DOCUMENTATION

GROUP MEMBERS:
ANANYASHREE GARG 2015A7PS117P
HARSHIT OBEROI 2013B1A7865P
ANSHUL CHHABRA 2013B2A7803P

Introduction

In this assignment, we have implemented a Cross-Lingual Document Translator based on statistical machine translation model. It is used for converting documents in english to french or vice-versa. We then calculate Jaccard Coefficient and Cosine similarity to find how close is our result to the correct result. Following documents give a detailed description of the entire project. We were able to achieve a Cosine Similarity of 0.41 and Jaccard coefficient of 0.23 for French to English translation and a Cosine Similarity of 0.34 and Jaccard coefficient of 0.23 for English to French Translation. The best case achieved by us in the number of iterations we tried is 0.5 as cosine similarity for French to English translation and its corresponding Jaccard coefficient is 0.32(this was the max value of Jaccard coefficient). In English to French translation also this test case had the maximum similarity- 0.47(~0.5) cosine similarity and 0.32 Jaccard coefficient. These values have been very clearly depicted in the Test Cases table below.

Language- Java
Corpus- https://drive.google.com/open?id=0B6v0ngBbAZWvcUstV3Z4bWJLRHM

Outline

1) Pre-Processing
The size of the corpus was very large to be handled in one go by the default heap space allocated by eclipse (the compiler used by us). So first we had to increase the heap space of eclipse depending on the ram size of our computers. Then we processed the document in slots of 10000 lines each.After running the algorithm for 10000 sentences 20 iterations at a time, mapping of a word to the parallel language word is written in file. The match with maximum probability was chosen and file of all such correct alignments with their corresponding probabilities is made. There maybe multiple mappings of a single word to other language' words as the probabilities can be same after 20 iterations. This is the pre-processing required by us.

2) Statistical Machine Translation
A statistical model is trained for alignment and translated using the indices built in the previous step. We have used IBM Model and Expectation Maximization (EM) Algorithm covered during class.

3) Compute Similarity
A class is made for this whose functions takes 2 documents of the same language as parameters and computes their Cosine Similarity and Jaccard Coefficient. Detailed descriptions of these functions are mentioned below.

DESIGN DOCUMENT

The List of Libraries used

- java.io.BufferedReader;
- java.io.BufferedWriter;
- java.io.FileNotFoundException;
- java.io.FileReader;
- java.io.FileWriter;
- java.io.IOException;
- java.util.ArrayList;
- java.util.HashMap;
- java.util.StringTokenizer;
- java.io.*;
- static java.lang.Math.log;
- static java.lang.Math.pow;
- java.util.*;
- java.text.*;
- java.math.*;

Main class (Main.java)

In this class we run the IBM Model once and compile the result into a file so that we don't have to run the Model code again and again

DataStructures Used

HashMap
We have used HashMap to map each unique english or french word to an integer to reduce the space complexity.

HashMap of HashMap
While reading the text files made, there could be many mappings of an integer to many different integers . So to efficiently store them we have used hashmat of integer that maps to another hashmap which is a hashmat of integer versus double. where double would be the probability of mapping those particular words with each other.

ArrayList of ArrayList
We have used arraylist of array lists of integer to store each line in parallel corpuses. for each corpus.

Converter.java

In this class we compile the data gathered by running the corpus over size of 10000 lines each . We create a HashMap of HashMap for the english to french translations and for french to english translation . The IBM TM model was run only once and now then we created a file which has all the conversions of words .
If a word is mapped to same word multiple times in different text files then the probabilties are added of all in our data structure. At the end when all files are read, we take the mapping which has maximum value and that becomes our final mapping.

Cosine_similarity.java

1. **public static void ComputeSimilarity(String name_doc1, String name_doc2) throws FileNotFoundException,IOException**

This function is used for computing cosine similarity and Jaccard coefficient. The similarity is computed line-by-line and then average over all lines is calculated. Instead of calculating similarities over the entire document in one go, we calculated it line by line. This was done because even if all the lines in the 2 documents are same but their corresponding positions in the document are different it would give us false positives. Computing similarity over entire document would call it a perfect match, but going line by line would give no match- which indeed is the correct result.

