll2002 format is pretty simple; The data files contain one word per line. Empty lines are used to mark sentence boundaries and a line containing '-DOCSTART-' mark article boundaries. Each non-empty line contains

1. The current word
2. The part-of-speech (POS) tag
3. The named entity tag

Example:

```
Het Art O
Hof N B-ORG
van Prep I-ORG
Cassatie N I-ORG
verbrak V O
het Art O
arrest N O
zodat Conj O
het Pron O
moest V O
worden V O
overgedaan V O
door Prep O
het Art O
hof N O
van Prep O
beroep N O
van Prep O
Antwerpen N B-LOC
. Punc O
```

Input format for classification

The software for classification can only read plain text. That means that if the document is in a certain format, say ALTO, there is a transformation required to plain text, and after classification the information about named entities needs to be transferred to the original document.

Output format

The output consists of a sequence of sentences which are marked as '(S …)'. Within these tags, all word tokens appear on a separate line.
Example:

```
(S
  Ik/Pron
  behoor/Prep
  niet/Adv
  tot/Prep
  de/Art
  50/Num
  beste/Adj
  kunstenaars/N
  van/Prep
  (LOC Nederland/N)
```

```
,/Punc
maar/Conj
als/Conj
allrounder/Adj
mocht/V
ik/Pron
ook/Adv
mee/Adv
doen/V
./Punc)
```

As can be seen in the example, the POS-tag is added after the word, and the NE-tag before the word embedded in parens.

Procedure

Below is an example for using the toolkit for named entity recognition. The used modules know many methods for statistical classification. In this example we use an algorithm called “Naïve Bayes”. It is not necessary that this algorithm produces the best result. It might be useful to experiment with different algorithms and options that the software makes available. However, since the software has a very broad application, some options will not be useful for NER.
The following commands will produce an overview of all available options (See appendix):

```
Python train_tagger.py --help
Python train_chunker.py -help
```

The procedure consists of two parts. First we need to train language models for the task of named entity recognition. That only needs to be done once for a data collection. The second step is using the models to classify the named entities in the documents of a collection. This has to be done for every document in the collection.
The procedure makes use of two models. The first model is used for tagging the separate words in a text. The second model is used for chunking words together in a group that forms a named entity.

Train language models

We train a model for tagging using the CONLL2002 corpus for Dutch with the following command:

```
python train_tagger.py conll2002 --fileids ned.train --classifier NaiveBayes
--filename /usr/share/nltk_data/taggers/conll2002_ned_NaiveBayes.pickle
```

This will save the model in the location ‘/usr/share/nltk_data/taggers’. Other locations can be used as long as they will be in the search path of your python impleme
ntation.
Next, we build the language model for chunking in a similar fashion:

```
python train_chunker.py conll2002 --fileids ned.train --classifier NaiveBayes
--filename /usr/share/nltk_data/chunkers/conll2002_ned_NaiveBayes.pickle
```

Classify named entities in a document

With these trained models it is possible to classify named entities in a document in plain text. Below is an example of a script that can be used for that purpose. The script is called from a command line thus:

```
python ne_chunk.py -f test_ne.txt
# script ne_chunk.py
```

Succeed is supported by the European Union under FP7-ICT and coordinated by Universidad de Alicante.

```
import sys
import getopt
from nltk.data import load
from nltk.tokenize import sent_tokenize, word_tokenize

opts, extraparams = getopt.getopt(sys.argv[1:], 'f:')

file = ''
for o,p in opts:
    if o in ['-f']:
        file = p

f = open(file, 'r')
txt = f.read()
sentences = sent_tokenize(txt.strip())
tagger = load('taggers/conll2002_ned_NaiveBayes.pickle')
chunker = load('chunkers/conll2002_ned_NaiveBayes.pickle')

for sentence in sentences:
    str_tags = tagger.tag(word_tokenize(sentence))
    str_chunks = chunker.parse(str_tags)
    print str_chunks
```

Input and output formats

The NE-tools described above can only be used with plain texts. If the workflow defines other (structured) formats for input and output, like ALTO and XML, a workaround has to be created. That workaround could exits of scripts that strip the markup from the original document in order to provide the plain format for NER. A second script can be created that injects the NE-tags in the original document based on the preserved order of the word tokens.

