Probabilistic Language Modeling

Katharina Kann — CSCI/LING5832

Evaluation of Embedding Models

Evaluation Methods

Intrinsic evaluation:

- Word similarity
- Word analogy

Extrinsic evaluation:

- Use embeddings in down-stream task like question answering or natural language inference

Word Similarity

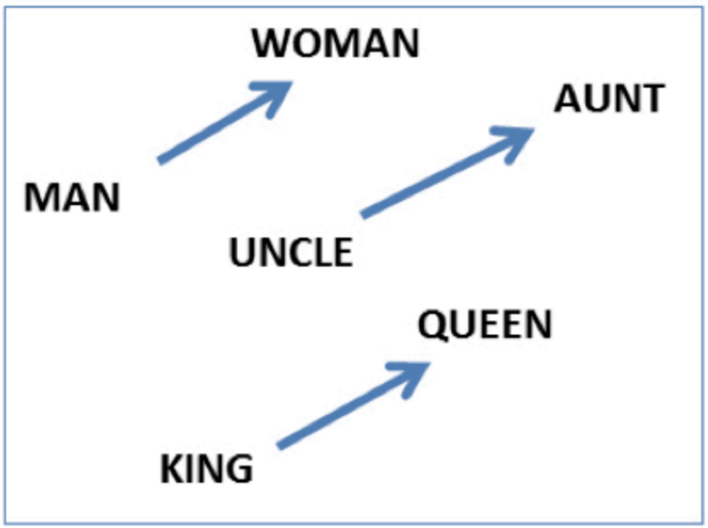
wordsim_similarity_goldstandard.txt

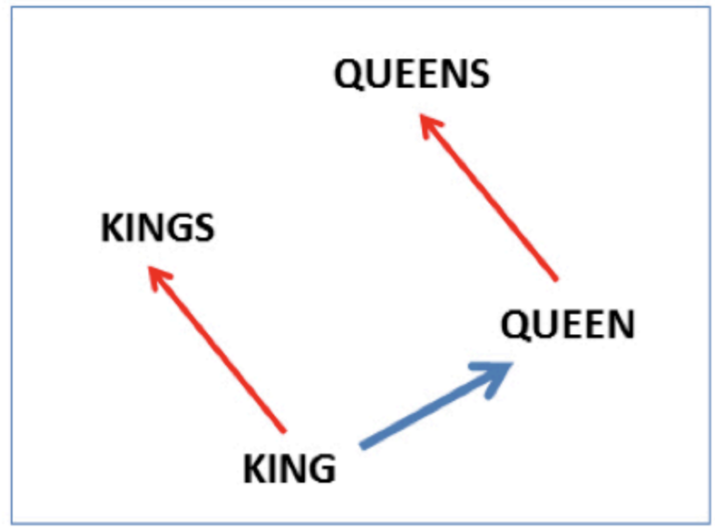
mile	kilometer	8.66
skin	eye	6.22
Japanese	American	6.50
century	year	7.59
announcement	news	7.56
doctor	personnel	5.00
Harvard	Yale	8.13
hospital	infrastructure	4.63
life	death	7.88
travel	activity	5.00
type	kind	8.97
street	place	6.44
street	avenue	8.88
street	block	6.88
cell	phone	7.81
dividend	payment	7.63
calculation	computation	8.44
profit	loss	7.63
dollar	yen	7.78
dollar	buck	9.22
phone	equipment	7.13
liquid	water	7.89
marathon	sprint	7.47
seafood	food	8.34
seafood	lobster	8.70
lobster	food	7.81
lobster	wine	5.70
championship	tournament	8.36
man	woman	8.30

Word Analogy (word2vec, Mikolov+ '13)

vector(*'king'*) - vector(*'man'*) + vector(*'woman'*) ≈ vector('queen')

vector(*'Paris'*) - vector(*'France'*) + vector(*'Italy'*) ≈ vector('Rome')





Tip: If you want to try this, you should exclude all 3 original vectors from the pool of candidates!

