CS3245

Information Retrieval

1

Lecture 1: Language Models

Live Q&A
https://pollev.com/jin

Modeling Language

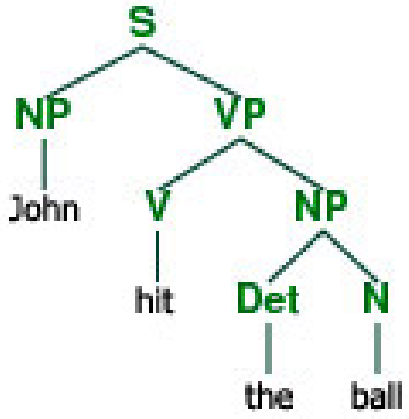
- **Book A** by Shakespeare
- **Book B** by J.K. Rowling

- *Which book is more likely to contain the following phrases?*

  1. A nice normal day
  2. Wherefore art thou

Modeling Language

- Traditionally, we model language with the notion of a formal syntax and semantics.
  - Vocabulary and grammar
  - Specify what is included in or excluded from a language
  - Help us to interpret
    the meaning (semantics) of the sentence



What's a language model?

- It can be helpful to have a **computational** (e.g., probabilistic) model of a language that is simple without the use of a grammar.

- A language model is a set of statistics
  - created based on a collection of text, and
  - used to assign a score (e.g., probability) to a sequence of words.

What's a language model?

- Example
  - chatlog_LangModel: created based on the chat logs in a messaging tool (e.g., Telegram)
  - Word sequence: "Forsooth, there is no one I trust more"
  - chatlog_LangModel: **low** probability

  - Shakespeare_LangModel: created based on the plays written by Shakespeare
  - Word sequence: "Forsooth, there is no one I trust more"
  - Shakespeare_LangModel: **high** probability

Applications of LMs

- Deciding between alternatives

  I either heard "Recognize speech" or "Wreck a nice beach", which is more likely?

  - Speech Recognition
  - Spelling Correction
  - Plagiarism Detection
  - Prediction of what products you'll browse next
  - Typeahead prediction on mobile devices
  - Result Ranking

The Unigram Model

- Views language as an unordered collection of tokens
  - Each of the *n* tokens contributes one count (or 1/n) to the model
  - Also known as a "bag of words"

- Outputs a count (or probability) of an input based on its individual tokens

  - Count(input) = $\sum_{n}$ Count (n)
  - P(input) = π_{n} P(n)

Aerosmith vs. Lady Gaga: A Simple Count Model

- Let's take a sentence from each of these artists and build two language models:

… I don't want to close my eyes // …

I	1	close	1
don't	1	my	1
want	1	eyes	1
to	1		

… I want your love and I want your revenge // …

I	2	love	1
want	2	and	1
your	2	revenge	1

Test: Input queries

- Q1: "I want"

I	1	close	1
don't	1	my	1
want	1	eyes	1
to	1		

I	2	love	1
want	2	and	1
your	2	revenge	1

Test: Input queries

- Q1: "I want"

Count (Aerosmith): 1 + 1 = 2

Count (LadyGaga): 2 + 2 = 4

Winner: **Lady Gaga**

- Q2: "I don’t want"

I	1	close	1
don't	1	my	1
want	1	eyes	1
to	1		

I	2	love	1
want	2	and	1
your	2	revenge	1

Test: Input queries

- Q1: "I want"

Count (Aerosmith): 1 + 1 = 2

Count (LadyGaga): 2 + 2 = 4

Winner: Lady Gaga

- Q2: "I don't want"

Count (Aerosmith): 1 + 1 + 1 = 3

Count (LadyGaga): 2 + 0 + 2 = 4

Winner: **Lady Gaga**

- Q3: "close my eyes"

I	1	close	1
don't	1	my	1
want	1	eyes	1
to	1		

I	2	love	1
want	2	and	1
your	2	revenge	1

Test: Input queries

- Q1: "I want"

Count (Aerosmith): 1 + 1 = 2

Count (LadyGaga): 2 + 2 = 4

Winner: Lady Gaga

- Q2: "I don't want"

Count (Aerosmith): 1 + 1 + 1 = 3

Count (LadyGaga): 2 + 0 + 2 = 4

Winner: Lady Gaga

- Q3: "close my eyes"

Count (Aerosmith): 1 + 1 + 1 = 3

Count (LadyGaga): 0 + 0 + 0 = 0

Winner: **Aerosmith**

I	1	close	1
don't	1	my	1
want	1	eyes	1
to	1		

I	2	love	1
want	2	and	1
your	2	revenge	1

Extending the example

- Imagine you take your music collection and for each song you get the lyrics from the web
- Then you can build unigram language models for all songs with the same artist or genre

Quick poll: What are your answers to:

Which artist is most likely to have written some input lyric?

