Advanced Methods in Text Analytics
Exercise 4: Language Models - Part 2
Solutions

Daniel Ruffinelli

FSS 2025

1 N-Gram Language Models

(a) See Jupyter notebook.

(b) See Jupyter notebook.

(c) See Jupyter notebook.

(d) Most popular n-grams are those that have padding on the left. This is because of the way we construct examples. More meaningful n-grams are found lower in the ranking of most common n-grams.

(e) The output looks very much like Shakespeare's work, but it is not actually meaningful most of the time. This has to do with our model's inability to capture patterns between more distant words, with our sampling approach, or both. Consequently, short chunks of text make sense, e.g. a subsequence of roughly the size of n.

(f) The output is still as before, with most generated sequences seeming like Shakespeare's work, but not being quite meaningful. The output does change considerably with different runs, mostly due to the way we sample. Different prompts also have a considerable impact in how meaningful the generated text appears.

(g) Recall that perplexity can be interpreted as proportional to the size of our vocabulary. In that sense, the reported perplexity is quite high, as it's below but close to 50% of the vocabulary size. However, without a proper baseline, it's always difficult (or impossible) to determine how well a model performs.

2 Language Models with Fully-Connected Neural Networks

(a) See Jupyter notebook.

(b) See Jupyter notebook.

(c) See Jupyter notebook.

(d) Perplexity is now much lower than when using an n-gram LM before. However, it is not exactly fair to make this comparison, as the set of examples used to compute the likelihood of the data are not the same. With the n-gram LM, we predicted every word

in the validation dataset using the last two words only, which is what that model can do. Here, however, we split the validation set into sequences of a fixed length and then just computed the likelihood of the last words in each sequence given all previous ones in the sequence. This is something that the n-grams model cannot do, but we should be able to compute likelihood as done with the n-gram model using this neural LM. Instead, we'll compare it with an RNN-based LM soon.

3 Language Models with RNNs

(a) It is more expensive to train due to the computation done by the network, which is considerably more involved than the simple forward pass of the neural LM from before. In addition, for a given input sequence, we are constructing many more training examples, which we average over.

(b) Perplexity is much lower compared to the neural LM using fully-connected neural networks. The RNN likely takes advantage of its architecture, designed to keep track of patterns across input sequences, but as mentioned before, it does use more training examples compared to the neural LM from before, which may account for some of this improvement. The neural LM can in principle be trained in this way as well, all we need for that is construct the training examples accordingly.

(c) The text generated with our n-gram models from Task 1 actually seems of higher quality, despite the RNN showing lower perplexity on validation data, and an inductive bias designed for processing sequences of tokens. This is likely because of the difference in sampling approaches. Here, we are using random sampling, i.e. sampling from the entire distribution over our vocabulary. This explains why we do sample tokens that aren't necessarily frequent, e.g. certain punctuation marks.

In Task 1, however, we sample from a small set of words in the vocabulary, not from the entire vocabulary. This small set of words is given by the words the n-gram model has seen in the same context during training, and even with add-1 smoothing, the vast majority of the probability mass is in the words seen next to each other during training. This is quite a strong bias and likely not good for generalization, but it does produce better quality text in this case compared to using random sampling.

In addition, the quality of the generated text is highly dependent on the temperature used with the softmax function. Higher temperatures mean that the softmax distribution becomes more uniform, whereas lower temperatures mean that the mass in the softmax distribution focuses more on elements which already have more mass. In this case, with a temperature value of 1.5, we can generate text that seems of much higher quality compared to using a temperature of 10 or 0.1. However, even in the best cases, the results are still sentences that seem *shakespearean*, but aren't quite meaningful.

(d) The choice of sampling method has a clear impact on the generated text. With some combinations of temperature and k, e.g. with temperature set to 2.2 and k to 100, the output can sometimes be similar to what the n-grams produced, but this is often not the case, especially with other values for temperature and k. The model often outputs names of characters or other words outside of dialogue, e.g. 'SONNETS' and similar words in all-caps, as well as punctuation marks in unusual places. It is unclear why this is the case, but it may be that the model is already overfitting to the corpus, which would explain why it produces words that pertain to the format rather than actual dialogue.

Such issues could be addressed by better pre-processing the data before training (e.g. remove character names, unless we want a model to generate scripts), or by applying different forms of regularization. These decisions are task-dependent. More generally, language models often generate text in undesired ways, which is why fine-tuning them on a more specific task is often done after pre-traning.

This issue also nicely illustrates the difference between intrinsic evaluation (in this case, perplexity) and extrinsic evaluation (in this case, text generation). While we are holistically and subjectively evaluating the quality of the generated text, it is quite clear that the generated text from the RNN model is of inferior quality, despite its much lower perplexity. In short, it can happen that improvements in an intristic evaluation metric do not translate to better performance in an extrinsic evaluation task.

(e) When training for 20 epochs as done with the FNN, the RNN achieves a PPL of about 350. We can see from PPL on training data that the model is indeed converging, as the PPL decreased slower in the last epochs. The model is not fully converged yet, so there is potential room for improvement, but the remaining improvement is likely not significant.