Tutorial 4: Welcome to POS Tagging

Alok Debnath

25 January 2019

1 Introduction

Part of speech tagging is the process of assigning the POS tag, or a grammatical category to a word. The idea behind this assignment of a category is to understand the behaviour of a class of words that belong to the same syntactic category or behave similarly in their morphology. As discussed before, the POS taggers require two independent parts, a tag set and a tagging algorithm. While some uses of a tagset had been previously discussed, we shall briefly answer the following questions:

- What is the right number of tags for a tagset in a POS tagger?
- What are some of the features of the Indian POS tagset?
- How do we use the tools that are available for our purposes?
- (extra) What are the principles of designing a POS tagger?

POS tagging, as we know, is a preprocessing task and therefore errors at this stage of analysis leads to propagation of error through the rest of the pipeline, which is why a very high accuracy is demanded in this task. More importantly, overall, POS tagging is a context sensitive classification task, which makes error analysis a crucial part of improving the POS tagger models.

2 The Right Number of Tags?

The question of the right number of tags, the right number of classifications of each broad POS has been a debate for the longest time. Like most classification tasks, POS tags have the problem granularity versus ambiguity in classification and representation of information. The gist of the debate is as follows:

Let us suppose we have x number of POS tags. These POS tags represent a certain amount of information and there are certain rules for classifying words into these tags. Now let us suppose that for some task, we require more information about a word of a particular class. We can create another class that represents the presence of this new feature, thereby having $x + 1$ tags. However, for rule based systems, new rules have to be defined for the inclusion of words in this new class AND rules for the exclusion from their previous class. For a data-centric approach, there have to be enough words in the new class in order for any system to draw a correlation between the word, the tag and the surrounding tags.

In this scenario, we see that there is a gain of information by increasing the granularity of the tagset (i.e. by increasing the number of tags). However, there is a strong need to handle ambiguity in the rules or the possible tags that can be given to a word given the context tags. This ambiguity can cause errors, which is a major fear in preprocessing tasks, as mentioned above. Therefore, it shows almost immediately that the tagset design and the choice of tags is not only dependent on the task at hand, but also on gaining the most amount of structural information with as little ambiguity as possible.

The CLAWS tagset and the Penn Treebank POS Tagset are good examples of this. The Penn Treebank Tagset was developed for the purpose of maintaining a high-quality treebank of English sentences. Therefore the purpose of POS tagging was to generate unambiguous parse trees, and the team at University of Pennsylvania came up with 36 such tags, which aided with parsing. The many versions of the CLAWS tagset. developed at the University of Lancester, was made for aiding with the constituent detection, releasing the one-to-one word-to-tag relationship. This allows them to have multiple tags, which is then used to determine the likelihood of a word belonging to a particular constituent. The result, the CLAWS systems have 60 tags (C5) or 160 tags (C7) (based on the constraints or tagging guidelines followed).

3 POS Tagging for Indian Languages

The Bureau of Indian Standards (BIS) created a fundamental set of POS Tags for some of the most common Indian languages. The design of the POS tags was based on well defined design principles, one of which is the independence from the burden of downstream tasks. POS Tagging for Indian Languages is morphological as much as it is syntactic, because a large number of Indian languages are agglutinative. Therefore the tag set is defined in terms of morpho-syntactic roles, and includes a wide range of characteristics such as plurality, verbalization and so on. The document on the BIS POS tagset has been designed to be flexible (so that the guidelines can be appended for the specific syntactic characteristics of each language), consistent in annotation, annotator friendly (unambiguous guidelines) and compatible with other existing schemes. Since the BIS establishes the POS tags for multiple languages, the guidelines have to be applicable across languages. Most importantly, POS tags enrich the amount of information available at the most basic level of linguistic information, therefore it should be easily portable to other tasks.

The BIS POS Tagset and their official documentation will be uploaded with this tutorial sheet.

4 [LAB WORK] Using the tools

4.1 English: NLTK

```
$ python3
Python 3.6.6
Type "help", "copyright", "credits" or "license" for more information.
>>> import nltk
>>> text = 'This can be whatever text you like, okay? Do not copy this line as it is.'
>>> words = nltk.word_tokenize(text)
>>> nltk.pos_tag(words)
[('This', 'DT'), ('can', 'MD'), ('be', 'VB'), ('whatever', 'WDT'), ('text', 'IN'),
('you', 'PRP'), ('like', 'VBP'), (',', ','), ('okay', 'FW'), ('?', '.'), ('Do', 'VBP'),
('not', 'RB'), ('copy', 'VB'), ('this', 'DT'), ('line', 'NN'), ('as', 'IN'),
('it', 'PRP'), ('is', 'VBZ'), ('.', '.')]
```

4.2 Indian Languages POS Tagging

Github repo: `https://github.com/rajesh-iiith/POS-Tagging-and-CYK-Parsing-for-Indian-Languages`

```
$ git clone https://github.com/rajesh-iiith/POS-Tagging-and-CYK-Parsing-for-Indian-Languages
Cloning into 'POS-Tagging-and-CYK-Parsing-for-Indian-Languages'...
remote: Enumerating objects: 38, done.
remote: Total 38 (delta 0), reused 0 (delta 0), pack-reused 38
Unpacking objects: 100% (38/38), done.

$ cd POS-Tagging-and-CYK-Parsing-for-Indian-Languages
$ python supervised.py X ./data/X_testing.txt
0.345607042313 seconds for training
0.622769117355 seconds for testing 100 Sentences

Kindly check ./output/X_tags.txt file for POS tags.
```

where X and X_testing.txt are 0 being "hindi", 1 being "telugu", 2 being "kannada" and 3 being "tamil".

for example, for Hindi, the command is:

```
$ python supervised.py 0 ./data/hindi_testing.txt
```

Change the sentences in the testing file in the data directory for your language.

5 [EXTRA] Designing a POS Tagger

So, one of the most important aspects of a POS Tagging System is the algorithm used to design it. I will provide you with a brief intuition on the design of a POS Tagger which is rule based and one which is statistical. The former deals with adequately defined rules and the latter with the statistics of word and tag occurrences. Interestingly, the parameters of both making the rules or the word-tag occurrences are simply whether ”the word belongs to a particular tag” and ”does a tag succeed another tag”. And this is the basic principle of designing a POS Tagger.