CHAPTER 3: N-gram Language Models

Instructor: PhD. Nguyen Thi Quy

Group 5: 3.4 - 3.5

November 15, 2023

Presentation Overview

Presentation Overview

Generalization and Zeros

N-gram model is dependent on training corpus.

Implication

- Probabilities encode specific facts about training corpus.
- N-grams do better job of modeling as we increase N.

Approximating Shakespeare

1 gram	–To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have
	–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.
	–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.
	–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;
	–It cannot be but so.

Figure 3.4 Eight sentences randomly generated from four n-grams computed from Shakespeare's works. All characters were mapped to lower-case and punctuation marks were treated as words. Output is hand-corrected for capitalization to improve readability.

Shakespeare as corpus

- $N = 884,647$
- $V = 29,066$
- Possible bigrams $V^2 = 844,000,000$
- Possible 4-grams $V^4 = 7 \times 10^{17}$

The Wall Street Journal

1 gram	Months the my and issue of year foreign new exchange's september were recession exchange new endorsed a acquire to six executives
2 gram	Last December through the way to preserve the Hudson corporation N. B. E. C. Taylor would seem to complete the major central planners one point five percent of U. S. E. has already old M. X. corporation of living on information such as more frequently fishing to keep her
3 gram	They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions

Figure 3.5 Three sentences randomly generated from three n-gram models computed from 40 million words of the *Wall Street Journal*, lower-casing all characters and treating punctuation as words. Output was then hand-corrected for capitalization to improve readability.

Generalization and Zeros

P: Statistical models are useless as predictors if training sets and test sets are different.

How should we deal with this problem when building n-gram models?

Generalization and Zeros

S1: Use a training corpus that has a similar **genre** to the task.

Generalization and Zeros

S1: Use a training corpus that has a similar **genre** to the task.

Example

- translating legal documents - legal documents
- question-answering system - questions

Generalization and Zeros

S2: Get training data in the appropriate **dialect** or **variety**, especially when processing social media posts or spoken transcripts.

Generalization and Zeros

S2: Get training data in the appropriate **dialect** or **variety**, especially when processing social media posts or spoken transcripts.

Example

Some tweets use features of **African American Language** (**AAL**)

- (3.19) Bored af den my phone finna die!!!
- (3.20) @username R u a wizard or wat gan sef: in d mornin - u tweet, afternoon - u tweet, nyt gan u dey tweet. beta get ur IT placement wiv twitter

Zeros

Our models may still be subject to the problem of **sparsity**.

Zeros

Our models may still be subject to the problem of **sparsity**.

Example

Training set:
denied the allegations
denied the speculation
denied the rumors
denied the report

Test set:
denied the offer
denied the loan

$$P(\text{offer}|\text{denied the}) = 0$$

Zeros

Our models may still be subject to the problem of **sparsity**.

Example

Training set:
denied the allegations
denied the speculation
denied the rumors
denied the report

Test set:
denied the offer
denied the loan

$$P(\text{offer}|\text{denied the}) = 0$$

These **zeros**:

- things that don't ever occur in the training set
- but do occur in the test set

Reasons

First,

- Underestimate the probability of all sorts of words that might occur
- Hurt the performance of any application we want to run on this data

Reasons

First,

- Underestimate the probability of all sorts of words that might occur
- Hurt the performance of any application we want to run on this data

Second,

- Zero probability n-grams, the entire probability of the test set is 0.
- Can't compute perplexity (can't divide by 0)

Unknown Words

- Stipulating that we already know all the words that can occur.
- In **closed vocabulary** system the test set can only contain words from known lexicon.

Unknown Words

- In real situations, we have to deal with **unknown** words, or **out of vocabulary** (**OOV**) words.
  - **OOV rate**: The percentage of OOV words that appear in test set
- Create **open vocabulary** system: model potential unknown words in test set by adding a pseudo-word called $< UNK >$.

Train the probabilities of the unknown word model $< UNK >$

The first one:
Turn the problem back into a closed vocabulary one

Train the probabilities of the unknown word model $< UNK >$

The first one:

Turn the problem back into a closed vocabulary one

1. **Choose** a vocabulary (word list) that is fixed in advance.

Train the probabilities of the unknown word model $< UNK >$

The first one:

Turn the problem back into a closed vocabulary one

1. **Choose** a vocabulary (word list) that is fixed in advance.
2. **Convert** in the training set any OOV word to the unknown word token $< UNK >$ in a text normalization step.

Train the probabilities of the unknown word model $< UNK >$

The first one:

Turn the problem back into a closed vocabulary one

1. **Choose** a vocabulary (word list) that is fixed in advance.
2. **Convert** in the training set any OOV word to the unknown word token $< UNK >$ in a text normalization step.
3. **Estimate** the probabilities for $< UNK >$ from its counts just like any other regular word in the training set.

