CS-AD 220 – Spring 2016

Natural Language Processing

Session 10: 1-Mar-16

Prof. Nizar Habash

Some slides are adapted from Jurafsky and Martin's course slides on Speech and Language Processing

NYUAD CS-AD 220 – Spring 2016
Natural Language Processing

Assignment #2
Finite State Machines
Assigned Feb 18, 2016
Due Mar 10, 2016 (11:59pm)

I. Grading & Submission

This assignment is about the development of finite state machines using the OpenFST and Thrax toolkits. The assignment accounts for 15% of the full grade. It consists of three exercises. The first is a simple "machine translation" system for animal sounds to help with learning the tools. The second is about modeling how numbers are read in English and French. And the third is about Spanish verb conjugation. The answers should be placed in a zipped folder with separate sub-directories for each exercise.

The assignment is due on March 10 before midnight (11:59pm). For late submissions, 10% will be deducted from the homework grade for any portion of each late day. The student should upload the answers in a single zipped to NYU Classes (Assignment #2).

Assignment #2 posted on NYU Classes

Moving Legislative Day Class

- Spring Break is March 18 – 25, 2016
- Sat March 26, 2016 is a Legislative *Thursday*
- Move to

Sat April 2, 2016 at 10am

Same Classroom C2-E049

Final Exam!

Monday May 16th

1pm-4pm

CR-002

This Week

- How is Assignment #2 progressing?
- Language Modeling

$$P(z|Qui) = 1.0$$

ABBA

- Money, money, money
- Must be funny
- In the rich man's world
- Money, money, money
- Always sunny
- In the rich man's world

https://www.youtube.com/watch?v=ETxmCCsMoD0

ABBA

<s> money money money </s>
<s> must be funny </s>
<s> in the rich man's world </s>
<s> money money money </s>
<s> always sunny </s>
<s> in the rich man's world </s>

P(money|money)=?

P(funny|be)=?

P(always|sunny)=?

P(</s>|world)=?

CS-AD 220 – Spring 2016
Quiz #2
Name:________________

<s> money money money </s>
<s> must be funny </s>
<s> in the rich man's world </s>
<s> money money money </s>
<s> always sunny </s>
<s> in the rich man's world </s>

P(money|money)=
P(funny|be)=
P(always|sunny)=
P(</s>|world)=

$$P(w_i \mid w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

6 money
6 <s>
6 </s>
2 world
2 the
2 rich
2 man's
2 in
1 sunny
1 must
1 funny
1 be
1 always

4	money	money
2	world	</s>
2	the	rich
2	rich	man's
2	money	</s>
2	man's	world
2	in	the
2	<s>	money
2	<s>	in
1	sunny	</s>
1	must	be
1	funny	</s>
1	be	funny
1	always	sunny
1	<s>	must
1	<s>	always

P(money|money)=
C(money,money)/C(money)= 4/6 => 0.66

P(funny|be)=
C(be,funny)/C(be)= 1/1 = 1.0

P(always|sunny)=
C(sunny,always)/C(always)= 0/1 = 0.0

P(</s>|world)=
C(world,</s>)/C(world) = 2/2 =1.0

This Week

- How is Assignment #2 progressing?
- Language Modeling

Evaluation:
How good is our (language) model?

- Does our language model prefer good sentences to bad ones?
  - Assign higher probability to “real” or “frequently observed” sentences
    - Than “ungrammatical” or “rarely observed” sentences?
- We train parameters of our model on a **training set**.
- We test the model’s performance on data we haven’t seen.
  - A **test set** is an unseen dataset that is different from our training set, totally unused.
  - An **evaluation metric** tells us how well our model does on the test set.
    - **Extrinsic vs Intrinsic evaluations**

Extrinsic evaluation of N-gram models

- Best evaluation for comparing models A and B
  - Put each model in a task
    - Spelling corrector, speech recognizer, MT system
  - Run the task, get an accuracy for A and for B
    - How many misspelled words corrected properly
    - How many words translated correctly
  - Compare accuracy for A and B

Difficulty of extrinsic (in-vivo) evaluation of N-gram models

- Extrinsic evaluation
  - Time-consuming; can take days or weeks
- So
  - Sometimes use **intrinsic** evaluation: **perplexity**
  - Bad approximation
    - unless the test data looks **just** like the training data
    - So **generally only useful in pilot experiments**
  - But is helpful to think about.

