N-Gram – Part 2
ICS 482 Natural Language Processing

Lecture 8: N-Gram – Part 2

Husni Al-Muhtaseb

بسم الله الرحمن الرحيم
ICS 482 Natural Language Processing

Lecture 8: N-Gram – Part 2

Husni Al-Muhtaseb

NLP Credits and Acknowledgment

These slides were adapted from presentations of the Authors of the book

SPEECH and LANGUAGE PROCESSING:

An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition

and some modifications from presentations found in the WEB by several scholars including the following

NLP Credits and Acknowledgment

If your name is missing please contact me
muhtaseb
At
Kfupm.
Edu.
sa

NLP Credits and Acknowledgment

- Husni Al-Muhtaseb
- James Martin
- Jim Martin
- Dan Jurafsky
- Sandiway Fong
- Song young in
- Paula Matuszek
- Mary-Angela Papalaskari
- Dick Crouch
- Tracy Kin
- L. Venkata Subramaniam
- Martin Volk
- Bruce R. Maxim
- Jan Hajič
- Srinath Srinivasa
- Simeon Ntafos
- Paolo Pirjanian
- Ricardo Vilalta
- Tom Lenaerts
- Heshaam Feili
- Björn Gambäck
- Christian Korthals
- Thomas G. Dietterich
- Devika Subramanian
- Duminda Wijesekera
- Lee McCluskey
- David J. Kriegman
- Kathleen McKeown
- Michael J. Ciaraldi
- David Finkel
- Min-Yen Kan
- Andreas Geyer-Schulz
- Franz J. Kurfess
- Tim Finin
- Nadjet Bouayad
- Kathy McCoy
- Khurshid Ahmad
- Staffan Larsson
- Robert Wilensky
- Feiyu Xu
- Jakub Piskorski
- Rohini Srihari
- Mark Sanderson
- Andrew Elks
- Marc Davis
- Ray Larson
- Jimmy Lin
- Marti Hearst
- Andrew McCallum
- Nick Kushmerick
- Mark Craven
- Chia-Hui Chang
- Diana Maynard
- James Allan
- Martha Palmer
- julia hirschberg
- Elaine Rich
- Christof Monz
- Bonnie J. Dorr
- Nizar Habash
- Massimo Poesio
- David Goss-Grubbs
- Thomas K Harris
- John Hutchins
- Alexandros Potamianos
- Mike Rosner
- Latifa Al-Sulaiti
- Giorgio Satta
- Jerry R. Hobbs
- Christopher Manning
- Hinrich Schütze
- Alexander Gelbukh
- Gina-Anne Levow
- Guitao Gao
- Qing Ma

Previous Lectures

- Pre-start questionnaire
- Introduction and Phases of an NLP system
- NLP Applications - Chatting with Alice
- Finite State Automata & Regular Expressions & languages
- Deterministic & Non-deterministic FSAs
- Morphology: Inflectional & Derivational
- Parsing and Finite State Transducers
- Stemming & Porter Stemmer
- 20 Minute Quiz
- Statistical NLP – Language Modeling
- N Grams

Today's Lecture

- NGrams
- Bigram
- Smoothing and NGram
  - Add one smoothing
  - Witten-Bell Smoothing

Simple N-Grams

- An N-gram model uses the previous N-1 words to predict the next one:
  - $P(w_n \mid w_{n-1})$
  - We'll be dealing with P(<word> | <some previous words>)
- unigrams: P(dog)
- bigrams: P(dog | big)
- trigrams: P(dog | the big)
- quadrigrams: P(dog | the big dopey)

Chain Rule

conditional probability: $$P(A \mid B) = \frac{P(A \wedge B)}{P(B)}$$

$$P(A \wedge B) = P(A \mid B)P(B)$$

and

$$P(A \wedge B) = P(B \mid A)P(A)$$

So: $$P(A \wedge B) = P(B \mid A)P(A)$$

“the dog”: $$P(The \wedge dog) = P(dog \mid the)P(the)$$

“the dog bites”:

$$P(The \wedge dog \wedge bites) = P(The)P(dog \mid The)P(bites \mid The \wedge dog)$$

Chain Rule

the probability of a word sequence is the probability of a conjunctive event.

$$P(w_1^n) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1^2)\ldots P(w_n \mid w_1^{n-1})$$

$$= \prod_{k=1}^{n} P(w_k \mid w_1^{k-1})$$

Unfortunately, that’s really not helpful in general. Why?

