CMPUT 497: Assignment 2 Report

Shouyang Zhou
University of Alberta
Edmonton, Alberta, Canada
shouyang@ualberta.ca

1 Implementation Details

The script contains three main sections, a section for constants, a class titled "N_model", and a small set of functions used to bulk evaluate files.

The language model is implemented through the "N_model" class. The class is initialized from a filepath containing the training data. Upon initialization, the class processes the training text into a frequency distribution of n-grams and interpolation coefficients (if needed). Once the model is initialized, it can be used to evaluate a text using the "eval_text(text)" method.

Internally, the model references three frequency distributions to produce probabilities. In general, the attribute "freq_dist" is referenced to obtain counts of the actual n-gram, and the "prev_dist" is used to obtain the count of the prefix (n-1 grams) of the n-gram. These two values are then used to calculate a conditional log probability for the n-gram. Additionally, there is a unigram distribution to act as a term list. The interpolation model uses a differing set of frequency distributions titled, "freqs", this was a later addition for the same purposes.

To evaluate a text, the model splits the evaluation text into n-grams according the model and evaluates each n-gram summing their log_2 probabilities. The sum of log-probs. are then transformed into a perplexity score. The evaluation text is preprocessed such that characters (unigrams) not found in the unigram distribution are replaced with the unknown token.

The functions provide a small evaluation framework. The function generate_evaluations_file, generates a list of models and evaluates each evaluation text using each model. It will select the model with the lowest perplexity score per evaluation text.

2 Issues Encountered

Dealing with unknown characters became an issue following the guide in J&M chapter 3. J&M recommend using an unknown token and to generate possible unknown token inclusive n-grams via preprocessing the training text, substituting infrequent terms (by some user specified criterion, ie low word occurrence) with the unknown token.

I decided to skip this unknown token preprocessing step. My rationale is that this preserves more of the actual training text (which is quite short) used to generate n-gram counts. Also, choosing the right criterion poses a confounding challenge, examining and tuning a threshold per language is too time intensive and may introduce too much bias. Using one general threshold may be too general to consider the unique characteristics in each language.

To deal with unknown terms, I will limit my unsmoothed model to an n value of 1 and seed any unigram models with a single occurrence of the unknown token.

3 Parameter Tuning

N values were tuned on the dev suffixed dataset. Iteratively, each model variant with increasing n from 1-4 was ran to produce the evaluation report upon this data. The accuracy of the model and relative perplexity scores where considered in tuning. My tests results are contained in the Excel

file. I aim to achieve a reasonable accuracy rate (95%) while minimizing perplexity scores. I suspect marginal increases in accuracy at substantial costs in perplexity will be an indicator of data sparseness and potential overfitting.

Mentioned earlier, the unsmoothed model has a fixed n value of 1. This model correctly identifies the 49 / 55 dev files and has a mean perplexity of 17.5.

Increases in N of Laplace models increased accuracy at the cost of "perplexity bloom". The model's accuracy increase (49/55, 53/55, 53/55, 54/55) however their mean assigned perplexity scores increase substantially (17.5, 16.75, 117, 426). N = 2 meets my selection criterion.

Increase of N of interpolation models were considered on the range of 2-4 given the nature of the model. The accuracy of the model increased over N (50/55, 52/55, 53/55), and the perplexity remained approximately constant (bounded between 10-20). Examining the weights placed by the deleted interpolation process, all models assign unigrams a weighting of approximately 90% with following ngrams being exponentially weighted less. I choose an N value of 3.

4 Evaluation & Error Analysis

Common errors for all models involve distinctions between dialects, for example deu_1996 vs deu_1991 and hat_kreyol vs hat_popular. Examining these two as case studies, both training datasets share very similar character sets and words. Notably, the deu training sets differ by the replacement of a single character. In these cases, the small variations in the training datasets, notably length, may favor certain models.

Errors were correlated in general; some dev files appear be simply harder to distinguish overall. Errors within the same model class, across n are most highly correlated.

Harder files are often texts that draw similarities to other languages. For example, "udhr-sco.txt.dev" appears to draw on many English like words. I suspect one confounding factor to be text forms that in some form are a spelling simplification of another text form. If this simplification favors removing idiosyncrasies for more frequently used rules, then this may make a character model consider it to be more closer to parent language.

5 Conclusion

The models implemented were able to achieve similar accuracy rates, of my threshold (95%) and mean perplexity scores. Laplace models exhibit a substantial increase in perplexity scores as n grows, likely due to the noise generated by its redistribution of probability mass. Interpolation models exhibit consistent perplexity scores, likely due to robustness added by the deleted interpolation process.

References

https://stackoverflow.com/questions/17531684/n-grams-in-python-four-five-six-grams

https://www.sltinfo.com/type-token-ratio/