LANGUAGE MODELS

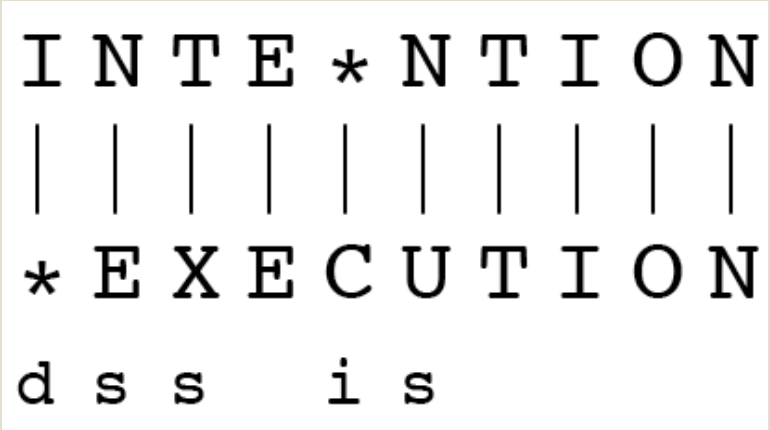
CS 445 - NATURAL LANGUAGE PROCESSING

Instructor: Dilara Keküllüoğlu
Fall 2024 – Week 4

Recap

1. Edit Distance
2. Text Analysis Steps
   - Dataset selection and exploration (Week 1)
   - Text Pre-processing (Week 1-2)
   - Feature Extraction (Week 4-5)
   - Classification (Week 5)
   - Evaluation (Week 5)
3. Sentiment Analysis Example

Attendance



Positive:

absolutely	beaming	calm
adorable	beautiful	celebrated
accepted	believe	certain
acclaimed	beneficial	champ
accomplish	bliss	champion
achieve	bountiful	charming
action	bounty	cheery
active	brave	choice
admire	bravo	classic
adventure	brilliant	classical
affirm	bubbly	clean
...		...

Negative:

abysmal	bad	callous
adverse	banal	can't
alarming	barbed	clumsy
angry	belligerent	coarse
annoy	bemoan	cold
anxious	beneath	collapse
apathy	boring	confused
appalling	broken	contradictory
atrocious		contrary
awful		corrosive
		corrupt
		...

From http://www.enchantedlearning.com/wordlist/

Slide references: Jurafsky and FNLP at UoE

Learning Goals (Week 4)

1. **Describe** language models
2. **Calculate** word-level, sentence-level prediction probabilities
3. **Define** N-gram models
4. **Evaluate** N-gram models

Introduction to Language Modeling

- N-gram language models – today
- Large language models – Week 12
  - Word prediction in large scale

Word Prediction

- The food they made in Masterchef Türkiye yesterday looked …

delicious
savory
complicated
flavorful

*unicorn
*wall
*car

Language Models

- Models that can predict upcoming words
  - Assigning a probability for each potential next word
  - Assigning a probability to a whole sentence

The food they made in Masterchef Türkiye yesterday looked …

- Given a language model in English, delicious would/should have more probability than wall as the next word

Why word prediction?

- Helpful tool for language tasks
- Grammar or spell checking
  - When there are two words with the same edit distance – which one to select
  - Get the one with higher probability of being the word considering surrounding words

    You're results are out - Your results are out.

- Speech recognition

  I will be bassoon dish - I will be back soonish

Why word prediction?

- The underlying mechanism for Large Language Models (LLMs)
- LLMs use lots of training data to predict words
  - Left-to-right (Autoregressive) LMs learn to predict next word
- LLMs generate text by predicting words
  - By predicting the next word over and over again
    - Sometimes causes them to get stuck repeating same things

Language Modeling (LM) Formalization

Goal: compute the probability of a sentence or sequence of words W:

$$P(W) = P(w_1, w_2, w_3, \ldots, w_n)$$

Goal: compute the probability of an upcoming word given previous n words:

$$P(w_n \mid w_1, w_2, w_3, \ldots, w_{n-1})$$

An LM computes either the sentence level prediction or the word prediction.

How to estimate probabilities?