Markov Assumption

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

- $P(w_n)$ can be approximated using only N-1 previous words of context
- This lets us collect statistics in practice
- Markov models are the class of probabilistic models that assume that we can predict the probability of some future unit without looking too far into the past
- Order of a Markov model: length of prior context

Language Models and N-grams

- Given a word sequence: $w_1\ w_2\ w_3\ \ldots\ w_n$
- **Chain rule**
  - $p(w_1\ w_2) = p(w_1)\ p(w_2|w_1)$
  - $p(w_1\ w_2\ w_3) = p(w_1)\ p(w_2|w_1)\ p(w_3|w_1 w_2)$
  - ...
  - $p(w_1\ w_2\ w_3 \ldots w_n) = p(w_1)\ p(w_2|w_1)\ p(w_3|w_1 w_2) \ldots\ p(w_n|w_1 \ldots w_{n-2}\ w_{n-1})$
- Note:
  - It's not easy to collect (meaningful) statistics on $p(w_n|w_{n-1} w_{n-2} \ldots w_1)$ for all possible word sequences
- **Bigram approximation**
  - *just look at the previous word only* (*not all the proceedings words*)
  - Markov Assumption: finite length history
  - 1st order Markov Model
  - $p(w_1\ w_2\ w_3 .. w_n) = p(w_1)\ p(w_2|w_1)\ p(w_3|w_1 w_2)\ ..p(w_n|w_1 \ldots w_{n-3} w_{n-2} w_{n-1})$
  - $p(w_1\ w_2\ w_3 .. w_n) \approx p(w_1)\ p(w_2|w_1)\ p(w_3|w_2) .. p(w_n|w_{n-1})$
- Note:
  - $p(w_n|w_{n-1})$ is a lot easier to estimate well than $p(w_n|w_1 .. w_{n-2}\ w_{n-1})$

