CS 181: Natural Language Processing

Lecture 6: N-Grams & PoS Tagging

Kim Bruce
Pomona College
Spring 2008

Disclaimer: Slide contents borrowed from many sources on web!

More Problems w/ N-grams

- Sparsity of data
  - Even common words don't occur very often
    - In a million words:
      - "kick" occurs about 10 times
      - "wrist" occurs about 5 times
      - Even common 3 word phrases are unlikely to appear!
    - How to cope with missing data?

It's bad!

count	2-grams	3-grams
1	8,045,024	53,737,350
2	2,065,469	9,229,958
3	970,434	3,654,791
> 4	3,413,290	8,728,789
> 0	14,494,217	75,349,888
possible	6.8×10^{10}	1.7×10^{16}

Taken from data set w/ 261,741 words
365,000,000 words training!

Too Many Zeroes

- 6.8×10^{10} possible bigrams, but only 3.65×10^{8} words in training set.
- Trigrams worse!
- Can't get data set large enough to get them all -- even those that could occur.
- Solution:
  - Redistribute probability to *save* some for those that haven't been encountered.

Getting Rid of Zeroes

- Zeroes for P(w | uv) come in two ways:
  - uvw doesn't exist in training data
  - Even vw doesn't occur!!
- Make counts non-zero - how?
- Must reduce other probabilities so that $\sum_{w' \text{ is word}} P(w' \mid uv) = 1$

LaPlace Smoothing

$$P_{LaPlace}(w|uv) = \frac{C(uvw)+1}{C(uv)+V}$$

- If add 1 to counts of each trigram, then must add V = size of vocabulary so still sums to 1.
- All moved from zero.
- Changes probability too much!

Surprising Results

- Suppose have 20,000 word vocabulary and "threw the" occurs 100 times and "threw the ball" 50 times in 1,000,000 words training
- P(ball | threw the) ≈ 50/100 = .5
- $P_{LaPlace}$(ball | threw the) ≈ (50+1)/(100 + 20,000) = .0025!
- Try $P_{LaPlace}(w|uv) = \frac{C(uvw) + \lambda}{C(uv) + \lambda * V}$ where λ < 1.

What is Problem

- Too much weight to unseen trigrams!
  - 19,900/20,000 given to unseen!!!
- Clearly too much
- How many are actually likely to occur in test text of size 10,000?

Sushi

- At Sushi bar. So far seen 10 tuna, 3 unagi, 2 salmon, 1 shrimp, 1 octopus, 1 yellowtail
- How likely is it for next item to be salmon?
  - 2/18? or ...
- How likely is it to be new kind?

Good-Turing Discounting

- How many types of sushi seen once? 3
- Use this to predict probability for new.
- Let Si = set of words that occur i times.
- Let N_1 = size of S_1. Initial estimate of prob of new words is N_1/N.
  - For sushi: 3/18
  - Must adjust other probabilities, too!
- Let S_2 = words occur twice, N_2 = size of S_2, ...

Good-Turing

- Normally best estimate is all words in S_c occur c times, but must adjust because gave probability to N_0, which not occur at all.
- Good-Turing says use
  - $c^\dagger = (c+1)*N_{c+1}/N_c$ for c>0
- If w in S_c, est prob at $c^\dagger/N$
- Easy exercise shows $\sum_{c \geq 1} c*N_c = N$ and $N_1 + \sum_{c\geq 0}(c^\dagger)*N_c = N$

Sushi

- Using Good-Turing:
  - P(new species) = 3/18 = .1666...
    - If know how many missing, can get prob of each
  - P_{GT}(yellowtail) *(= P_{GT}(octopus) = P_{GT}(shrimp))* = (2*(1/3))/18 = .0372..., compared with P(yellowtail) = 1/18 = .0555...
  - P_{GT}(salmon) = (3*(1/1))/18 = .1666... compared with P(salmon) = 2/18 = .111...
- Works better if lots of data ...
- What about P_{GT}(tuna)?

