Design Document

CS F469 IR ASSIGNMENT - 3

# Machine Translation

# - IBM Model 1 and EM Implementation

# - Comparison of IBM1 and IBM2 models

# - Phrase based extraction and scoring

…………………………………………………………………………………………………………………………………………………………….

2016A7PS0006H R MONITH SOURYA

2016A7PS0033H SANJAY DEVPRASAD

2016A7PS0049H NIKHIL SREEKUMAR NAIR

2016A7PS0086H N VISHNU TEJA

# **Custom Corpus Lang - Dutch**

We used a small set of 8 sentences in dutch to compare all possibilities and easily identify the errors and explain the results.

**True Alignment:**

1: [(0, 0), (1, 1), (2, 2), (3, 3), (4, 4)]

2: [(0, 0), (1, 1), (2, 2)]

3: [(0, 0), (1, 1), (2, 2), (3, 3)]

4: [(0, 0), (1, 1), (2, 2), (3, 3), (4, 4), (5, 5)]

5: [(0, 0), (1, 1), (2, 2), (3, 3), (4, 4), (5, 5)]

6: [(0, 0), (1, 1), (2, None), (3, 3), (4, 4), (5, 5), (6, 6)]

7: [(0, 0), (1, 1), (2, 2), (3, 3), (4, 4)]

8: [(0, 0), (1, 1), (2, 2), (3, 3)]

# **Packages used**

* **numpy:** NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
* **nltk:** The tokenize and translate modules of nltk were directly used wherever required.
* **Collections:** we used this model for efficient use of data structures.
* **math:** for calculating basic mathematical expressions.

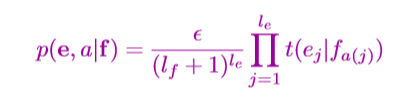
# **Formulas**

**IBM Model 1**

Generative model: break up translation process into smaller steps – IBM Model 1 only uses lexical translation

Translation probability

* for a foreign sentence f = (f1,...,flf) of length lf
* to an English sentence e = (e1,...,ele) of length le
* with an alignment of each English word ej to a foreign word fi according to the alignment function a : j → i



**EM Algorithm**

Incomplete data

* if we had complete data, would could estimate model
* if we had model, we could ﬁll in the gaps in the data

Expectation Maximization (EM) in a nutshell

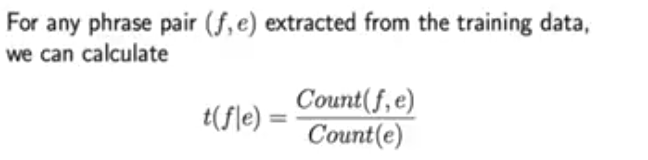
1. Initialize model parameters (e.g. uniform)

2. Assign probabilities to the missing data

3. Estimate model parameters from completed data

4. Iterate steps 2–3 until convergence

**Phrase-ranking**



**Running Time**

1.py - 0.02s

2.py - 0.02s

3.py - 0.03s

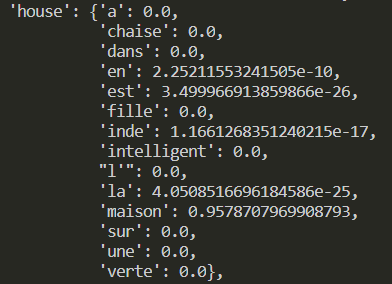
# **Observations, inferences and other findings**

The IBM Model 2 does better than IBM1 as expected. It predicts with a higher accuracy compared to IBM Model 1.

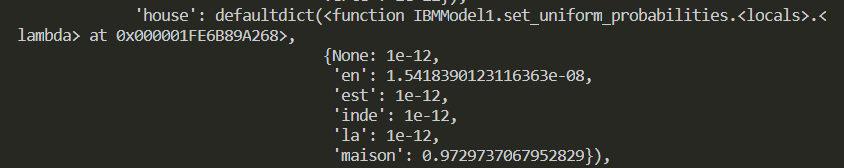
**Eg:-**

**French**

IBM Model 1(Custom)

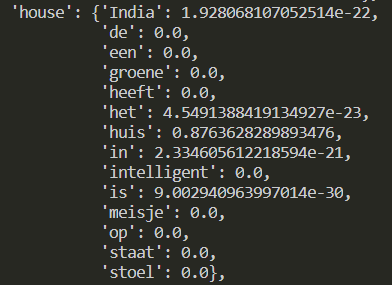


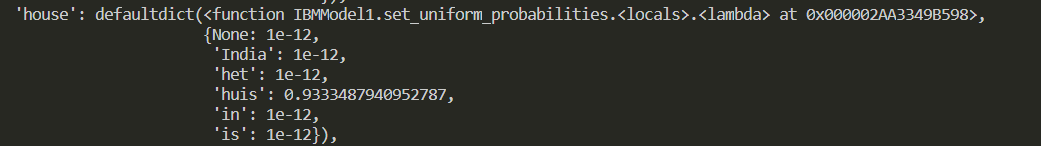
IBM Model 1 (NLTK)



The nltk implantation showed very similar results to our custom implementation and there was only a difference in accuracy maybe due to different epsilon values in our EM algorithm other than that the alignments are identical.

**Dutch**

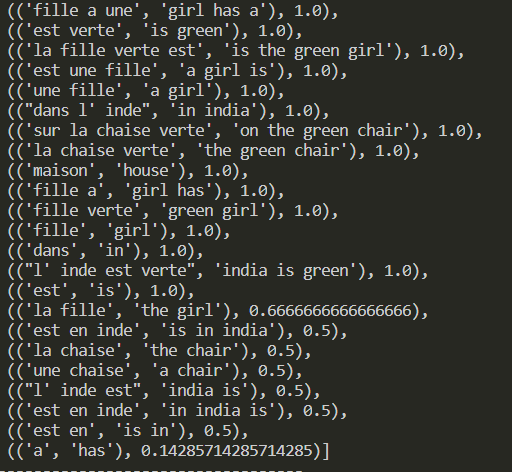




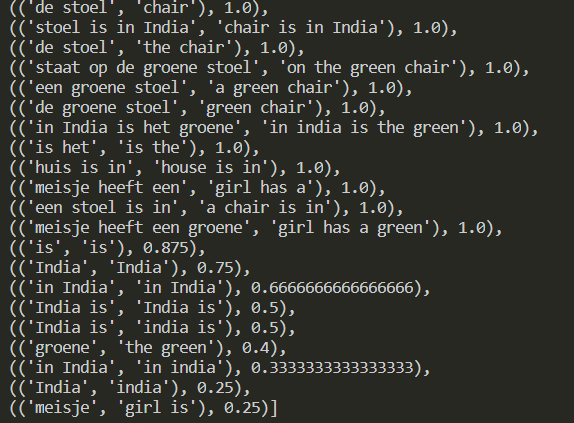
The nltk implantation showed very similar results to our custom implementation and there was only a difference in accuracy maybe due to different epsilon values in our EM algorithm other than that the alignments are identical

**3.py**

**Example (a few phrases) - French**



**Example (a few phrases) - Dutch**



The phrases extracted are ranked based on the phrase–ranking formula given.

Since the corpus is small and the phrases are aligned very well by the algorithm, most of the phrases have a rank 1