Language Models and N-grams

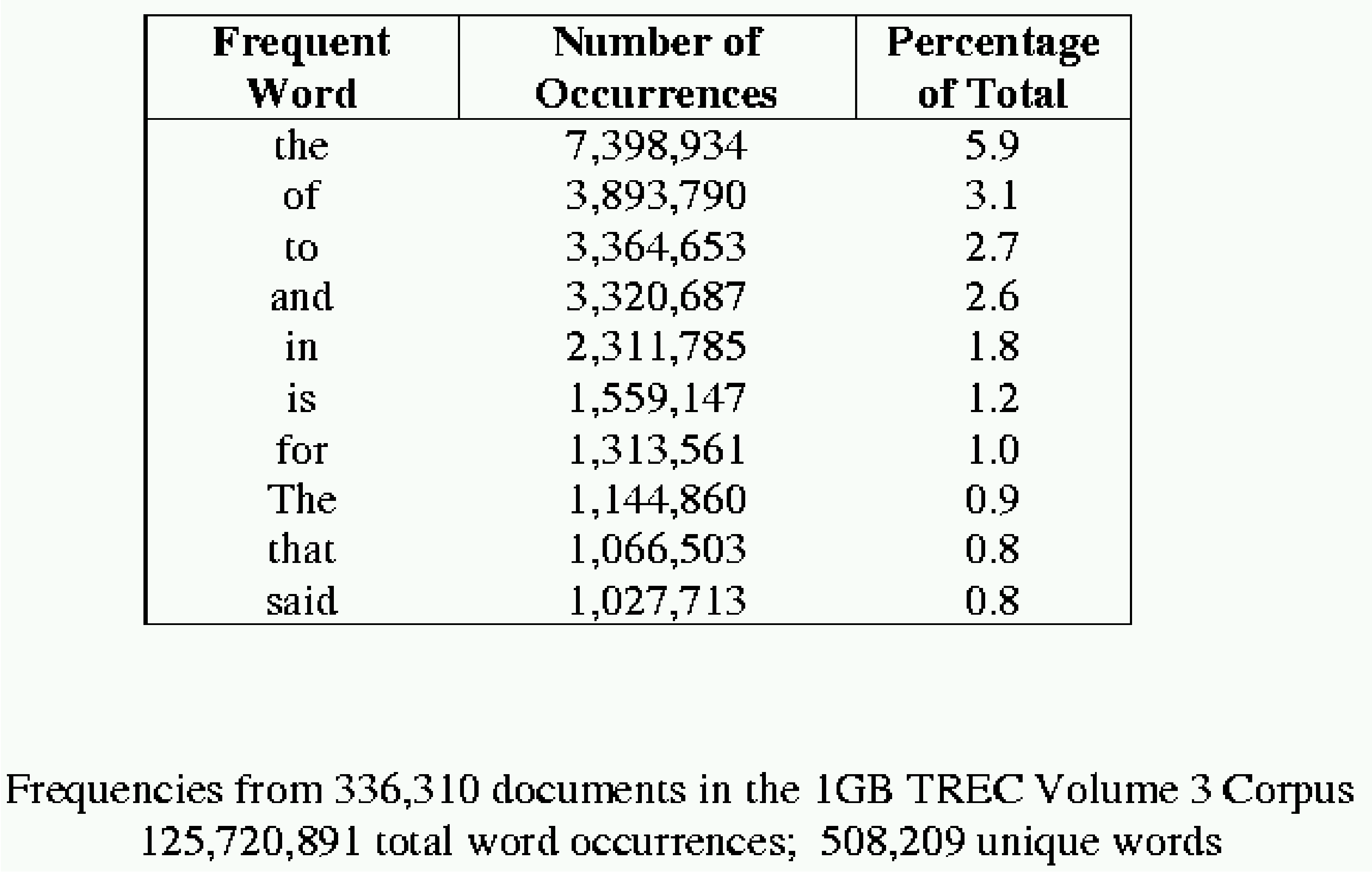
- Given a word sequence: $w_1\ w_2\ w_3\ \ldots\ w_n$
- **Chain rule**
  - $p(w_1\ w_2) = p(w_1)\ p(w_2|w_1)$
  - $p(w_1\ w_2\ w_3) = p(w_1)\ p(w_2|w_1)\ p(w_3|w_1 w_2)$
  - ...
  - $p(w_1\ w_2\ w_3 \ldots w_n) = p(w_1)\ p(w_2|w_1)\ p(w_3|w_1 w_2) \ldots\ p(w_n|w_1 \ldots w_{n-2}\ w_{n-1})$
- **Trigram approximation**
  - 2nd order Markov Model
  - *just look at the preceding two words only*
  - $p(w_1\ w_2\ w_3\ w_4 \ldots w_n) = p(w_1)\ p(w_2|w_1)\ p(w_3|w_1 w_2)\ p(w_4|w_1 w_2 w_3) \ldots p(w_n|w_1 \ldots w_{n-3} w_{n-2} w_{n-1})$
  - $p(w_1\ w_2\ w_3 \ldots w_n) \approx p(w_1)\ p(w_2|w_1)\ p(w_3|w_1 w_2) p(w_4|w_2 w_3) \ldots p(w_n\ |w_{n-2}\ w_{n-1})$
- Note:
  - $p(w_n|w_{n-2} w_{n-1})$ is a lot easier to estimate well than $p(w_n|w_1 \ldots w_{n-2}\ w_{n-1})$ but harder than $p(w_n|w_{n-1}\ )$

Corpora

- Corpora are (generally online) collections of text and speech
- e.g.
  - Brown Corpus (1M words)
  - Wall Street Journal and AP News corpora
  - ATIS, Broadcast News (speech)
  - TDT (text and speech)
  - Switchboard, Call Home (speech)
  - TRAINS, FM Radio (speech)

Sample Word frequency (count)Data

(The Text REtrieval Conference) - (from B. Croft, UMass)



Frequent Word	Number of Occurrences	Percentage of Total
the	7,398,934	5.9
of	3,893,790	3.1
to	3,364,653	2.7
and	3,320,687	2.6
in	2,311,785	1.8
is	1,559,147	1.2
for	1,313,561	1.0
The	1,144,860	0.9
that	1,066,503	0.8
said	1,027,713	0.8

Frequencies from 336,310 documents in the 1GB TREC Volume 3 Corpus
125,720,891 total word occurrences; 508,209 unique words

Counting Words in Corpora

- Probabilities are based on counting things, so ….
- What should we count?
- Words, word classes, word senses, speech acts …?
- What is a word?
  - e.g., are cat and cats the same word?
  - September and Sept?
  - zero and oh?
  - Is seventy-two one word or two? AT&T?
- Where do we find the things to count?

