Natural Language Processing, Problem Set 3

This programming problem focuses on machine translation. The goal is to implement two translation models, IBM model 1 and IBM model 2, and apply these models to predict English/Spanish word alignments. Both models estimate the conditional probability of a foreign sentence $f_1 \cdots f_m$ and an alignment $a_1 \cdots a_m$ given a particular English sentence $e_1 \cdots e_l$ and length m. The simpler model, IBM model 1, defines the conditional probability with a single set of parameters t as

$$p(f_1 \cdots f_m, a_1 \cdots a_m | e_1 \cdots e_l, m) = \frac{1}{(l+1)^m} \prod_{i=1}^{m} t(f_i | e_{a_i})$$

The richer model, IBM model 2, defines the same conditional probability with two sets of parameters t and q

$$p(f_1 \cdots f_m, a_1 \cdots a_m | e_1 \cdots e_l, m) = \prod_{i=1}^{m} q(a_i | i, l, m) t(f_i | e_{a_i})$$

Training Data

The main task of this assignment is to estimate these parameters from data. We estimate the parameters from a parallel corpus of aligned sentences. Each entry in the corpus contains two versions of the same sentence, one in English and one in Spanish. The parallel corpus we are using is a subset of the Europarl corpus [Koehn, 2005] accessed from `http://www.statmt.org/europarl/` which contains aligned sentences from the proceedings of the European Parliament.

The training corpus is split into two files, *corpus.en* and *corpus.es*, which contain English and Spanish sentences respectively. The i-th sentence in the English file is a translation of the i-th sentence in the Spanish file. The files are in UTF-8, contain one sentence per line, and words are separated by a space.

```
> head -n 3 corpus.en
resumption of the session
i declare resumed the session of the european parliament adjourned on ...
Although , as you will have seen , the dreaded ' millennium bug ' failed  ...
```

```
> head -n 3 corpus.es
reanudación del período de sesiones
declaro reanudado el período de sesiones del parlamento europeo ...
como todos han podido comprobar , el gran " efecto del año 2000 " no se ...
```

IBM Model 1

The full algorithm for IBM model 1 is given on page 21 in the notes on machine translation `http://www.cs.columbia.edu/~mcollins/ibm12.pdf`. We recommend closely reading those notes for a full description of the problem. The core portion of the algorithm is summarized here.

Recall that IBM model 1 only has word translation parameters $t(f|e)$, which can be interpreted as the conditional probability of generating a foreign word f from an English word e (or from NULL).

We can estimate $t(f|e)$ using the EM algorithm, which iterates through the parallel corpus repeatedly. For the k-th sentence pair and each index i in the foreign sentence $f^{(k)}$ and j in the English sentence $e^{(k)}$, we define

$$\delta(k,i,j) = \frac{t(f_i^{(k)}|e_j^{(k)})}{\sum_{j=0}^{l_k} t(f_i^{(k)}|e_j^{(k)})}$$

where l_k is the length of the English sentence. The delta function is used to update the *expected counts* for the current iteration:

$$c(e_j^{(k)}, f_i^{(k)}) \leftarrow c(e_j^{(k)}, f_i^{(k)}) + \delta(k,i,j)$$
$$c(e_j^{(k)}) \leftarrow c(e_j^{(k)}) + \delta(k,i,j)$$

After each iteration through the parallel corpus, we revise our estimate for t parameters:

$$t(f|e) = \frac{c(e,f)}{c(e)}$$

for all possible foreign words f and English words e (and NULL).

IBM Model 2

The complete algorithm for estimating IBM model 2 parameters is given on page 13 of the notes. Be sure to understand the details of this algorithm before beginning to code the assignment.

We note here that IBM model 2 extends the implementation of the EM algorithm for IBM model 1. The main additional step is adapting the delta function to include $q(j|i,l,m)$ parameters

$$\delta(k,i,j) = \frac{q(j|i,l_k,m_k)t(f_i^{(k)}|e_j^{(k)})}{\sum_{j=0}^{l_k} q(j|i,l_k,m_k)t(f_i^{(k)}|e_j^{(k)})}.$$

and computing expected counts $c(j|i,l,m)$ and $c(i,l,m)$.

