Cross Lingual Document Translator: Design Document

Submitted By

Vedant Patwary, 2016A7PS0031P
Anupam Aggarwal, 2016A7PS0033P
Rahul Khandelwal, 2016A7PS0128P
Satyavrat Sharma, 2016B4A70322P
Nikita N. Singh, 2016B4A70509P

Link to IBM Model English to Dutch & Dutch to English
Link to Complete Assignment on Drive

Corpus

Bilingual Text Corpora with sentence pairs in English and Dutch Language for Statistical Machine Translation.

Model

IBM Model with further enhancements trained to predict text translations from English Language to Dutch Language and vice-versa. Steps Involved to complete this task:

Data Sampling

Owing to the huge data availability, we filtered out sentences with word count between 3 to 8 only. Then, random selection of 1 lakh sentences amongst the filtered sentences was done in order to train the model. This helped us to improve accuracy as longer sentences tend to exponential increase in possible number of alignments.

Text Cleaning

Implemented in `processSentence(st)`
Input: text obtained from the two files

Return: processed/cleaned text

1. Special Characters are removed and replaced with space character ‘ ’.
2. Text is completely converted into lowercase.

Example:

Given: Fundamental rights and citizenship (2007-2013) (vote)

Transforms to: Fundamental rights and citizenship 2007 2013 vote

Reasons:

1. ***Different forms of a word due to special characters:*** This allows us to treat ***(vote)*** & ***vote*** not as two different words, but just the same word ***vote*** during training. Hence, avoiding loss of information.
2. ***Conjunction of more than one meaningful words:*** Here, removal of hyphen (and in general, removal of special characters) leads to better training results because we are able to break single word ***(2007-2013)*** into ***2007*** and ***2013*** which is more meaningful.
3. ***Uppercase words leading to information loss:*** Here, ***Fundamental***, if not converted into lowercase ***fundamental***, will lead to two possible words during the training of model, however, we would prefer them to be treated as same word semantically. Hence, we choose to convert it into ***fundamental*** in order to facilitate training.

<u>Word Dictionary</u>

Implemented in `wordDict(listData)`

Input: text (in source/target language)

Return: text (processed), word dictionary (in same language)

Depends on: `processSentence(st)` in order to clean text

After cleaning of text, sentences are then processed and each new word encountered is saved into the dictionary. This allows us to maintain a dictionary of unique words used in English and Dutch language (source and target language in general).

<u>Training Model</u>

Implemented in `trainModel(engData,fnData,engDict,fnDict)`

Input: the English and Dutch sentences along with their respective vocabulary.
Return:depending on whether we want English(Dutch) to Dutch(English) translation or vice versa- probability of each possible translation of English(Dutch) words to Dutch(English) words considering all the alignments of each pair of sentences after each iteration.

This function incorporates the implementation of IBM model in our program. Initially the ***probability of each alignment possible*** is calculated using the probability of translation of any English(Dutch) word to any Dutch(English) word. The probability of each alignment is normalized. The normalized alignment probabilities are further used to update probability of all the possible translation of English words to Dutch words.These translation probabilities are normalized too. At the end of each iteration, the probability of alignment tends to converge and allows us to reach to a solution. This happens due to continuous application of **Expectation Step** [assigning hidden values to complete the model] and then **Maximisation Step**[using statistics (in terms of number of occurrences) to estimate the model].

These updated probabilities are saved via `np.save('data/IBM_Model_dut_to_eng',probDutToEng)` so that it could be used in the next iteration. As we near the end of the training process, it is observed that the correct translations' probability converges to 1 and the incorrect ones converges to 0. This data can be later used to predict the translations from source text to target text by taking the **maximum argument** (alignment with maximum probability given the source).

<u>Translation</u>
In the beginning, saved data (dictionaries and word translation probabilities) is loaded. Then, each word from dictionary of source language is mapped to a particular word of target language by taking

maximum argument of probability value of transition from given source word to all possible target words.
Implemented in: `translateDutToEng(st)` & `translateEngToDut(st)`
Input: Text Document to be translated (source text)
Return: Text Document (translated) (target text)
Depends on: Training Process & `processSentence(st)`
File path to the source file is taken as input by the user, contained text is passed in the above-mentioned function. This text is processed (same way as in text cleaning, using `processSentence(st)`) and word mappings mentioned above are used to obtain translated document.

<u>Cost/Similarity Measures</u>
We used **Jaccard Coefficient** and **Cosine Similarity** as the two similarity measures in order to determine the accuracy of prediction (wrt original text).
Jaccard Coefficient
Implemented in `jaccCoeff(ori,trans)`
Input: Expected Translation , Translated Text(using model)
Return: Calculated Jaccard Coefficient
Computes similarity between two text documents as ratio between ***number of common words*** and ***total number of distinct words***. However, this is not a good measure of similarity as it does not consider the number of occurences of words.

Cosine Similarity
Implemented in `cosDistance(ori, trans)`
Input: Expected Translation , Translated Text(using model)
Return: Calculated Cosine Similarity
Depends on: `textToVector(text)`
Computes similarity between two text documents as cosine of angle between vector representation of common words which is calculated by taking dot products and normalizing it by L2 norm.