Terminology

- Sentence: unit of written language
- Utterance: unit of spoken language
- Wordform: the inflected form that appears in the corpus
- Lemma: lexical forms having the same stem, part of speech, and word sense
- Types: number of distinct words in a corpus (vocabulary size)
- Tokens: total number of words

Training and Testing

- Probabilities come from a training corpus, which is used to design the model.
  - narrow corpus: probabilities don't generalize
  - general corpus: probabilities don't reflect task or domain
- A separate test corpus is used to evaluate the model

Simple N-Grams

- An N-gram model uses the previous N-1 words to predict the next one:
  - $P(w_n \mid w_{n-1})$
  - We'll be dealing with P(<word> | <some prefix>)
- unigrams: P(dog)
- bigrams: P(dog | big)
- trigrams: P(dog | the big)
- quadrigrams: P(dog | the big red)

Using N-Grams

- Recall that
  - $P(w_n \mid w_{1..n-1}) \approx P(w_n \mid w_{n-N+1..n-1})$
- For a bigram grammar
  - P(sentence) can be approximated by multiplying all the bigram probabilities in the sequence
  - P(I want to eat Chinese food) = P(I | <start>) P(want | I) P(to | want) P(eat | to) P(Chinese | eat) P(food | Chinese) P(<end>|food)

Chain Rule

- Recall the definition of conditional probabilities

$$P(A \mid B) = \frac{P(A \wedge B)}{P(B)}$$

- Rewriting

$$P(A \wedge B) = P(A \mid B) P(B)$$

- Or…

$$P(The\ big) = P(big \mid the) P(the)$$

- Or…

$$P(The\ big) = P(the) P(big \mid the)$$

Example

- The big red dog
- P(The)*P(big|the)*P(red|the big)*P(dog|the big red)
- Better P(The| <Beginning of sentence>) written as P(The | <S>)
- Also <end> for end of sentence

General Case

- The word sequence from position 1 to n is w_1^n
- So the probability of a sequence is

$$P(w_1^n) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1^2)\ldots P(w_n \mid w_1^{n-1})$$

$$= P(w_1)\prod_{k=2}^{n} P(w_k \mid w_1^{k-1})$$

Unfortunately

- That doesn’t help since its unlikely we’ll ever gather the right statistics for the prefixes.

Markov Assumption

- Assume that the entire prefix history isn’t necessary.
- In other words, an event doesn’t depend on all of its history, just a fixed length near history

Markov Assumption

- So for each component in the product replace each with its approximation (assuming a prefix (Previous words) of N)

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

N-Grams
The big red dog

- Unigrams: P(dog)
- Bigrams: P(dog|red)
- Trigrams: P(dog|big red)
- Four-grams: P(dog|the big red)

In general, we’ll be dealing with

P(Word| Some fixed prefix)

Note: prefix is Previous words

N-gram models can be trained by counting and normalization

Bigram: $$P(w_n \mid w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

Ngram: $$P(w_n \mid w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n-1}w_n)}{C(w_{n-N+1}^{n-1})}$$

An example

- <s> I am Sam <\s>
- <s> Sam I am <\s>
- <s> I do not like green eggs and meet <\s>

$$P(w_n|w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n-1}w_n)}{C(w_{n-N+1}^{n-1})}$$

$$P(I\ |<s>) = \frac{2}{3} = 0.67$$

$$P(Sam\ |<s>) = \frac{1}{3} = 0.33$$

$$P(am\ |\ I) = \frac{2}{3} = 0.67$$

$$P(<\backslash s>|\ Sam) = \frac{1}{2} = 0.5$$

$$P(<s>|\ Sam) = \frac{1}{2} = 0.5$$

$$P(Sam\ |\ am) = \frac{1}{2} = 0.5$$

$$P(do\ |\ I) = \frac{1}{1} = 1.0$$

BERP Bigram Counts

BErkeley Restaurant Project (speech)

	I	Want	To	Eat	Chinese	Food	lunch
I	8	1087	0	13	0	0	0
Want	3	0	786	0	6	8	6
To	3	0	10	860	3	0	12
Eat	0	0	2	0	19	2	52
Chinese	2	0	0	0	0	120	1
Food	19	0	17	0	0	0	0
Lunch	4	0	0	0	0	1	0

BERP Bigram Probabilities

- Normalization: divide each row's counts by appropriate unigram counts

I	Want	To	Eat	Chinese	Food	Lunch
3437	1215	3256	938	213	1506	459

- Computing the probability of I I
  - C(I|I)/C(all I)
  - p = 8 / 3437 = .0023
- A bigram grammar is an NxN matrix of probabilities, where N is the vocabulary size