After each iteration through the corpus we re-estimate the $t(f_i,e_j)$ parameters as before and add new updates for the q parameters:

$$q(j|i,l,m) = \frac{c(j|i,l,m)}{c(i,l,m)}.$$

Evaluation

We provide aligned development and test data, `dev.en / dev.es` and `test.en / test.es` in order to evaluate alignment accuracy. The format of these files is identical to the corpus training files, one sentence per line with aligned sentences.

We also provide a file, `dev.key`, which contains the manually annotated gold alignments for the development sentences. The key is adapted from Lambert et al. [2005] and accessed at `http://gps-tsc.upc.es/veu/LR/epps_ensp_alignref.php3`. The file contains lines of the form

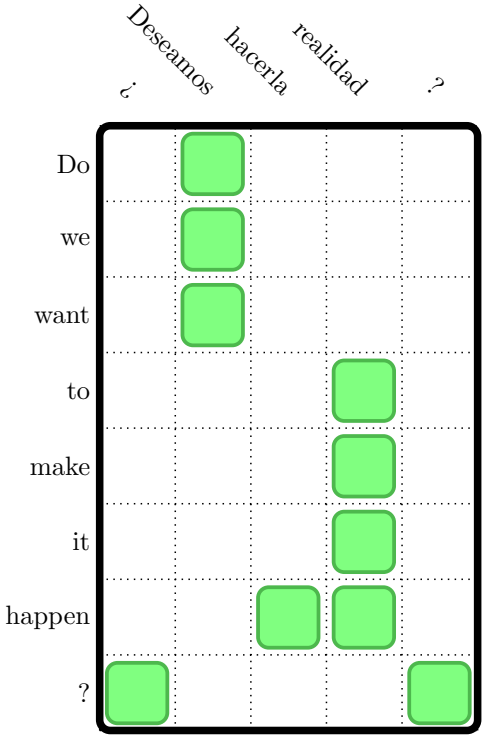
```
...
6 7 3
6 7 4
6 8 5
...
```

Each line specifies a word alignment of the form `SentenceIndex EnglishIndex ForeignIndex`. The first line of the example specifies that in the 6th pair of sentences the 7th English word is aligned to the 3rd Spanish word. Note that the gold alignments do not include NULL word alignments, all three indices start from 1, and unlike previous assignments there are no blank lines between sentences.

As an example, consider the 6th sentence pair in the development set.

- $e =$ Do we want to make it happen ?
- $f =$ ¿ Deseamos hacerla realidad ?

The gold word alignment, `6 7 3`, indicates that the English word "happen" is aligned with the Spanish word "hacerla". The full alignment of the sentence pair is



For all of the questions in this assignment you should output predicted alignments in this format. We have included a script `eval_alignment.py` to evaluate the accuracy of these alignments. Assuming that you have written alignments to a file `dev.out`, running `python eval_alignment.py dev.key dev.out` will give the F-Score of the predicted alignments compared to the gold alignments.

Note: The alignments given in the key file often map single English words to multiple foreign words, whereas our models do not. For this reason it is impossible to get perfect recall or F-Score on this data set. For questions 1 and 2 we ignore this issue. Question 3 discusses a method to fix this problem.

Question 1 (25 points) - IBM Model 1

The first problem is to estimate the parameters of IBM model 1 using `corpus.en` and `corpus.es` as input. Before implementing please read the following notes carefully.

