Narrative Portion- Prachi Patel and Kim Ngo

1. N-grams are a sliding window of words/letters. They can range from 1 to more tokens. These n-grams are generally collected from a text or corpus. They are used to help understand the probability of a certain word that may appear after another word. The n-grams are used in statistical language processing, because with a large corpus, one can generally accurately predict the probability of the succeeding word/letter.
2. We can use n-grams in processes like language generation, in which given a few words, the computer has to generate the next word/words. We can also use n-grams in things like spelling error detection, autocompletion of sentences, semantic analysis, and language detection.
3. When calculating the probability of a unigram, all you have to do is count the number of times the unigram appear in the corpus and divide by the length of the corpus. Essentially, you are getting the percentage of times a word appears out of the total words in the text. When calculating the probability of a bigram, you have to calculate the percent of times, the bigram appears in the corpus, which is the number of times the two words appear together divided by the length of the corpus. You also calculate the percentage of times the second word appear in the corpus. Then you divide the just bigram percentage, by the unigram percentage to get the probability of the bigram.
4. Language models learn from a source text and apply learned patterns to new texts. For ngrams, the source text is used to calculate the most likely sequence of words following any given word in the source text vocabulary. If the source text is large and representative of future texts, it is likely that predictions made by the model will be accurate.
5. Sometimes, we want to calculate the probability of an ngram in testing text that never occurred in the training corpus. Without smoothing, this will always result in a likelihood of zero. Zero-probability n-grams can cause problems with language models. For example, we may have two n-grams which both contain a bigram which never appeared in the training corpus. Say that, excluding this bigram, these n-grams have significantly different probabilities. Without smoothing, our language model will incorrectly assign both n-grams the same probability: zero. This can also be interpreted as a case of overfitting, one which smoothing can help avoid. To avoid this, we can treat novel n-grams as highly unlikely, instead of impossible. There are several methods to smooth probability calculations in language models, the simplest being Laplace smoothing or add-one smoothing. Laplace smoothing is accomplished by incrementing the count of a word by one during probability calculations. By doing this, we ensure that our language model will never consider any combination of words impossible.
6. Language models can be used for text generation by continuing text. Using bigrams, we consider a context of one word, calculate the most likely next word, append it, and repeat. We can do the same with trigrams and calculate the most next likely word given a context of two words, and so on. Generating text with language models has limitations, however. First, the output text may be nonsensical. If text is generated using n-grams, such that the text has many more words than n, the text at the end will not take into context the text from the beginning. This can lead to contradicting statements. To reduce this issue, n can be increased, but only so far. As n approaches the number of words in the training corpus, less and less n-grams are created, and the ones that are will likely be specific to the training corpus. This will result in overfitting. Similar problems face other language models which incorporate context, and it is a fundamental limitation of text generalization with language models.
7. Language models can be evaluated intrinsically or extrinsically. For example, an extrinsic evaluation may consist of human annotation of generated text, while an intrinsic evaluation may consist of one or more internal metrics, such as perplexity.
8. Google’s n-gram viewer allows users to see the occurrence of a given n-gram over time in their in-house corpus. It can be used by navigating to the page and entering a term.

Graphical user interface, application

Description automatically generated