A Bigram Grammar Fragment from BERP

Eat on	.16	Eat Thai	.03
Eat some	.06	Eat breakfast	.03
Eat lunch	.06	Eat in	.02
Eat dinner	.05	Eat Chinese	.02
Eat at	.04	Eat Mexican	.02
Eat a	.04	Eat tomorrow	.01
Eat Indian	.04	Eat dessert	.007
Eat today	.03	Eat British	.001

<start> I	.25	Want some	.04
<start> I’d	.06	Want Thai	.01
<start> Tell	.04	To eat	.26
<start> I’m	.02	To have	.14
I want	.32	To spend	.09
I would	.29	To be	.02
I don’t	.08	British food	.60
I have	.04	British restaurant	.15
Want to	.65	British cuisine	.01
Want a	.05	British lunch	.01

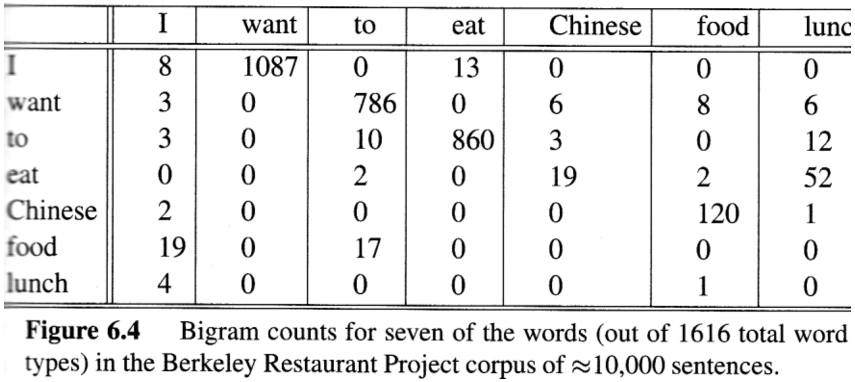
Language Models and N-grams

- **Example**:

$w_{n-1}w_n$ bigram frequencies

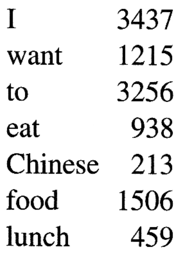
w_n

W_{n-1}



	I	want	to	eat	Chinese	food	lunc
I	8	1087	0	13	0	0	0
want	3	0	786	0	6	8	6
to	3	0	10	860	3	0	12
eat	0	0	2	0	19	2	52
Chinese	2	0	0	0	0	120	1
food	19	0	17	0	0	0	0
lunch	4	0	0	0	0	1	0

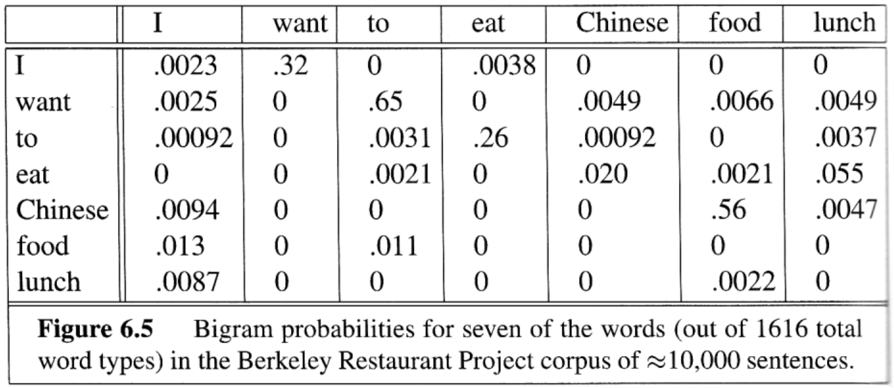
Figure 6.4 Bigram counts for seven of the words (out of 1616 total word types) in the Berkeley Restaurant Project corpus of ≈10,000 sentences.