Still Problems

- Can't compute $c^\dagger = (c+1)N_{c+1}/N_c$ if N_c is 0
- Smooth data by fitting $\log(N_c)$ to linear regression on c: Find a, b to find best fit for $\log(N_c) = a + b \log c$
- Tend not to use $c^\dagger$ for large values of c (> k) (e.g. c > 5). Must readjust:
- $c^\dagger = \dfrac{(c+1)\frac{N_{c+1}}{N_c} - c\frac{(k+1)N_{k+1}}{N_1}}{1 - \frac{(k+1)N_{k+1}}{N_1}}$ for $1 \leq c \leq k$

Other Attempts

- Linear interpolation: Estimate prob as an average of lower-order n-grams:

  $$\hat{P}(z|x,y) = \lambda_1 P(z|x,y) + \lambda_2 P(z|y) + \lambda_3 P(z)$$

  where $\lambda_1 + \lambda_2 + \lambda_3 = 1$
- Fit data to find optimal λ's.

Backoff

$$P_{katz}(z|x,y) = \begin{cases} P^\dagger(z|x,y), & \text{if } C(xyz) > 0 \\ \alpha(x,y)P_{katz}(z|y), & \text{else if } C(x,y) > 0 \\ P^\dagger(z), & \text{otherwise} \end{cases}$$

where

$$P_{katz}(z|y) = \begin{cases} P^\dagger(z|y), & \text{if } C(yz) > 0 \\ \alpha(y)P^\dagger(z), & \text{otherwise} \end{cases}$$

α's required to get true probability

POS Tagging

Parts of Speech

- Predict behavior of previously unseen words.
- Divide into classes that behave similarly
- Traditionally: noun, verb, pronoun, preposition, adverb, conjunction, adjective, and article
- Brown (87), Penn (45), Susanne (353)

What is use of POS?

- Tell us what words likely occur in neighborhood:
  - adjective -> noun
  - personal pronoun -> verbs
  - possessive pronoun -> nouns
- Speech synthesis:
  - Ex.: object, content, discount
- Speech recognition
- Help in info retrieval

Parts of Speech

- Closed Classes (fixed membership):
  - prepositions, determiners, pronouns, conjunctions, aux. verbs, particles, numerals
  - usually function words - freq. occurring, often short. Differ more from lang to lang.
- Open classes
  - nouns (proper/common, count/mass), verbs, adjectives, adverbs
  - adverbs a mess:
    - *Unfortunately,* John walked *home extremely slowly yesterday.*

Penn Tagset

Tag	Description	Example	Tag	Description	Example
CC	Coordin. Conjunction	*and, but, or*	SYM	Symbol	+,%, &
CD	Cardinal number	*one, two, three*	TO	"to"	*to*
DT	Determiner	*a, the*	UH	Interjection	*ah, oops*
EX	Existential 'there'	*there*	VB	Verb, base form	*eat*
FW	Foreign word	*mea culpa*	VBD	Verb, past tense	*ate*
IN	Preposition/sub-conj	*of, in, by*	VBG	Verb, gerund	*eating*
JJ	Adjective	*yellow*	VBN	Verb, past participle	*eaten*
JJR	Adj., comparative	*bigger*	VBP	Verb, non-3sg pres	*eat*
JJS	Adj., superlative	*wildest*	VBZ	Verb, 3sg pres	*eats*
LS	List item marker	*1, 2, One*	WDT	Wh-determiner	*which, that*
MD	Modal	*can, should*	WP	Wh-pronoun	*what, who*
NN	Noun, sing. or mass	*llama*	WP$	Possessive wh-	*whose*
NNS	Noun, plural	*llamas*	WRB	Wh-adverb	*how, where*
NNP	Proper noun, singular	*IBM*	$	Dollar sign	*$*
NNPS	Proper noun, plural	*Carolinas*	#	Pound sign	*#*
PDT	Predeterminer	*all, both*	"	Left quote	' or "
POS	Possessive ending	*'s*	"	Right quote	' or "
PRP	Personal pronoun	*I, you, he*	(	Left parenthesis	[, (, {, <
PRP$	Possessive pronoun	*your, one's*	)	Right parenthesis	],), }, >
RB	Adverb	*quickly, never*	,	Comma	,
RBR	Adverb, comparative	*faster*	.	Sentence-final punc	. ! ?
RBS	Adverb, superlative	*fastest*	:	Mid-sentence punc	: ; ... – -
RP	Particle	*up, off*			

Figure 5.6 Penn Treebank part-of-speech tags (including punctuation).