What words are most popular in a specific genre?

What are the significant phrases used in this genre?

Of Words Matter Order The

- Unigrams LM don’t model word order (hence "bag of words")
  - "close my eyes" is as likely as "eyes close my"
- We must introduce additional context to model order

Blanks on slides, you may want to fill in

Ngram LM

- An ngram LM remembers sequences of *n* tokens
  - Unigram is just a special case of n=1
  - **Bigrams** are ngram LMs where n=2, **trigrams** where n=3

e.g. "I don't want to close my eyes"

START I		START START I
I don't		START I don't
don't want		I don't want
Want to		don't want to
to close		want to close
close my		to close my
my eyes		close my eyes
eyes END		my eyes END
		eyes END END

Use special START and END symbols for encoding beyond the text boundary

Blanks on slides, answers on next slide

Ngram LM

- A ngram model can predict a current word from the *n*-1 previous context words.
  - P(?? | "Please turn off your hand")

prediction context of n=5

Probability of predicting "??" after seeing "Please turn off your hand".
What's your guess about the next word?

How would the unigram, bigram and trigram models predict "??"

- Unigram (n=1):
- Bigram (n=2):
- Trigram (n=3):

Ngram LM

- A ngram model can predict a current word from the n-1 previous context words.
  - P(?? | "Please turn off your hand")

prediction context of n=5

Probability of predicting "??" after seeing "Please turn off your hand".
What's your guess about the next word?

How would the unigram, bigram and trigram models predict "??"

- Unigram (n=1): P(??)
- Bigram (n=2): P(??|"hand")
- Trigram (n=3): P(??|"your hand")

Markov Assumption

- The ***Markov assumption*** is the presumption that the future behavior of a dynamical system **only** depends on its **recent** history. In particular, in a ***k*th-order *Markov model***, the next state only depends on the ***k*** **most recent** states

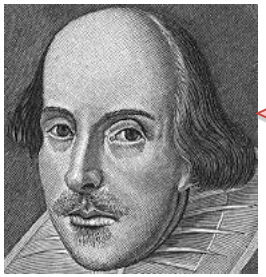
- Therefore, an N-gram model is a (*N*-1)-order Markov model.

Blanks on slides, you may want to fill in

From 1 to n

- Longer ngram models are more accurate but exponentially more costly to construct (why?).

E.g., Shakespeare ngram models



Yeah, that guy

a) To him swallowed confess hear both.

b) What means, sir. I confess she? Then all sorts, he is trim, captain.

c) Sweet Prince, Falstaff shall die. Harry of Monmouth's grave.

d) Will you not tell me who I am? It cannot be but so.

Complexity

- Let |V| stand for the size of the vocabulary used in a language. For English, let's use |V| = 30,000

- For a unigram LM we need to store counts/probabilities for |V| words
- For a bigram LM, we need to store counts/probabilities for (up to) |V|*|V| ordered length 2 phrases
- Check your understanding:
  What about a trigram model?

Gets expensive very quickly!

Probability-based LM

- Q1: "I want"

Prob(Aerosmith): .14 * .14 = 1.9E-2

Prob(LadyGaga): .22 * .22 = 4.8E-2

Winner: **Lady Gaga**

- Q2 : "I don’t want"

I	1 (0.14)	close	1 (0.14)
don't	1 (0.14)	my	1 (0.14)
want	1 (0.14)	eyes	1 (0.14)
to	1 (0.14)		

I	2 (0.22)	love	1 (0.11)
want	2 (0.22)	and	1 (0.11)
your	2 (0.22)	revenge	1 (0.11)

Probability-based LM

- Q1: "I want"