I	3437
want	1215
to	3256
eat	938
Chinese	213
food	1506
lunch	459

unigram frequencies

bigram probabilities



	I	want	to	eat	Chinese	food	lunch
I	.0023	.32	0	.0038	0	0	0
want	.0025	0	.65	0	.0049	.0066	.0049
to	.00092	0	.0031	.26	.00092	0	.0037
eat	0	0	.0021	0	.020	.0021	.055
Chinese	.0094	0	0	0	0	.56	.0047
food	.013	0	.011	0	0	0	0
lunch	.0087	0	0	0	0	.0022	0

Figure 6.5 Bigram probabilities for seven of the words (out of 1616 total word types) in the Berkeley Restaurant Project corpus of ≈10,000 sentences.

sparse matrix

zeros probabilities unusable

(*we'll need to do* ***smoothing***)

Example

- P(I want to eat British food) = P(I|<start>) P(want|I) P(to|want) P(eat|to) P(British|eat) P(food|British) = .25*.32*.65*.26*.001*.60 = 0.0000081 *(different from textbook)*
- vs. I want to eat Chinese food = .00015

Note on Example

- Probabilities seem to capture “syntactic” facts, “world knowledge”
  - eat is often followed by a NP
  - British food is not too popular

What do we learn about the language?

- What's being captured with ...
  - P(want | I) = .32
  - P(to | want) = .65
  - P(eat | to) = .26
  - P(food | Chinese) = .56
  - P(lunch | eat) = .055

Some Observations

- P(I | I)
- P(want | I)
- P(I | food)

- I I I want
- I want I want to
- The food I want is

- What about
  - P(I | I) = .0023 I I I I want
  - P(I | want) = .0025 I want I want
  - P(I | food) = .013 the kind of food I want is ...

To avoid underflow use Logs

- You don’t really do all those multiplies. The numbers are too small and lead to underflows
- Convert the probabilities to logs and then do additions.
- To get the real probability (if you need it) go back to the antilog.

Generation

- Choose N-Grams according to their probabilities and string them together

BERP

- I want

  want to

  to eat

  eat Chinese

  Chinese food

  food .

Some Useful Observations

- A small number of events occur with high frequency
    - You can collect reliable statistics on these events with relatively small samples
- A large number of events occur with small frequency
    - You might have to wait a long time to gather statistics on the low frequency events

Some Useful Observations

- Some zeroes are really zeroes
  - Meaning that they represent events that can’t or shouldn’t occur
- On the other hand, some zeroes aren’t really zeroes
  - They represent low frequency events that simply didn’t occur in the corpus

Problem

- Let’s assume we’re using N-grams
- How can we assign a probability to a sequence where one of the component n-grams has a value of zero
- Assume all the words are known and have been seen
  - Go to a lower order n-gram
  - Back off from bigrams to unigrams
  - Replace the zero with something else

Add-One

- Make the zero counts 1.
- Justification: They’re just events you haven’t seen yet. If you had seen them you would only have seen them once. so make the count equal to 1.

Add-one: Example

unsmoothed bigram counts:

2nd word

1st word

	I	want	to	eat	Chinese	food	lunch	…	**Total (N)**
I	8	1087	0	13	0	0	0		3437
want	3	0	786	0	6	8	6		1215
to	3	0	10	860	3	0	12		3256
eat	0	0	2	0	19	2	52		938
Chinese	2	0	0	0	0	120	1		213
food	19	0	17	0	0	0	0		1506
lunch	4	0	0	0	0	1	0		459
…									

unsmoothed normalized bigram probabilities:

	I	want	to	eat	Chinese	food	lunch	…	**Total**
I	.0023 (8/3437)	.32	0	.0038 (13/3437)	0	0	0		1
want	.0025	0	.65	0	.0049	.0066	.0049		1
to	.00092	0	.0031	.26	.00092	0	.0037		1
eat	0	0	.0021	0	.020	.0021	.055		1
Chinese	.0094	0	0	0	0	.56	.0047		1
food	.013	0	.011	0	0	0	0		1
lunch	.0087	0	0	0	0	.0022	0		1
…									

Add-one: Example (con’t)

add-one smoothed bigram counts:

	I	*want*	*to*	*eat*	*Chinese*	*food*	*lunch*	…	***Total (N+V)***
I	~~8~~ 9	~~1087~~ 1088	1	14	1	1	1		~~3437~~ 5053
want	~~3~~ 4	1	787	1	7	9	7		2831
to	4	1	11	861	4	1	13		4872
eat	1	1	23	1	20	3	53		2554
Chinese	3	1	1	1	1	121	2		1829
food	20	1	18	1	1	1	1		3122
lunch	5	1	1	1	1	2	1		2075

add-one normalized bigram probabilities:

