Advanced Methods in Text Analytics
Exercise 3: Language Models - Part 1

Daniel Ruffinelli

FSS 2025

1 Language Modeling

Language models are able to predict the next word in a given sequence of n words. Similarly, language models can give the probability of a given sequence of n words.

(a) Give a formal expression for the probability of word $n + 1$ given the previous n words in a sequence.

(b) Give a formal expression for the probability of a sequence of n words and specify how this probability is computed using the marginal probabilities of each word in the sequence.

(c) What is the main assumption that n-gram language models make? What is the name of this assumption? What is a bigram model? And a trigram model? Give the expressions for computing the probability of a sequence of words (i.e. your answer to b)) using a bigram model, and a trigram model.

(d) How is the probability of predicting the next word given previous words *estimated* in n-gram models?

(e) What are some of the problems that n-gram language models have?

(f) Neural language models can address some of the issues with n-gram models. Describe the neural language model designed by Bengio et al., which was discussed in the lecture. What kind of neural network was it? How many hidden layers did it have? How did they represent the input sequence? How did it compute the relevant probabilities for language modeling? How was it parameterized? And how many parameters does it have with a vocabulary V and word embedding of size d? For simplicity, you may ignore the optional linear layer and all biases.

(g) What is *self-supervision*? Describe the self-supervised training approach used by Bengio et al. to train their neural language model.

(h) What problems in n-gram models were no longer an issue with neural language models? And what problems still persisted with such models? Explain why the existing limitations are indeed limitations.

2 Language Model Evaluation

In this task, we discuss how language models (LMs) can be evaluated.

(a) LMs are often used as part of various *downstream tasks.* Give two examples of downstream tasks that may use LMs in their pipeline, and describe how may use LMs.

(b) LMs can be evaluated extrinsically and intrinsically. Describe the difference between intrinsic and extrinsic evaluation. Discuss some advantages and disadvantages of each.

(c) Say you are tasked with designing the simplest intrinsic evaluation for language models using the basic machine learning principle of *held-out data* and a given text corpus. What sort of evaluation would you propose?

(d) The most common form of intrinsic evaluation of language models is computing its *perplexity* on held-out data. Perplexity is defined as follows:

$$ppl(w_1, w_2, \dots, w_n) = p(w_1, w2, \dots, w_n)^{-\frac{1}{n}},$$

where p is the *likelihood* of held-out sequence $x_1, x_2, \dots, x_n$. Describe the intuition of perplexity from the perspective of the likelihood of the data. How does one change when the other changes?

(e) Assume a uniform unigram model over some vocabulary V. What probability does such a model assign to each word in the vocabulary? Compute the likelihood of this model and use it to compute its perplexity over a held-out sequence of n words.

3 Recurrent Neural Networks

In this section, we review some basic concepts about RNNs, including how they can be used to implement language models.

(a) Describe the components of an RNN and discuss how it differs from fully-connected neural networks. What are the parameters of an RNN? Are RNNs deep? In what way(s)?

(b) How can a RNN be used for a sequence classification task? And what about for a task where each token in the sequence requires a predicted label, e.g. part-of-speech (POS) tagging? Describe the input, output and general architecture of the model.

(c) How can RNNs be used for language modeling? Describe the model's architecture. What is the fundamental problem that RNNs solve over fully-connected neural networks for language modeling?

(d) What are bidirectional RNNs? What are their advantages and disadvantages compared to standard (unidimensional) RNNs?

(e) Describe the main problem with RNNs and how LSTMs were designed to address this issue. Give a high-level intuition for each of the gates in an LSTM unit.

(f) Describe an encoder-decoder architecture and explain how it can be used for machine translation, i.e. the task of predicting a sequence of words in a target language given a sequence of words in a source language. When implementing such an architecture with RNNs, what is the input and output of the hidden state in the encoder at each time step? And in the decoder?

4 Sampling for Text Generation

An important aspect of using language models to (autoregressively) generate text is the step of *sampling* the next word from the distribution over words in our vocabulary that is produced by a language model.

(a) What is autoregressive text generation? What are its advantages compared non-autoregressive text generation? And its disadvantages?

(b) In a machine translation setting, let $X = \{x_1, x_2, \ldots, x_n\}$ be the input sequence in some source language, $Y = \{y_1, y_2, \ldots, y_m\}$ the output sequence in some target language, and $p(y_t|X, \boldsymbol{\theta})$ the distribution for the t-th word in a non-autoregressive setting using a model parameterized by $\boldsymbol{\theta}$. Give the corresponding expression for p in an autoregressive model.

(c) Let sample(p) be a function to sample a word from the autoregressive expression for p you provided in the previous question, and $< s >, < /s >$ be the start-of-sequence and end-of-sequence tokens, respectively. Write an algorithm in pseudocode to autoregressively generate a sequence of words of arbitrary length.

(d) The most straightforward way to implement the *sample* function used in the previous question is by choosing the word with the highest probability. This is known as *greedy* sampling. Can you think of advantages and disadvantages to such an approach? Use the following two incomplete sentences to reason about this.

  (i) The capital of Germany is ...

  (ii) The title of my talk is ...

(e) Another way to implement the *sample* function is by *randomly* sampling from p. This means we sample a word w with probability $p(w)$, so that words with higher probability get sampled more often, while words with lower probability get sampled less often. Can you think of advantages and disadvantages to such an approach? **Hint:** the size of the vocabulary is often very large, and for a given input sequence, the distribution over words is very skewed.

(f) Can you think of an approach that can be used to sample from the distribution over words in a more balanced way compared to both *greedy* and *random* sampling? What are its advantages and disadvantages compared to the approaches described in the previous questions?

(g) *Temperature* sampling is another approach, where instead of truncating a model distribution, we reshape it. It is defined in the context of softmax distributions as follows:

$$\boldsymbol{y} = \text{softmax}\left(\frac{\boldsymbol{u}}{\tau}\right),$$

where $\boldsymbol{u}$ is the unnormalized distribution over words, i.e. a vector of logits, τ is the temperature parameter, and we do an element-wise division with it. Describe the impact that different values of τ have over the resulting model distribution that we will then use to sample the next word from. Specifically, how does the distribution change as τ is less than 1 and approaches zero? And as it approaches infinity?