Train the probabilities of the unknown word model $< UNK >$

The second alternative:

- We don't have a prior vocabulary in advance.
- Create a vocabulary implicitly.
- Replace words in the training data by $< UNK >$ based on their frequency.

Unknown Words

- Choice of $< UNK >$ affects metrics like perplexity.
- Small vocabulary with high $< UNK >$ probability leads to low perplexity.
- Perplexities can only be compared across language models with the same vocabularies.

Presentation Overview

The intuition of smoothing

When we have sparse statistics:

$P(w \mid \text{denied the})$

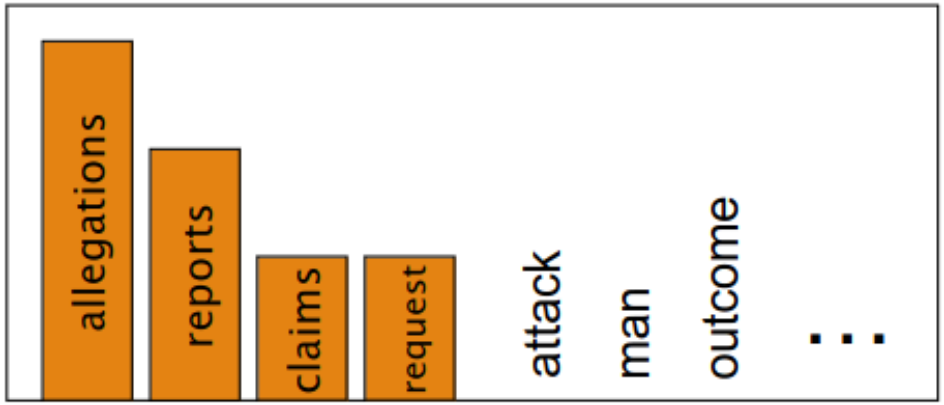
3 allegations

2 reports

1 claims

1 request

7 total



Steal probability mass

$P(w \mid \text{denied the})$

2.5 allegations

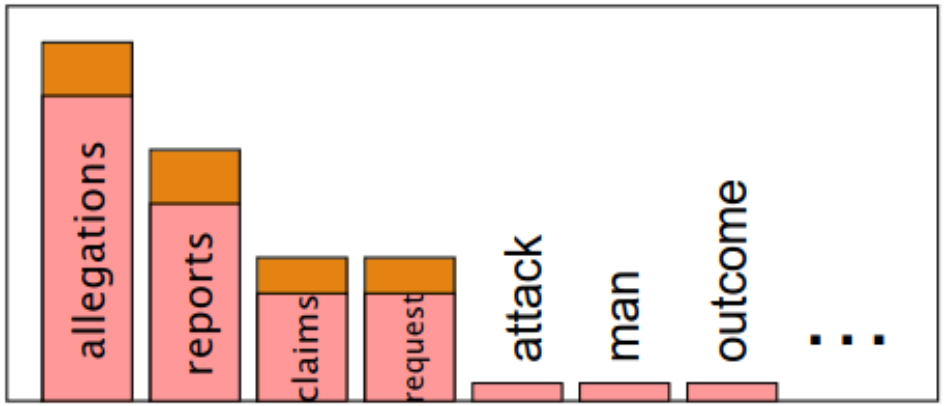
1.5 reports

0.5 claims

0.5 request

2 other

7 total



Add-one estimation

Unsmoothed maximum likelihood estimate

$$P(w_i) = \frac{c_i}{N}$$

Laplace smoothing merely adds one to each count:

$$P_{Laplace}(w_i) = \frac{c_i + 1}{N + V}$$

Add-one estimation

It is convenient to describe how a smoothing algorithm affects the numerator, by defining an adjusted count c^*:

$$c_i^* = (c_i + 1)\frac{N}{N + V}$$

A related way to view smoothing is as discounting (lowering) some non-zero counts in order to get the probability mass that will be assigned to the zero counts:

$$d_c = \frac{c^*}{c}$$

Add-one estimation

- MLE estimate:

$$P_{MLE}(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

- Add-1 estimate:

$$P_{Add-1}(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i) + 1}{c(w_{i-1}) + V}$$

Berkeley Restaurant Corpus

Raw bigram counts

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Figure: Bigram counts for eight of the words (out of V = 1446) in the Berkeley Restau- rant Project corpus of 9332 sentences. Zero counts are in gray.