2. public static float cosine_sim(String str1, String str2)

This function is mainly directed to calculating cosine similarity. The frequency of each term in 2 sentences is calculated. A 2-D matrix is made which has term frequencies of the corresponding sentences for which we are calculating the similarity. Their values are then converted to log (1+tf). Dividing them by their LNorms then normalizes these values. Corresponding final values for each term in 2 sentences are then multiplied and added to give the final cosine similarity value.

3. public static int intersect(String str1, String str2), public static int union(String str1, String str2)

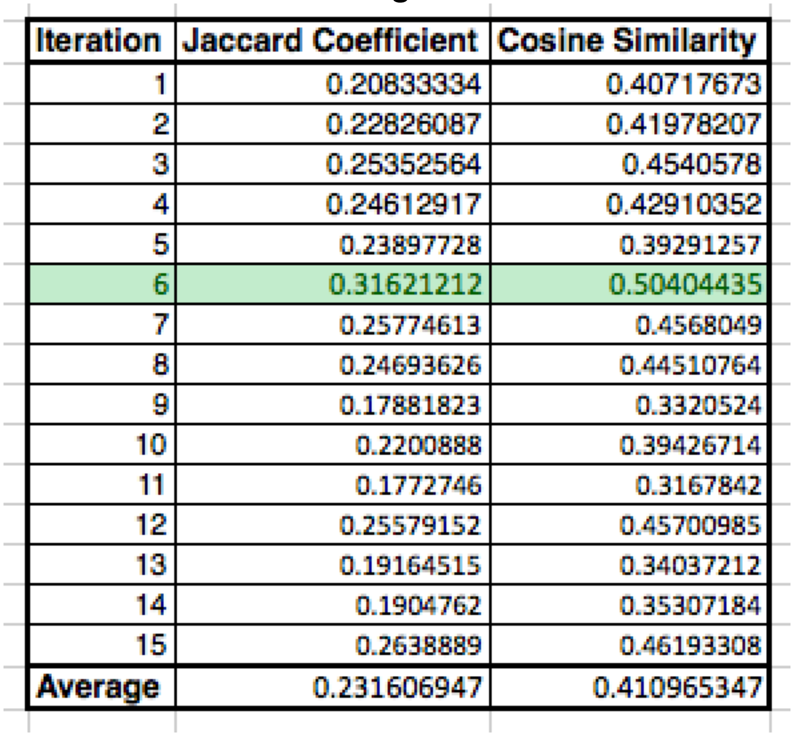
These functions are mainly directed to calculating Jaccard coefficient. Intersect function counts the number of common terms in 2 String passed as parameters. Union function counts the total number of terms in both the sentences after removing duplicates if any. Then these 2 values are then divided to give the Jaccard coefficient.

README FILE

This translator is used for language translation between two languages- French and English. After compiling and running the code you get an option to choose between languages- as to from which language do you want to translate? Translated language is automatically assumed to be the other one. Then you get an option to either continue with the process (1) or exit (0).
If you wish to continue next option you get is to enter the name of the file, which contains the sentences, you wish to translate (mention the entire path or just name of the file if it is in the same folder as the code). The sentences are translated and stored in another file in the same folder. This translated version is then compared with an already existing file in the same folder that contains the correct translation in term of Jaccard coefficient and cosine similarity.
Then you are given the option to continue (1) or exit (0). And the entire above process repeats.
A separate folder is made which contains all the processed files- mapping of English words to their corresponding French word according to the algorithm used along with their probabilities of matching. Consecutive files of all these maps are made each containing 10000 words in succession as there were memory constraints in mapping all words in one go (there were about 18 lakh lines in the parallel corpus). These files are taken from our dictionary and then used for translating the query document.