Difference between probability theory vs estimation

- Probability theory can solve problems like:
  - I have deck of cards with 5 hearts and 4 spades.
  - If I choose a card uniformly at random, what's the probability it's a hearts?

- Often we don't know the true probabilities, only have data:
  - I have a deck of cards..
  - I repeatedly choose a card uniformly at random and then replace it before choosing again.
  - In ten draws, I get 6 hearts marbles and 4 spades.
  - On the next draw, what's the probability I get a spade card?

- First three facts are evidence.
- The question requires estimation theory.

How to estimate probabilities?

Should we count all occurrences and divide by the total? – Maximum-Likelihood Estimation (MLE)

P (delicious | The food they made in Masterchef Türkiye yesterday looked) =

$$\frac{C(\text{The food they made in Masterchef Türkiye yesterday looked delicious})}{C(\text{The food they made in Masterchef Türkiye yesterday looked})}$$

Sentences that have never occurred

Danilo chef devoured the cabbages hastily

vs

hastily pumpkins the loch walk

- Neither of these sentences show up in any corpus

→ C(sentence) = 0 for both of these cases if we use MLE

- However, one of them is grammatical and makes sense but other does not.
- Estimating probabilities by sentence-level MLE does not work well.

The problem with MLE

- Too many possible sentences and counts to compute
- Data sparsity
- We looked at corporas in the first week and there are many words which appear only once
- Do they have equal probability?
- Not enough observation data to estimate these probabilities by just counting
- Especially whole sentences - specific word order might not appear at all.

Compute P(W) or P(w_n | w_1, w_2, w_3,…, w_{n-1})

- Can we combine the smaller parts of the sentence to compute probability?
  - Reasoning behind the N-gram language models

- P(The, food, they, made, in, Masterchef, Türkiye, yesterday, looked, delicious)

We can use the Chain Rule of Probability

Chain Rule reminder

- Recall the definition of conditional probabilities

$$P(B \mid A) = P(A,B)/P(A) \rightarrow P(A,B) = P(A)\,P(B \mid A)$$

- More variables:

$$P(A,B,C,D) = P(A)\,P(B \mid A)\,P(C \mid A,B)\,P(D \mid A,B,C)$$

- The Chain Rule in General

$$P(x_1,x_2,x_3,\ldots,x_n) = P(x_1)P(x_2 \mid x_1)P(x_3 \mid x_1,x_2)\ldots P(x_n \mid x_1,\ldots,x_{n-1})$$

Joint probability of words in sentence with Chain Rule

$$
\begin{aligned}
P(w_{1:n}) &= P(w_1)P(w_2|w_1)P(w_3|w_{1:2})\ldots P(w_n|w_{1:n-1}) \\
&= \prod_{k=1}^{n} P(w_k|w_{1:k-1})
\end{aligned}
$$

Joint probability

P("The food they made in Masterchef Türkiye yestarday looked") =
P(The) x P(food | The) x P(they | The food) x P(made | The food they) x
P(in | The food they made) x P(Masterchef | The food they made in) x
P(Türkiye | The food they made in Masterchef) x
P(yesterday | The food they made in Masterchef Türkiye) x
P(looked | The food they made in Masterchef Türkiye yesterday)

P(looked | The food they made in Masterchef Türkiye yesterday)

Too many probabilities that may be sparse in the data!

Markov Assumption

- Future state relies solely on the current state – no past states considered
- Memoryless
- Simplifies the probability estimation

P(delicious | The food they made in Masterchef Türkiye yesterday looked) ≈ P(delicious | looked)

$$P(w_n|w_{1:n-1}) \approx P(w_n|w_{n-1})$$

N-gram Independence Assumption

- Instead of relying only the last word let's say the next words probability rely on the last N words

$$P(w_n|w_{1:n-1}) \approx P(w_n|w_{n-N+1:n-1})$$

- Unigram model – P(delicious)
- Bigram model – P(delicious | looked)
- Trigram model – P(delicious | yesterday looked)

Unigram model generation

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

Months the my and issue of year foreign new exchange’s September

were recession exchange new endorsed a acquire to six executives

Bigram model generation

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.

