CS3245

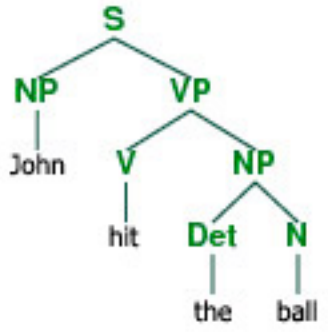
Information Retrieval

1

Lecture 1: Language Models

Modeling Language

- In traditional A.I., we model language with the notion of a formal syntax and semantics
  - From A.I.: Grammar and lexicon
  - Specific rules to specify what is included in or excluded from a language
  - The grammar helps us to interpret the meaning (semantics) of the sentence



What's a language model?

- In some aspects of Information Retrieval (and A.I.), it can be helpful to give a **probabilistic** model of language that is simple without the use of a grammar.

- A language model assigns a probability to a sequence of words.

  e.g. Input: "Forsooth, there is no one I trust more"

  - SMS_LangModel: low probability
  - 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
  - Optical Character Recognition (Scanning)
  - Computer generation of language
  - Typeahead prediction on mobile devices

The Unigram Model

- View language as a 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 token

  - Count(input) = Σ_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"

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

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

Winner: Lady Gaga

- Q3: "love and revenge"

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"

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

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

Winner: Lady Gaga

- Q3: "love and revenge"

P(Aerosmith): 0 + 0 + 0 = 0

P(LadyGaga): 1 + 1 + 1 = 3

Winner: Lady Gaga

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

Would any of these predictions change if we were to use probabilities instead? Why might it be better if we used probabilities?

Blanks on slides, you may want to fill in

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

Can you ask:

- 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

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

Blanks on slides, answers on next slide

Ngram LM

- We can also define a ngram model as a one that 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 "Phone" after seeing "Please turn off your hand"

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

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

Ngram LM

- We can also define a ngram model as a one that 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?

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 exponentially more costly to construct (why?) but yield more accurate models

E.g., Shakespeare n-gram models

1. To him swallowed confess hear both.
2. What means, sir. I confess she? Then all sorts, he is trim, captain.
3. Sweet Prince, Falstaff shall die. Harry of Monmouth's grave.
4. 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 all |V| words
- For a bigram LM, we need to store counts/ probabilities for all |V|*|V| ordered length 2 phrases
- Check your understanding:
  What about a trigram model?

Gets expensive very quickly!

Problem with 0 probabilities

- Q2 (By **Count**): "I don't want"

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

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

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

- Q2 (By **Probability**) : "I don't want"

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

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

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

e.g., assuming |V| = 11

Total # of word types in both lines

I	2	eyes	2
don't	2	your	1
want	2	love	1
to	2	and	1
close	2	revenge	1
my	2	**Total Count**	18

- Q2 (By **Probability**) : "I don't want"

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

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

Winner: Aerosmith

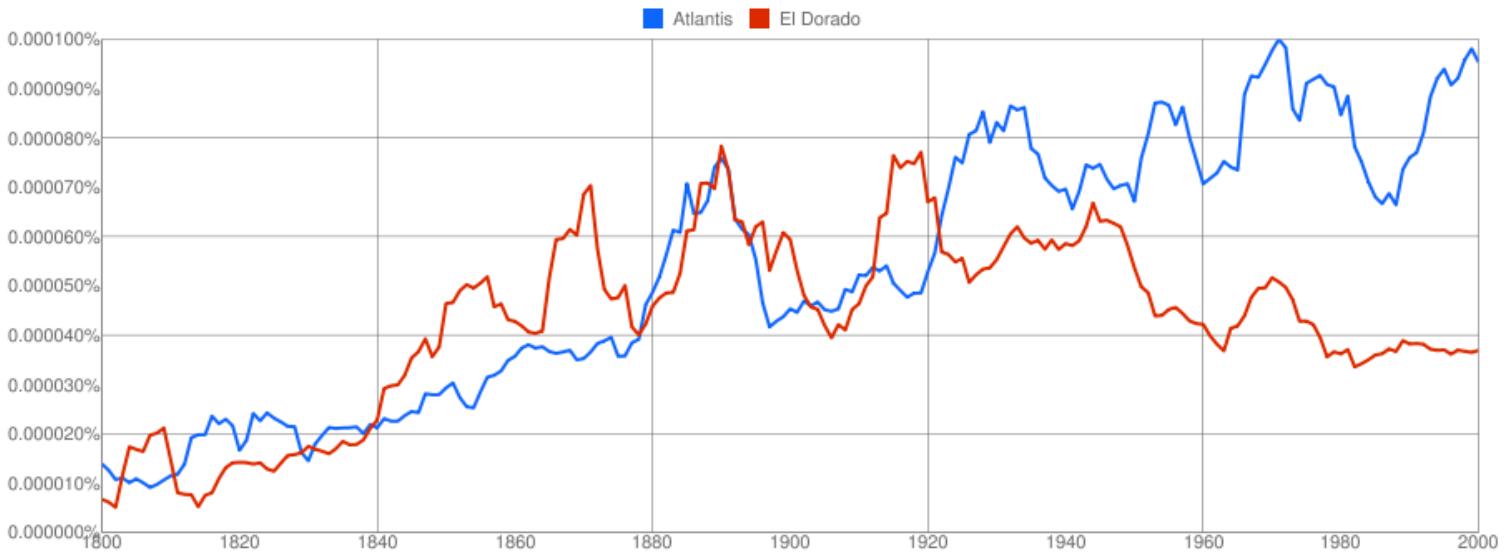
I	3	eyes	1
don't	1	your	3
want	3	love	2
to	1	and	2
close	1	revenge	2
my	1	**Total Count**	20

Now that you know…

Google labs Books Ngram Viewer

http://ngrams.googlelabs.com/

Graph these **case-sensitive** comma-separated phrases: Atlantis,El Dorado
between 1800 and 2000 from the corpus English with smoothing of 3.
Search lots of books



Run your own experiment! Raw data is available for download here.

Try some searches

- Is it from first principals or first principles?
- When did tofu become an acknowledged ingredient in Western cooking?
- Is "stupefy" more likely to occur in English fiction or in general English?
- What language does "arbeit" belong to?

Summary

- Ngram LMs are simple but powerful models of language
- Probabilistic computation, with attention to missing or unseen data
- Diminishing returns for larger n-gram 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