	I	*want*	*to*	*eat*	*Chinese*	*food*	*lunch*	…	***Total***
I	.0018 (9/5053)	.22	.0002	.0028 (14/5053)	.0002	.0002	.0002		1
want	.0014	.00035	.28	.00035	.0025	.0032	.0025		1
to	.00082	.00021	.0023	.18	.00082	.00021	.0027		1
eat	.00039	.00039	.0012	.00039	.0078	.0012	.021		1
Chinese	.0016	.00055	.00055	.00055	.00055	.066	.0011		1
food	.0064	.00032	.0058	.00032	.00032	.00032	.00032		1
lunch	.0024	.00048	.00048	.00048	.00048	.0022	.00048		1

The example again

unsmoothed bigram counts:

V= 1616 word types

	I	*want*	*to*	*eat*	*Chinese*	*food*	*lunch*	…	**Total (N)**
I	8	1087	0	13	0	0	0		3437
want	3	0	786	0	6	8	6		1215
to	3	0	10	860	3	0	12		3256
eat	0	0	2	0	19	2	52		938
Chinese	2	0	0	0	0	120	1		213
food	19	0	17	0	0	0	0		1506
lunch	4	0	0	0	0	1	0		459

V= 1616

Smoothed P(I eat)
= (C(I eat) + 1) / (nb bigrams starting with "I" + nb of possible bigrams starting with "I")
= (13 + 1) / (3437 + 1616)
= 0.0028

Smoothing and N-grams

I	3437
want	1215
to	3256
eat	938
Chinese	213
food	1506
lunch	459

- **Add-One Smoothing**
  - add 1 to all frequency counts
- **Bigram**
  - $p(w_n|w_{n-1}) = (C(w_{n-1}w_n)+1)/(C(w_{n-1})+V)$
  - $(C(w_{n-1}\ w_n)+1)^*\ C(w_{n-1})\ /(C(w_{n-1})+V)$
- **Frequencies**

	I	**want**	**to**	**eat**	**Chinese**	**food**	**lunch**
I	8	1087	0	13	0	0	0
want	3	0	786	0	6	8	6
to	3	0	10	860	3	0	12
eat	0	0	2	0	19	2	52
Chinese	2	0	0	0	0	120	1
food	19	0	17	0	0	0	0
lunch	4	0	0	0	0	1	0

	I	**want**	**to**	**eat**	**Chinese**	**food**	**lunch**
I	6.12	740.05	0.68	9.52	0.68	0.68	0.68
want	1.72	0.43	337.76	0.43	3.00	3.86	3.00
to	2.67	0.67	7.35	575.41	2.67	0.67	8.69
eat	0.37	0.37	1.10	0.37	7.35	1.10	19.47
Chinese	0.35	0.12	0.12	0.12	0.12	14.09	0.23
food	9.65	0.48	8.68	0.48	0.48	0.48	0.48
lunch	1.11	0.22	0.22	0.22	0.22	0.44	0.22

Remarks:
add-one causes large changes in some frequencies due to relative size of *V* (1616)

want to: 786 ⇒ 338
= (786 + 1) * 1215 / (1215 + 1616)

$$\left(c_i+1\right)\frac{N}{N+V}$$

Problem with add-one smoothing

- bigrams starting with *Chinese* are boosted by a factor of 8 ! (1829 / 213)

unsmoothed bigram counts:

1^{st} *word*

	I	*want*	*to*	*eat*	*Chinese*	*food*	*lunch*	*…*	***Total (N)***
I	8	1087	0	13	0	0	0		**3437**
want	3	0	786	0	6	8	6		**1215**
to	3	0	10	860	3	0	12		**3256**
eat	0	0	2	0	19	2	52		**938**
Chinese	2	0	0	0	0	120	1		**213**
food	19	0	17	0	0	0	0		**1506**
lunch	4	0	0	0	0	1	0		**459**

add-one smoothed bigram counts:

1^{st} *word*

	I	*want*	*to*	*eat*	*Chinese*	*food*	*lunch*	*…*	***Total (N+V)***
I	9	1088	1	14	1	1	1		**5053**
want	4	1	787	1	7	9	7		**2831**
to	4	1	11	861	4	1	13		**4872**
eat	1	1	23	1	20	3	53		**2554**
Chinese	3	1	1	1	1	121	2		**1829**
food	20	1	18	1	1	1	1		**3122**
lunch	5	1	1	1	1	2	1		**2075**

Problem with add-one smoothing (con’t)

- Data from the AP from (Church and Gale, 1991)
  - Corpus of 22,000,000 bigrams
  - Vocabulary of 273,266 words (i.e. 74,674,306,756 possible bigrams)
  - 74,671,100,000 bigrams were unseen
  - And each unseen bigram was given a frequency of 0.000295

Freq. from training data

Freq. from held-out data

Add-one smoothed freq.

f_{MLE}	$f_{empirical}$	$f_{add\text{-}one}$
0	0.000027	**0.000295**
1	0.448	0.000274
2	1.25	0.000411
3	2.24	0.000548
4	3.23	0.000685
5	4.21	0.000822

too high

too low

- Total probability mass given to unseen bigrams =
  (74,671,100,000 x 0.000295) / 22,000,000 ~**99.96** !!!!