POS Tagging

- Assignment of POS tag to each word & punctuation marker in corpus:
  - "/" The/DT guys/NNS that/WDT make/VBP traditional/JJ hardware/NN are/VBP really/RB being/VBG obsoleted/VBN by/IN microprocessor-based/JJ machines/NNS ,/, "/" said/VBD Mr./NNP Benton/NNP ./.
- Must resolve ambiguities
- Brown corpus: 11.5% of word types & 40% of tokens are ambiguous
  - though some easily recognizable!

Ambiguity in Brown Corpus

- Unambiguous (1 tag): 35,340
- Ambiguous (2-7): 4,100

2 tags	3,760
3 tags	264
4 tags	61
5 tags	12
6 tags	2
7 tags	1

Determining Tags

- Some tags more likely than others.
- Assign most likely - gives 90% accuracy
- Use POS tags of adjacent words:
  - the/AT red/JJ drink/NN *versus*
  - the/AT red/JJ drink/VBP

Kinds of Taggers

- Rule-Based Tagger - English Two Level Analysis
- Stochastic Tagger: Hidden Markov Model
- Transformation-based Tagger

Rule-Based Taggers

- Basic idea:
  - Use dictionary to assign all reasonable tags to words
  - Remove tags according to set of rules:
    - *if word+1 is an adj, adv, or quantifier and the following is a sentence boundary and word-1 is not a verb like "consider" then eliminate non-adv else eliminate adv.*
    - Typically more than 1000 hand-written rules, but may also be machine-learned.

ENGTWOL Lexicon

Word	POS	Additional POS features
smaller	ADJ	COMPARATIVE
entire	ADJ	ABSOLUTE ATTRIBUTIVE
fast	ADV	SUPERLATIVE
that	DET	CENTRAL DEMONSTRATIVE SG
all	DET	PREDETERMINER SG/PL QUANTIFIER
dog's	N	GENITIVE SG
furniture	N	NOMINATIVE SG NOINDEFDETERMINER
one-third	NUM	SG
she	PRON	PERSONAL FEMININE NOMINATIVE SG3
show	V	PRESENT -SG3 VFIN
show	N	NOMINATIVE SG
shown	PCP2	SVOO SVO SV
occurred	PCP2	SV
occurred	V	PAST VFIN SV

Figure 5.11 Sample lexical entries from the ENGTWOL lexicon described in Voutilainen (1995) and Heikkilä (1995).

Stage 1

- Run through lexicon transducer to get all parts of speech
- Ex.: *Pavlov had shown that salivation ...*
  - Pavlov PAVLOV N NOM SG PROPER
  - had HAVE V PAST VFIN SVO
    ~~HAVE PCP2 SVO~~
  - shown SHOW PCP2 SVOO SV
  - that ~~ADV~~
    ~~PRON DEM SG~~
    ~~DET CENTRAL DEM SG~~
    CS
  - salivation N NOM SG

Stage 2

- Apply constraints to rule out cases:
- Ex: Adverbial "that" rule:

  Given input: "that"

  If next word is adj, adverb, or quantifier and following next is a sentence boundary and previous word is not a verb like "consider" which allows adjs as object complements

  then eliminate non-ADV tags

  else eliminate ADV tag

NLTK & Tagging

- Simplest possible tagger assigns all "noun"

```
import nltk

inputText = "You've made that same mistake 16 times now!"
inputTokens = inputText.split()

defaultTagger = nltk.DefaultTagger('NN')
for t in defaultTagger.tag(inputTokens):
    print t
```

Regular Expression Taggers

- Use regular expressions to select:

```
import nltk

default_pattern = (r'.*', 'NN')
cd_pattern = (r'\b[0-9]+(?:\.[0-9]+)?\b', 'CD')
patterns = [cd_pattern, default_pattern]
NN_CD_tagger = nltk.RegexpTagger(patterns)
print NN_CD_tagger.tag(inputTokens):

# [("You've", 'NN'), ('made', 'NN'), ('that', 'NN'),
('same', 'NN'), ('mistake', 'NN'), ('16', 'CD'), ('times',
'NN'), ('now!', 'NN')]
```

Any Questions?