# Sumarizzing

Programming Club

Introduction to NLP(1/3)

## Content to be covered...

1. N-gram models
2. Sentiment Analysis (using Logistic Regression and Naive Bayes)

## Content to be covered..

1. N-gram models

2. Sentiment Analysis (using Logistic Regression and Naive Bayes)

## N-gram models

A bit about text-generation:

- We shall discuss models that assign probability to each next token.
- The models discussed here are based on a greedy approach to generate text. Later we shall discuss more sophisticated techniques for tex generation once we have the probability distribution over the **corpus**.
- However, firstly onto obtaining this probability distribution.
- This distribution is important for almost all NLP sub-routines (speech recognition, machine translation etc.)

- One way of assigning probabilities can be:

$$P(w|h) = \frac{C(w,h)}{C(h)}$$

  here $w$ is the word, $h$ is the history. However, since language is creative, this way of assigning probability is very rigid, and fails.
- Consider the following probability of an entire sequence of words (**chain rule of probability**):

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

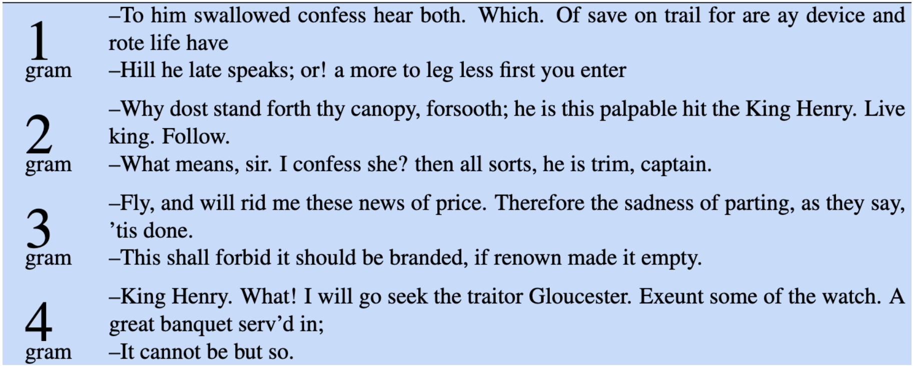
- The intuition behind **n-gram models** is that, this probability can be approximated by only certain previous tokens, instead of taking the entire history. For eg. a **bigram model** will approximate $P(w_n|w_{1:n-1}) = P(w_n|w_{n-1})$

- This assumption that the probability depends on only the previous state is called the **Markov** assumption. (read about Markov chains a bit, if time permits)
- Generalizing, $P(w_n|w_{1:n-1}) \approx P(w_n|w_{w-N+1:n-1})$ for a N-gram model.
- An intuitive way to obtain these probabilities is to use **Maximum Likelihood Estimation (MLE)**, i.e., hetting counts from the corpus and normalizing it. For eg.:

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

- We extend this approximation and estimation to find the probability of the entire sequence.

- Eg. Some n-gram model trained on a some Shakespeare:



| | |
|---|---|
| 1 gram | –To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have<br>–Hill he late speaks; or! a more to leg less first you enter |
| 2 gram | –Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow.<br>–What means, sir. I confess she? then all sorts, he is trim, captain. |
| 3 gram | –Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, ’tis done.<br>–This shall forbid it should be branded, if renown made it empty. |
| 4 gram | –King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv’d in;<br>–It cannot be but so. |

- The model $P$ must not have any knowledge of the test set, or the vocab of the test set. (What is a test set in this scenario: we measure how well the model performs based off of how high a probability it assigns to test sentences, apart from seeing coherent sentence generation).
- Things that do not occur in the training set, but occur in the test set, would be assigned 0 probability, and hence the whole test set would be assigned a zero probability (multiplication of these probabilities), we need a way to handle these instances.
  - Unknown words
  - Smoothing

## Unknown words

- In a closed vocabulary system, test data can only take items from the vocab defined by the training set, or the vocab defined.
- However, to deal with out-of-vocabulary (OOV) words, we might use a new token $< UNK >$ to put all the unseen words and/or words below a particular threshold.

## Smoothing

- To deal with words that are in our vocabulary, but appear in the test set with unseen history (context). To keep the language model from assigning our language model from assigning zero probability to these unseen events, a bit of probability mass is shoved from more frequent events and given to events never seen before. This is called **smoothing** or **discounting**.
- **LAPLACE SMOOTHING**: All the counts that were used to be zero, will now have a count of 1.