Appendix

Options available in tagger

```
usage: train_tagger.py [-h] [--filename FILENAME] [--no-pickle]
                       [--trace TRACE] [--reader READER] [--fileids FILEIDS]
                       [--fraction FRACTION] [--default DEFAULT]
                       [--simplify_tags] [--backoff BACKOFF]
                       [--sequential SEQUENTIAL] [-a AFFIX] [--brill]
                       [--template_bounds TEMPLATE_BOUNDS]
                       [--max_rules MAX_RULES] [--min_score MIN_SCORE]
                       [--classifier
[{NaiveBayes,DecisionTree,Maxent,GIS,IIS,CG,BFGS,Powell,LBFGSB,Nelder-
Mead,MEGAM,TADM,sklearn.BernoulliNB,sklearn.DecisionTreeClassifier,sklearn.ExtraTre
esClassifier,sklearn.GaussianNB,sklearn.GradientBoostingClassifier,sklearn.KNeighbo
rsClassifier,sklearn.LinearSVC,sklearn.LogisticRegression,sklearn.MultinomialNB,skl
earn.NuSVC,sklearn.RandomForestClassifier,sklearn.SVC}
[{NaiveBayes,DecisionTree,Maxent,GIS,IIS,CG,BFGS,Powell,LBFGSB,Nelder-
Mead,MEGAM,TADM,sklearn.BernoulliNB,sklearn.DecisionTreeClassifier,sklearn.ExtraTre
esClassifier,sklearn.GaussianNB,sklearn.GradientBoostingClassifier,sklearn.KNeighbo
rsClassifier,sklearn.LinearSVC,sklearn.LogisticRegression,sklearn.MultinomialNB,skl
earn.NuSVC,sklearn.RandomForestClassifier,sklearn.SVC} ...]]]
                       [--cutoff_prob CUTOFF_PROB] [--metaphone]
                       [--double-metaphone] [--soundex] [--nysiis]
                       [--caverphone] [--max_iter MAX_ITER] [--min_ll MIN_LL]
                       [--min_lldelta MIN_LLDELTA]
```

```
                        [--entropy_cutoff ENTROPY_CUTOFF]
                        [--depth_cutoff DEPTH_CUTOFF]
                        [--support_cutoff SUPPORT_CUTOFF] [--no-eval]
                        corpus

Train a NLTK Classifier

positional arguments:
  corpus                The name of a tagged corpus included with NLTK, such as
treebank,
                        brown, cess_esp, floresta, or the root path to a corpus
directory,
                        which can be either an absolute path or relative to a
nltk_data directory.

optional arguments:
  -h, --help            show this help message and exit
  --filename FILENAME   filename/path for where to store the pickled tagger.
                        The default is {corpus}_{algorithm}.pickle in
~/nltk_data/taggers
  --no-pickle           Don't pickle and save the tagger
  --trace TRACE         How much trace output you want, defaults to 1. 0 is no
trace output.

Corpus Reader Options:
  --reader READER       Full module path to a corpus reader class, such as
                        nltk.corpus.reader.tagged.TaggedCorpusReader
  --fileids FILEIDS     Specify fileids to load from corpus
  --fraction FRACTION   Fraction of corpus to use for training, defaults to
1.000000

Tagger Choices:
  --default DEFAULT     The default tag "-None-". Set this to a different tag, such
as "NN",
                        to change the default tag.
  --simplify_tags       Use simplified tags
  --backoff BACKOFF     Path to pickled backoff tagger. If given, replaces default
tagger.

Sequential Tagger:
  --sequential SEQUENTIAL
                        Sequential Backoff Algorithm. This can be any combination
of the following letters:
