TDT4310/2021S: Intelligent Text Analytics and Language Understanding Prof. Björn Gambäck

Lab Exercise 2

Lab Date: 5th of February 2021

1. **Guidelines**

   Deadline for submitting your solution: **23:59 18th February 2021**.
   Submission is a zipped folder with {*your name*} containing:

   (a) Source files (Python): format name as Lab{LabNumber}_{Exercise Number}.py.

   (b) A summary/report file formatted as a pdf which explains and presents the results with respect to the input values.

2. **Exercises** Use the *Brown Corpus* or the *NPS Chat Corpus* as stated in the exercises below. The *Brown Corpus* contains sentences, the *NPS Chat Corpus* contains posts.

   Exercise 1: Write code to do the following tasks, using the *Brown Corpus*.

   (a) Write out the most frequent tag in the corpus.

   (b) How many words are ambiguous, in the sense that they appear with at least two tags?

   (c) What is the percentage of ambiguous word in the corpus?

   (d) For the 10 words with the highest number of distinct tags, print out some sentences from the *Brown Corpus* that contain each of the words.

   Exercise 2: We are going to train our own taggers. We want to compare different ratios of training and test data with different taggers, to find out which combinations has the best performance. Using the corpora (*Brown*, *NPS Chat*) we want to use the ratios 90/10 and 50/50. (i.e., all four permutations, *Brown 50/50*, *Brown 90/10*, *NPS 50/50*, *NPS 90/10*)

   (a) Tag each word with a *DefaultTagger* using the most common tag in the corpus as the default tag. Calculate the accuracy for each test set, and comment the results.

   (b) Use a regular expression (*RegEx*) tagger[1] and use n-gram taggers (*UnigramTagger*, *BigramTagger*) to perform backoff. You can freely chose ordering. Print the accuracy of each combined tagger.

   Exercise 3:

   (a) Instead of using the *DefaultTagger* for unknown words as noun, create a tagger which uses the *LookupTagger* [2] as your default backoff with the model on the 200 most frequent words in the *Brown Corpus*. Apply this tagger to all your test sets and print the accuracies. Comment the results in comparison to the results from the last exercise.

   (b) Try varying the size of the data sets (i.e., the total amount of sentences, not their ratios). How does this affect your results, and why?

   Exercise 4: The Hidden Markov Model (HMMs) are usually used to solve the task of Part-of-Speech(POS) tagging.[3] In this exercise we will use HMMs to calculate probabilities for words and tags, and use apply the Viterbi algorithm to efficiently find the most likely tag for sequences of words.

   You can find the code for the exercise as an attachment.

   (a) Print the probability of a *Noun(NN)* being 'we'.[4]
   Print the probability of a *Verb(VB)* being 'like'.
   Print the probability of a *Preposition (PP)* being followed by a *Verb (VB)*.

   (b) What is the probability of the tags *PP VB PP NN* used for the following sequence of words "I conduct my conduct"?

   (c) What is the best tags for this sentence "You should invest in the stock market" and its probability? (that is not a serious endorsement)

[1] Chapter 5, Section 4.2 contains the default patterns for the *Regex* Tagger
[2] Chapter 5, Section 4.3 contains the default patterns for the *Lookup* Tagger
[3] `https://www.nltk.org/_modules/nltk/tag/hmm.html`
[4] Both the returns from probDist are ConditionalFreqDist from Nltk.
They have a built-in *.prob* method :)