Prob(Aerosmith): .14 * .14 = 1.9E-2

Prob(LadyGaga): .22 * .22 = 4.8E-2

Winner: Lady Gaga

I	1 (0.14)	close	1 (0.14)
don't	1 (0.14)	my	1 (0.14)
want	1 (0.14)	eyes	1 (0.14)
to	1 (0.14)		

- Q2 : "I don't want"

Prob(Aerosmith): .14 * .14 * .14 = 2.7E-3

Prob(LadyGaga): .22 * 0 * .22 = 0

Winner: **Aerosmith**

I	2 (0.22)	love	1 (0.11)
want	2 (0.22)	and	1 (0.11)
your	2 (0.22)	revenge	1 (0.11)

Problem: The probability that Lady Gaga would use "don't" in a song isn't really 0, but that's what our limited data says.

Add 1 Smoothing

- Not used in practice, but most basic to understand

I	1 (0.14)	close	1 (0.14)
don't	1 (0.14)	my	1 (0.14)
want	1 (0.14)	eyes	1 (0.14)
to	1 (0.14)		

I	1 (0.14)	eyes	1 (0.14)
don't	1 (0.14)	**your**	**0 (0)**
want	1 (0.14)	**love**	**0 (0)**
to	1 (0.14)	**and**	**0 (0)**
close	1 (0.14)	**revenge**	**0 (0)**
my	1 (0.14)		

Show the zero entries

I	2 (0.22)	love	1 (0.11)
want	2 (0.22)	and	1 (0.11)
your	2 (0.22)	revenge	1 (0.11)

I	2 (0.22)	eyes	**0 (0)**
don't	**0 (0)**	your	2 (0.22)
want	2 (0.22)	love	1 (0.11)
to	**0 (0)**	and	1 (0.11)
close	**0 (0)**	revenge	1 (0.11)
my	**0 (0)**		

Add 1 Smoothing

- Idea: add 1 count to all entries in the LM, including those that are not seen

I	1 (0.14)	eyes	1 (0.14)
don't	1 (0.14)	your	0 (0)
want	1 (0.14)	love	0 (0)
to	1 (0.14)	and	0 (0)
close	1 (0.14)	revenge	0 (0)
my	1 (0.14)		

I	**2 (0.11)**	eyes	**2 (0.11)**
don't	**2 (0.11)**	your	**1 (0.06)**
want	**2 (0.11)**	love	**1 (0.06)**
to	**2 (0.11)**	and	**1 (0.06)**
close	**2 (0.11)**	revenge	**1 (0.06)**
my	**2 (0.11)**		

Add 1 count to all entries and recompute the probabilities

I	2 (0.22)	eyes	0 (0)
don't	0 (0)	your	2 (0.22)
want	2 (0.22)	love	1 (0.11)
to	0 (0)	and	1 (0.11)
close	0 (0)	revenge	1 (0.11)
my	0 (0)		

I	**3 (0.15)**	eyes	**1 (0.05)**
don't	**1 (0.05)**	your	**3 (0.15)**
want	**3 (0.15)**	love	**2 (0.10)**
to	**1 (0.05)**	and	**2 (0.10)**
close	**1 (0.05)**	revenge	**2 (0.10)**
my	**1 (0.05)**		

Add 1 Smoothing

- Q2: "I don't want"

Prob (Aerosmith): .11 * .11 * .11 = 1.3E-3

Prob (LadyGaga): .15 * .05 * .15 = 1.1E-3

Winner: Aerosmith

I	2 (0.11)	eyes	2 (0.11)
don't	2 (0.11)	your	1 (0.06)
want	2 (0.11)	love	1 (0.06)
to	2 (0.11)	and	1 (0.06)
close	2 (0.11)	revenge	1 (0.06)
my	2 (0.11)		

I	3 (0.15)	eyes	1 (0.05)
don't	1 (0.05)	your	3 (0.15)
want	3 (0.15)	love	2 (0.10)
to	1 (0.05)	and	2 (0.10)
close	1 (0.05)	revenge	2 (0.10)
my	1 (0.05)		

