Itamar Trainin
315425967

# Ex. 1 – Word Alignment: Report

In this exercise I have implemented IBM models 1 and 2 and experimented with different parameter to understand how they influence the results.

## Part 1:

In the first part I have trained the original models with no special configuration. I have achieved the following results:

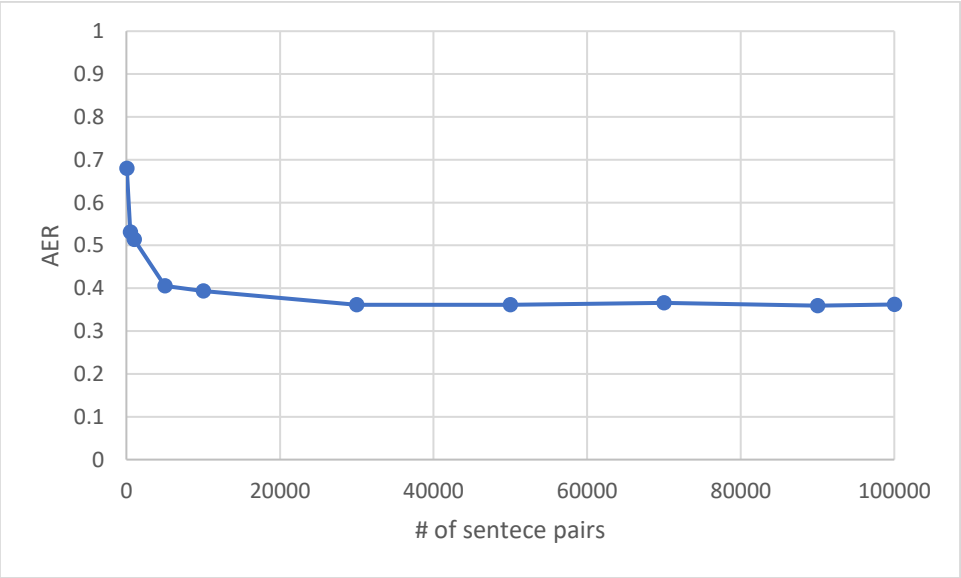
| Model | AER | Precision | Recall |
|---|---|---|---|
| **IBM Model 1 (50 itr.)** | 0.360753 | 0.587183 | 0. 742603 |
| **IBM Model 2 (50 itr.)** | 0.296506 | 0.651872 | 0.813609 |
| **IBM Model 2 + pre-train (50 itr.)** | 0.288951 | 0.657420 | 0.825443 |

It can be seen from the results that IBM Model 2 greatly improves the results over IBM Model 1, an additional improvement is achieved by initializing the parameters of model 2 using the output parameters of model 1.

## Part 2:

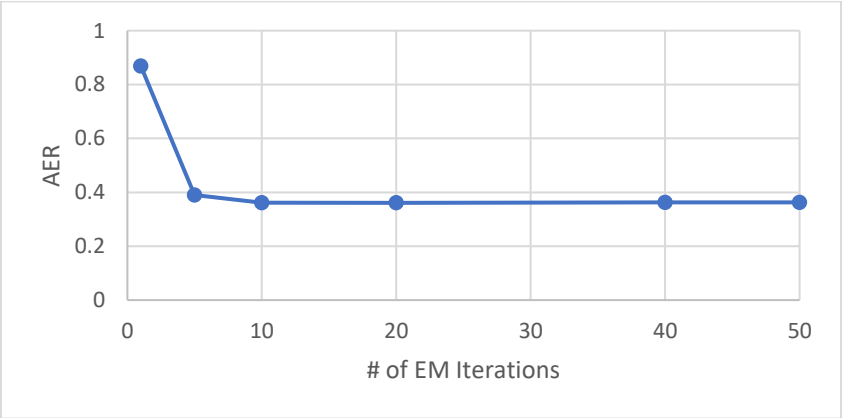
In the second part, I have experimented with different configurations in order to test the influence of the parameters on IBM Model 1.

A. To begin with, I have experimented with the influence of the **number of sentence pairs** i.e. the size of the dataset on the *AER*. To perform this experiment, I have trained IBM Model 1 (for 50 EM iteration) multiple times, each time using the same initialization, however limiting the number of sentence pairs.
The following graph is showing the AER as a function of the number of sentence pairs used. It can be concluded from the graph that the first sentences are more crucial for better results, however **a correlation between high number of sentences to a lower AER is clear**.

