Language modeling

Natural Language Processing LIN350, CS378 Spring 2011 Jason Baldridge

(Remix and augmentation of slides from Jason Eisner, Katrin Erk, Jonas Kuhn, James Martin, Detmar Meurers, and Ray Mooney)





- Formal grammars (e.g. regular, context free) give a hard "binary" model of the legal sentences in a language.
- For NLP, a probabilistic model of a language that gives a probability that a string is a member of a language is typically more useful.
- There are many ways of constructing such models using ngram language models is one of the simplest, most efficient and most effective ways.



- Given a sequence of words, what will be the next word?
- Hard to be sure but if we don't demand extremely high accuracy, we can list many plausible possibilities.
- "I could eat a ..."

horse	cake	car	cavalo
very	the	а	gloink

• Clearly, word prediction is not entirely random.

© 2008 Jason M Baldridge



- I could eat a ...
- Predictions on the basis of
 - language: *horse* vs. *cavalo*
 - syntax: *I could eat a the...
 - selectional preferences: *I could eat a horse/cake/?car*
 - discourse context: If I were a giant metal-crushing machine, I guess I could eat a car without trouble.



Applications

- Speech recognition
- Optical character recognition
- Augmentative communication systems for the disabled
- Language identification
- Context-sensitive spelling error correction
- Statistical machine translation
- N-gram predictions are often used to choose a best candidate out of a set of alternative hypotheses, or rank them.

© 2008 Jason M Baldridge





It's Miller Time!

http://www.youtube.com/watch?v=hhuUIb0xW-M"



- It's Miller Time!
 - http://www.youtube.com/watch?v=hhuUIb0xW-M"
- They said...
 - ... wear the fox hat.
 - ... where the f**k's that?



- It's Miller Time!
 - http://www.youtube.com/watch?v=hhuUIb0xW-M"
- They said...
 - ... wear the fox hat.
 - ... where the f**k's that?
- A recalcitrant robot
 - <u>http://videolectures.net/aaai07_zender_htsn/</u>
 - Why did the robot have so much trouble?

© 2008 Jason M Baldridge

NLP (LIN350/CS378), UT Austin, Fall 2008



- We can attempt to classify documents according to the language a document is (mostly) written in.
- Can sometimes tell by
 - which characters are used, e.g. Liebe Grüße uses ü and $\beta \rightarrow$ German
 - which character encoding is being used e.g., ISO 8859-8 is used to encode Hebrew characters → text is written in Hebrew
- But how can you tell if you are reading English vs. Japanese transliterated into the Roman alphabet? Or Swedish vs. Norwegian? And all phonetically transcribed text is encoded in the same IPA encoding!
- Consider what you base your guess on regarding whether the following is Portuguese or Polish:
 - Czy brak planów zagospodarowania hamuje rozwój Warszawy?



- Histogram of letters?
- Histogram of bigrams?
- Compare to histograms from known language text
- But how to aggregate the evidence?
 - Could steal techniques from information retrieval: Treat every language as a huge document and input text as a query
- **Better solution**: *How likely is this text to be generated randomly?*

© 2008 Jason M Baldridge



• Similar to language ID: how likely is the text according to models based on texts from known topics?



 Similar to language ID: how likely is the text according to models based on texts from known topics?

Topic I sample: In the beginning God created ...

Topic 2 sample:

The history of all hitherto existing society is the history of class struggles. ...



• Similar to language ID: how likely is the text according to models based on texts from known topics?





 Similar to language ID: how likely is the text according to models based on texts from known topics?

Topic I sample: In the beginning God created ...

Capitalism is unfair and has been ruining the lives of millions of people around the world. The profits from the workers' labor ...

Topic 2 sample:

The history of all hitherto existing society is the history of class struggles. ...



• Similar to language ID: how likely is the text according to models based on texts from known topics?





 Similar to language ID: how likely is the text according to models based on texts from known topics?

Topic I sample: In the beginning God created ...

Capitalism is unfair and has been ruining the lives of millions of people around the world. The profits from the workers' labor ...