LMs over time…

NUS National University of Singapore

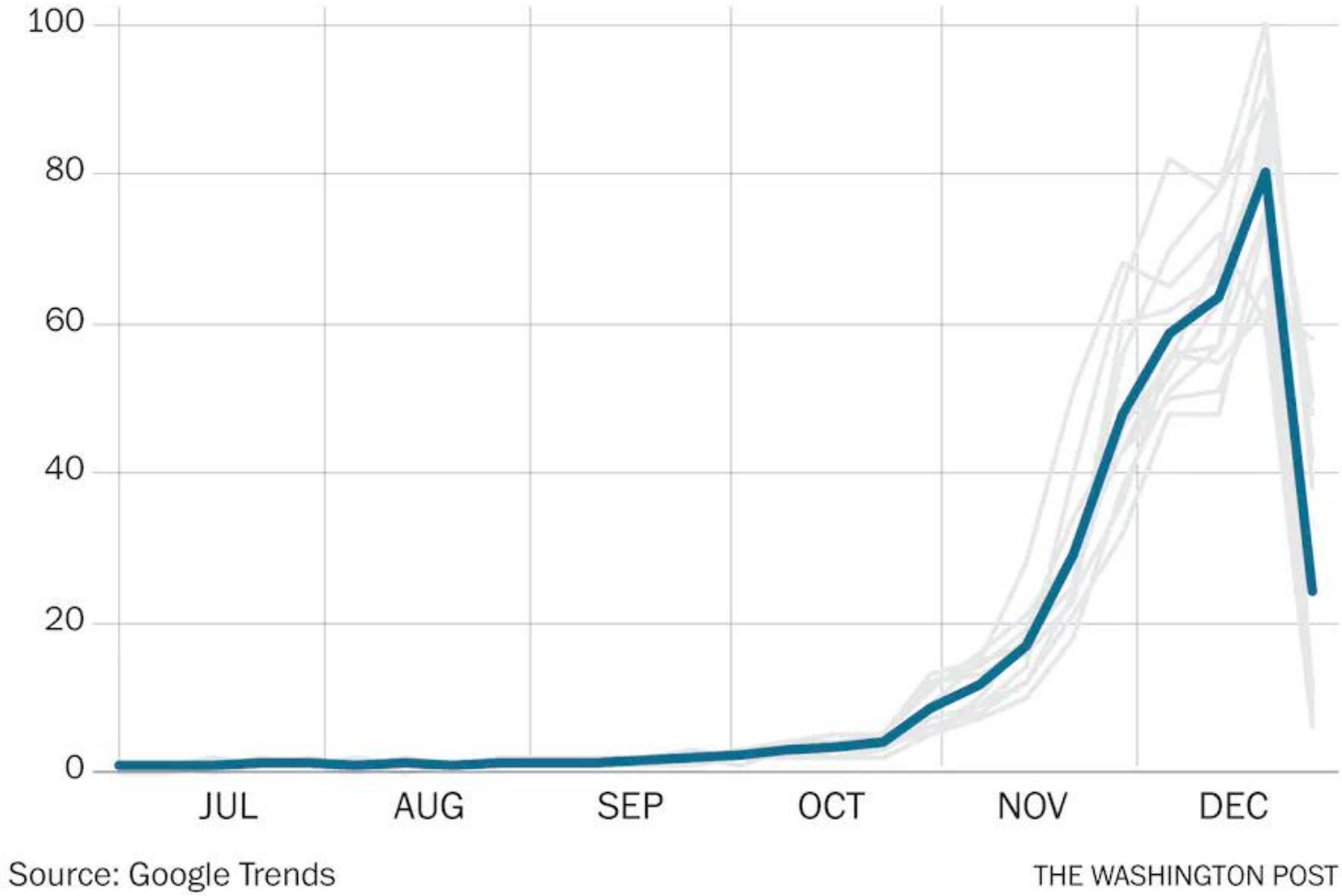
Google Books Ngram Viewer

https://books.google.com/ngrams

Graph these comma-separated phrases: Albert Einstein,Sherlock Holmes,Frankenstein case-insensitive

between 1800 and 2000 from the corpus English with smoothing of 3 . Search lots of books

What 7-gram is this?



Summary

- Ngram LMs are simple but powerful models of language
- Probabilistic computation, with attention to missing or unseen data
- Diminishing returns for larger ngram contexts
- Applicable to many classification tasks

References

- Jurafsky and Martin. Chap 6, Speech and Language Processing
- You'll likely learn this again in CS 4248 Natural Language Processing