                          a: AffixTagger
                          u: UnigramTagger
                          b: BigramTagger
                          t: TrigramTagger
                        The default is "aubt", but you can set this to the empty
string
                        to not train a sequential backoff tagger.
  -a AFFIX, --affix AFFIX
                        Add affixes to use for one or more AffixTaggers.
                        Negative numbers are suffixes, positive numbers are
prefixes.
                        You can use this option multiple times to create multiple
AffixTaggers with different affixes.
                        The affixes will be used in the order given.

Brill Tagger Options:
```

```
  --brill               Train a Brill Tagger in front of the other tagger.
  --template_bounds TEMPLATE_BOUNDS
                        Choose the max bounds for Brill Templates to train a Brill
Tagger.
                        The default is 1.
  --max_rules MAX_RULES
  --min_score MIN_SCORE

Classifier Based Tagger:
  --classifier
[{NaiveBayes,DecisionTree,Maxent,GIS,IIS,CG,BFGS,Powell,LBFGSB,Nelder-
Mead,MEGAM,TADM,sklearn.BernoulliNB,sklearn.DecisionTreeClassifier,sklearn.ExtraTre
esClassifier,sklearn.GaussianNB,sklearn.GradientBoostingClassifier,sklearn.KNeighbo
rsClassifier,sklearn.LinearSVC,sklearn.LogisticRegression,sklearn.MultinomialNB,skl
earn.NuSVC,sklearn.RandomForestClassifier,sklearn.SVC}
[{NaiveBayes,DecisionTree,Maxent,GIS,IIS,CG,BFGS,Powell,LBFGSB,Nelder-
Mead,MEGAM,TADM,sklearn.BernoulliNB,sklearn.DecisionTreeClassifier,sklearn.ExtraTre
esClassifier,sklearn.GaussianNB,sklearn.GradientBoostingClassifier,sklearn.KNeighbo
rsClassifier,sklearn.LinearSVC,sklearn.LogisticRegression,sklearn.MultinomialNB,skl
earn.NuSVC,sklearn.RandomForestClassifier,sklearn.SVC} ...]]
                        ClassifierBasedPOSTagger algorithm to use, default is None.
                        Maxent uses the default Maxent training algorithm, either
CG or iis.
  --cutoff_prob CUTOFF_PROB
                        Cutoff probability for classifier tagger to backoff to
previous tagger

Phonetic Feature Options for a Classifier Based Tagger:
  --metaphone           Use metaphone feature
  --double-metaphone    Use double metaphone feature
  --soundex             Use soundex feature
  --nysiis              Use NYSIIS feature
  --caverphone          Use caverphone feature

Maxent Classifier:
  These options only apply when a Maxent classifier is chosen.

  --max_iter MAX_ITER   maximum number of training iterations, defaults to 10
  --min_ll MIN_LL       stop classification when average log-likelihood is less
than this, default is 0
  --min_lldelta MIN_LLDELTA
                        stop classification when the change in average log-
likelihood is less than this.
                          default is 0.100000

Decision Tree Classifier:
  These options only apply when the DecisionTree classifier is chosen

  --entropy_cutoff ENTROPY_CUTOFF
                        default is 0.05
  --depth_cutoff DEPTH_CUTOFF
                        default is 100
  --support_cutoff SUPPORT_CUTOFF
                        default is 10

Tagger Evaluation:
  Evaluation metrics for part-of-speech taggers

  --no-eval             don't do any evaluation
```

