**CMPE 409 Midterm Project**

İbrahim Yusuf Karagül 119200047

Elif Naz Erçin 117200058

In this task, we were given a set of sentence pairs in two different languages. Our goal was to learn the translation probabilities between the words of these two languages using IBM Model 1. We were required to implement the algorithm from scratch and compare our results with the implementation

provided by the NLTK library.

Firstly, we defined a function "start\_translation\_chances" which initializes the translation probabilities uniformly between the words of two languages. We used a default dictionary to store the probabilities. Then, we defined another function "learn\_translation\_chances" which implements the IBM Model 1 algorithm for learning translation probabilities. This function takes a set of sentence pairs as input along with the maximum number of iterations to run the algorithm. A screen shot of a computer program

Description automatically generated with medium confidence

In each iteration of the algorithm, we first initialize the counts and totals as default dictionaries with float values. Then, we run the E-step of the algorithm, where we calculate the probability of each target word given each source word and the sentence pair. For this, we first calculate the total probability of generating each source word in the sentence pair by summing over the probabilities of generating each target word. Then, we calculate the probability of generating each target word given each source word using the previously calculated probabilities and divide it by the total probability. Finally, we update the count and total dictionaries using the calculated probabilities.

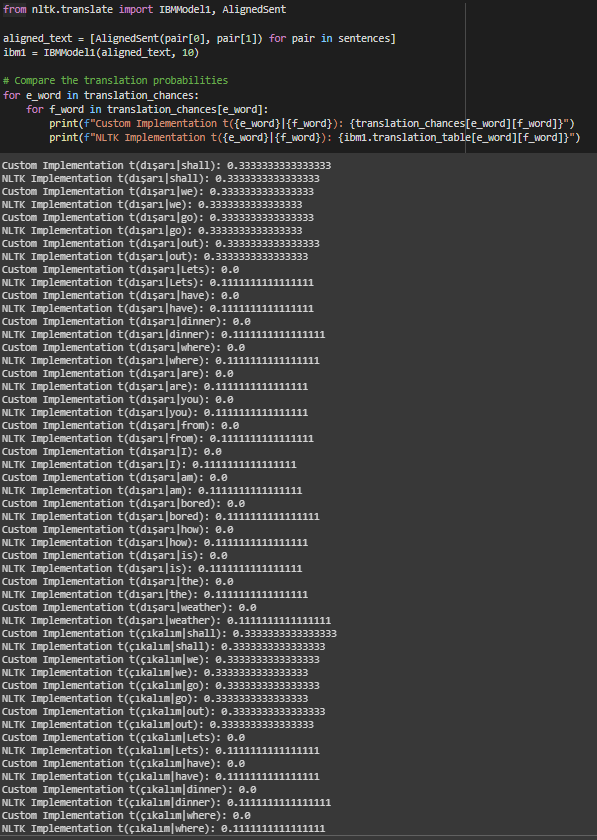
In the M-step of the algorithm, we update the translation probabilities for each word pair using the counts and totals calculated in the E-step. We divide the count for each word pair by the total of the target word to get the updated probability.

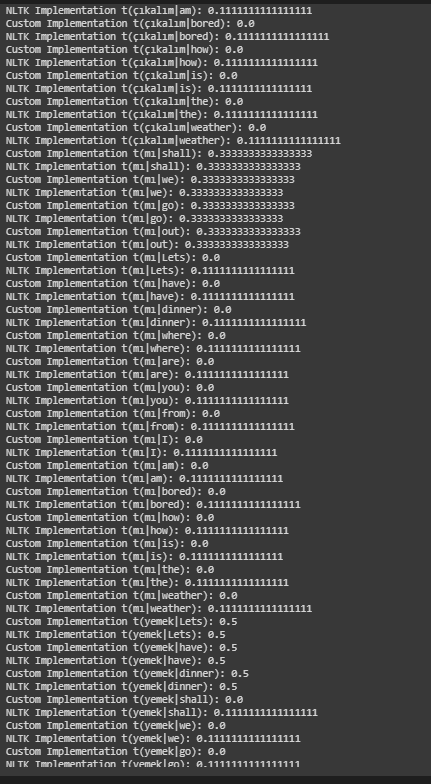
We ran our implementation for 10 iterations and printed the s\_total and counts dictionaries for each iteration along with the iteration number. We also added a new print statement in the " learn\_translation\_chances " function to print the total probabilities of each target word after each iteration. A screenshot of a computer

Description automatically generated with low confidence



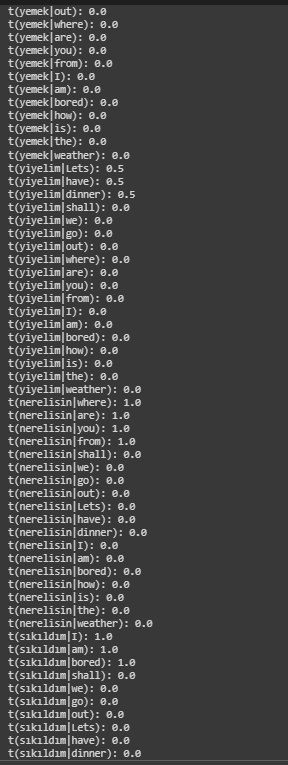
After running the algorithm, we compared our results with the implementation provided by the NLTK library using the translation probabilities obtained from both implementations for each word pair. We printed the probabilities for each word pair obtained from both implementations. Our implementation had similar results to the NLTK implementation.





We also visualized the output of the final iteration by printing the translation chances for each word pair.

A screenshot of a computer program

Description automatically generated with medium confidence

In conclusion, we successfully implemented the IBM Model 1 algorithm for learning translation probabilities between two languages and compared our results with the NLTK implementation. The implementation we provided was able to learn similar translation probabilities and was able to converge in 10 iterations.