ABOUT THE PROJECT

Our project aims at primarily implementing a Cross Lingual Document Translator, using Statistical Machine Translation model to be trained for alignment and translation. Statistical Machine Translation is an empirical machine translation technique using which translations are generated on the basis of statistical models trained on bilingual text corpora.

We were given two text files to train our model. One file had English text while the second file was the same text translated in Dutch. Taking the two files as our training data, we have implemented our own IBM model and Expectation Maximization (EM) algorithm without any external libraries in python, which outputs the results upon request. It generates a translation from English to Dutch and the other way around as well as per the user request.

The generated translation would then be compared with an accurate translation, to be given by the user, using performance metrics cosine similarity and Pearson's correlation coefficient. For this, we built a module which takes input two documents of the same language at a time and outputs the cosine similarity and Pearson's correlation coefficient for the two.

Our project comprises two codes. The first code named FinalCode.py, was used to train our model on the test data and generate a dictionary which maps a dutch word to its english translation. The second code named Interface.py is to handle the user request and output the result on the console.

COMPONENTS

INTERFACE.PY

This python function is run by the user to test our translation model. It comprises three functions, namely:

1. *readFile*
2. *parameterGeneration*
3. *Tester*

On running the program, tester is the first function which runs followed by readFile while parameterGeneration runs the last. All the functions are defined in detail below.

1. *readFile(file1, file2)*

This function takes 2 files as input arguments where,

file1 is the test file to be passed by user which can either be in English or Dutch.

file2 contains the actual output as desired by the user.

The function readFile reads both the files and splits them into their constituent sentences. Two empty lists are created and the sentences generated from file1 are put into one list while the sentences generated from file2 are put into the other.

Finally, the function returns the two lists, finalDoc1 and finalDoc2 which are passed as input parameters to the following function.

2. *parameterGeneration(finalDoc1, finalDoc2)*

This function takes 2 lists as input parameters where,

1. finalDoc1 contains sentence pairs obtained from translation by our model
2. finalDoc2 contains actual sentence pairs as required by the user

On getting the 2 lists, the function generates a bag of words model for both the documents to compute two scores:

a) jaccard coefficient
b) cosine similarity

using which, the user can test the accuracy of our model.

3. *tester(testFile, transDirection)*

This function takes 2 parameters as input where,

1. testFile is the file which the user wishes to test
2. transDirection is an integer to be taken as input from the user.

If the user wishes to translate from Dutch to English, value of transDirection should be 0. If it is any other integer then the user would get anEnglish to Dutch translation.

The function uses the dictionary created by our model to translate the given test file as per the user requirements. The translated text is then written into the file, translatedData.txt.

FINALCODE.PY

The main function of this code is to generate the two dictionaries,

1. English to Dutch
2. Dutch to English

It takes English_Updated.txt and Dutch_Updated.txt as inputs and removes all the punctuations except the apostrophe(‘) and splits the document into sentences. Next, it compares the corresponding sentences in the two documents and generates all possible alignments of the words and fills the probability matrix according to the EM(Expectation Maximization) algorithm. This is the implementation of IBM model 1. To improve the accuracy of our model we have also implemented IBM model 2.

INNOVATION:

1. To optimize the code it has been written in a numpy framework which enables faster execution hence, saving training time.
2. Further, due to computational constraints we have implemented batch training. The corpus was divided into batches of one and a half lakh sentences each.
3. IBM Model 2, which takes into consideration the positional probability of the Translation Model, has been implemented in the training code.