DESIGN DOCUMENT

GROUP NO: 3

DISHITA MALAV: 2017A7PS0164P

IESHAAN SAXENA: 2017A7PS0180P

KUSHAGRA RAINA: 2017A7PS0161P

NITIN VINAYAK AGRAWAL: 2017A4PS0415P

SHEFALI TRIPATHI: 2017A7PS0139P

TABLE OF CONTENTS

INTRODUCTION

We have implemented a Cross Lingual Document Translator, using Statistical Machine Translation model. The statistical model, IBM Model 1, has been trained for alignment and translation. Performance metrics such as cosine similarity and Pearson's correlation coefficient, have also been implemented in the translator.

ASSUMPTIONS

1. Taking into consideration the large size of the data set, as a part of preprocessing the documents, we have:
   - Removed all punctuation marks and replaced them with "".
   - Converted all Uppercase characters to Lowercase for uniformity.
   - Removed all numerals and replaced them with "".
   - Removed a list of stop-words.
     List of Stop Words for English: ['of', 'the', 'I', 'on', 'and', 'would', 'to', 'you', 'a', 'in', 'that', 'as', 'have', 'for', 'be', 'from', 'it', 'at', 'can', 'an', 'has', 'The', 'It', 'is', 'not', 'with', 'We', 'by', 'This', 'we', 'are', 'more', 'our', 'or', 'also', 'these', 'but', 'must']
     List of Stop Words for Dutch: ['van', 'de', 'ik', 'Aan', 'en', 'Zou', 'naar', 'u', 'een', 'in', 'dat', 'als', 'hebben', 'voor', 'worden',

'van', 'het', 'Bij', 'kan', 'een', 'heeft',
'De', 'Het', 'is', 'niet', 'met', 'Wij',
'door', 'Deze', 'wij', 'zijn', 'meer', 'onze',
'of', 'ook', 'deze', 'maar', 'moet']

2. An SQL Database is created to store translational probabilities for mapping English to Dutch (Foreign Language). The Database Columns are:

ENG_WORD	DUT_WORD	PROBABILITY

Primary Key: The composite key of ENG_WORD and DUT_WORD.

The Clustered Index is based on the Primary Key.

LIMITATIONS

Space and Time were the two major limitations in the project.

1. Due to the large size of the dataset, we found that storing the translational probabilities, counts and other metadata related to the datasets, in data structures implemented in python (like a dictionary) was highly space consuming. Hence, we stored them in files and SQL Databases.
2. The EM Algorithm used in training the IBM Model also takes a significant amount of time. In order to reduce the time taken, we used SQL Database in which reduced the time taken in seeks. Stop-words were also removed in order to reduce the amount of time taken in processing these commonly occurring words.

INNOVATION:

The large size of the dataset increases the numbers of seeks, in order to make the model most time and space efficient we used an SQL Database to store the translational probabilities t(e|f). We also used a clustered index in the database to further reduce the retrieval time.

ALGORITHMS AND FORMULAE UTILIZED:

1. Expectation-Maximization Algorithm:

```
initialize t(e|f) uniformly
 do until convergence
   set count(e|f) to 0 for all e,f
   set total(f) to 0 for all f
   for all sentence pairs (e_s,f_s)
     set total_s(e) = 0 for all e
     for all words e in e_s
       for all words f in f_s
         total_s(e) += t(e|f)
     for all words e in e_s
       for all words f in f_s
         count(e|f) += t(e|f) / total_s(e)
         total(f)   += t(e|f) / total_s(e)
   for all f
     for all e
       t(e|f) = count(e|f) / total(f)
```

2. Pearson Coefficient Calculation:

$$sim(a,b) = \frac{\sum_{p \in P}(r_{a,p} - \bar{r}_a)(r_{b,p} - \bar{r}_b)}{\sqrt{\sum_{p \in P}(r_{a,p} - \bar{r}_a)^2}\sqrt{\sum_{p \in P}(r_{b,p} - \bar{r}_b)^2}}$$

3. Cosine Similarity Calculation:

$$\cos(\vec{q},\vec{d}) = \vec{q} \bullet \vec{d} = \sum_{i=1}^{|V|} q_i d_i$$

DESCRIPTION OF VARIOUS MODULES IMPLEMENTED:

1. Function: readfiles

   Description: Function opens the source files for English and Dutch languages, loads all their sentences to return variables ‘eng’ and ‘dutch’ respectively.

   Input Variables:

   Eng_File: Name of the file containing English Corpus.

   Dutch_File: Name of the file containing Dutch Corpus.

   no_of_sentences: Total number of sentences in the Corpus’.

   Return Variables:

   eng: List of all English sentences.

   dutch: List of all Dutch sentences.

