Tutorial 5: Indian Languages and Types of POS Tagging

Alok Debnath

1st February 2019

1 Introduction

The ideas covered in class so far deal with using a tagset and developing the rules for a given language and a given task. As it has been proven, these tasks are not simple ones and therefore, research in the field of POS tagging has been faced with multiple challenges, ony few of which have been encountered so far. In this tutorial, we expand on some of the challenges that POS tagging brings, especially with resource poor and resource starved languages like ours, and some ideas on the algorithmic and development ideas that are geared towards tackling these problems.

This tutorial aims to answer the following questions:

1. What is India as a Linguistic Area? How does that affect POS tagging?
2. What are "levels of analysis" and how are they relevant to POS tagging?
3. What are the different types of taggers? (HMMs, transformational taggers, and so on)

2 India as a Linguistic Area

This term 'linguistic area' may be defined as meaning an area which includes languages belonging to more than one family but showing traits in common which are found not to belong to the other members of (at least) one of the families. We know that by definition, India is a linguistic area, because of the multiple language families that interact with one another in this region, two major ones including the Indo-Aryan and Dravidian language families. The other language families include Austro-Asiatic and Tibeto-Burman.

While defining the family of these languages after centuries of diffusion is a challenge (one concerned with historical linguistics in particular), this heavy diffusion of linguistic traits permeates simply the vocabulary, into the syntactic and morphological characteristics of the various languages in this area, which is a double edged sword as far as preprocessing tasks are considered. While most

computational preprocessing tasks, such as POS tagging, chunking, morphanalysis are data intensive, the similarity in the languages is that a model created for one of the languages can be successfully applied to other languages of this region, even across language families. Because Indian languages are resource poor or resource starved, the non-data oriented models, such as rule based or tranformational taggers and parsers can be used to bootstrap data creation and aid in developing resources with less data and human effort.

Some of the problems with this approach is that a linguistic area gives rise to unique language variations, each of which have different properties than the 'standard' variation of the language. A standard language is one that is used for formal purposes, such as education, legislation and so on. While computational tasks require either a standard language or data in each of the possible variations, using the standard language rids the language of richness and diversity, which harm language preservation. Furthermore, a small amount of annotated corpus and a method of verification of bootstrapped data are necessary, which is difficult for most Indian languages.

3 POS Tagging and Levels of Analysis

One of the interesting contributions that BIS POS tagset made and used was making the concept of 'levels of linguistic analysis' a concrete idea. Linguistic analysis can be broadly defined as the extraction of features and information which is latent in the surface text. For example, POS analysis is a type of linguistic analysis, because it merely classifies the words, giving more information about it. Other such tasks include morphanalysis, phrase structure parsing, dependency parsing, word sense disambiguation, sentiment analysis and so on. Note that none of these tasks generate any new text. They are just analyzing the surface form an gathering more information about the text. Levels in linguistic analysis, therefore, implies the existence of a hierarchy, where a a task at a higher level requires certain information from the lower level of the task, and the lowest level requires information directly from the corpus.

POS tagging is one of the lowest levels of linguistic analysis, and therefore, designing a tagset that can be used for multiple higher levels of analysis is a challenge. However, if a tagset were nested, in the sense that a coarse tag has more granular tags as subclassses (along with a model that supports this nesting, of course), can be used for multiple tasks. For example, sentiment analysis, which does not require fine grained tag information, but needs the part of speech and the word sense disambiguation, can be given that information, while word sense disambiguation itself, which needs fine grained tag information, can be given that information instead.

4 Models of POS Tagging

With the completion of Assignment 2 Part A, it should be abundantly clear that there is no POS tagging method that can be done without at least some form of annotated data. A basic end-to-end pipeline for having a POS Tagger in a language goes as follows:

1. Collect and clean data.
2. Decide on a minimal number of tags.
3. POS tag the data based on this set.
4. Gather annotator data.
5. Analyze the information lost and discrepancies in the annotation.
6. Decide the new tags and what should be annotated in those tags.
7. Repeat steps 3 through 6 until a POS tag set is created and the data is tagged according to the latest tag set.

At this point, an unambiguous tag set and the corpus is ready. As you are familiar with developing rules for a rule based system, the idea of developing a rule based system is pretty much clear. The systems in which these rules are implemented are called constraint grammars. However, now, we explore two basic models of POS tagging which are more often used and integrated with other, larger systems. These two methods are transformation based taggers and stochastic taggers.

4.1 Transformation based Taggers

A transformation based tagger (e.g. Brill Tagger: error driven) is a POS tagger which has a unique algorithm. The Brill tagger tags each lexical item with its most common tag, and then, given the context, it changes the tag of the focus word by a set of predefined rules. The Brill tagger, therefore, depends highly on the accuracy and neutrality of the dataset, primarily because an unobserved context will result in incorrect tagging of the words.

More powerful (data-centric) transformation based taggers 'learn' the context sensitivity of a tag and derives the rules of transformation, which then, given the test data, provides the most common tag and then changes it based on the rules it has learned. The rules are learned by providing parameters such as (word, tag), (word, context) and (tag, context), where the context can be a few words or tags on either side of the focus word.

4.2 Stochastic Taggers

A stochastic process is defined in mathematics as a process where data (x) from a sample space $\{x : x \in S\}$ is indexed by a set T. POS tagging can

be modelled as a stochastic process. Therefore, it can be solved by modelling the classification of words in a particular tag and identifying the parameters required in this process. This is the broad understanding of POS Tagging. I shall cover the basic idea of one of the most popular models of tagging, which uses the concept of Hidden Markov Models (HMM).

An HMM is a model which is based on an assumption known as the Markov assumption. The Markov assumption suggests that the probability of a future event depends only on the current event, and not on the history. Of course the Markov assumption is an exaggerated simplification of natural language syntax. In practice, the Markov assumption is a bit more relaxed, such that the model takes into account a certain window of history larger than just the previous event.

We define an observation as the occurrence of a word in a sequence. We define a state as a member of the set of labels used to classify the word, aka the tags. As has been mentioned before, there are two probabilities that are to be maximized for assigning the word a tag. These probabilities are:

1. Emission Probability: The probability that a lexical item "emits" a tag (e.g. `P(was is a VBD)`)

2. Transition Probability: The probability that a tag occurs after another tag or sequence of tags (e.g. `P(VBD is followed by VBD)`

These probabilities act as parameters for multiple algorithms including Viterbi and Baum-Welch, which are the most widely used. The Viterbi algorithm is a dynamic programming algorithm which aims at maximizing the product of emission and transition probabilities for a given window size. The Baum-Welch algorithm is an algorithm that creates a probability distribution of a sequence of transitions and maximizes over it.