Assignment 1: Word Alignment

(This assignment is based on the 2014 version of this course by prof. Yoav Goldberg, which was inspired by and based on HW1 in JHU's 2012 mt-class).

Word alignment is a core component in statistical machine translation. Even today, when neural machine translation systems are usually the go-to method, word alignment models are used to produce alternative translations and other features in production systems.

In class, we discussed methods for learning word-alignments from parallel text.

In this assignment, you will experiment with word alignments by:

1. Manually aligning some sentences between Hebrew and English.
2. Implementing a word-alignment software.

Part 1 - The Manual Alignment Task (10 pts)

In this task, you will experiment with word alignment of some "real world" sentences, to get a feeling for the task.

In the following files: hebrew.txt, english.txt you will find 10 Hebrew and 10 English sentences. The files are in unicode/utf8 encoding. You need to manually align the words in these sentences. You are allowed to have many-many mapping in both directions, as well as un-aligned words. You can have both "sure" and "probable" alignments.

What to submit: For this part, you need to submit your manual alignments. You can either print them out and draw lines manually, or submit a text file in the format described in the README file provided in part 2.

Part 2 - Statistical Word Alignment (70 pts)

Download and extract the contents of this file.

You will find French data, English data, some gold alignments, an example of an alignment file output, some scripts, and a README file. Read the README file.

Goal: You need to write a program that takes the English and French data, aligns it, and produces an alignment file in the format described in the README file. In addition, you should write the translation parameters t(e,f) to a file.

For 80% of the credit, implement IBM model 1. For the additional 20%, implement model 2. You can find the details of model 2's EM procedure in Michael Collins' notes, available here (also useful for model 1 - see Figure 4).

The supplied .a file contains manual alignments for the first few sentence pairs. This file should not be used in training your aligner (and it is too small anyways). But you can use it to measure the quality of your alignments in terms of AER using the supplied evaluation script.

What to submit: For this part, you need to submit a zip file containing:

a. Your code.
b. A README file, describing how to run your code to align the data with model 1 and model 2.
c. Alignment output file for model 1 and alignment output file for model 2.

A note about memory: While developing/debugging your code, you can use only the first k lines of the data. However, your final submission is expected to run on the entire dataset. This is a relatively small dataset, and should easily fit in the memory of a relatively modern personal computer. If it does not fit, maybe you should restructure your program.

The recommended language for this assignment is Python. To get faster and more memory efficient implementation it is recommended to use the numpy library to represent long lists of numbers. If you are using Java, make sure you allocate enough heap space for the JVM, as the default is very small. The head size can be specified using the -Xmx flag: “java -Xmx1g YourClassName" will allocate 1gb of heap size.

Part 3 - Looking at and improving the produced alignments and process (20 pts)

After running your aligner, look at the alignments and the learned parameters.

Experiment with your aligners and produce a report. Use graphs to showcase your results in each experiment. **In all cases, include results to justify your conclusions.**

1. Run your aligner several times, with various percentages of the training data. How does the AER change when you change the amount of training material?
2. How does the AER change when you change the number of EM iterations?
3. Does initialization affect the AER? Run model 1 several times, with different random initializations. Run model 2 with different random initializations, and with an initialization based on model 1. How does the initialization affect the AER?
4. Can you produce a better AER score by running your models in both directions ("e-to-f" and "f-to-e") and combining the results?

5. Can you produce better AER scores by trying some of the suggestions in this paper?

What to submit:

For this part, you need to provide:

A short report **(in .pdf format)** describing your experiments and the effect of the different training conditions on the AER.

Submission details, deadlines, etc:

You can program the assignment in any programming language you want. Your code should be able to run on linux, from the command line, without installing or using an IDE.

Your submission should include a .zip or .tar.gz file with your code, as well as a README file that explains how to run your program.

The input file names should be specified on the commandline, either as a parameter or through a pipe (your program may be run on a different file than the one provided to you).

Submit your assignment by email. The subject of your email should be "**mt course ex1**" (without the quotes).

roee.aharoni@gmail.com

You should submit the assignment by April 26th.