A Few Results

win	Method	WordSim Similarity	WordSim Relatedness	Bruni et al. MEN	Radinsky et al. M. Turk	Luong et al. Rare Words	Hill et al. SimLex	Google Add / Mul	MSR Add / Mul
2	PPMI	.732	**.699**	.744	.654	.457	.382	.552 / .677	.306 / .535
	SVD	.772	.671	**.777**	.647	**.508**	.425	.554 / .591	.408 / .468
	SGNS	**.789**	.675	.773	**.661**	.449	**.433**	.676 / **.689**	.617 / **.644**
	GloVe	.720	.605	.728	.606	.389	.388	.649 / .666	.540 / .591
5	PPMI	.732	**.706**	.738	**.668**	.442	.360	.518 / .649	.277 / .467
	SVD	.764	.679	**.776**	.639	**.499**	**.416**	.532 / .569	.369 / .424
	SGNS	**.772**	.690	.772	.663	.454	.403	.692 / **.714**	.605 / **.645**
	GloVe	.745	.617	.746	.631	.416	.389	.700 / .712	.541 / .599
10	PPMI	.735	**.701**	.741	.663	.235	.336	.532 / .605	.249 / .353
	SVD	.766	.681	.770	.628	**.312**	.419	.526 / .562	.356 / .406
	SGNS	**.794**	.700	**.775**	**.678**	.281	**.422**	.694 / .710	.520 / **.557**
	GloVe	.746	.643	.754	.616	.266	.375	.702 / **.712**	.463 / .519
10	SGNS-LS	.766	.681	**.781**	**.689**	**.451**	.414	.739 / **.758**	.690 / **.729**
	GloVe-LS	.678	.624	.752	.639	.361	.371	.732 / .750	.628 / .685

Table 5: Performance of each method across different tasks using 2-fold cross-validation for hyperparameter tuning. Configurations on large-scale (LS) corpora are also presented for comparison.

Levy et al. ‘15: Improving Distributional Similarity with Lessons Learned from Word Embeddings

Results: Extrinsic Evaluation

Dataset	Random	GloVe
SST-2	84.2	88.4
SST-5	48.6	53.5
IMDb	88.4	91.1
TREC-6	88.9	94.9
TREC-50	81.9	89.2
SNLI	82.3	87.7
SQuAD	65.4	76.0

Model	Squad
GloVe Wiki + news	77.7%
fastText Wiki + news	78.8%
GloVe Crawl	78.9%
fastText Crawl	**79.8%**

McCann+ '17; Mikolov+ '17

Problems and Challenges

How about Antonyms?

- Antonyms appear in very similar contexts.
- Their word embeddings are similar.
- However, meaning can be very different, depending on the task at hand.

How about Antonyms?

- Antonyms appear in very similar contexts.
- Their word embeddings are similar.
- However, meaning can be very different, depending on the task at hand.

The movie I watched yesterday was really good.

vs.

The movie I watched yesterday was really bad.

How about Antonyms?

- Antonyms appear in very similar contexts.
- Their word embeddings are similar.
- However, meaning can be very different, depending on the task at hand.

Give me the red block! vs. Give me the green block!

Pitfalls of Unsupervised Learning

Word similarity:

- Occupations most similar to *she*:
    - *nurse*, *librarian, nanny, stylist, dancer*
- Occupations most similar to *he*:
    - *architect, captain, philosopher, legend, hero*

Source: Bolukbasi et al. '16, Quantifying and Reducing Stereotypes in Word Embeddings

Pitfalls of Unsupervised Learning

Word analogy:

- doctor - father + mother: nurse

Source: Bolukbasi et al. '16, Quantifying and Reducing Stereotypes in Word Embeddings

Pitfalls of Unsupervised Learning

Additionally:

- African American names have a higher GloVe cosine with unpleasant words.
- European American names (‘Brad’, ‘Greg’, ‘Courtney’) have a higher cosine with pleasant words.