2. Function: remove_punc

Description: Function calls remove_stopwords and remove_stuff to remove stopwords and punctuation, sentence by sentence, respectively.

Input Variables:

l: List of English or Dutch sentences

lang: Name of the language passed.

Return Variables:

l: Containing all the sentences without the stop-words and punctuation.

3. Function: remove_stopwords

   Description: Removes stop-words in a sentence.
   Input Variables:

   eng_sw: List of English stop-words.

   dut_sw: List of Dutch stop-words.

   lang: Name of the language the input sentence belongs to.

   l: A single English or Dutch sentence.

   Return Variables:

   l: Sentence without stop-words.

4. Function: remove_stuff
   Description: Removes all punctuation in a sentence.
   Input Variables:
   l: A single English or Dutch sentence.
   Return Variables:

l: Sentence with punctuations removed.

5. Function: assign_line_no

   Description: Creates an inverted index where each word is matched to all the lines it is present in.

   Input Variables:

   doc: Contains list of the cleaned sentences of either Dutch or English (i.e. without the punctuations or stop-words).

   Return Variables:

   dict_lo: Contains the inverted index (in the form of a dictionary) of all the words in the document.

6. Function: initialize

   Description: Creates Dutch-English word pairs then initializes their probabilities. It then writes then in an SQL Database.

   Input Variables:

   foreign_no_of_words : Total number of Dutch words.

   foreign_l: All Dutch words.

   english_l: All English words.

   Return Variables:

Returns True.

7. Function: finding_probabilities
   Description: Updates count variable for each Dutch-English word pair.
   Input Variables:
   dutch_sentences: List of clean Dutch sentences retrieved
   eng_sentences: List of english sentences retrieved
   no_of_sentences: Total number of sentences
   total: Normalized fractional counts
   Return Variables:
   total:Normalized fractional counts of every dutch-english word pair.

8. Function: running_function

   Description: Function repeatedly calls finding_probabilities to update count for every Dutch-English word pair.

   Input Variables:

   foreign_l: List of all Dutch

   words.

   english_l : List of all English words.

   dutch_sentences : List of all clean Dutch sentences retrieved

eng_sentences : List of all clean English sentences retrieved

no_of_iterations : Number of times the count is to be updated.

Return Variables:

None.

9. Function: retrieve_max

Description: Function to retrieve the translaton with maximum probabilities for each word to be translated, i.e. to obtain the most probable translation for each word

Input Variables:

num_dict_dutch : Dictionary of Inverted index of all Dutch words

num_dict_eng : Variable which stores line number of English words.

Return Variables:

translation_etof: Translation from English to Dutch.

translation_ftoe: Translation from English to Dutch.

10. Function: pearson_coefficient

Description: Calculates the Pearson coefficient. It calculates normalized weight of individual sentences by calling maintain_count().

Input Variables:

dutchWord_line_no: Dictionary of Dutch words mapping to individual line numbers.

dut_cleaned: Dictionary of cleaned Dutch lines.

result_dut: List of lines produced by our model.

Return Variable:

Returns Pearson Coefficient between two Documents

11. Function: cosine_similarity

Description: Function calculates cosine similarity. It calculates normalized weight of individual sentences by calling maintain_count().

Input Variables:

dutchWord_line_no: Dictionary of Dutch words mapping to individual line numbers.

dut_cleaned: List of cleaned Dutch lines.

result_dut: List of lines produced by our model.

Return Variables:

Total_sim: Cosine similarity between two dutch lines.

12. Function maintain_normalized_tf:

Description: Function to find normalized T_f weights of a Dutch sentence with a sentence produced by our model.

Input Variables:

doc_dict_line_no: Single Dutch sentence.

data_clean: List that contains Dutch sentences produces by the IBM model.

Return Variables:

og_dutch_tf: Normalized T_f weights 2 sentences at a time.

13. Function: produce_sentence

Description: Translates an English sentence to Dutch and adds the punctuation to form a complete sentence.

Input Variables:

eng_sentence: A single English sentence.

translated_dict: Dictionary of Dutch translation of all English words.

doc_lang: Variable gives user option to either translate English to Dutch or vice-versa. Default is English to Dutch.

Return Variables:

s: Dutch translation of an English sentence.

14. Function: translate_doc

Description: Function calls produce_sentence repeatedly to translate the entire English document to Dutch.

Input Variables:

eng_doc: Variable contains either Dutch or English list of sentences.

translated_dict: Variable contains Dutch translation of all English words.

rewrite_file: The file in which all the translated words are stored.

doc_lang: Variable gives user option to either translate English to Dutch or vice-versa. Default is English to Dutch.

Return Variables:

result_doc: List of all translated sentences.

15. Function: delete_files

Description: Deleting previously present files responsible for maintaining translational probabilities and count.