Predicting the Future

Lecture 8: Language Models

Harito

September 15, 2025

What is a Language Model?

Definition

A **Language Model (LM)** is a probability distribution over sequences of words. It assigns a probability to a sequence of words, or, more commonly, predicts the probability of the next word in a sequence given the preceding words.

Mathematically, an LM computes $P(W_1, W_2, \ldots, W_m)$ for a sequence of words $W_1, \ldots, W_m)$.

Example:

- $P(\text{The cat sat on the mat})$
- $P(\text{mat}|\text{The cat sat on the})$

Applications of Language Models

LMs are fundamental to many NLP tasks:

- **Speech Recognition**: Choosing the most likely word sequence given an acoustic signal.
- **Machine Translation**: Selecting the most probable translation.
- **Text Generation**: Generating coherent and grammatically correct text.
- **Spelling Correction**: Suggesting the most likely correct word.
- **Information Retrieval**: Ranking search results.

The Chain Rule of Probability

To compute the probability of a sequence of words, we use the chain rule:

$$P(W_1, \ldots, W_m) = P(W_1)P(W_2|W_1)P(W_3|W_1, W_2)\ldots P(W_m|W_1, \ldots, W_{m-1})$$

This means we need to estimate the probability of each word given all preceding words.

However, estimating $P(W_i|W_1, \ldots, W_{i-1})$ is hard due to data sparsity (too many possible sequences).

The Markov Assumption

To simplify, N-gram models make the **Markov Assumption**:

The probability of a word depends only on the previous $N-1$ words.

So, instead of $P(W_n|W_1,\ldots,W_{n-1})$, we approximate it as:

$$P(W_n|W_1,\ldots,W_{n-1}) \approx P(W_n|W_{n-(N-1)},\ldots,W_{n-1})$$

Examples:

- **Bigram (N=2)**: $P(W_n|W_{n-1})$
- **Trigram (N=3)**: $P(W_n|W_{n-2},W_{n-1})$

N-gram Models

N-gram is a contiguous sequence of N items from a given sample of text or speech.

Types of N-grams:

- **Unigram (N=1)**: Individual words. $P(W_n)$
- **Bigram (N=2)**: Pairs of words. $P(W_n|W_{n-1})$
- **Trigram (N=3)**: Triplets of words. $P(W_n|W_{n-2}, W_{n-1})$

We often add special start-of-sentence ($< s >$) and end-of-sentence ($< /s >$) tokens.

Example: "I love NLP." $\rightarrow < s >$ I love NLP $< /s >$

Maximum Likelihood Estimation (MLE)

How do we estimate the probabilities for N-gram models?

MLE: Count and Divide

The probability of an N-gram is estimated by its frequency in the training corpus.

For a bigram $P(W_n|W_{n-1})$:

$$P(W_n|W_{n-1}) = \frac{\text{Count}(W_{n-1}W_n)}{\text{Count}(W_{n-1})}$$

Example Corpus: "I love NLP. I love programming."

- $\text{Count}(\text{"love NLP"}) = 1$
- $\text{Count}(\text{"love"}) = 2$
- $P(\text{NLP}|\text{love}) = 1/2 = 0.5$

The Problem of Zero Probabilities

What if an N-gram never appeared in our training corpus?

- $P(\text{fantastic}|\text{I love}) = 0/\text{Count}(\text{I love}) = 0$
- This is problematic: a single zero probability makes the entire sequence probability zero.

Solution: Smoothing

Smoothing techniques adjust the MLE probabilities to give some probability mass to unseen N-grams.

Add-one Smoothing (Laplace Smoothing):

$$P(W_n|W_{n-1}) = \frac{\text{Count}(W_{n-1}W_n) + 1}{\text{Count}(W_{n-1}) + V}$$

Where V is the size of the vocabulary.

Time for Lab 8!

Objective:

- Implement an ‘NgramLanguageModel‘.
- Train it on a corpus.
- Predict the next word and (bonus) generate text.