Homework 4 - Report

Quanze Chen

March 26, 2015

Motivation/Models Used For this task, I mainly experimented with feature engineering for PRO. The motivation behind choosing PRO is its relative ease in implementation, resistance against noise and fast computation time. For feature engineering I mainly experimented with the following features:

- Default translation, lexical and language models $p(e|f), p_{lex}(f|e), p(e)$
- Length of hypothesis sentence
- Number of Cyrillic words in hypothesis sentence
- Number of matched brackets in hypothesis
- IBM Model 1 Alignment Scores against source sentence (untrained)
- Unigram Features
- Bigram Features

I was able to get some observable improvement by incorporating length, untranslated word count and Unigram features but the remaining features proved to be of limited help. With these features I was not able to beat the baseline but this may be due to the classifier model used (analyzed at the end).

Description of Each Feature and Results :

Default Features + PRO
The default probability features follow from the default model. I used PRO training with a perceptron as the backing classifier to learn weights. Sentences were then scored by the weights and the best scoring one was picked. The default model under PRO training scored at 0.2536 on the dev set locally (compared to the default model at 0.2735) and got a score slightly below the default in the test set at around 0.24.

+ Length of Hypothesis and Untranslated Word Count
Adding these two features achieved local scores of 0.2542 to 0.2617 (close to without the features). On the test set online, these were able to get close but still slightly below the default at around 0.25.

+ Unigram Features
Unigram features proved to be the most help. All unigrams were extracted from the target list and added as a feature. I experimented with each feature's value being binary (0 if has not occurred, 1 if there is at least 1 occurance) or counts (number of occurances). Both proved to be similar in result. I was able to get to 0.3085 on the local dev set and 0.2760 on the test set. This later proved to be the highest score I was able to achieve. It may be that unigrams will reflect the context of the sentences and thus causing somewhat an increase in score.

+ Bigram Features
Bigram features proved to be helpful in much more limited ways. With the local training data I was able to get to 0.3475 (unigram + bigram) and 0.3432 (just bigrams) on dev test set. However, this success

was not mirrored for the test set and Bigram features caused the score to drop to around 0.25. This may be because the classifier is overfitting the dev data locally when optimizing features and bigrams will carry information of the optimal sentence into the result.

+ Matched Parenthesis/Unmatched Parenthesis

I also experimented with various parenthesis checks. Again this feature took on many different values: (1) Unmatched parenthesis binary (2) Unmatched parenthesis count (3) Matched parenthesis count. With this feature as type (1) and without ngram features, I saw a 0.01 improvement locally (0.2728 over length + untranslated). Later I experimented with the two other definitions, but neither influenced the score much (with unmatched parenthesis count even decreasing score).

+ IBM Model-1 Alignments

Lastly I tried incorporating model 1 alignments by appending it to the feature list in the input. Alignment score was calculated as number of alignments from Model 1 (against sentences in src) with no extra alignment training data. This gave a score of around 0.2638 locally and also around 0.26 in the online test set.

Analysis and Possible Problems During testing scores were not consistent when introducing new features (sometimes scores go down with the introduction of a new feature) and parameters influenced the results also to some extent. By default the system generates 5000 random sample pairs and picks the top 100 with best BLEU improvements (BLEU is calculated as smoothed single sentence BLEU) to train a perceptron. When BLEU improvements are < 0.1 the record will not be counted. The perceptron is trained for 5 rounds. It is often the case that the weight values do not converge and will tend to switch between certain states. This may be due to the fact that we cannot divide the data well enough with the features. This also means that different runs will produce different results due to the random nature of selection.

It is suspected that the variation in perceptron training caused variation in the scores which proved to be not insignificant. By only running 5 - 10 rounds of training, it may also be the case that the perceptron cannot/has not converged. The multitude of configurations also does not help in identifying the perfect input parameters.

Towards the end I was able to swap out the default perceptron implementation with an averaged perceptron implementaton, MegaM classifier and logistic regression (sklearn). However I was only able to test without unigram features as it took too long (and too much memory) to run with them. With all features above not including IBM 1 and ngram features, I was able to get a consistent 0.26 score on both the test and local data. Adding IBM 1 did not help and with some classifiers decreased the value.

The training feature vector is $(x_i - x_j)$ (the difference between two feature vectors). Labels are assigned on whether x_i or x_j got a higher BLEU score. Best sentences are picked by predicting the label of each pair of hypothesis (1 if the first is better, -1 if reversed) and outputting the hypothesis that has the most positive lables against all others. More consistent classifier results lead me to believe that better scores may be achieved using the same features (such as unigrams) on a better trained classifier.

Also som problems of inaccuracy may cause discrepancies. For example, to identify cyrillic I used regular expressions to match words not completely made up of the English alphabet or common symbols. As this is a kind of whitelisting for characters, it may lead to some false positives identified as untranslated. However I do not expect this to contribute much as it still correlates to the actual number of untranslated words.