**Software Design Specification**

**CS F469- Assignment 3-**

**IBM Alignment Models**

**Rohan Tammara- 2016A8PS0332H**

**Vaibhav Balloli- 2016AAPS0165H**

**Karthik Nagaraj- 2016A7PS0804H**

**1. Introduction**

This project is based on the IBM 1 & 2 alignment models and phrase-based translation.

(Github Repository link: <https://github.com/vbbphc/IBM-Alignment> )

**1.1 Purpose of this document**

This document has been created to give a pre-implementation design outline for the application.

This document will contain all the concepts and packages used along with a brief description.

**1.2 Packages Used**

**NLTK**

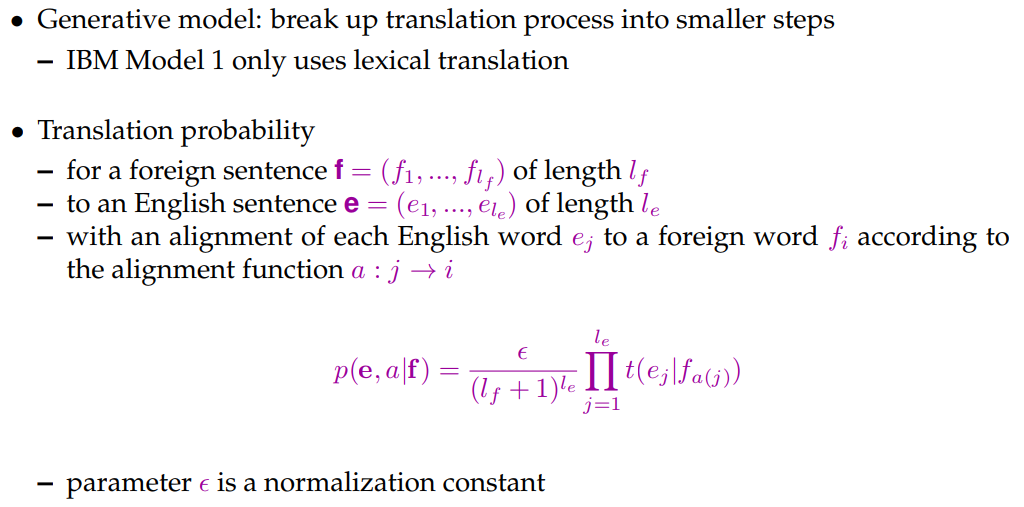
NLTK is a leading platform for building Python programs to work with human language data. It provides a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, and various other language processing capabilities.

**JSON**

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. JSON is built on two structures: A collection of name/value pairs. (In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array.) and an ordered list of values. In most languages, this is realized as an array, vector, list, or sequence.

**1.3 Definitions**

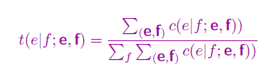
**IBM Model 1**



IBM Model 1 utilizes the EM model. (Estimation Maximization.) In this model, initially, all translations are initialized to an equal probability. The model is then fed the corpus iteratively until convergence.

IBM Model 1 is weak in terms of conducting reordering or adding and dropping words. In most cases, words that follow each other in one language would have a different order after translation, but IBM Model 1 treats all kinds of reordering as equally possible. Another problem while aligning is the fertility (the notion that input words would produce a specific number of output words after translation). In most cases one input word will be translated into one single word, but some words will produce multiple words or even get dropped (produce no words at all).

After applying this and simplifying, the IBM Model 1 with EM is:

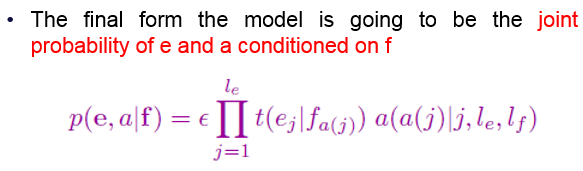


**IBM Model 2**

The IBM Model 2 addressed this issue by modeling the translation of a foreign input word in position i to a native language word in position j using an alignment probability distribution defined as:

C:\Users\Karthik Nagaraj\Desktop\ibm2.PNG

Translation in the IBM Model 2 can be viewed as a two-step process. Lexical Translation, modeled as t(e/f), followed by the alignment step.



**Phrase Based Models**

Phrase-Based Models translate phrases as atomic units. Foreign input is segmented into phrases. Each phrase is translated into English. Phrases are reordered. It is advantageous as words may not be the best atomic units for translation, due to frequent one-to-many mappings. Translating word groups instead of single words helps to resolve translation ambiguities. If we have large training corpora, we can learn longer and longer useful phrases, sometimes even memorize the translation of entire sentences.

**1.4 References**

* [https://www.nltk.org/api/nltk.translate.html#module-nltk.translate.phrase\_based](https://www.nltk.org/api/nltk.translate.html" \l "module-nltk.translate.phrase_based)
* <https://en.wikipedia.org/wiki/IBM_alignment_models>
* <https://docs.python.org/3/>

**2 Execution**

**2.1 System Requirements:**

Python3 with nltk installed.

**2.2 Executing the application**

To run the application, simply run the ‘run.sh’ script. It will check for all dependencies, including the dataset and the libraries, install or update them if need be, and then run the application which will output the results.

**2.3 Run Times**

**Q1) 0.031s**

**Q2) ~0.001s**

**Q3) 0.312s**

**2.4 Alignments**

**"en": "the car",**

**"fr": "La voiture"**

**(1-1, 2-2)**

**"en": "the season",**

**"fr": "La saison"**

**(1-1, 2-2)**

**"en": "blue car",**

**"fr": "Voiture bleue"**

**(1-2, 2-1)**

**"en": "blue house",**

**"fr": "Bleu maison"**

**(1-1, 2-2)**