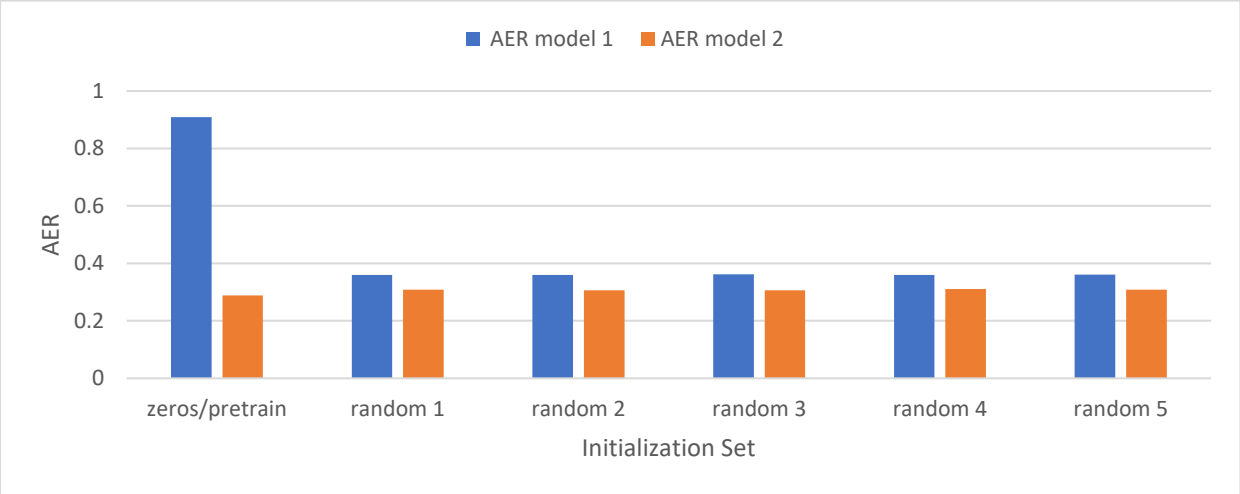


Itamar Trainin
315425967

B. Next, I have experimented with the influence of the **number of EM iterations** on the AER. To perform this experiment, I have trained the model multiple times, each time using the same initialization and all the available data, changing the number of EM iterations each time.
The following graph is showing the AER as a function of the number of EM iterations used. From the graph we see **that the model converges after about 20 iterations, that is, there is almost no improvement afterwards**.




C. In this experiment, I have tested the influence of **initialization** on the AER for both Model 1 and 2. To perform this experiment, I have trained the model multiple times, each time initializing the parameters to different random values. In addition, I have experimented with initializing the parameters to zero in Model 1 and initialize Model 2's parameters to Model 1's output parameters.
In this experiment I have used 50 EM iterations and all the available data. In the supplementary paper '*Improving IBM Word-Alignment Model 1, by Robert C. Moore*', it was strongly assumed that better results can be achieved by using better initialization for the parameters, however I did not apply their method in this part.
The following graph is showing the AER induced by the different initializations sets.




From the results I can deduce the following conclusions:

a. Initializing the parameters to zero performs poorly.
b. The different random initializations don't have much influence on the results.
c. Model 2 performs similarly, where pretrained parameters obtain slightly better results.

Itamar Trainin
315425967

D. In this experiment, I have tested the influence of the **direction of translation** on the AER. I have experiment with setting the source language to English and the Target language to French and I have received **lower results**:

AER - **0.503503** / Precision - **0.429652** / Recall - **0.627218**

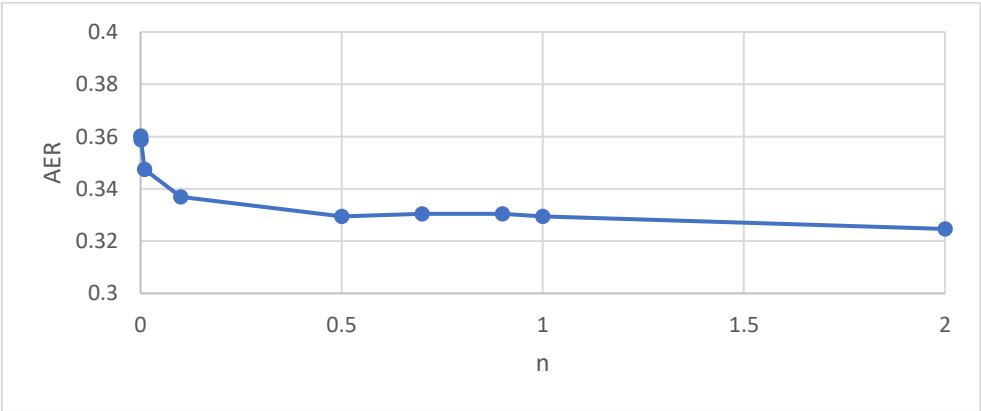
I believe that these poor results due to the test alignments provided in the test data. Those alignments were generated in the opposite direction (French to English alignments) and therefore are not as good when applied in the opposite direction without any review. However, one could build a combined model that uses both directions to improve the results. This was not tested here.

E. Lastly, I have referred to the paper '*Improving IBM Word-Alignment Model 1, by Robert C. Moore*', where multiple methods are suggested to improve the AER of IBM Model 1. The methods included:
   a. A way to increase the probability of NULL word alignment by adding more NULL word to each target sentence.
   b. Decreasing the likeliness of rare word alignments by smoothing the pair-probabilities ($t$) on each EM iteration
   c. Using a better initialization method for the model parameters.