Berkeley Restaurant Corpus

Laplace smoothed bigram counts

	i	want	to	eat	chinese	food	lunch	spend
i	6	828	1	10	1	1	1	3
want	3	1	609	2	7	7	6	2
to	3	1	5	687	3	1	7	212
eat	1	1	3	1	17	3	43	1
chinese	2	1	1	1	1	83	2	1
food	16	1	16	1	2	5	1	1
lunch	3	1	1	1	1	2	1	1
spend	2	1	2	1	1	1	1	1

Laplace-smoothed bigrams

$$P^*(w_n|w_{n-1}) = \frac{c(w_{n-1}w_n) + 1}{c(w_{n-1}) + V}$$

	i	want	to	eat	chinese	food	lunch	spend
i	0.0015	0.21	0.00025	0.0025	0.00025	0.00025	0.00025	0.00075
want	0.0013	0.00042	0.26	0.00084	0.0029	0.0029	0.0025	0.00084
to	0.00078	0.00026	0.0013	0.18	0.00078	0.00026	0.0018	0.055
eat	0.00046	0.00046	0.0014	0.00046	0.0078	0.0014	0.02	0.00046
chinese	0.0012	0.00062	0.00062	0.00062	0.00062	0.052	0.0012	0.00062
food	0.0063	0.00039	0.0063	0.00039	0.00079	0.002	0.00039	0.00039
lunch	0.0017	0.00056	0.00056	0.00056	0.00056	0.0011	0.00056	0.00056
spend	0.0012	0.00058	0.0012	0.00058	0.00058	0.00058	0.00058	0.00058

Reconstituted counts

$$c^*(w_{n-1}w_n) = \frac{[C(w_{n-1}w_n)+1] \times C(w_{n-1})}{C(w_{n-1})+V}$$

	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

Figure: Add-one reconstituted counts for eight words (of V = 1446) in the BeRP corpus of 9332 sentences. Previously-zero counts are in gray

Berkeley Restaurant Corpus

Compare with raw bigram counts

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

	i	**want**	**to**	**eat**	**chinese**	**food**	**lunch**	**spend**
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

Add-k smoothing

Instead of adding 1 to each count, we add a **fractional** count k (.5? .05? .01?).

$$P^*_{Add-k}(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n) + k}{C(w_{n-1} + kV)}$$

Add-k smoothing requires that we have a method for choosing k; this can be done, for example, by optimizing on a **devset**.

Add-1 estimation is a blunt instrument

So add-1 isn't used for N-grams:

- We'll see better methods

But add-1 is used to smooth other NLP models

- For text classification
- In domains where the number of zeros isn't so huge.

Backoff and Interpolation

Sometimes it helps to use **less** context

- Condition on less context for contexts you haven't learned much about

Backoff

- Use trigram if you have good evidence
- Otherwise bigram or unigram

Interpolation

- Mix unigram, bigram, trigram

Linear Interpolation

Simple Interpolation

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

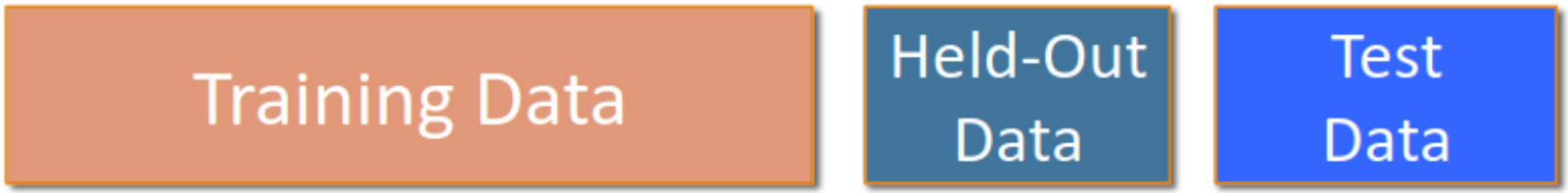
$$\sum_i \lambda_i = 1$$

Lambdas Conditional on Context

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

How to set the lambdas?

Use a **held-out** corpus



Choose λs to maximize the probability of held-out data:

- Fix the N-gram probabilities (on the training data)
- Then search for λs that give largest probability to held-out set:

$$\log P(w_1 \ldots w_n | M(\lambda_1 \ldots \lambda_k)) = \sum_i \log P_{M(\lambda_1 \ldots \lambda_k)}(w_i | w_{i-1})$$

Backoff

In a **backoff** n-gram model

- If the n-gram we need has zero counts, we approximate it by backing off to the (n-1)-gram.
- We have to discount the higher-order n-grams to save some probability mass for the lower order n-grams.
- If we don't, the total probability assigned to all possible strings by the language model would be greater than 1.

Katz Backoff

We'll need a function alpha to distribute this probability mass to the lower order n-grams.

$$P_{BO}(w_n|w_{n-N+1:n-1}) = \begin{cases} P^*(w_n|w_{n-N+1:n-1}), & \text{if } C(w_{n-N+1:n}) > 0 \\ \alpha(w_{n-N+1:n-1})P_{BO}(w_n|w_{n-N+2:n-1}), & \text{otherwise} \end{cases}$$

Reference

Speech and Language Processing (3rd ed. draft)

Dan Jurafsky and James H. Martin

Part I: Fundamental Algorithms, *Chapter 3: N-gram Language Models*

Thanks for listening!

Q&A section