Source: Bolukbasi et al. ‘16, Quantifying and Reducing Stereotypes in Word Embeddings

Pitfalls of Unsupervised Learning

Impossible to avoid these issues altogether when learning from naturally occurring text.

Mitigating bias will usually require identifying explicitly, and the best method will depend on the task at hand.

Source: Bolukbasi et al. '16, Quantifying and Reducing Stereotypes in Word Embeddings

Probabilistic Language Modeling

Predicting the Next Word…

- Guess the next word...
  - So I notice three guys standing on the ___

Predicting the Next Word…

- Guess the next word...
  - So I notice three guys standing on the ___

What kinds of knowledge did you use to come up with those predictions?

Predicting the Next Word…

- We can formalize this task as a problem in discrete probability
    - Given a vocabulary, compute a probability distribution over that vocabulary given the previous words.
    - Or assign a probability to a sequence.
    - We'll call this a **probabilistic language model**.

Predicting the Next Word…

- It turns out that assessing the probability of a sequence is an extremely useful thing to be able to do.

Predicting the Next Word…

- It turns out that assessing the probability of a sequence is an extremely useful thing to be able to do.
- Used in many applications:
  - Automatic speech recognition
  - Handwriting and character recognition
  - Spelling correction
  - Machine translation

Application: Speech Recognition

- Initial acoustic/signal system proposes two hypotheses for an input sentence
  - Its hard to wreck a nice beach

Application: Speech Recognition

- Initial acoustic/signal system proposes two hypotheses for an input sentence
  - Its hard to wreck a nice beach
  - Its hard to recognize speech

Application: Speech Recognition

- Initial acoustic/signal system proposes two hypotheses for an input sentence
  - Its hard to wreck a nice beach
  - Its hard to recognize speech
- Job of the language model is to say which of those is more likely

Recap: Probabilities

- Probabilities are beliefs about an event outcome expressed as a number between 0 an 1.

Recap: Probabilities

- Probabilities are beliefs about an event outcome expressed as a number between 0 an 1.
- The sample space is the set of all possible outcomes.

Recap: Probabilities

- Probabilities are beliefs about an event outcome expressed as a number between 0 an 1.
- The sample space is the set of all possible outcomes.
- An event is some particular outcome.

Recap: Probabilities

- Probabilities are beliefs about an event outcome expressed as a number between 0 an 1.
- The sample space is the set of all possible outcomes.
- An event is some particular outcome.
- A prior is a probability we hold in the absence of any other evidence.

Recap: Probabilities

- Probabilities are beliefs about an event outcome expressed as a number between 0 an 1.
- The sample space is the set of all possible outcomes.
- An event is some particular outcome.
- A prior is a probability we hold in the absence of any other evidence.
- A conditional probability is a probability given some particular piece of evidence.

Probabilities and Language Models

- With respect to “language models” we’ll mainly be concerned with the probability of sentences (sequences of words)

Probabilities and Language Models

- With respect to “language models” we’ll mainly be concerned with the probability of sentences (sequences of words)
  - The utterance of a sentence is the event

Probabilities and Language Models

- With respect to “language models” we’ll mainly be concerned with the probability of sentences (sequences of words)
  - The utterance of a sentence is the event
  - The sample space is the space of all possible sentences
    - (how many possible sentences are there?)

Probabilities and Language Models

- With respect to “language models” we’ll mainly be concerned with the probability of sentences (sequences of words)
  - The utterance of a sentence is the event
  - The sample space is the space of all possible sentences
    - (how many possible sentences are there?)
  - We’d like to assign a probability to that event
    - (this is a strange notion)

Chomsky

- “… it must be recognized that the notion of "probability of a sentence" is an entirely useless one, under any known interpretation of this term.”

Chomsky

- “… it must be recognized that the notion of "probability of a sentence" is an entirely useless one, under any known interpretation of this term.”

“Entirely useless” is a pretty strong claim. One that turns out to be incorrect.

