Language Models

Language Modeling

Language Model: A probability distribution over a sequence of words.

Assigns a probability to a sequence or phrase.

Can be used for predicting the next word in a sentence.

Eg. Fill in the blank:

its water is so transparent that _______

Language Modeling

Applications:

- Machine Translation:
  - P(**high** winds tonight) > P(**large** winds tonight)
- Spelling Correction
- Speech Recognition (audio to text)
- Summarization (long texts -> short texts)
- Dialogue systems (convert input -> response)

Language Modeling

Language models can be quite complex.

BERT and GPT-3 are two examples of very powerful language models that are built using deep learning.

Today, we'll look at a simpler type of language model, the **n-gram model**.

N-gram Model

Big Idea: Can we find the probability of the next word *w* in a sentence given some history *h*?

$$P(w \mid h)$$

Eg. What word most likely fills in the blank?

its water is so transparent that ______

the? -> P(the | its water is so transparent that)

she? -> P(she | its water is so transparent that)

N-gram Model

Example:

P(the|its water is so transparent that)

This could be estimated using counts (if a large enough corpus is available):

$$P(the|its\ water\ is\ so\ transparent\ that) = \frac{C(its\ water\ is\ so\ transparent\ that\ the)}{C(its\ water\ is\ so\ transparent\ that)}$$

N-gram Model

Problems with a count-based approach:

- We need a *very* large corpus for it to work.
- Language is creative, and new sentences are created all the time, so you'll have a lot of zero counts for what could be plausible sentences.

Instead, we can estimate this probability using an n-gram approach.

N-gram Model

Given a sequence with *n* words ($w_1, w_2, \ldots, w_n$), how do we find the probability of that sequence?

Recall from probability:

$$P(A \text{ and } B) = P(A) \cdot P(B \mid A)$$

N-gram Model

Given a sequence with n words ($w_1, w_2, \ldots, w_n$), how do we find the probability of that sequence?

Recall from probability:

$$P(A \text{ and } B) = P(A) \cdot P(B \mid A)$$

This can be extended (**chain rule of probability**):

$$P(A \text{ and } B \text{ and } C) = P(A) \cdot P(B \mid A) \cdot P(C \mid A \text{ and } B)$$

N-gram Model

Let $X_{1:k} = X_1 X_2 \ldots X_k$.

N-gram Model

Let $X_{1:k} = X_1X_2\ldots X_k$.

$$P(X_1 \ldots X_n) \;=\; P(X_1)P(X_2|X_1)P(X_3|X_{1:2})\ldots P(X_n|X_{1:n-1})$$

N-gram Model

Let $X_{1:k} = X_1 X_2 \ldots X_k$.

$$\begin{aligned} P(X_1 \ldots X_n) &= P(X_1)P(X_2|X_1)P(X_3|X_{1:2})\ldots P(X_n|X_{1:n-1}) \\ &= \prod_{k=1}^{n} P(X_k|X_{1:k-1}) \end{aligned}$$

N-gram Model

Let $X_{1:k} = X_1 X_2 \ldots X_k$.

$$
\begin{aligned}
P(X_1 \ldots X_n) &= P(X_1)P(X_2|X_1)P(X_3|X_{1:2}) \ldots P(X_n|X_{1:n-1}) \\
&= \prod_{k=1}^{n} P(X_k|X_{1:k-1})
\end{aligned}
$$

P(its water is so transparent) =

P(its)·P(water | its)·P(is | its water)·P(so | its water is)·P(transparent | its water is so)

N-gram Model

Let $X_{1:k} = X_1 X_2 \ldots X_k$.

$$
\begin{aligned}
P(X_1 \ldots X_n) &= P(X_1)P(X_2|X_1)P(X_3|X_{1:2}) \ldots P(X_n|X_{1:n-1}) \\
&= \prod_{k=1}^{n} \boxed{P(X_k|X_{1:k-1})}
\end{aligned}
$$

What if we don't use the whole history at each word?

Unigram Model:

$$P(w_{1:n}) \approx \prod_{i=1}^{n} P(w_i)$$

P(its water is so transparent) = P(its)·P(water)·P(is)·P(so)·P(transparent)

Unigram Model:

$$P(w_{1:n}) \approx \prod_{i=1}^{n} P(w_i)$$

P(its water is so transparent) = P(its)·P(water)·P(is)·P(so)·P(transparent)

What are the advantages and disadvantages of this approach?

N-gram Model

The n-gram model says instead of using the full history, estimate using just the last few words.

Eg. the bigram model uses (**Markov assumption**):

$$P(X_n|X_{1:n-1}) \approx P(X_n|X_{n-1})$$

P(its water is so transparent) =

$$P(\text{its})\cdot P(\text{water} \mid \text{its})\cdot P(\text{is} \mid \text{water})\cdot P(\text{so} \mid \text{is})\cdot P(\text{transparent} \mid \text{so})$$

Bigram Model:

P(its water is so transparent) =

$$P(\text{its}) \cdot P(\text{water} \mid \text{its}) \cdot P(\text{is} \mid \text{water}) \cdot P(\text{so} \mid \text{is}) \cdot P(\text{transparent} \mid \text{so})$$

N-gram Model

The trigram model uses:

$$P(X_n|X_{1:n-1}) \approx P(X_n|X_{n-2:n-1})$$

P(its water is so transparent) =

$$P(\text{its})\cdot P(\text{water} \mid \text{its})\cdot P(\text{is} \mid \text{its water})\cdot P(\text{so} \mid \text{water is})\cdot P(\text{transparent} \mid \text{is so})$$

N-gram Model

This assumption results in (for the bigram model), the estimate that:

$$P(w_{1:n}) \approx \prod_{k=1}^{n} P(w_k | w_{k-1})$$

This allows us, instead of counting phrases to count bigrams.

$$P(w_n | w_{n-1}) = \frac{C(w_{n-1} w_n)}{C(w_{n-1})}$$

Evaluating Language Models

A common way to evaluate a language model is **perplexity**, which looks at the estimated inverse probability of a test set (lower perplexity is better).

$$
\begin{aligned}
\mathrm{PP}(W) &= P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} \\
&= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}
\end{aligned}
$$

Evaluating Language Models

Notice that if our language model assigns a zero probability to any word that appears in the test set, perplexity becomes undefined.

$$
\begin{aligned}
\mathrm{PP}(W) &= P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} \\
&= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}
\end{aligned}
$$

Language Models - Smoothing

Laplace smoothing/add-one smoothing: add one to all the n-gram counts.

$$P_{\text{Laplace}}(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)+1}{\sum_w (C(w_{n-1}w)+1)} = \frac{C(w_{n-1}w_n)+1}{C(w_{n-1})+V}$$

Backoff: Use a lower order n-gram if there are zero counts.

Interpolation: Mix the probability estimates at different levels.

Kneser-Ney Smoothing