Topic 2 sample:

The history of all hitherto existing society is the history of class struggles. ... And they have beat their swords to ploughshares, And their spears to pruning-hooks. Nation doth not lift up sword unto nation, neither do they learn war any more. ...



- Break big document or media stream into indexable chunks
- From NPR's All Things Considered:

The U. N. says its observers will stay in Liberia only as long as West African peacekeepers do, but West African states are threatening to pull out of the force unless Liberia's militia leaders stop violating last year's peace accord after 7 weeks of chaos in the capital, Monrovia ... Human rights groups cite peace troops as among those smuggling the arms. I'm Jennifer Ludden, reporting. Whitewater prosecution witness David Hale began serving a 28-month prison sentence today. The Arkansas judge and banker pleaded guilty two years ago to defrauding the Small Business Administration. Hale was the main witness in the Whitewater-related tral that led to the convictions ...



- Break big document or media stream into indexable chunks
- From NPR's *All Things Considered*:

The U. N. says its observers will stay in Liberia only as long as West African peacekeepers do, but West African states are threatening to pull out of the force unless Liberia's militia leaders stop violating last year's peace accord after 7 weeks of chaos in the capital, Monrovia ... Human rights groups cite peace troops as among those smuggling the arms. I'm Jennifer Ludden, reporting. Whitewater prosecution witness David Hale began serving a 28-month prison sentence today. The Arkansas judge and banker pleaded guilty two years ago to defrauding the Small Business Administration. Hale was the main witness in the Whitewater-related tral that led to the convictions ...

Real world spelling errors (from Kukich 1992):

- They are leaving in about fifteen **minuets** to go to her house.
- The study was conducted mainly **be** John Black.
- The design **an** construction of the system will take more than a year.
- Hopefully, all **with** continue smoothly in my absence.
- Can they lave him my messages?
- I need to **notified** the bank of [this problem.]
- He is trying to **fine** out.

© 2008 Jason M Baldridge

- Choose randomly among outputs:
 - Visitant which came into the place where it will be Japanese has admired that there was Mount Fuji.
- Top 10 outputs according to bigram probabilities:

Visitors who came in Japan admire Mount Fuji. Visitors who came in Japan admires Mount Fuji. Visitors who arrived in Japan admire Mount Fuji. Visitors who arrived in Japan admires Mount Fuji. Visitors who came to Japan admire Mount Fuji. A visitor who came in Japan admire Mount Fuji. The visitor who came in Japan admire Mount Fuji. Visitors who came in Japan admire Mount Fuji. The visitor who came in Japan admire Mount Fuji. Mount Fuji is admired by a visitor who came in Japan.

		-
-	-	
X		
	_	

	good English? (n-gram)	good match to French?
Jon appeared in TV.		\checkmark
Appeared on Jon TV.		
In Jon appeared TV.		\checkmark
Jon is happy today.	\checkmark	
Jon appeared on TV.	\checkmark	\checkmark
TV appeared on Jon.	\checkmark	
TV in Jon appeared.		
Jon was not happy.	\checkmark	



- Estimate probability of each word given prior context.
 - P(phone | Please turn off your cell)
- Number of parameters required grows exponentially with the number of words of prior context.
- An N-gram model uses only N–1 words of prior context.
 - Unigram: P(phone)
 - Bigram: P(phone | cell)
 - Trigram: P(phone I your cell)
- The Markov assumption is the presumption that the future behavior of a dynamical system only depends on its recent history. In particular, in a kth-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.

^{© 2008} Jason M Baldridge

- To estimate probabilities, we need to count frequencies.
- What do we count?
 - word forms: inflected word as it appears in the corpus
 - lemmas: morphological root
- The type/token distinction:
 - types: the distinct words in a corpus, i.e. the vocabulary
 - tokens: each instance of a word being seen

For the bird bathed in the pool in the yard:

- 6 types: the, bird, bathed, in, pool, yard
- 9 tokens: the, bird, bathed, in, the, pool, in, the, yard