Last December through the way to preserve the Hudson corporation N. B. E. C. Taylor would seem to complete the major central planners one gram 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

Trigram Independence

- Let's say we use trigram assumption, the following sentences have equal probabilities:

P(delicious | Her food yesterday looked)
P(delicious | The cat yesterday looked)
P(delicious | The exam yesterday looked)

- All equal to P(delicious| yesterday looked)

- So it is not always a good assumption but it does reduce the data sparsity problem!

Coming back to estimation with counts

- In the example with trigram independence assumption,

P(delicious | yesterday looked) =

$$\frac{C(yesterday, looked, delicious)}{C(yesterday, looked)}$$

Beginning / End of Sequence

- What happens when we have short sequence of words for trigram modeling?
- What will happen at the start and end of the sequence?
- We can put identifiers for start and end - <s> and <\s>
- P(<\s> | the, short) would be low, signifying it is not a good sentence end.

Problems with N-gram models

- Trigram seems better than unigram or bigram
- Still N-gram models cannot handle long-distance dependencies

The **food** they made in Masterchef Türkiye yesterday **looked** delicious.

Solution: Large language models!

- can handle much longer contexts
- embedding – generate better novel strings

N-gram Probability Estimation

- Let's say we use MLE with Bigram model

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

An example

<s> I am Sam </s>
<s> Sam I am </s>
<s> I do not like green plums and salt </s>

$$P(\texttt{I}\,|\,\texttt{<s>}) = \frac{2}{3} = .67 \qquad P(\texttt{Sam}\,|\,\texttt{<s>}) = \frac{1}{3} = .33 \qquad P(\texttt{am}\,|\,\texttt{I}) = \frac{2}{3} = .67$$

$$P(\texttt{</s>}\,|\,\texttt{Sam}) = \frac{1}{2} = 0.5 \qquad P(\texttt{Sam}\,|\,\texttt{am}) = \frac{1}{2} = .5 \qquad P(\texttt{do}\,|\,\texttt{I}) = \frac{1}{3} = .33$$

Berkeley Restaurant Project sentences

- Raw bigram count out of 9222 sentences

	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

Raw bigram probabilities

- Let's normalize by unigrams:

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Sentence Probability

P(<s> I want english food </s>) =

P(I|<s>)

× P(want|I)

× P(english|want)

× P(food|english)

× P(</s>|food)

= .000031

Additional Notes

- Word probabilities are mostly small other than few common words (the, a, an, …)
- Multiplying small probabilities cause really tiny results where representation is problematic
- Negative log probabilities are used
  - Probabilities range from 0-1 – negative log probabilities range from 0 to ∞: lower cost = higher probability
  - Add on negative log probabilities instead of multiplying

N-gram LM Toolkits

- SRILM
  - http://www.speech.sri.com/projects/srilm/
- KenLM
  - https://kheafield.com/code/kenlm/

Evaluation of N-gram models

- How do we evaluate an N-gram model?
  - Is trigram model really better than bigram
- Extrinsic and Intrinsic evaluation methods

- Extrinsic methods:
  - Put both models in a real task
  - Compare the downstream tasks performance
- Expensive and time-consuming
- Does not always generalize to other applications – the performance might be OK for one application but not the other

Intrinsic Evaluation - Perplexity

- Directly measure the performance on word prediction
- Might not correlate with real application performance
- Single general metric to use for all
- The test data should look just like training data for it to be meaningful!
    - Not always the case.

Training and Test Sets

- Train parameters of the model on a training set
- We test the performance on data we haven't seen.
- No training on the test set – higher probability to seen n-grams – falsely high probability
- Overfitting to test set if we test too many times and iterate on the model
- Use test sets only once at the end
- Dev sets – Use these to test during development time

Choosing training data

- If task-specific, use a training corpus that has a similar genre to your task.
  - If legal or medical, need lots of special-purpose documents
- Make sure to cover different kinds of dialects and speaker/authors.
  - Example: *African-American Vernacular English (AAVE)*
    - One of many varieties that can be used by African Americans and others
    - Can include the auxiliary verb **finna** that marks immediate future tense:
    - "My phone finna die"

