Interpolated n-gram models

Marco Kuhlmann

While neural models define the state of the art in language modelling, they require substantial computational resources. Where these are not available, the older generation of probabilistic language models can make a strong baseline. Your task in this advanced lab is to evaluate one of these models on the WikiText-2 dataset.

Problem description

Read the section on Interpolation in Eisenstein's book (section 6.2.3).

0. Implement a statistical n-gram model and write code for setting the probabilities of that model using maxmimum likelihood estimation. You can do this in pure Python or use NumPy; no other non-standard libraries are allowed. Your implementation should be able to handle arbitrary values of n.

1. Extend your implementation into an interpolated n-gram model. One way to do this is by following the *composite pattern* of software engineering, essentially building a container for a list of standard n-gram models. Instantiate your interpolated model for $n = 3$ and report the perplexity of this model on the test section of the data. Choose a uniform distribution for the interpolation weights (λ).

2. Implement the Expectation–Maximization algorithm for finding the best values of the interpolation weights. Tune the weights on the validation section and report the perplexity of your best trigram model on the test section.

3. Summarise your results and observations in a short report (PDF, ca. 1 page). Submit a single zip-file with your code, instructions for how to run your code in order to reproduce your results, and your report.