UNIX tools



- Ken Church, Unix for Poets:
 - http://research.microsoft.com/users/church/wwwfiles/tutorials/unix_for_poets.ps
- Brew and Moens, Data-Intensive Linguistics:
 - http://www.ling.ohio-state.edu/~cbrew/dilbook.ps
- Use Unix tools to…
 - tokenize a document, i.e. split it into words
 - count the occurrences of each word in a corpus
 - count n-grams of words in a corpus
 - remove stop-words from a corpus
 - and more.
- Just the UNIX command line part of Brew and Moens:
 - http://comp.ling.utexas.edu/wiki/lib/exe/fetch.php/commandlinetoolsfornlp.pdf



• One short Unix pipeline can do quite a bit:

```
$ cat holmes.txt | tr -cs '[:alpha:]' '\n' | tr '[:upper:]' '[:lower:]' |
sort | uniq -c | sort -nr | more
5725 the
3063 and
3038 i
2813 to
2719 of
2675 a
1792 in
1762 that
1754 it
1563 you
1483 he
1410 was
...
```

 Each program is a simple one that takes a stream of data from the previous program and passes it on to the next (indicated via the "pipe" l).



• **cat**: list the contents of a file

\$ cat holmes.txt
The Project Gutenberg EBook of The Adventures of Sherlock Holmes
by Sir Arthur Conan Doyle
(#15 in our series by Sir Arthur Conan Doyle)

Copyright laws are changing all over the world. Be sure to check the copyright laws for your country before downloading or redistributing this or any other Project Gutenberg eBook.

This header should be the first thing seen when viewing this Project Gutenberg file. Please do not remove it. Do not change or edit the header without written permission.

Please read the "legal small print," and other information about the eBook and Project Gutenberg at the bottom of this file. Included is important information about your specific rights and restrictions in how the file may be used. You can also find out about how to make a donation to Project Gutenberg, and how to get involved.

• • •



• **tr**: translate a set of characters (**-c** = complement, **-s** = squeeze)

```
$ cat holmes.txt | tr -cs '[:alpha:]' '\n'
The
Project
Gutenberg
EBook
of
The
Adventures
of
Sherlock
Holmes
by
Sir
Arthur
Conan
Doyle
in
our
series
by
Sir
```



• tr to convert upper case letters to lower case

```
$ cat holmes.txt | tr -cs '[:alpha:]' '\n' | tr '[:upper:]' '[:lower:]'
the
project
gutenberg
ebook
of
the
adventures
of
sherlock
holmes
by
sir
arthur
conan
doyle
in
our
series
by
sir
```

```
© 2008 Jason M Baldridge
```



• sort: sort the lines (by default, alphabetically)

```
$ cat holmes.txt | tr -cs '[:alpha:]' '\n' | tr '[:upper:]' '[:lower:]' |
sort
а
а
а
а
а
а
а
а
а
а
а
а
а
а
а
а
а
а
а
```



• **uniq**: merge lines that are the same (**-c** = count)

```
$ cat holmes.txt | tr -cs '[:alpha:]' '\n' | tr '[:upper:]' '[:lower:]' |
sort | uniq -c
2675 a
   3 abandoned
   1 abandons
   1 abbots
   2 aberdeen
   1 abhorrent
   1 abiding
   1 abjure
  31 able
   1 abnormal
   1 abnormally
   1 abode
   1 abominable
   1 abomination
   1 abound
 178 about
  28 above
   2 abroad
   1 abrupt
```



• **sort**: now, by reverse numerical order (**-n**=numeric, **-r**=reverse)

```
$ cat holmes.txt | tr -cs '[:alpha:]' '\n' | tr '[:upper:]' '[:lower:]' |
sort | uniq -c | sort -nr
5725 the
3063 and
3038 i
2813 to
2719 of
2675 a
1792 in
1762 that
1754 it
1563 you
1483 he
1410 was
1159 his
1147 is
1007 my
 933 have
 867 as
 838 with
 830 had
```



- The previous example showed how to count unigrams (1-grams).
- It is straightforward to extend this to bigrams, trigrams, and so on.