Smoothing and N-grams

- **Witten-Bell Smoothing**
  - equate zero frequency items with frequency 1 items
  - use frequency of things seen once to estimate frequency of things we haven’t seen yet
  - *smaller impact than Add-One*
- **Unigram**
  - a zero frequency word (unigram) is “an event that hasn’t happened yet”
  - count the number of words (T) we’ve observed in the corpus (Number of types)
  - p(w) = T/(Z*(N+T))
    - w is a word with zero frequency
    - Z = number of zero frequency words
    - N = size of corpus

Distributing

- The amount to be distributed is $$\frac{T}{N+T}$$
- The number of events with count zero $$Z$$
- So distributing evenly gets us $$\frac{1}{Z}\frac{T}{N+T}$$

Smoothing and N-grams

- **Bigram**
  - $p(w_n|w_{n-1}) = C(w_{n-1}w_n)/C(w_{n-1})$ (*original*)
  - $p(w_n|w_{n-1}) = T(w_{n-1})/(Z(w_{n-1})*(T(w_{n-1})+N))$ for zero bigrams (*after* Witten-Bell)
    - $T(w_{n-1})$ = number of bigrams beginning with w_{n-1}
    - $Z(w_{n-1})$ = number of unseen bigrams beginning with w_{n-1}
    - $Z(w_{n-1})$ = total number of possible bigrams beginning with w_{n-1} minus the ones we've seen
    - $Z(w_{n-1}) = V - T(w_{n-1})$
  - $T(w_{n-1})/ Z(w_{n-1}) * C(w_{n-1})/(C(w_{n-1})+ T(w_{n-1}))$
    - estimated zero bigram frequency
  - $p(w_n|w_{n-1}) = C(w_{n-1}w_n)/(C(w_{n-1})+T(w_{n-1}))$
    - for non-zero bigrams (*after* Witten-Bell)

Smoothing and N-grams

- **Witten-Bell Smoothing**
  - use frequency (count) of things seen once to estimate frequency (count) of things we haven’t seen yet
- **Bigram**
  - $T(w_{n-1})/ Z(w_{n-1}) * C(w_{n-1})/(C(w_{n-1})+ T(w_{n-1}))$ estimated zero bigram frequency (count)
    - $T(w_{n-1})$ = number of bigrams beginning with w_{n-1}
    - $Z(w_{n-1})$ = number of unseen bigrams beginning with w_{n-1}

	I	want	to	eat	Chinese	food	lunch
I	8	1087	0	13	0	0	0
want	3	0	786	0	6	8	6
to	3	0	10	860	3	0	12
eat	0	0	2	0	19	2	52
Chinese	2	0	0	0	0	120	1
food	19	0	17	0	0	0	0
lunch	4	0	0	0	0	1	0

Remark:
smaller changes

	I	want	to	eat	Chinese	food	lunch
I	7.785	1057.763	0.061	12.650	0.061	0.061	0.061
want	2.823	0.046	739.729	0.046	5.647	7.529	5.647
to	2.885	0.084	9.616	826.982	2.885	0.084	11.539
eat	0.073	0.073	1.766	0.073	16.782	1.766	45.928
Chinese	1.828	0.011	0.011	0.011	0.011	109.700	0.914
food	18.019	0.051	16.122	0.051	0.051	0.051	0.051
lunch	3.643	0.026	0.026	0.026	0.026	0.911	0.026

Distributing Among the Zeros

- If a bigram “w_x w_i” has a zero count

Number of bigram types starting with wx

$$P(w_i \mid w_x) = \frac{1}{Z(w_x)} \frac{T(w_x)}{N(w_x)+T(w_x)}$$

Number of bigrams starting with wx that were not seen

Actual frequency (count)of bigrams beginning with wx

Thank you

- السلام عليكم ورحمة الله