Perplexity

- What is the difference between the estimation and the real probability?
- How surprised the model on the test set?
- Lower the perplexity – better the model
- Probability depends on the size of the test set – really small longer the text
- Perplexity is the inverse probability of the test set, normalized by the number of words

$$PP(W) = P(w_1 w_2 ... w_N)^{-\frac{1}{N}}$$

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 ... w_N)}}$$

Perplexity

- Bigram language mode perplexity :

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-1})}}$$

- Training 38 million words, test 1.5 million words, Wall Street Journal data

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

Shannon's Method

- Understanding how probable a sequence of words is all good but it is more fun to generate sentences with these models
- Using a language model, we can generate sentences with the estimated probabilities
- Start with a random bigram for sentence starting using probability distributions - (<s>, w)
- Using this, generate the next word with sampling - (w,x)
- Continue until we stumble into sentence ender – (y,<\s>)
- String the words together

Shannon's Method

<s> I
I want
want to
to eat
eat Chinese
Chinese food
food </s>

I want to eat Chinese food

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.

Shakespeare as corpus

N=884,647 tokens, V=29,066
Shakespeare produced 300,000 bigram types out of V^2= 844 million possible bigrams.

- So 99.96% of the possible bigrams were never seen (have zero entries in the table)
- That sparsity is even worse for 4-grams, explaining why our sampling generated actual Shakespeare.

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

Guess the author

1) 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 gram Brazil on market conditions
2) This shall forbid it should be branded, if renown made it empty.
3) “You are uniformly charming!” cried he, with a smile of associating and now and then I bowed and they perceived a chaise and four to wish for.

Learning Goals (Week 4) - revisited

1. **Describe** language models
2. **Calculate** word-level, sentence-level prediction probabilities
3. **Define** N-gram models
4. **Evaluate** N-gram models

Following lectures

- Smoothing, Interpolation, and Backoff
- Word Embeddings
- Classification

Further resources

- 3rd chapter of Jurafsky & Martin

Programming Assignment #1(5 points)

Create your own tokenizer

- Deadline: October 22nd , 5pm
- Submission: To the SUCourse
- The assignment is going to be a Jupyter Notebook
- I will share the project specification file/notebook which will have the details of the assignment after the lecture.

Assignment Details

- I want you to create your own word tokenizer without the use of any external tokenizers.
- Select one of the given corpus from nltk corpora
- Write your tokenizer (regex, BPE, …)
- Evaluation block is already written – you need to give the corpus name you have used and the token list you have extracted.
- Please do not touch the evaluation block – you will get zero points

Assignment Report

- At the end of the code, please write an assignment report including information:
  - Selection of the corpus
  - The reasons behind the design of your tokenizer
  - Challenges you have faced while writing the tokenizer
  - Limitations of your approach
  - Any possible improvements to the system
- Please make sure to include all the information above but be brief. (500 words max)

Project Groups

- Please start making project groups
  - Undergrads – 4 people
  - Grads – 2,3 people
- No exceptions for people who want to create bigger groups.
- Next week, I will announce the tasks – 3 tasks
- You can only select a task after you form your group.

LANGUAGE MODELS

SMOOTHING

CS 445 - NATURAL LANGUAGE PROCESSING

Instructor: Dilara Keküllüoğlu

Fall 2024 – Week 4.2

Language Modeling (LM) Formalization

Goal: compute the probability of a sentence or sequence of words W:

$$P(W) = P(w_1, w_2, w_3, \ldots, w_n)$$

Goal: compute the probability of an upcoming word given previous n words:

$$P(w_n \mid w_1, w_2, w_3, \ldots, w_{n-1})$$

An LM computes either the sentence level prediction or the word prediction.

