Exercises

Statistical Processing of Natural Language

Winter 2012

1 Language Models – MLE & Smoothing

Retrieve the exercises done in class about MLE and Smooting, and modify them to perform Linear Interpolation smoothing. Proceed as follows:

1. Extend the program `mle.py` to estimate the coefficients λ_1,λ_2,λ_3 for a linear Interpolation smoothing. Write the coefficients into the first line of the model file, followed by the trigram parameters.

   Coefficient estimation via deleted interpolation:

   ```
   $\lambda_1$=$\lambda_2$=$\lambda_3$=0
   foreach trigram $xyz$ with $count(xyz) > 0$
      depending on the maximum of the following three values:
         case $\frac{count(xyz)-1}{count(xy)-1}$ :   increment $\lambda_1$ by $count(xyz)$
         case $\frac{count(yz)-1}{count(y)-1}$ :   increment $\lambda_2$ by $count(xyz)$
         case $\frac{count(z)-1}{N-1}$ :   increment $\lambda_3$ by $count(xyz)$
   normalize $\lambda_1$,$\lambda_2$,$\lambda_3$
   ```

2. Extend the program `smooth.py` to load the Linear Interpolation coefficients in the first line of the file, load the rest of the model normally, and use Linear Interpolation to smooth the trigram probabilitites:

   $$P(z|xy) = \lambda_1 P(z) + \lambda_2 P(z|y) + \lambda_3 P(z|xy)$$

   Compare the results with those obtained in the smoothing versions used in class.

2 Supervised Methods – Max. Entropy Classifiers

1. (a) Use the encoded corpus `corpus/efe/f50/train.0` to learn a Maximum Entropy Model using the `megam_i686.opt` executable:

```
./megam_i686.opt -quiet -fvals multiclass corpus/efe/f50/train.f0 > f50.mem
```

   (b) Test the performance of the module running `megam` in test mode on the corpus `corpus/efe/f50/test.f0`:

```
./megam_i686.opt -fvals -predict f50.mem multiclass corpus/efe/f50/test.f0 >out
```

   (c) Complete the program `classifier.py` to compute the probability of each class for each input example, and produce the same output than `megam` test mode. Use the correct answer in the test files to compute the accuracy statistics.

   The probability that the ME model assigns to a class a given a document b is computed as:

$$p(a \mid b) = \frac{\exp(\sum_{j=1}^{k} \lambda_j f_j(a,b))}{Z(b)} \quad ; \qquad \text{where} \quad Z(b) = \sum_a \exp(\sum_{j=1}^{k} \lambda_j f_j(a,b))$$

   Each λ_j corresponds to a combination $j = (feature, class)$. $f_j(a,b)$ is the active value of j for document b and class a (note that $f_j(a,b) = 0$ if $a \neq j.class$, and that it is the value of the feature in the document otherwise).

   NOTES:

   - The corpus files contain one document example per line. The first field is the right answer (document class) used in train and in evaluation. The other fields are pairs <`feature,value`> representing that document
   - The produced model file `f50.mem` has the following format: The first field in each line is a feature name x. The other fields are the λ_j values for each class $j = (x, i); \forall i = 0 \ldots 12$.

2. (a) Modify the program `classifier.py` to output not only the most likely class, but all classes with a probability over a given threshold. Modifiy the evaluation to compute also precision, recall, and F1. Check how results vary depending on the given threshold.

   (b) Train and test a classifier using the corpus `corpus/efe/f100/train.f0` for training and the corpus `corpus/efe/f100/test.f0` for testing. Compare the performance of this classifier with that of the classifier obtained in the previous exercise using corpus `f50`. Perform a hypothesis test to find out whether the difference is statistically significant.

(c) Perform a cross-validation evaluation for the same cases above, using corpus `corpus/efe/f50/train.*` and `corpus/efe/f50/test.*` to train and test five folds of one classifier, and `corpus/efe/f100/train.*` and `corpus/efe/f100/test.*` for the other. Discuss the changes in the statistical significance of the difference between both models.

> NOTE: Five-fold cross-validation consists of repeating the train-test cycle five times, using different partitions of the corpus. That is, train with corpus `train.`i and test with corpus `test.`i for $i = 0 \ldots 4$.