The results I have achieved on the original IBM Model 1 where not far from the results achieved in this paper. The yet existing gap may have resulted from the difference in the amount of data used. The paper's author used 5 times more sentence pairs ($500000$) for training than I had available ($100000$).
I have chosen to implement the second method ('*Smoothing Translation Counts*') suggested in the paper to see if I could improve my results as well. The paper suggested to use the following equation when updating the parameters after each EM iteration:

$$tr(t|s) = \frac{C(t,s) + n}{C(s) + n \cdot |V|}$$

where $|V|$ is the target vocabulary size and $n$ is the smoothing coefficient.
To perform this experiment, I used the all the available data and trained for 50 EM iterations. The graph below shows the AER as a function of $n$:




We can see that applying this modification **does** in fact greatly improves the result as was discovered in the paper.

Itamar Trainin
315425967

**Part 3:**

To summarize, we have seen that the model benefits from more data, it converges rather fast, and that any random initialization (not zero matrix) is ok. In addition, we have seen that the 'smoothing' method used in the supplementary paper was efficient.

I have used all conclusions to train the best model. The parameters can be found in the attached files as well as the corresponding alignments. The scores achieved for those configurations are specified below:

| Model | AER | Precision | Recall |
|---|---|---|---|
| **IBM Model 1 (50 itr, n=2)** | 0.323643 | 0.629682 | 0.772189 |
| **IBM Model 2 (50 itr, n=2)** | 0.295561 | 0.653259 | 0.813609 |
| **IBM Model 2 + pre-train (50 itr, n=2)** | 0.288951 | 0.657420 | 0.825443 |

These results could have been farther improved by adding more data and applying the remaining two methods mentioned in the supplementary paper.

Itamar Trainin
315425967

## Commands used to construct this report:

```
python WordAlignment.py --ep 50 -m 1
python WordAlignment.py --ep 50 -m 2
python WordAlignment.py --ep 50 -m 2 -t data/t_init_model_1.npy

# Number of sentnences
python WordAlignment.py --ep 50 -m 1 --ll 50 -o -t data/t_init_random.npy
python WordAlignment.py --ep 50 -m 1 --ll 500 -o -t data/t_init_random.npy
python WordAlignment.py --ep 50 -m 1 --ll 1000 -o -t data/t_init_random.npy
python WordAlignment.py --ep 50 -m 1 --ll 5000 -o -t data/t_init_random.npy
python WordAlignment.py --ep 50 -m 1 --ll 10000 -o -t data/t_init_random.npy
python WordAlignment.py --ep 50 -m 1 --ll 30000 -o -t data/t_init_random.npy
python WordAlignment.py --ep 50 -m 1 --ll 50000 -o -t data/t_init_random.npy
python WordAlignment.py --ep 50 -m 1 --ll 70000 -o -t data/t_init_random.npy
python WordAlignment.py --ep 50 -m 1 --ll 90000 -o -t data/t_init_random.npy
python WordAlignment.py --ep 50 -m 1 --ll 100000 -o -t data/t_init_random.npy

# Epochs
python WordAlignment.py --ep 1 -m 1 -o -t data/t_init_random.npy
python WordAlignment.py --ep 5 -m 1 -t data/t_init_random.npy
python WordAlignment.py --ep 10 -m 1 -t data/t_init_random.npy
python WordAlignment.py --ep 20 -m 1 -t data/t_init_random.npy
python WordAlignment.py --ep 40 -m 1 -t data/t_init_random.npy
python WordAlignment.py --ep 50 -m 1 -t data/t_init_random.npy

# Initialization
python WordAlignment.py --ep 50 -m 1 -o
python WordAlignment.py --ep 50 -m 1
python WordAlignment.py --ep 50 -m 1
python WordAlignment.py --ep 50 -m 1
python WordAlignment.py --ep 50 -m 1
python WordAlignment.py --ep 50 -m 1 -t data/t_init_zero.npy

python WordAlignment.py --ep 50 -m 2
python WordAlignment.py --ep 50 -m 2
python WordAlignment.py --ep 50 -m 2
python WordAlignment.py --ep 50 -m 2
python WordAlignment.py --ep 50 -m 2

# Translation direction
python WordAlignment.py --ep 50 -m 1 -f data/hansards.e -e data/hansards.f

# Smoothing
python WordAlignment.py --ep 50 -m 1 -o -n 0.0001
python WordAlignment.py --ep 50 -m 1 -n 0.001
python WordAlignment.py --ep 50 -m 1 -n 0.01
python WordAlignment.py --ep 50 -m 1 -n 0.1
python WordAlignment.py --ep 50 -m 1 -n 0.5
python WordAlignment.py --ep 50 -m 1 -n 0.7
python WordAlignment.py --ep 50 -m 1 -n 0.9
python WordAlignment.py --ep 50 -m 1 -n 1
python WordAlignment.py --ep 50 -m 1 -n 2

# Final
python WordAlignment.py --ep 50 -m 1 -n 2
python WordAlignment.py --ep 50 -m 2 -n 2
python WordAlignment.py --ep 50 -m 2 -n 2 -t data/t_init_model_1.npy
```