N-gram Independence Assumption

- Instead of relying only the last word let's say the next words probability rely on the last N words

$$P(w_n|w_{1:n-1}) \approx P(w_n|w_{n-N+1:n-1})$$

- Unigram model – P(delicious)
- Bigram model – P(delicious | looked)
- Trigram model – P(delicious | yesterday looked)

Raw bigram probabilities

- Let's normalize by unigrams:

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

Second word

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

First word

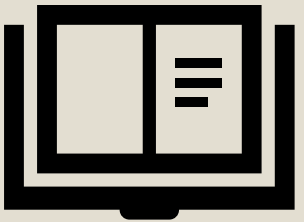
	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Some Questions from the Previous Lecture

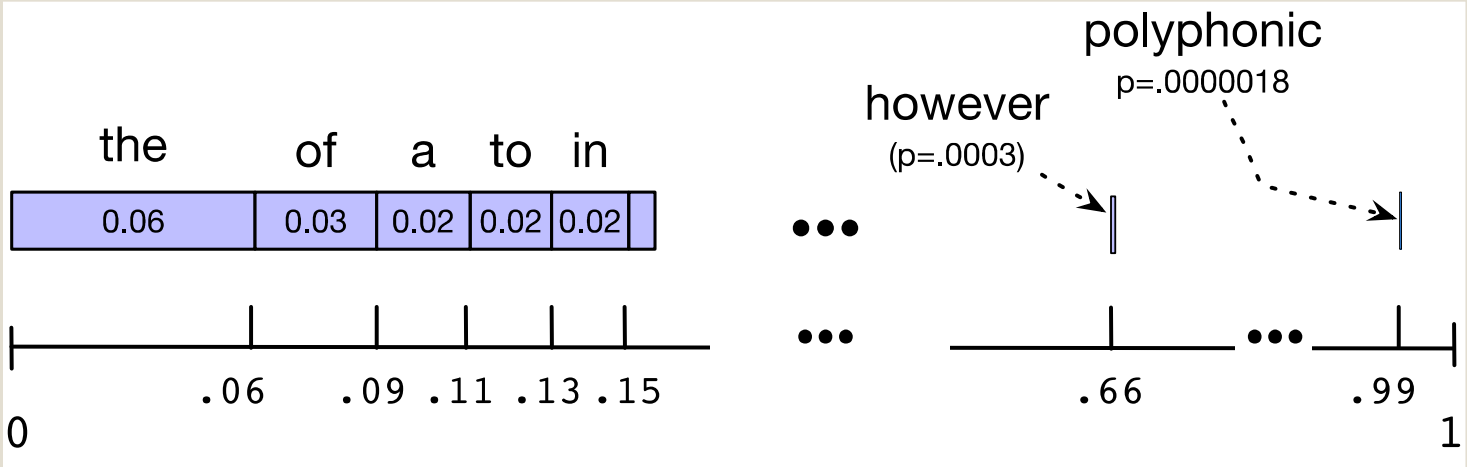
- What happens when we sample from a dataset with dominating word frequencies?
  - Probabilities by count will be dominated by ‘the’,’a’,’an’, etc.
  - How do we sample?

Sampling – How did Shannon sample?

"Open a book at random and select a letter at random on the page. This letter is recorded. The book is then opened to another page and one reads until this letter is encountered. The succeeding letter is then recorded. Turning to another page this second letter is searched for and the succeeding letter recorded, etc."



Sample from a distribution



- Select a random value between 0-1
- Find the corresponding area for the word
- Do this until you randomly generate a end of sentence token. <\s>
- Generated text might be a combination of most common words!

Another Question

- Could we have n-gram for characters instead of words?
  - Depends on the task and storage space
  - N-gram characters – will take much less space but will also have less context
  - It works well to find stylistic preferences or the native language of the author
    - Could be used in authorship recognition – more on classification next week

Another Question

What happens when we need to compute probability for a word we have not seen in the test data?

- It will make the whole probability 0
- We cannot take the log probability as log0 is undefined
- We cannot compute perplexity – zero division
- What can we do?

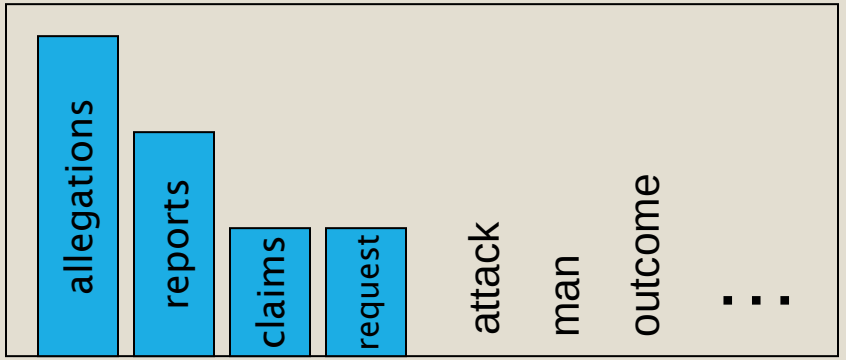
Sparse Data

- Sparsity is an issue once again!
- There are n-grams that the model have not seen before
- This is a problem even with unigrams – tail end of the word frequency table
- The flaw of MLE – it will estimate the probabilities that make the training data maximally probable!
- Everything else will be minimally probable – unseen data
- To avoid this, we use smoothing!