Language Modeling

- How might we go about calculating a conditional probability?
  - One way is to use the definition of conditional probabilities and look for counts. So to get
    - P(the | its water is so transparent that)

Language Modeling

- How might we go about calculating a conditional probability?
  - One way is to use the definition of conditional probabilities and look for counts. So to get
    - P(the | its water is so transparent that)
- By definition that's
  - $$\frac{P(\text{its water is so transparent that the})}{P(\text{its water is so transparent that})}$$

Language Modeling

- How might we go about calculating a conditional probability?
  - One way is to use the definition of conditional probabilities and look for counts. So to get
    - P(the | its water is so transparent that)
- By definition that's
  - $$\frac{P(\text{its water is so transparent that the})}{P(\text{its water is so transparent that})}$$
- We can get each of those from counts in a large corpus.

Very Easy Estimate…

$$P(\text{the} \mid \text{its water is so transparent that}) = \frac{\text{Count(its water is so transparent that the)}}{\text{Count(its water is so transparent that)}}$$

Very Easy Estimate…

- (Let’s assume that:) According to Google the counts are 1320 and 1420.

Very Easy Estimate…

- (Let’s assume that:) According to Google the counts are 1320 and 1420.
- Then the conditional probability of interest is...
    - P(the | its water is so transparent that) = 0.93

Very Easy Estimate…

its water is so transparent that

- How about “matrix”
- That gives you a 0. 0/1420 = ?

Very Easy Estimate…

$\boxed{\text{its water is so transparent that}}$

- How about "matrix"
- That gives you a 0. 0/1420 = ?
- How about "you"
- That gives you a 1. 1/1420 = 0.0007

Very Easy Estimate…

its water is so transparent that

- How about "matrix"
- That gives you a 0. 0/1420 = ?
- How about "you"
- That gives you a 1. 1/1420 = 0.0007
- How about "she"
- That gives you a 0. 0/1420 = hmmm

Language Modeling

- Unfortunately, for most sequences, and for most text collections, we won’t get good estimates from this method.
  - What we’re likely to get are 0 counts.

Language Modeling

- Unfortunately, for most sequences, and for most text collections, we won’t get good estimates from this method.
  - What we’re likely to get are 0 counts.
- We’ll have to be a little more clever to make this scheme work.
  - Let’s first use the chain rule for probability.
  - And then apply a particularly useful independence assumption.

The Chain Rule

- Recall the definition of conditional probabilities
  - $P(A|B) = P(A, B)/P(B)$

The Chain Rule

- Recall the definition of conditional probabilities
  - $P(A|B) = P(A, B)/P(B)$
- Rewriting:
  - $P(A, B) = p(A|B)P(B)$

The Chain Rule

- Recall the definition of conditional probabilities
  - $P(A|B) = P(A, B)/P(B)$

- Rewriting:
  - $P(A, B) = p(A|B)P(B)$

- For sequences...
  - $P(A,B,C,D) = P(A)P(B|A)P(C|A,B)P(D|A,B,C)$

The Chain Rule

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

P(its water was so transparent)=

P(its)*

P(water|its)*

P(was|its water)*

P(so|its water was)*

P(transparent|its water was so)

Unfortunately

- There are still a lot of problematic long sequences in there.
- In general, we'll never be able to get enough data to compute the statistics for those longer prefixes
  - Same problem we had for the original sequence.

Markov Assumption

Independence Assumption

- Make the simplifying assumption
  - P(lizard|the,other,day,I,was,walking,along,and,saw,a) = P(lizard|a)
- Or maybe
  - P(lizard|the,other,day,I,was,walking,along,and,saw,a) = P(lizard|saw,a)

Independence Assumption

- Make the simplifying assumption
  - P(lizard|the,other,day,I,was,walking,along,and,saw,a) = P(lizard|a)
- Or maybe
  - P(lizard|the,other,day,I,was,walking,along,and,saw,a) = P(lizard|saw,a)

- That is, the probability in question is to some degree **independent** of its earlier history.