Succeed is supported by the European Union under FP7-ICT and coordinated by Universidad de Alicante.

Options available in chunker

```
usage: train_chunker.py [-h] [--filename FILENAME] [--no-pickle]
                        [--trace TRACE] [--reader READER] [--fileids FILEIDS]
                        [--fraction FRACTION] [--flatten-deep-tree]
                        [--shallow-tree] [--simplify_tags]
                        [--sequential SEQUENTIAL]
                        [--classifier
[{NaiveBayes,DecisionTree,Maxent,GIS,IIS,CG,BFGS,Powell,LBFGSB,Nelder-
Mead,MEGAM,TADM,sklearn.BernoulliNB,sklearn.DecisionTreeClassifier,sklearn.ExtraTre
esClassifier,sklearn.GaussianNB,sklearn.GradientBoostingClassifier,sklearn.KNeighbo
rsClassifier,sklearn.LinearSVC,sklearn.LogisticRegression,sklearn.MultinomialNB,skl
earn.NuSVC,sklearn.RandomForestClassifier,sklearn.SVC}
[{NaiveBayes,DecisionTree,Maxent,GIS,IIS,CG,BFGS,Powell,LBFGSB,Nelder-
Mead,MEGAM,TADM,sklearn.BernoulliNB,sklearn.DecisionTreeClassifier,sklearn.ExtraTre
esClassifier,sklearn.GaussianNB,sklearn.GradientBoostingClassifier,sklearn.KNeighbo
rsClassifier,sklearn.LinearSVC,sklearn.LogisticRegression,sklearn.MultinomialNB,skl
earn.NuSVC,sklearn.RandomForestClassifier,sklearn.SVC} ...]]]
                        [--max_iter MAX_ITER] [--min_ll MIN_LL]
                        [--min_lldelta MIN_LLDELTA]
                        [--entropy_cutoff ENTROPY_CUTOFF]
                        [--depth_cutoff DEPTH_CUTOFF]
                        [--support_cutoff SUPPORT_CUTOFF] [--no-eval]
                        corpus

Train a NLTK Classifier

positional arguments:
  corpus                The name of a chunked corpus included with NLTK, such as
treebank_chunk or
                        conll2000, or the root path to a corpus directory, which
can be either an
                        absolute path or relative to a nltk_data directory.

optional arguments:
  -h, --help            show this help message and exit
  --filename FILENAME   filename/path for where to store the pickled tagger.
                        The default is {corpus}_{algorithm}.pickle in
~/nltk_data/chunkers
  --no-pickle           Don't pickle and save the tagger
  --trace TRACE         How much trace output you want, defaults to 1. 0 is no
trace output.

Corpus Reader Options:
  --reader READER       Full module path to a corpus reader class, such as
                        nltk.corpus.reader.chunked.ChunkedCorpusReader
  --fileids FILEIDS     Specify fileids to load from corpus
  --fraction FRACTION   Fraction of corpus to use for training, defaults to
1.000000
  --flatten-deep-tree   Flatten deep trees from parsed_sents() instead of
chunked_sents().
                        Cannot be combined with --shallow-tree.
  --shallow-tree        Use shallow trees from parsed_sents() instead of
chunked_sents().
                        Cannot be combined with --flatten-deep-tree.
  --simplify_tags       Use simplified tags

Chunker Options:
  --sequential SEQUENTIAL
```

```
                        Sequential Backoff Algorithm for a Tagger based Chunker.
                        This can be any combination of the following letters:
                          u: UnigramTagger
                          b: BigramTagger
                          t: TrigramTagger
                        The default is "ub". If you specify a classifier, this
option will be ignored.
  --classifier
[{NaiveBayes,DecisionTree,Maxent,GIS,IIS,CG,BFGS,Powell,LBFGSB,Nelder-
Mead,MEGAM,TADM,sklearn.BernoulliNB,sklearn.DecisionTreeClassifier,sklearn.ExtraTre
esClassifier,sklearn.GaussianNB,sklearn.GradientBoostingClassifier,sklearn.KNeighbo
rsClassifier,sklearn.LinearSVC,sklearn.LogisticRegression,sklearn.MultinomialNB,skl
earn.NuSVC,sklearn.RandomForestClassifier,sklearn.SVC}
[{NaiveBayes,DecisionTree,Maxent,GIS,IIS,CG,BFGS,Powell,LBFGSB,Nelder-
Mead,MEGAM,TADM,sklearn.BernoulliNB,sklearn.DecisionTreeClassifier,sklearn.ExtraTre
esClassifier,sklearn.GaussianNB,sklearn.GradientBoostingClassifier,sklearn.KNeighbo
rsClassifier,sklearn.LinearSVC,sklearn.LogisticRegression,sklearn.MultinomialNB,skl
earn.NuSVC,sklearn.RandomForestClassifier,sklearn.SVC} ...]]
                        ClassifierChunker algorithm to use instead of a sequential
Tagger based Chunker.
                        Maxent uses the default Maxent training algorithm, either
CG or iis.

Maxent Classifier:
  These options only apply when a Maxent classifier is chosen.

  --max_iter MAX_ITER   maximum number of training iterations, defaults to 10
  --min_ll MIN_LL       stop classification when average log-likelihood is less
than this, default is 0
  --min_lldelta MIN_LLDELTA
                        stop classification when the change in average log-
likelihood is less than this.
                          default is 0.100000

Decision Tree Classifier:
  These options only apply when the DecisionTree classifier is chosen

  --entropy_cutoff ENTROPY_CUTOFF
                        default is 0.05
  --depth_cutoff DEPTH_CUTOFF
                        default is 100
  --support_cutoff SUPPORT_CUTOFF
                        default is 10

Chunker Evaluation:
  Evaluation metrics for chunkers

  --no-eval             don't do any evaluation
```

Licensing

NLTK is licensed under the Apache License, Version 2.0 (http://www.apache.org/licenses/LICENSE-2.0).

Succeed is supported by the European Union under FP7-ICT and coordinated by Universidad de Alicante.