NLA Assignment 1 Report

Name: Shashank Gupta
Date: 1/2/16

Problem Statement

Given a parallel corpus, find the alignment matrix using EM algorithm.

Approach

Given a parallel corpus alignment matrix can be estimated using EM algorithm by first initializing the alignment matrix with uniform probability or with random values and then performing EM steps.

E step consists of finding estimated counts matrix for each sentence from the corpus and M step consists of re-estimating the transition matrix from the expected counts. These steps are for some iterations.

Code README and observations

REQUIREMENTS
1. Python numpy
2. Python scipy

The program takes as argument parallel corpus file with the following scheme:

{Source Language sentences} $\|\{Target Language sentences\}$

It also takes as argument the initial mode which can be used to initialize the alignment matrix.
It can be Uniform or Random. Command to run the code is:

python construct_alignment.py /path/to/parallelCorpus init_mode

NOTE

The schema for Alignment matrix is: Rows corresponds to French words and Columns corresponds to English words

Design for Scalability

This program can scale for large corpus. In the EM process for alignment estimation for each sentence a V1 * V2 matrix is constructed (V is the size of vocabulary). This makes total of N* V1 * V2 matrices at the end. This will blow the system memory.

To handle this issue only 3 matrices are used. One matrix is to store alignment matrix values at start of each iteration, other matrix is a temporary matrix where values of expected counts are updates after encountering each sentence. So we have only 3* V1 * V2 matrix for any number of sentences.

This is the design choice for scalability.

Convergence and Entropy Observation

EM converges after 27 iteration for the dataset given in assignment (2 sentence pair). The convergence criteria choosen is norm of matrix difference between alignment matrix and it's re-estimate matrix.

After each iteration of EM the entropy values per column keep decreasing. The reason is that initially they were random/uniform (entropy high for both cases) and as more information is feed to system it attains some stable state (which corresponds to low entropy)