Markov Assumption

So for each component in the product replace with the approximation (assuming a prefix of size N - 1)

$$P(w_n \mid w_1^{n-1}) \approx P(w_n \mid w_{n-N+1}^{n-1})$$

Bigram version

$$P(w_n \mid w_1^{n-1}) \approx P(w_n \mid w_{n-1})$$

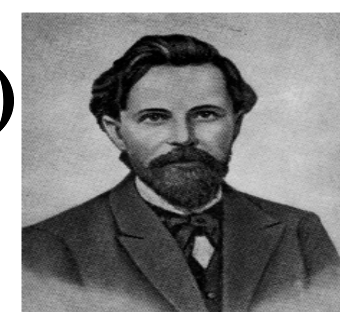
Markov Assumption

So for each component in the product replace with the approximation (assuming a prefix of size N - 1)

$$P(w_n \mid w_1^{n-1}) \approx P(w_n \mid w_{n-N+1}^{n-1})$$

Bigram version

$$P(w_n \mid w_1^{n-1}) \approx P(w_n \mid w_{n-1})$$



Language Models

- Bigram language models:
  - 1 preceding word
- Trigram language models:
  - 2 preceding words
- N-gram language models:
  - N-1 preceding words

Estimating Bigram Probabilities

- The Maximum Likelihood Estimate (MLE)

$$P(w_i \mid w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

Example

$$P(w_i \mid w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

- <s> I am Sam </s>
- <s> Sam I am </s>
- <s> I do not like green eggs and ham </s>

- p(I | <s>) =
- P(Sam | <s>) =
- P(am | I) =
- P(</s> | Sam) =
- P(Sam | am) =
- P(do | I) =

Example

$$P(w_i \mid w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

- <s> I am Sam </s>
- <s> Sam I am </s>
- <s> I do not like green eggs and ham </s>

- p(I | <s>) = 2/3 = 0.67
- P(Sam | <s>) =
- P(am | I) =
- P(</s> | Sam) =
- P(Sam | am) =
- P(do | I) =

Example

$$P(w_i \mid w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

- <s> I am Sam </s>
- <s> Sam I am </s>
- <s> I do not like green eggs and ham </s>

- p(I | <s>) = 2/3 = 0.67
- P(Sam | <s>) = 1/3 = 0.33
- P(am | I) =
- P(</s> | Sam) =
- P(Sam | am) =
- P(do | I) =

Example

$$P(w_i \mid w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

- <s> I am Sam </s>
- <s> Sam I am </s>
- <s> I do not like green eggs and ham </s>

- p(I | <s>) = 2/3 = 0.67
- P(Sam | <s>) = 1/3 = 0.33
- P(am | I) = 2/3 = 0.67
- P(</s> | Sam) =
- P(Sam | am) =
- P(do | I) =

Example

$$P(w_i \mid w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

- <s> I am Sam </s>
- <s> Sam I am </s>
- <s> I do not like green eggs and ham </s>

- p(I | <s>) = 2/3 = 0.67
- P(Sam | <s>) = 1/3 = 0.33
- P(am | I) = 2/3 = 0.67
- P(</s> | Sam) = 1/2 = 0.5
- P(Sam | am) =
- P(do | I) =

Example

$$P(w_i \mid w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

- <s> I am Sam </s>
- <s> Sam I am </s>
- <s> I do not like green eggs and ham </s>

- p(I | <s>) = 2/3 = 0.67
- P(Sam | <s>) = 1/3 = 0.33
- P(am | I) = 2/3 = 0.67
- P(</s> | Sam) = 1/2 = 0.5
- P(Sam | am) = 1/2 = 0.5
- P(do | I) =

Example

$$P(w_i \mid w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

- <s> I am Sam </s>
- <s> Sam I am </s>
- <s> I do not like green eggs and ham </s>

- p(I | <s>) = 2/3 = 0.67
- P(Sam | <s>) = 1/3 = 0.33
- P(am | I) = 2/3 = 0.67
- P(</s> | Sam) = 1/2 = 0.5
- P(Sam | am) = 1/2 = 0.5
- P(do | I) = 1/3 = 0.33

Berkeley Restaurant Project

- can you tell me about any good cantonese restaurants close by
- mid priced thai food is what i'm looking for
- tell me about chez panisse
- can you give me a listing of the kinds of food that are available
- i'm looking for a good place to eat breakfast
- when is caffe venezia open during the day

Bigram Counts

- Vocabulary size is 1446 |V|
- Out of 9222 sentences
  - "I want" occurred 827 times

	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

Bigram Probabilities

- Divide bigram counts by the prefix unigram counts to get bigram probabilities.

