Table of Contents

General Overview:

In Programming Assignment 3, the task was to implement IBM translation models 1 and 2 and use the implementation to learn word alignments in an English/ German parallel corpus.

The code is present in a file called ***'code4.py'*** where all the tasks of the assignment have been carried out in functions or modules.

Shell script "run.sh" will run all necessary scripts for the programming assignment and produce the required outputs inside a directory called ***'output'*** under the current directory.

For executing the code for the assignment, go to the current directory and type the following command:

>>./run.sh

Question 4:

Initialization of the 't' parameters

Only possible combinations of English and German word pairs have been considered while initializing 't'. Also 'NULL' has been appended at the beginning of every English sentence so that it can be aligned to every German word in the corpus.

EM Algorithm for IBM Model 1

5 iterations of the EM Algorithm have been implemented for IBM Model 1. It takes approximately 1.7 minutes to execute the 5 iterations.

Then for every English word in *'devwords.txt'* the 10 most probable German words are printed and stored in ***'log_devwords.txt'*** in the ***'output'*** folder. This computation is carried on the basis of the 't' parameters calculated after 5 iterations of the EM algorithm on IBM Model 1.

We observe that English word 'I' 'man' and 'anniversary' have 'ich', 'mann' and 'jahrestag' respectively, as the most probable German words. This indicates that for some English words the most probable German word predicted is the actual German word for the corresponding English word.

In other cases for words like 'depicted' (actual translation 'dargestellt') and 'anxiety' (actual translation 'angst') the prediction does not present the actual translation as the word with the highest 't' parameter, yet the actual translation still comes up amongst the top 10 German words.

Thus it can be concluded that the most appropriate German word corresponding to an English word comes up in the top 10 words even though it might not be the one with the highest 't' parameter.

Compute Alignments for IBM Model 1

Based on the t(f|e) calculated for every (e,f), after 5 iterations of the EM algorithm for IBM Model 1, alignments have been compared for the first 20 sentence pairs of the training data. The generated alignments have been stored in ***'alignments4IBM1.txt'*** in the ***'output'*** folder. These computed alignments have been compared with the *'alignment_sample_model1.txt'* file in the current folder.

It is observed that *'alignment_sample_model1.txt'* had alignments for 10 sentences. So first 10 alignments generated in ***'alignments4IBM1.txt'*** were compared against *'alignment_sample_model1.txt'* and they were found to be a perfect match. The comparison was done using the *'ndiff'* function of the *'difflib'* library. The results of the comparison are in file ***'alignmentlog1'*** in the ***'output'*** folder.

Question 5:

Initialization of the 'q' parameters

Only possible (i, l, m) tuples, where l and m are length of each English and German sentence respectively have been considered while initializing 'q'. The 'q' parameters have been initialized as 1/(l+1). The initialized 't' parameters are the 't' parameters generated by the implementation of EM algorithm for IBM Model 1.

EM Algorithm for IBM Model 2

5 iterations of the EM Algorithm have been implemented for IBM Model 2. It takes approximately 3.5 minutes to execute the 5 iterations.

Compute Alignments for IBM Model 1

Based on the t(f|e) * q(j|i, l ,m) calculated for every (e,f), after 5 iterations of the EM algorithm for IBM Model 1, alignments have been computed for the first 20 sentence pairs of the training data.

The generated alignments have been stored in ***'alignments4IBM2.txt'*** in the ***'output'*** folder. These computed alignments have been compared against the *'alignment_sample_model2.txt'* file in the current folder.

It is observed that *'alignment_sample_model2.txt'* had alignments for 10 sentences. So first 10 alignments generated in ***'alignments4IBM2.txt'*** were compared against *'alignment_sample_model1.txt'*. The comparison was done using the *'ndiff'* function of the *'difflib'* library. The results of the comparison are in file ***'alignmentlog2'*** in the ***'output'*** folder.

This time some of the alignments differ. We can see in ***'alignmentlog2'*** that sentence number 5, 6 and 8 do not have a perfect match. Sentence 5, for example, shows better results after the IBM model 2 has been implemented. For sentence 5 *'alignment_sample_model2.txt'* shows all alignments as 1 (which is the same after implementing IBM Model 1) whereas IBM model 2 generates [1,2,1] which is a better result. Similar observations can be made for sentence 6 and 8 too.

Thus IBM model 2 is more computationally expensive and needs more running time but is a better model for translation as it generates better and more realistic alignments.

Question6

Finding Translations

Issues encountered

The original German transcripts and the scrambled English transcripts will have many (e,f) pairs that have not been encountered within the training data. Similarly, for every l(length of English sentence) and m(length of German sentence), several (i, l, m) tuples will be missing. All such missing values have been set to 'zero'.

Efficiency

It takes approximately 38 seconds to generate the unscrambled English transcripts after the IBM models have been implemented.

Both the models and the unscrambling would cumulatively take 6 minutes (probably less!!)

Accuracy

The accuracy of the unscrambled English transcript was evaluated using *'eval_scramble.py'.* **The calculated accuracy is 92%.**