- Your implementation should only store t parameters for possible pairs of foreign and English words, i.e. words that occur together in some parallel translation, and the special English word NULL. We recommend implementing t as a collection of sparse maps for each English word e where the keys of the map are the words f that are occur with e and the values are the corresponding parameters $t(f|e)$. Using a non-sparse implementation will likely result in memory issues.
- In the initialization step, set $t(f|e)$ to be the uniform distribution over all foreign words that could be aligned to e in the corpus. More specifically

$$t(f|e) = \frac{1}{n(e)},$$

  where $n(e)$ is the number of different words that occur in any translation of a sentence containing e. Note that the special English word NULL can be aligned to any foreign word in the corpus.
- Starting from the initial $t(f|e)$ parameters, run 5 iterations of the EM algorithm for IBM model 1 (this may take a while). When your model is completed, save your t parameters to a file, as you will need them for the next problem.
- Finally, use your model to find alignments for the development sentence pairs `dev.en / dev.es`. For each sentence, align each foreign word f_i to the English word with the highest $t(f|e)$ score, i.e.

$$a_i = \arg \max_{j \in \{0 \cdots l\}} t(f_i|e_j).$$

  Write your alignments to a file in the format of `dev.key` described above. Check the accuracy with `eval_alignments.py .`

When you are satisfied with development performance, run your model on `test.en` and `test.es` to produce `alignment_test.p1.out`. Submit your output with `submit.py`.

Question 2 (25 pts) - IBM Model 2

We now extend the alignment model to IBM model 2 by adding alignment parameters $q(j|i, l, m)$.

- Initialize the q parameters to the uniform distribution over all j for each i, l, and m, i.e.

$$q(j|i,l,m) = \frac{1}{l+1}$$

  You only need to store parameters for pairs of sentence lengths l and m that occur in the corpus.

- To initialize the $t(f|e)$ parameters, use the last set of parameters (after 5 iterations) produced by your implementation of IBM model 1.

- Run 5 iterations of EM for IBM model 2.

- As before, use the model to compute alignments for the development sentence pairs in the corpus. For each foreign word f_i, the best alignment is

$$a_i = \arg \max_{j \in \{0 \cdots l\}} q(j|i,l,m)t(f_i|e_j).$$

  Write your alignments to a file in the format of `dev.key` described above. Check the accuracy with `eval_alignments.py` .

When you are satisfied with development performance, run your model on `test.en` and `test.es` to produce `alignment_test.p2.out`. Submit your output with `submit.py`.

Optional Question 3 (1 point) - Growing Alignments

(This question is an optional extension of the first two problems.)

As we noted above, the gold alignments allow English words to be aligned with multiple Spanish words. This problem explores a method to get around this issue and generate more complete alignments. This process is a crucial first step in order to extract a lexicon for phrase-based translation.

The recipe we use is described on slide 11 of the lectures on phrase-based translation. It consists of the following steps

1. Estimate IBM model 2 for $p(f|e)$ (exactly as in question 2).
2. Estimate IBM model 2 for $p(e|f)$ (as in question 2 **except** with English as the foreign language).
3. Calculate the best alignments with $p(f|e)$ and $p(e|f)$.
4. Calculate the intersection and union of these alignments.
5. Starting from the intersection, apply a heuristic to grow the alignment.

A good example heuristic is sketched out on slide 12.

- Only explore alignment in the union of the $p(f|e)$ and $p(e|f)$ alignments.
- Add one alignment point at a time.
- Only add alignment points which align a word that currently has no alignment.

- At first, restrict ourselves to alignment points that are "neighbors" (adjacent or diagonal) of current alignment points. Later, consider other alignment points.

For this problem, you should estimate both $p(f|e)$ and $p(e|f)$ and use these models to generate new alignments. We recommend starting with the heuristic from the lecture and begin with the intersection alignment while growing towards the union alignment. However, feel free to try out different alignment heuristics to improve the score of the model. The auto-grader for this problem will not have an upper-bound on accuracy.

When you are satisfied with development performance, run your model on `test.en` and `test.es` to produce `alignment_test.p3.out`. Submit your output with `submit.py`.

References

Philipp Koehn. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, 2005.

Patrik Lambert, Adrià De Gispert, Rafael Banchs, and José B Mariño. Guidelines for word alignment evaluation and manual alignment. *Language Resources and Evaluation*, 39(4):267–285, 2005.