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	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

Bigram Probabilities

- Divide bigram counts by the prefix unigram counts to get bigram probabilities.

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	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

$$P(\text{want} \mid \text{I}) = 827/2533 = .336$$

Bigram Probabilities

- Divide bigram counts by the prefix unigram counts to get bigram probabilities.

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

Bigram Estimates of Sentence Probabilities

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

  P(I|<s>)*

  P(want|I)*

  P(english|want)*

  P(food|english)*

  P(</s>|food)*

  =.000031

What Can This Do?

- As crude as they are, N-gram probabilities capture a range of interesting facts about language.

- P(english|want) = .0011
- P(chinese|want) = .0065
- P(to|want) = .66
- P(eat | to) = .28
- P(food | to) = 0
- P(want | spend) = 0
- P (i | <s>) = .25

What Can This Do?

- As crude as they are, N-gram probabilities capture a range of interesting facts about language.

- P(english|want) = .0011
- P(chinese|want) = .0065
- P(to|want) = .66
- P(eat | to) = .28
- P(food | to) = 0
- P(want | spend) = 0
- P (i | <s>) = .25

World knowledge

What Can This Do?

- As crude as they are, N-gram probabilities capture a range of interesting facts about language.

- P(english|want) = .0011
- P(chinese|want) = .0065
- P(to|want) = .66
- P(eat | to) = .28
- P(food | to) = 0
- P(want | spend) = 0
- P (i | <s>) = .25

World knowledge

Syntax

In-Class Exercise

- *the movie was really boring*
- *the last scene was really boring*
- *I really liked the other movie more*

In-Class Exercise

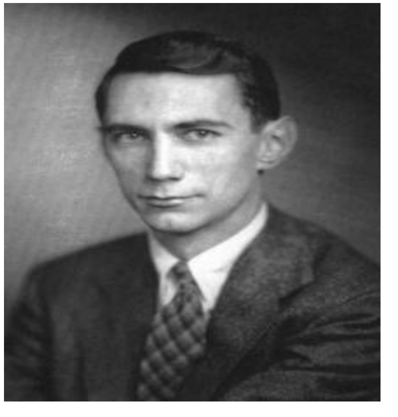
- ***the movie was really boring***
- ***the last scene was really boring***
- ***I really liked the other movie more***

1. Compute the following probabilities:
    1. P(boring | really)
    2. P(movie | the)
2. Compute the probability of the sentence “I really liked the last scene”
    1. With a bigram language model
    2. With a trigram language model

Generating text with language models

Shannon’s Method

- Assigning probabilities to sentences is all well and good, but it’s not terribly illuminating.
- A more entertaining (and useful?) task is to turn the model around and use it to **generate** random sentences that are **similar** to the sentences from which the model was derived.
- Generally attributed to
- Claude Shannon.



Shannon's Method

- Sample a random bigram (<s>, w) according to the probability distribution over bigrams
- Now sample a new random bigram (w, x) according to its probability. Where the prefix w matches the suffix of the first bigram chosen.
- And so on until we randomly choose a (y, </s>)
- Then string them together
  - <s> I
    - I want
      - want to
        - to eat
          - eat Chinese
            - Chinese food
              - food </s>

Shakespeare

Unigram	• To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have • Every enter now severally so, let • Hill he late speaks; or! a more to leg less first you enter • Are where exeunt and sighs have rise excellency took of.. Sleep knave we. near; vile like
Bigram	• What means, sir. I confess she? then all sorts, he is trim, captain. •Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow. •What we, hath got so she that I rest and sent to scold and nature bankrupt, nor the first gentleman? •Enter Menenius, if it so many good direction found'st thou art a strong upon command of fear not a liberal largess given away, Falstaff! Exeunt
Trigram	• Sweet prince, Falstaff shall die. Harry of Monmouth's grave. • This shall forbid it should be branded, if renown made it empty. • Indeed the duke; and had a very good friend. • Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done.
Quadrigram	• King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in; • Will you not tell me who I am? • It cannot be but so. • Indeed the short and the long. Marry, 'tis a noble Lepidus.