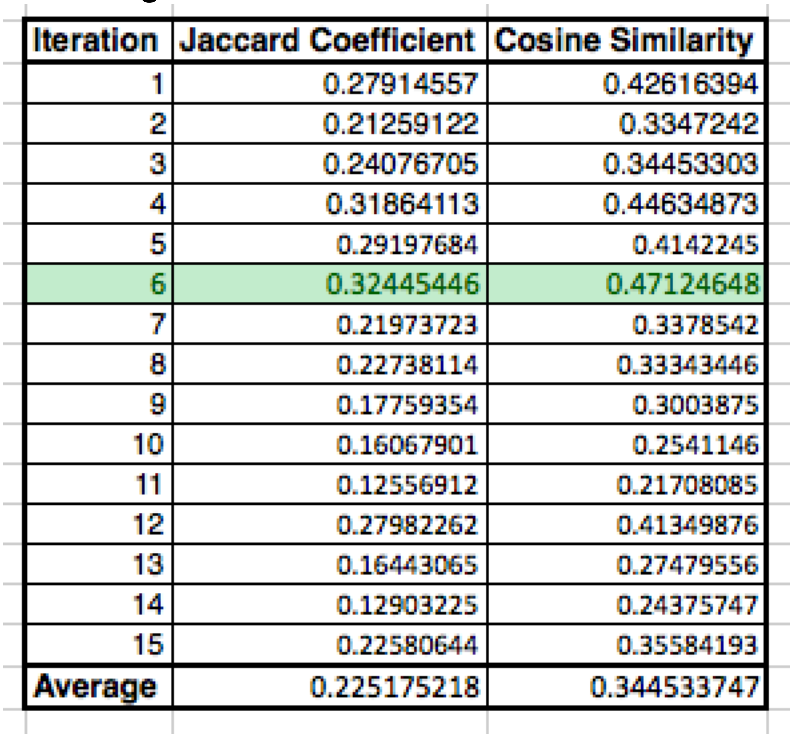
<u>TEST CASES</u>

French to English Translation



Iteration	Jaccard Coefficient	Cosine Similarity
1	0.20833334	0.40717673
2	0.22826087	0.41978207
3	0.25352564	0.4540578
4	0.24612917	0.42910352
5	0.23897728	0.39291257
6	0.31621212	0.50404435
7	0.25774613	0.4568049
8	0.24693626	0.44510764
9	0.17881823	0.3320524
10	0.2200888	0.39426714
11	0.1772746	0.3167842
12	0.25579152	0.45700985
13	0.19164515	0.34037212
14	0.1904762	0.35307184
15	0.2638889	0.46193308
Average	0.231606947	0.410965347

English to French Translation



Iteration	Jaccard Coefficient	Cosine Similarity
1	0.27914557	0.42616394
2	0.21259122	0.3347242
3	0.24076705	0.34453303
4	0.31864113	0.44634873
5	0.29197684	0.4142245
6	0.32445446	0.47124648
7	0.21973723	0.3378542
8	0.22738114	0.33343446
9	0.17759354	0.3003875
10	0.16067901	0.2541146
11	0.12556912	0.21708085
12	0.27982262	0.41349876
13	0.16443065	0.27479556
14	0.12903225	0.24375747
15	0.22580644	0.35584193
Average	0.225175218	0.344533747

The test cases shown above are in such a way that each index refers to corresponding translations- i.e. if a test case was tried for French to English, same test case was provided for English to French and their corresponding values were noted down in their respective tables.
It can be clearly seen that both the cosine similarity and Jaccard coefficient of French to English translation is more than that of English to French translation. This is because French has words with lot of accents, which English do not. French also has many words for one word in English- this could be other reason for this disparity. However when seen from bird's eye view the values are almost equivalent only.
The best case achieved by us has been highlighted and its corresponding test case has also been provided in this folder.

<u>LIMITATIONS</u>

Like any other project, this project has scope for improvements.

We are translating only between 2 languages; this code can later be easily extended to other languages also.

In English, adjectives come before the nouns whereas in French adjectives come after the nouns. Each noun in French has a gender (male, female, neutral) and each noun is also associated with an article (le, la, l', les) depending on gender and count (singular and plural); unlike English. So this can sometimes lead to inefficiency of the translator as such minute details are not captured. However, this problem is tackled up to some extent in the code, as this reordering of adjectives and nouns do not affect the Jaccard Coefficient and cosine similarity as they work on term frequencies rather than term positions. So although the translation may not be exact the Jaccard Coefficient and cosine similarity values will be correct.

In French there are many times where we do not have one word for one word in English; rather there are a group of words for one word in English- this would account for mapping problems in this translator.

Slight errors in the parallel corpus can also create problems. E.g. in English we have “Mr Berenguer Fuster, we shall check all this.” but it's corresponding french translation is “Cher collègue nous allons vérifier tout cela.” which although correct semantically but according to the translator would give less values of Jaccard coefficient and Cosine similarity.