INF5820, fall 2016

Assignment 2: Alignment for Stat. MT

Deadline 21 Oct. at 6 pm, to be delivered in Devilry

In this set we will familiarize ourselves with the first steps in the construction of a statistical machine translation system, learning word alignments.

We will first implement the first steps ourselves to understand the EM-algorithm better. Then we will use the well known tool GIZA++ for the remaining steps, before we consider how the results could be used for phrase alignment. You are advised to read the whole set before you start. You might find it easier to work on exercise 2 before exercise 1.

1 Implement EM training of IBM model 1

Write a program that implements the algorithm for training IBM Model 1. Input should be two files with the same number of sentences (lines); normally they will be from two different languages and one will be a translation of the other. Output should be a file with the word translation probabilities. Since we will not run model 1 to convergence, but a fixed number of times before higher models take over, the program should take as extra argument a number expressing the number of iterations.

You may use any of the programming languages: Python, Java, Scheme or CommonLisp. There is a pseudo code implementation in the K:SMT-book, figure 4.3, but there are some problems with this code (cf. the errata page) so you should make sure that you understand what is going on.

For testing during development, you may use the simple example from the lecture, and make two files *eng* and *nor* with two lines each:

```
dog barket
dog bit dog
```

and

```
hund bjeffet
hund bet hund
```

Repeat the experiment from the lecture and see whether you get the same results after 1,2, 5, 25, 100 iterations.

Submission: Code, the files *nor* and *eng*, files showing the results of running with 1, 2, 5, 25, 100 iterations.

2 GIZA++

We will experiment with the well known alignment tool GIZA++. You will find the relevant material in the folder /projects/nlp/external-bin-dir (which you may download to your own file area).

Part A: Preparartions

To familiarize ourselves with the tool, we will use the same simple example files *eng* and *nor*. To prepare the corpus for GIZA++, we have to compute some additional files. Run first

```
./plain2snt.out nor eng
```

Take a look on the newly generated files and try to understand what they contain. An .vcb-file assigns a unique numerical identifier to each word. The second number, e.g., 3 in

```
2 hund 3
```

counts the number of ocurrences of this word. In the .snt files, the words in the source files are replaced by the identifiers from the .vcb files.

We also need one more initial preparation step

```
./snt2cooc.out eng.vcb nor.vcb eng_nor.snt > corp.cooc
```

We are not concerned with the content of this file, but it is used in the next step. (We could have used a different name for this file, e.g., to distinguish between different .cooc files.)

Part B: Running GIZA, inspecting the results.

We can then run the alignment program

```
./GIZA++ -S eng.vcb -T nor.vcb -C eng_nor.snt \
-CoocurrenceFile corp.cooc
```

This generates a series of files with a longish prefix. You may instead use a prefix of your own choice by the option "-o", e.g.,

```
./GIZA++ -S eng.vcb -T nor.vcb -C eng_nor.snt \
-CoocurrenceFile corp.cooc -o experiment1
```

To see what is going on and what the files contain, we may take a look at the screen output from the run (which we could dump to a file by)

```
./GIZA++ -S eng.vcb -T nor.vcb -C eng_nor.snt \
-CoocurrenceFile corp.cooc -o experiment1 > output1
```

We see that the program by default has run 5 iterations of IBM model1 followed by 5 iterations of model3 and 5 iteraions of model4.

The names of the produced files indicate

- whether it is an alignment file (a/A), a lexical probabilities file (t) etc.
- which IBM model has produced it
- and after how many iterations

For example "experiment1.t3.final" is the word probabilities after the program has finished model 3. To make this readable, we have made a script which works for small files.

```
./nums2words.py eng.vcb nor.vcb experiment.t3.final
```

Take a look at the produced file. Also have a look at "experiment1.A3.final" and try to understand it.

Part C: Changing parameters

We may change the default parameters (which we see in output1). For example, to study the convergence behavior of Model 1, we may iterate it a hundred times by

```
./GIZA++ -S eng.vcb -T nor.vcb -C eng_nor.snt \
-CoocurrenceFile corp.cooc -o experiment2 \
-model1iterations 100
```

But this doesn't change which files are produced. How can we see the effect of the change? By asking the program to dump more intermediate results e.g.,

```
./GIZA++ -S eng.vcb -T nor.vcb -C eng_nor.snt \
-CoocurrenceFile corp.cooc -o experiment2 \
-model1iterations 100 -model1dumpfrequency 10
```

Submission: Check whether GIZA++ produces the same results as the implementation you made yourself in the first exercise—and the ones we reported at the lecture—and report the numbers, i.e., the translation probabilities after 1, 2, 5, 25 and 100 iterations,

Compare the translation probabilities from 100 iterations of model 1 to the output of model 3 with the default settings (i.e. 5 iterations of model 1 followed by 5 iterations of model 3).

Part D: The alignments

Consider the A1 files after each of the first 5 runs of model1. Then consider the result after 100 model 1 iterations and after 5 model1 followed by 5 model 3 iterations.

Submission: Report the alignments after each of the five rounds in the standard format for alignments. The results are a little surprising, how?

Then report the alignment after 100 model 1 iterations and after 5 model1 followed by 5 model 3 iterations. What do you see?

3 Phrase alignment

Part A

We have made a slightly larger corpus to study the effects of alignments which are not one-to-one. You find the texts in /projects/nlp/inf5820/alignment. First of all you should tokenize the texts and also lowercase them. The latter may be done by

```
/projects/nlp/mosesdecoder/scripts/tokenizer/lowercase.perl \
     < nor.tok > nor.lower
```

Then run GIZA++ in both directions with the default settings.

Submission: The resulting A3- and t3-files.

Part B

Consider the resulting alignments A3. Draw figures similar to figure 4.13 in the SMT book for the first three sentences.

Submission: The figures.

Part C

Submission: Extract all the phrase pairs that may be extracted from the three sentences on the basis of the figures.

4 Translational challenges

The following are examples of exercises you may get at the exam. We include them to get som useful practice. Use a half to one page to answer each of them.

Part A

The most immediate idea for MT is to use a dictionary and translate word-by-word. Some of the problems for this strategy have to do with typological differences between languages. Describe 3 typological differences between languages which make problems for this simplistic approach and explain how they are problematic.

Part B

The most immediate idea for MT is to use a dictionary and translate word-by-word. Some of the problems for this strategy have to do with typological differences between languages, but there are also other problems which do not reside in typological differences. What sort of problems are these? Illustrate with examples and explain why they are problematic.

End