Shakespeare as a 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)
  - This is the biggest problem in language modeling; we'll come back to it.
- Quadrigrams are worse: What's coming out looks like Shakespeare because it is Shakespeare

In-Class Exercise 2

love is patient love is kind
it is kind

In-Class Exercise 2

love is patient love is kind
it is kind

1. Compute the parameters of a bigram language model from the sentences above.
2. Use the bigram language model to generate *the most likely* sentence (break ties alphabetically!).
3. *Sample* a sentence from the bigram language model.

Evaluation of language models

Model Evaluation

- How do we know if our models are any good?
  - …and how do we know if one model is better than another?
- Well Shannon's game gives us an intuition.
  - The generated texts from the higher order models surely sound better.
    - That is, they sound more like the text the model was obtained from.
- But what does that mean? How can we make that notion operational?

Evaluating N-Gram Models

- Best evaluation for a language model
  - Put model A into an application
    - For example, a machine translation system
  - Evaluate the performance of the application with model A
  - Put model B into the application and evaluate
  - Compare performance of the application with the two models
    - **Extrinsic** evaluation

Evaluation

- Extrinsic evaluation
  - This is really time-consuming and can be hard
  - Not something you want to do unless you’re pretty sure you’ve got a good solution

Evaluation

- Extrinsic evaluation
  - This is really time-consuming and can be hard
  - Not something you want to do unless you're pretty sure you've got a good solution
- So
  - As a intermediate evaluation, in order to run rapid experiments we evaluate N-grams with an **intrinsic** evaluation
  - An evaluation that tries to capture how good the model is intrinsically, not how much it improves performance in some larger system

Evaluation

- Standard method
  - Train parameters of our model on a training set.
  - Evaluate the model on some new data: a test set.
    - A dataset which is different from our training set, but drawn from the same source

Perplexity

- The intuition behind perplexity as a measure is the notion of surprise.
  - How surprised is the language model when it sees the test set?
    - The more surprised the model is, the lower the probability it assigned to the test set.
    - The higher the probability, the less surprised it was.

Perplexity

- Perplexity is just the probability of a test set (assigned by the language model), as normalized by the number of words:

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

- Chain rule:

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_1 \ldots w_{i-1})}}$$

- For bigrams:

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

Perplexity

- Perplexity is just the probability of a test set (assigned by the language model), as normalized by the number of words:

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

- Chain rule:

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1 \ldots w_{i-1})}}$$

- For bigrams:

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

- Minimizing model perplexity is the same as maximizing probability of a test set.

Lower Perplexity is Better

- Training 38 million words, test 1.5 million words, WSJ

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

Practical Issues

- Once we start looking at test data, we'll run into words that we haven't seen before. So our models won't work. Standard solution:
  - Given a corpus
    - Create a fixed lexicon L, of size V
      - Say from a dictionary or
      - A subset of terms from the training set
    - Any word not in L is changed to <UNK>
    - Collect counts for that as for any normal word
  - At test time
    - Use <UNK> counts for any word not seen in training

Practical Issues

- Multiplying a bunch of really small numbers < 0 is a really bad idea.
    - Underflow is likely
- So do everything in log space
    - Avoids underflow (and adding is faster than multiplying)

Wrapping up

- Discussed today:
  - Evaluation of word embeddings
  - Problems with and challenges for word embeddings
  - N-gram language models
  - Assigning a probability to a sentence
  - Generating sentences using a language model
- Next Monday: Language modeling with neural models