- <: direct input from a file
- >: direct output to a file

```
$ tr -cs '[:alpha:]' '\n' < holmes.txt | tr '[:upper:]' '[:lower:]' >
token_per_line.txt
```

- tail: cut off the bottom part of a file (+2 = keep everything but the first line)
- \$ tail +2 token_per_line.txt > token_per_line2.txt
 - **paste**: put the lines of multiple files together (-d=delimiter)
- \$ paste -d ' ' token_per_line.txt token_per_line2.txt > bigrams.txt

● less: list the contents of a file (less is more!)
\$ less bigrams.txt
the project
project gutenberg
gutenberg ebook
ebook of

. . .

• Same as before, but now using bigrams.txt:

```
$ sort < bigrams.txt | uniq -c | sort -nr | more</pre>
732 of the
511 in the
335 it is
312 to the
301 i have
279 it was
257 that i
238 at the
218 and i
200 and the
199 to be
196 upon the
186 i was
184 with a
182 i am
169 of a
168 i had
160 was a
154 that he
 152 that the
```

• grep: display lines matching a pattern

```
$ sort < bigrams.txt | uniq -c | sort -nr | grep holmes</pre>
                                                             more
 111 said holmes
 101 sherlock holmes
  71 mr holmes
  31 holmes i
  24 holmes and
  16 holmes had
  15 holmes was
  14 holmes that
  13 holmes the
  13 holmes as
  12 asked holmes
  11 remarked holmes
   9 holmes you
   8 holmes sat
   8 holmes said
   8 holmes laughing
   8 holmes it
   6 holmes with
   6 holmes when
   6 holmes this
```

- With such frequencies, we can now easily answer questions about the probability of specific unigrams and bigrams.
 - $P(\text{``holmes''}) = Count(\text{``holmes''}) / \sum_{w} Count(w)$

• P("sherlock", "holmes") = Count("sherlock", "holmes")/ $\sum_{w1,w2}$ Count(w1,w2)





- With such frequencies, we can now easily answer questions about the probability of specific unigrams and bigrams.
 - $P(\text{``holmes''}) = Count(\text{``holmes''}) / \sum_{w} Count(w)$

```
$ sort < token_per_line.txt | uniq -c | sort -nr > counts_unigrams.txt
$ grep holmes unigram_counts.txt
466 holmes
$ wa_l token per_line_tyt
```

- \$ wc -l token_per_line.txt
 108206 token_per_line.txt
 - P("sherlock", "holmes") = Count("sherlock", "holmes")/ $\sum_{w1,w2}$ Count(w1,w2)



• $P(\text{``holmes''}) = Count(\text{``holmes''}) / \sum_w Count(w)$

- \$ sort < token_per_line.txt | uniq -c | sort -nr > counts_unigrams.txt
 \$ grep holmes unigram_counts.txt
 466 holmes = 466/108,206 = .00431
- \$ wc -l token_per_line.txt
 108206 token_per_line.txt
 - P("sherlock", "holmes") = Count("sherlock", "holmes")/ $\sum_{w1,w2}$ Count(w1,w2)



• $P(\text{``holmes''}) = Count(\text{``holmes''})/\sum_w Count(w)$

```
$ sort < token_per_line.txt | uniq -c | sort -nr > counts_unigrams.txt
$ grep holmes unigram_counts.txt
466 holmes = 466/108,206 = .00431
$ wc -l token_per_line.txt
108206 token_per_line.txt
```

• P("sherlock", "holmes") = Count("sherlock", "holmes")/ $\sum_{w1,w2}$ Count(w1,w2)

```
$ sort < bigrams.txt | uniq -c | sort -nr > bigram_counts.txt
$ grep "sherlock holmes" bigram_counts.txt
101 sherlock holmes
$ wc -l bigrams.txt
```

108206 bigrams.txt



• $P(\text{``holmes''}) = Count(\text{``holmes''}) / \sum_{w} Count(w)$



• $P(\text{``holmes''}) = Count(\text{``holmes''})/\sum_w Count(w)$





• $P(\text{``holmes''}) = Count(\text{``holmes''})/\sum_w Count(w)$



- We'll see that for language models, the major interest is in the conditional probabilities of a word given the previous one(s), e.g.:
 - P("holmes"l "sherlock")