Tutorial 3: Introduction to Preprocessing Tasks and Tools

Alok Debnath

18 January 2019

1 Introduction

The next section of the lectures in this course will aim to introduce preprocessing tasks in NLP, such as part-of-speech tagging, chunking and local word grouping, multiword expressions and morphanalysis. While the course touches upon the other tasks as well, such as parsing, the ideas in this tutorial should give you a brief understanding of some lexical preprocessing tasks.

The subject of computational linguistics is based upon extracting more data from the text than is readily available, in order to perform "larger" tasks such as translation, any analysis task, question answering and so on. However, as we shall soon explore, the preprocessing tasks are well researched and still expanding areas of research. With this in mind, this tutorial is geared towards answering the following questions:

- What is part-of-speech tagging?
- What are chunking, local word grouping, multiword expressions and shallow parsing?
- What is morphanalysis? What are some of the basic subtasks in morphanalysis?
- What tools do we use for these tasks?

2 What is POS Tagging?

Parts of speech are a very familiar grammatical concept. The idea is that words in a sentence have different roles, which can be abstracted over, to understanding syntactic patterns in the language. Parts of speech are one such abstraction. Nouns, pronouns, adjectives, verbs, adverbs, prepositions, conjunctions and interjections are the most well known parts of speech. And while to a laymen, these are enough to understand the structure of a sentence, as computational linguists, this minimal set is not nearly enough. Why?

POS tagging is an important preprocessing task, as it opens up avenues of using latent syntactic information for later, larger tasks (also known as downstream tasks). A POS tagger, as a system, consists of two basic components:

- **Tag set**: A tag set is the set of tags used for the task. The most basic set known to us has already been introduced above, but as mentioned, it is hardly enough. The most common tag set, the Penn Treebank POS Tagset has as many as 36 tags. Other tag sets have more than 50 tags, others have approximately 15-20. The number of tags in a tag set makes the process more coarse or fine grained. Why are there such variations?
- **Algorithm**: No task is complete without an algorithm. POS Tagging algorithms have been tested upon extensively, and can roughly be divided into three major classes, rule based, statistical and neural POS Taggers. Rule based systems are extensively used in domain specific applications.

Think about this: What is the role of context in POS Tagging?

3 What are chunking, local word grouping, multiword expressions?

Often, words individually do not contribute much towards the analysis of a corpus, but words, in combination with other words, are far more useful. Let us take the example of "post office" or "telephone box". These words behave syntactically as one unit, and are also semantically closely related. The study of such groups of words occurs at different levels of semantic information, and have been classified as three similar tasks.

Chunking, also known as shallow parsing, is the most syntactic of these processes, in theory. A chunk is just the group of words that can be observed together at the first level of the phrase structure tree bottom up. Chunks are identifiable using POS tags.

Local word grouping is a similar concept for Indian and other semantically restrictive languages. A local word group can loosely be defined as the that group of words that behaves like a single unit semantically, but exists as different words in the language. These are not strongly bound to the part of speech, but certain patterns can be abstracted from the data.

Multiword Expressions are almost entirely semantic units, which can be idiomatic, phrasal or simply referring to a single action or entity, but using multiple lexical items. Processing of natural language is sequential, and more often than not, words are the units used for this purpose.

4 What is morphanalysis? What are some of the subtasks in morphanalysis?

One of the most difficult and ardently language specific tasks in the computational linguistics community is the task of morphological analysis. Morphanalysis refers to the task of analyzing the root form, inflections and derivations that are a part of a word. English and Indian languages have very distinct morphological characteristics, and even in Indian languages, each language inflects and derives from the root form differently. However, there are some patterns that *most* languages follow, which are the list of subtasks as follows:

- Stemming/Lemmatization: The goal of both stemming and lemmatization is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form.
- Affix identification: Most languages have either a common set of affixes or phonetic/orthographic rules for the combination of words into a single form. This process is called affix identification.

These are mostly carried out in parallel, and therefore the task of morphanalysis becomes an overloaded one. For Indian languages, concepts such as *samas* and *sandhi* are morphological characteristics that have to be at least partially encoded into the system due to lack of extensive data for machine learning algorithms to train on.

5 What tools do we use for these tasks?

5.1 Downloading NLTK

5.1.1 Prerequisites

Please run the following commands:

```
sudo apt-get install pip
sudo apt-get install pip3
```

`pip` is a Python Package manager, which is used to download most python packages in their latest stable release. Now that you have `pip`, run the following

```
pip install setuptools
pip3 install setuptools
```

Now, you have the basic tools for setting up the tools in the NLTK toolkit.

Installing NLTK Instructions taken from: https://www.nltk.org/install.html

1. Install NLTK: Run the following `sudo pip3 install -U nltk`
2. Install NumPy (note that this is recommended, though optional): Run `sudo pip3 install -U numpy`

3. Test the installation: Run `python3`. Your screen should look like this.

   ```
   Python 3.6.6 (default)
   Type "help", "copyright", "credits" or "license" for more information.
   >>> import nltk
   ```

4. If there are no errors at this stage, you have installed NLTK correctly.

5.2 Getting NLTK Data

Instructions taken from: https://www.nltk.org/data.html No project is complete without the data on which the tools is to be used. Luckily, we have the data provided by NLTK, which includes multiple corpora, the most important of which is the brown corpus. All preliminary experiments will be carried out on the brown corpus. Follow the following instructions:

1. Launch `python3`

2. Run the commands:

   ```
   >>> import nltk
   >>> nltk.download()
   ```

3. A popup box should show up, click on the option `book`.

The NLTK book details many of the stable developments in NLTK, and the corpora used in the book and therefore, we shall be using these for the time being.

5.3 Testing it out

Let us test out our downloads, and understand what we have so far.

```
$ 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’), (’.’, ’.’)]
```

5.4 Indian Languages NLP Tools

You can download the tools for NLP in Indian Languages from this URL: http://183.82.119.160/ilmt-panel (run on college internet please):

Username: guest

Password : guest@ilmt123

Please follow download instructions as mentioned in the class:

Login Select category > Select Language Pair (Y2X) > Download

X can be anything except "pan", so do not use hin2pan for any of the installations. Y is your native tongue or the indian language you are most comfortable with.