notebook.community

EDIT AND RUN

Named Entity Recognition

This is a tutorial for NER (named entity recognition). In this tutorial you will see

- how to apply a pre-trained named entity recognition model to your text

It is assumed that you have some general knowledge on

- .. no particular knowledge required. You should be able to read texts, though ;-)

Prerequisites. We first need to install the Stanford NER tagger from here. And java also has to be installed. You have to figure out

- where the jar file `stanford-ner.jar` is located
- where the pretrained models (e.g. `english.all.3class.distsim.crf.ser.gz`) is located, this is the subdirectory classifiers
- whether the right version of java is installed. On a command line type `java -version` to see the version. Refer back to the documentation on the stanford nlp page to see which version is needed.

You can also test the NER tagger online here.

In [7]:

```
from nltk.tag import StanfordNERTagger
from nltk.tokenize import word_tokenize

# Adapt those lines to your installation
jar_location = '/Users/sech/stanford-ner-2018-10-16/stanford-ner.jar
model_location_3classes = '/Users/sech/stanford-ner-2018-10-16/classi
model_location_7classes = '/Users/sech/stanford-ner-2018-10-16/classi
st3 = StanfordNERTagger(model_location_3classes,jar_location,encoding
st7 = StanfordNERTagger(model_location_7classes,jar_location,encodin

print(st3)
print(st7)
```

```
<nltk.tag.stanford.StanfordNERTagger object at 0x1a1d98af60>
<nltk.tag.stanford.StanfordNERTagger object at 0x1a1d98af28>
```

Let's take a paragraph from the Wikipedia page of Ada Lovelace as an example. We need to put the text in triple quotes since the text itself contains quoting characters.

```
In [10]:

text = '''Lovelace became close friends with her tutor Mary Somervill
print(text)
```

```
Lovelace became close friends with her tutor Mary Somerville, who in
```

First we need to tokenize the text and then we apply the NER tagger. Let's try both, the 3 class version and the 7 class version.

```
In [11]:

tokenized_text = word_tokenize(text)
text_ner3 = st3.tag(tokenized_text)
text_ner7 = st7.tag(tokenized_text)

print(text_ner3)
print(text_ner7)
```

```
[('Lovelace', 'PERSON'), ('became', 'O'), ('close', 'O'), ('friends',
[('Lovelace', 'O'), ('became', 'O'), ('close', 'O'), ('friends', 'O')
```

We see that each word is tagged. Tags are for instance `ORGANIZATION` or `PERSON`. Very prominently, the `O` tag appears often. This is the `other` class (everything that is not an organisation or person, etc.). But it is still an aweful lot of text. Let's just have a look at the non-other entities detected. We do this assuming that adjacent words having the same tag should be collapsed into one named entity.

```
In [13]:

from itertools import groupby

print("**** 3 classes ****")
for tag, chunk in groupby(text_ner3, lambda x:x[1]):
    if tag != "O":
        print("%-12s"%tag, " ".join(w for w, t in chunk))

print("**** 7 classes ****")
for tag, chunk in groupby(text_ner7, lambda x:x[1]):
    if tag != "O":
        print("%-12s"%tag, " ".join(w for w, t in chunk))
```

```
**** 3 classes ****
PERSON       Lovelace
PERSON       Mary Somerville
PERSON       Charles Babbage
LOCATION     Somerville
PERSON       Andrew Crosse
PERSON       David Brewster
PERSON       Charles Wheatstone
PERSON       Michael Faraday
PERSON       Charles Dickens
PERSON       Ada
PERSON       John Hobhouse
PERSON       Byron
PERSON       Ada
PERSON       Hobhouse
**** 7 classes ****
PERSON       Mary Somerville
PERSON       Charles Babbage
DATE         1833
LOCATION     Somerville
PERSON       Andrew Crosse
PERSON       David Brewster
PERSON       Charles Wheatstone
PERSON       Michael Faraday
PERSON       Charles Dickens
DATE         1834
ORGANIZATION Ada
PERSON       John Hobhouse
PERSON       Byron
DATE         February 1834
ORGANIZATION Ada
PERSON       Hobhouse
```

We see that while this is pretty impressive, it still makes errors. For example, one occurrence of Ada is tagged a `ORGANISATION`. You should take the non-perfect nature into account if you use those tags further in your nlp pipeline.

That's all.

```
In [ ]:
```

Content source: chseifert/tutorials

Similar notebooks:

- Named-Entity-Recognition
- class14

- Text-Preprocessing
- HW7
- fizzbuzz two ways
- inherit
- test_tensorflow_and_jupyter_install
- UsingNERCs
- keras_preprocessing_image
- Sunoikisis - Named Entity Extraction

1b-VV

notebook.community | gallery | about