## Laplace Smoothing

$$\begin{aligned}
P(w_n|w_{n-1}) &= \frac{C(w_n, w_{n-1})}{C(w_{n-1})} \\
P_{laplace}(w_n|w_{n-1}) &= \frac{C(w_n, w_{n-1}) + 1}{\sum_w (C(w, w_{n-1}) + 1)} \\
&= \frac{C(w_n, w_{n-1}) + 1}{C(w_{n-1}) + V} \\
giving, \\
c^*(w_n, w_{n-1}) &= \frac{[C(w_n, w_{n-1}) + 1] * C(w_{n-1})}{C(w_{n-1}) + V}
\end{aligned}$$

as the new apparent count.

## Backoff and Interpolation

- If we are trying to estimate $P(w_n|w_{n-1}w_{n-2})$ and we don't have any example of trigram $w_{n-2}w_{n-1}w_n$, we estimate it using the bigram $P(w_n|w_{n-1})$.
- In **backoff** we use the trigram (or a higher gram model) if evidence is sufficient, otherwise the bigram, otherwise the unigram.
- In **interpolation** we use a mix of probability estimates from all the lower n-grams:

$$P(w_n|w_{n-2}w_{n-1}) = \lambda_1 P(w_n) + \lambda_2 P(w_n|w_{n-1}) + \lambda_3 P(w_n|w_{n-2}w_{n-1})$$

  such that $\sum_i \lambda_i = 1$ These $\lambda's$ can also be context conditioned. They are learned from a validation set.
- (Context-conditioned backoff with discounting- **Katz-backoff**:

$$P_{BO}(w_n|w_{n_{N+1:n-1}}) = \left\{ \begin{array}{cc} P^*(w_n|w_{n-N+1}) & C(w_{n-N+1}) > 0 \\ \alpha(w_{n-N+1}) P_{BO}(w_n|w_{n-N+2}) & otherwise \end{array} \right\}$$

## Kneser-Ney Discounting

- (**Absolute discounting**): To give sme weight to unseen events we cut some slack from the prevelant n-grams and subside. This idea is formalized in absolute discounting, by subtracting a fixed amount *d* (=0.75 generally), from these counts. (Absolute discounting generally applied to smaller gram models, trigrams and bigrams).

$$P_{absoluteDiscounting}(w_i|w_{i-1}) = \frac{C(w_{i-1}w_i) - d}{\sum_v C(w_{i-1}v)} + \lambda(w_{i-1})P(w_i)$$

- Finally, in Kneser Ney discounting, another term is included **continuation probability**: how likely is w to appear as a novel continuation.

$$P_{continuation}(w) = \frac{|\{v : C(vw) > 0\}|}{\sum_{w'} |\{v : C(vw') > 0\}|}$$

$$P_{KN}(w_i|w_{i-1}) = \frac{max((C(w_{i-1}w_i) - d), 0)}{C(w_{i-1})} + \lambda(w_{i-1})P_{continuation}(w_i)$$

## Content to be covered..

1. N-gram models
2. Sentiment Analysis (using Logistic Regression and Naive Bayes)

## Naive Bayes

- In a supervised situation, we have a training set of $N$ documents, each of which have been hand-labeled with a class. $\{(d_1, c_1), (d_2, c_2)...(d_N, c_N)\}$. Our goal is to learn a classifier capable of mapping from a new document $d$ to its correct class $c \in C$ where $C$ is a set of useful document classes.
- Bag of Words: representation of text documents (read about it)
- $$\begin{aligned}\hat{c} &= \arg\max_{c \in C} P(c|d) \\ &= \arg\max_{c \in C} \frac{P(d|c).P(c)}{P(d)} \\ &= \arg\max_{c \in C} P(d|c).P(c) \\ &= \arg\max_{c \in C} P(f_1, f_2, ..., f_n|c).P(c)\end{aligned}$$

  the first term of the product is the likelihood and the second is called the prior, and where $f_i$'s are the features of the document.

## Naive Bayes

- In a bag of words model, these features are the words itself and position does not matter in this model thus it reduces to the following:

$$\begin{aligned} c_{NB} &= \arg\max_{c \in C} P(c). \prod_{f \in F} P(f|c) \\ &= \arg\max_{c \in C} P(c). \prod_{i} P(w_i|c) \\ &= \arg\max_{c \in C} \log P(c) + \sum_{i} \log P(w_i|c) \end{aligned} \tag{1}$$

## Logistic Regression

- We have discussed Logistic regression before. Read about regularization, priors, posterior, predictive posterior and simple correlation of these terms in the Gaussian context.

Thank you!