COM6513 Lab 4: Structured Prediction and Part-of-Speech Tagging
Registration Number: 160122440

1. INTRODUCTION

The aim of this lab session was to implement part-of-speech tagging using the structured perceptron.
The dataset used for this particular task were sentences of up to length 5, which were imported from Brown corpus. The first 2500 sentences were used as training data and the remaining 733 were used as the testing data.
For the feature function $\Phi(x, y)$ in the algorithm, assuming each sentence-pos tag sequence (x, y) is represented as $(word_1, tag_1 ... word_N, tag_N)$, two kinds of features were implemented: $word - tag_i$ and $tag - tag_{+1}$.
For the implementation of the algorithm, 8 functions were used. A function named *split_data_to_train_and_test* was first created in order to split the dataset to training and testing parts. The second function implemented ($features$) was used to extract the features explained above. Moving on, a function to get all the possible part-of-speech tags was created (*tags*), and a total of 14 tags were collected (['.', 'END', 'PRON', 'VERB', 'NOUN', 'ADJ', 'DET', 'ADV', 'PRT', 'CONJ', 'NUM', 'ADP', 'X']). Then, a function to extract all the words from each sentence was implemented (*sentence_words*), along with a function to create all possible *word_tag* (*combinations*). Additionally, the function *combinations_tag_tag* was used to create all the possible *tag_tag* combinations. Finally, *training* and *testing* functions were created to train the structured perceptron and test it by evaluating the prediction accuracy.

2. IMPROVEMENTS

2.1 Shuffling

Since dictionaries were used in this assignment, for both the features and the weights, then shuffling would not have made any difference. This is because dictionaries do not have a defined order, therefore, shuffling was not implemented as an improvement for this particular assignment. Shuffling is generally a very good improvement (if dictionaries are not included), since the results provided after randomising are considered to be more reliable. This is because data partitions normally come from different sources.

2.2 Multiple Passes

Multiple passes was implemented by inserting the whole training function in a for loop, which caused the function to repeat the training procedure for 5 iterations. The benefit for this improvement is that as the algorithm repeats the training procedure, the accuracy is improved in every iteration.

2.3 Averaging

The final improvement was averaging, where the average was found by calculating the sum of the correctly predicted operations and dividing with the total number of predictions. Averaging provides a better overall representation of the the weights and more precise accuracy.

2.4. Accuracy

Even though the accuracy did not improve a lot after the improvements, there was still a small increase from 0.50 to 0.54 indicating that the improvements do indeed provide a better accuracy.

3. ANSWERS TO QUESTIONS

3.1 Accuracy

The final word-level accuracy was 0.54.

3.2 Highest-Weighted Features

The most positively-weighted features for each class were found, in order to visualise whether or not they seem sensible; which in fact they do. These can be shown in the following table (Table 1):

features:	weight:
._END:	1129.6
NOUN_END	1048.4
VERB_.	507.8
ADJ_NOUN	486.4
NOUN_NOUN	523.0
NOUN_.	833.8
NUM_.	713.2
PRON_VERB	342.0
NOUN_VERB	456.0
._.	875.4

Table 1: Highest-weighed features.

As shown from the table most of the features do make sense. For example the highest weighted one is a full stop and the tag 'END' which is basically a feature that is included at the end of almost all sentences (with the exceptions of the sentences that end with other punctuation marks). The second highest is the 'Noun' with and 'END' tag which again this can be the end of many sentences (if punctuation marks are not used). Another highly weighted feature is the 'ADJ'_'NOUN' which again is very common since an adjective is a word naming an attribute of a noun, such as: red ball, sweet cake, etc. Finally, another features shown on the table is the 'PRON'_'VERB' which again is very common since pronouns usually come before a verb, such as: I hear, you sleep.

3.3 Better Features

A better feature could have been to create features based on the tag and the previous tag, instead of the next tag, or even a combination of the two. Additionally, another feature could have been to include trigrams, since this creates bigger features with more related words/tags together.

3.4 Computational Bottleneck

The following table shows the required execution times for three different tasks (Table 2) :

Bottleneck:	training(argmax)	data splitting	testing
Execution Time (seconds):	891.48	0.03	17.40

Table 2: Computational Bottlenecks Execution Time.

As seen in the table above (table 2) the main computational bottleneck is the training procedure where the argmax is performed. One possible reason due to the vast amount of feature combinations generated, along with argmax computation which is included in the training section. For example if we have a sentence with 2 words and there are 3 possible part-of-speech tags, then there are (3^2) 9 combinations generated.