Smoothing Intuition

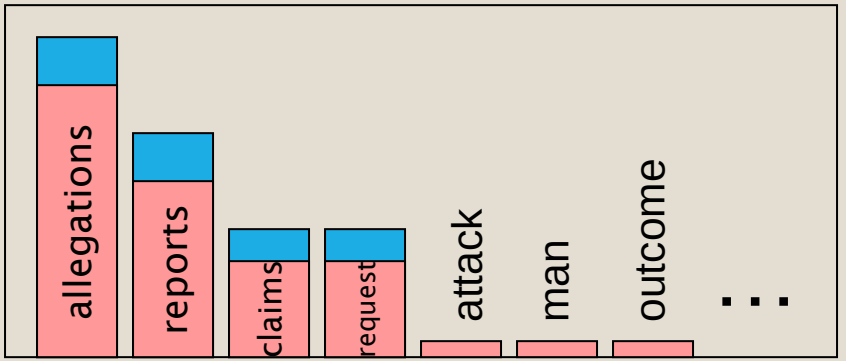
- When we have sparse statistics:

  P(w | denied the)
  3 allegations
  2 reports
  1 claims
  1 request
  7 total



- Steal probability mass to generalize better:

  P(w | denied the)
  2.5 allegations
  1.5 reports
  0.5 claims
  0.5 request
  2 other
  7 total



Credit: Dan Klein

Smoothing

- Transfers probability mass from the seen data to unseen data
- There are many with different assumptions
- One of them is Add-one (Laplace) Smoothing

Add-one Smoothing

- Pretend that every word is seen at least once – add one occurrence to the count
- MLE estimate :

$$P_{\text{MLE}}(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

- Add-1 estimate :

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

Number of the words in the vocabulary – probabilities should all up to 1.

Example

- Let's say in our corpus:
  - V = 10000
  - C(savory | yesterday looked) = 0
  - C(delicious | yesterday looked) = 1
    - "yesterday looked delicious" is the only trigram that starts with "yesterday looked"

	MLE Estimate	Add-1 Estimate
P(savory | yesterday looked)	0	1/(1+10000)
P(delicious | yesterday looked)	1	2/(1+10000)

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

Going back to 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

Comparison

	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

Another Example

- Considering the corpus with V=10000
- Let's say we have a more common bigram – yesterday looked was only one
  - C(w1,w2) = 100
  - C(w1,w2,w3) = 10

	MLE Estimate	Add-1 Estimate
P(w3|w1,w2)	10/100	11/(100+10000)

Limitations of Add-1

- Add-1 smoothing takes too much away!
- Other approaches of smoothing Add-k → train a system to approximate k instead of giving 1.
- Still a blunt tool when the vocabulary size is large.
- Text classification and domains where the number of zeros are not common – good to use add-1 or add-k
- Not very good for others

Another one!

- Let's say we have "Turkish tea" in the training corpus but we do not have the following:
  - Turkish tea drinkers
  - Turkish tea eaters

- Which one is true for a trigram model smoothed with add-1 or add-k?
  - P(Turkish tea drinkers) > P(Turkish tea eaters)
  - P(Turkish tea eaters) > P(Turkish tea drinkers)
  - P(Turkish tea drinkers) = P(Turkish tea eaters)

Unseen data problem

- In this case, the probabilities would be the same
- We expect Turkish tea drinkers to be more probable
- Smoothing does not solve this problem.
- We use interpolation or backoff instead

- Instead of looking for probabilities of the whole trigram, we can look at the probabilities of “tea drinkers” vs “tea eaters”

