Incomplete Text Predictor – EECS 349 Northwestern final report – Talpes Nicoara

Large texts that are transcribed on electronic format by hand or using OCR techniques contain errors, or have incomplete transcriptions. An example is the set of digital images on the EEBO website of early medieval works. The goal of this project and the learning task was to come up with a novel way of finding corrections for incompletely transcribed words. The research question to answer was how accurate can our corrector be? We also tested active learning, an active research field, and measured success if the learned model was better than just passive learning. Professor Downey had a large contribution to the ideas tackled in this project.

The novel solution has two parts. First, a language model looks at the context of the incomplete word and the vocabulary of the corpus and comes up with a set of corrections for each word. Then the most probable corrections are compared to a set of provided solutions to construct the data set, which is now composed of 9123 corrections. The four attributes for each instance are: the probability that the correction is right, the probability difference to the next most probable correction, the number of other corrections, and the number of characters missing from the incomplete word.

The second step was to use a classifier to calculate the accuracy of the training set. For most learning algorithms we would select a sample of the data - the train data, and train the model on it. Active learning is selective in how to select the training examples, and can predict corrections better. Active learning is a technique that can drive down the exponential dependency on the number of attributes to linear dependency, and the target concept can be acquired much faster. As an algorithm, linear regression worked well since our data requires nominal values for the class attribute: 0 or 1. I performed active learning by setting first a training set of 100 examples. Then I added the most uncertain example in the test set and retrained 900 times.

There was a bias present in the model if the elements that were chosen for active learning were chosen from the test set, since we peeked at the test set. Over fitting was avoided by splitting the data set into a training set of 100 examples, a pool set of 5000 examples to choose examples from, and a test set of 4023 examples. I did not use cross validation, since it would not have helped the results for our specific data set.

I looked at the classification performance at different times during the process (nine times), after each time 100 examples were added to the model. I compared this with 3 other techniques. First, the baseline using ZeroR was the number of correct examples in the training set, 78%. Second, the performance of the initial train set on the test set, 76.8% (3090 correct/933 incorrect). Third, the performance of the model with 100 randomly selected examples added to it every time, instead of based on uncertainty. Active learning seems to be dependent on the training set, and after playing around a bit with the distribution, we get similar results with what follows.

The results are in Figure 1 below. The x-axis represents the times when 100, 200 etc examples are added to the initial train set. The y-axis represents the accuracy of the model on the test set. Comparing the results at the mentioned benchmarks showed the accuracy of a classifier trained on active learning is consistently better, confirming our expectations. The active learning scores above 78% except once, while the randomly-selected examples technique oscillates between

76% and 77.5%. There is a 0.75-2 percent gap between them at all times. This is not big, but consistency proves the validity of the argument.

A different number that can point us to how the question of how well accuracy can get is 80.2%, the precision of a decision tree run on a training set of 6000 examples, and on a test set of 3123.

One suggestion for future work is employing caching inside the language model. This would tailor the provided corrections to a dictionary of the time around which the play was written, or from the author's vocabulary. Also, a natural step would be to try this on a larger data set. Both these methods could be tested to see if they drive up accuracy.

The goal of the incomplete text predictor tool would be to show where errors are, give suggestions, and fix errors. The application could be extended to finding errors in existing completely transcribed texts, in different languages. The endeavor is useful since it gives highly accurate corrections, minimizing the current human involvement in the process.

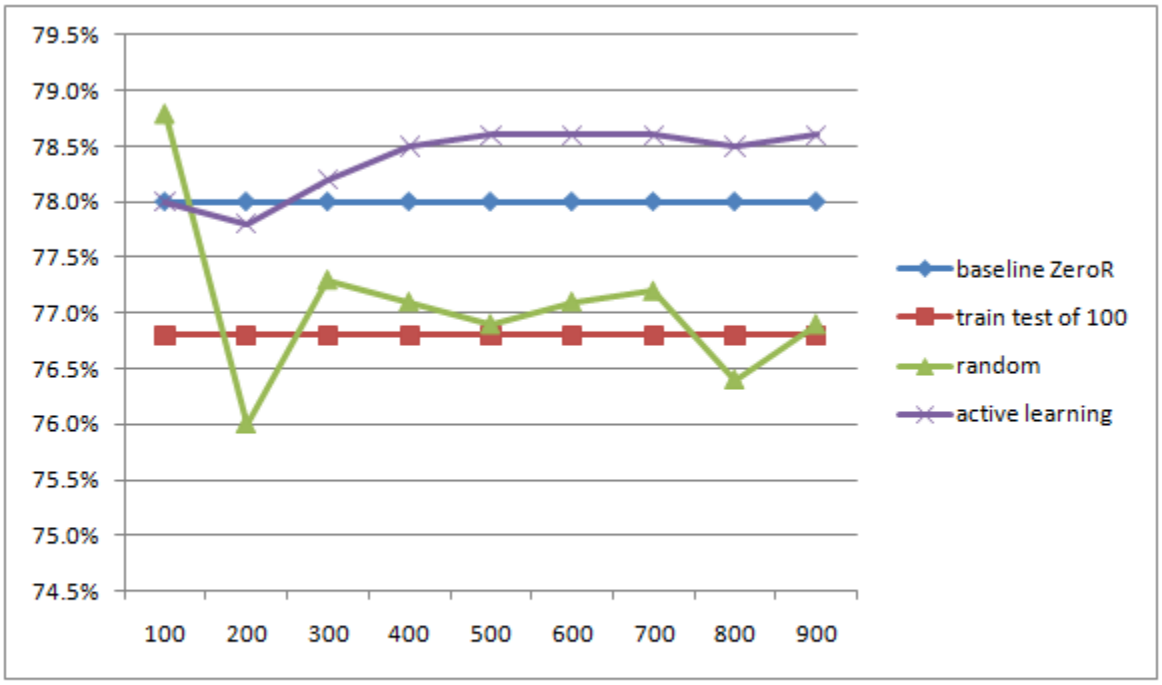

Figure 1 - the accuracy of a model starting with 100 examples when adding to it the numbers of examples on the x-axis