Interpolation and Backoff

- Use less context when the occurrences are low for n-gram using interpolation or backoff

Considering a trigram model,

- Interpolation
  - use a mix of trigram, bigram, and unigram probabilities

- Backoff
  - use trigram if you have good evidence,
  - Otherwise bigram, otherwise unigram

Interpolation

- Use higher and lower n-gram models together
  - High-order n-grams: more context, sparse counts
  - Low-order n-grams: less context, robust counts

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

$$\sum_i \lambda_i = 1$$

P_{INT}(Turkish tea drinkers) = λ_1P(drinkers | Turkish tea) +
λ_2P(drinkers | tea) +
λ_3P(drinkers)

How to decide the λ_i?

- We choose λ so that perplexity is optimized(lower) on the development set (also called held-out set).
  - Fix the n-gram probabilities on the training data
  - Solve for λ where the probabilities are largest (perplexity is lowest) for the development set.

Backoff

- Backoff is simpler.
- If P(drinkers | Turkish tea) = 0 then use the bigram P(drinkers | tea)
- If also P(drinkers | tea) = 0 then use the unigram P(drinkers)

- Problem – Probabilities will sum up higher than 1!
- We need to discount the higher-order n-gram (e.g. Katz backoff)
  - Not really easy to estimate correctly
  - Simpler backoff algorithm that works well in practice – Stupid backoff

Stupid Backoff

- Backoff without discounting (not a true probability)

$$S(w_i \mid w_{i-k+1}^{i-1}) = \begin{cases} \dfrac{\text{count}(w_{i-k+1}^{i})}{\text{count}(w_{i-k+1}^{i-1})} & \text{if } \text{count}(w_{i-k+1}^{i}) > 0 \\ 0.4\,S(w_i \mid w_{i-k+2}^{i-1}) & \text{otherwise} \end{cases}$$

$$S(w_i) = \frac{\text{count}(w_i)}{N}$$

Another example

- Given the work “York”
  - Frequent word comparable with foods, indicates, providers in Europarl corpus (477 times)
  - However, it is almost always preceded by “New” (473 times)

- In unseen bigram context, “York” should have low probability
  - Using interpolation/backoff will give “York” higher probability than expected with unigrams!
- Diversity of usage is important to consider

Kneser-Ney Smoothing – Only the Idea

- Takes diversity of the histories into account
- Intuition:
  - Take the counts of distinct histories for a word
  - Replace that with the count of the unigram
  - How many distinct bigrams “York” appears as the second word?
  - Take that as the count of unigram “York”
  - Divide it by the count of distinct histories of every word
- Real equation is more complex but one of the best smoothing method for n-grams.

Smoothing – could we do better?

- Uniform probabilities (add-1, add-k)
- Probabilities for lower-order n-grams (interpolation, backoff)
- Probability of appearing in new contexts (Kneser-Ney)

- Anything else to consider?

Word similarity

- Suppose we have two words with $C(w_1) >> C(w_2)$
  - Seagull
  - Puffin

- If we know **P(seagulls | fish caught by)**, will that help with **P(puffins | fish caught by)?**
- N-grams → No!

Photos by Till Rottmann, Kenneth

Word similarity

- We know both are sea birds that catch fish.
- Early solutions: class-based language models
  - Define classes for words, compute probability depending on the class.
- Recently: **distributed** language models
  - Project words to continuous space where words with similar contexts appear closer – word embeddings such as **Word2Vec**

Photos by Till Rottmann, Kenneth

Summary

- N-gram models can have many unseen word sequences that will make the probability 0
- **Smoothing** to give some probability to unseen n-grams
  - **Interpolation/Backoff** – use less context when higher order n-grams are not informative
  - **Kneser-Ney** – diversity of history – probability of a n-gram coming in new contexts
  - **Distributed representations** – words with similar contexts are accounted in calculations

Learning Goals (Week 4) - revisited

1. **Describe** language models
2. **Calculate** word-level, sentence-level prediction probabilities
3. **Define** N-gram models
4. **Evaluate** N-gram models
5. **Apply** smoothing, interpolation, and backoff to n-gram probability estimation

Following lectures

- Classification
- Word Embeddings