Intuition of Perplexity

- The Shannon Game:
  - How well can we predict the next word?

    I always order pizza with cheese and ____

    The 33rd President of the US was ____

    I saw a ____

  - Unigrams are terrible at this game. (Why?)

mushrooms 0.1

pepperoni 0.1

anchovies 0.01

….

fried rice 0.0001

….

and 1e-100

- A better model of a text
  - is one which assigns a higher probability to the word that actually occurs

Perplexity

The best language model is one that best predicts an unseen test set

- Gives the highest P(sentence)

Perplexity is the inverse probability of the test set, normalized by the number of words:

$$PP(W) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}}$$

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

Chain rule:

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1 \ldots w_{i-1})}}$$

For bigrams:

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-1})}}$$

Minimizing perplexity is the same as maximizing probability

The Shannon Game intuition for perplexity

- From Josh Goodman
  - **“average branching factor”**
- How hard is the task of recognizing digits ‘0,1,2,3,4,5,6,7,8,9’
  - Perplexity 10
- How hard is recognizing (30,000) names at Microsoft.
  - Perplexity = 30,000
- If a system has to recognize
  - Operator (1 in 4)
  - Sales (1 in 4)
  - Technical Support (1 in 4)
  - 30,000 names (1 in 120,000 each)
  - Perplexity is 53
- Perplexity is weighted equivalent branching factor

Perplexity as branching factor

- Let's suppose a sentence consisting of random digits
- What is the perplexity of this sentence according to a model that assign P=1/10 to each digit?

$$
\begin{aligned}
\mathrm{PP}(W) &= P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} \\
&= \left(\frac{1}{10}^{N}\right)^{-\frac{1}{N}} \\
&= \frac{1}{10}^{-1} \\
&= 10
\end{aligned}
$$

Lower perplexity = better model

- Training 38 million words, test 1.5 million words, WSJ

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

The Shannon Visualization Method

- Choose a random bigram
  (<s>, w) according to its probability
- Now choose a random bigram
  (w, x) according to its probability
- And so on until we choose </s>
- Then string the words together

```
<s> I
    I want
      want to
           to eat
              eat Chinese
                  Chinese food
                          food  </s>
I want to eat Chinese food
```

Approximating Shakespeare

Unigram

To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have

Every enter now severally so, let

Hill he late speaks; or! a more to leg less first you enter

Are where exeunt and sighs have rise excellency took of.. Sleep knave we. near; vile like

Bigram

What means, sir. I confess she? then all sorts, he is trim, captain.

Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow.

What we, hath got so she that I rest and sent to scold and nature bankrupt, nor the first gentleman?

Trigram

Sweet prince, Falstaff shall die. Harry of Monmouth's grave.

This shall forbid it should be branded, if renown made it empty.

Indeed the duke; and had a very good friend.

Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done.

Quadrigram

King Henry.What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in;

Will you not tell me who I am?

It cannot be but so.

Indeed the short and the long. Marry, 'tis a noble Lepidus.

Shakespeare as corpus

- N=884,647 tokens, V=29,066
- Shakespeare produced 300,000 bigram types out of V^2= 844 million possible bigrams.
  - So 99.96% of the possible bigrams were never seen (have zero entries in the table)
- Quadrigrams worse: What's coming out looks like Shakespeare because it ***is*** Shakespeare

The Wall Street journal is not Shakespeare

Unigram

Months the my and issue of year foreign new exchange's september were recession exchange new endorsed a acquire to six executives

Bigram

Last December through the way to preserve the Hudson corporation N. B. E. C. Taylor would seem to complete the major central planners one point five percent of U. S. E. has already old M. X. corporation of living on information such as more frequently fishing to keep her

Trigram

They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions

The Perils of Overfitting

- N-grams only work well for word prediction if the test corpus looks like the training corpus
  - In real life, it often doesn't
  - We need to train robust models that generalize!
  - One kind of generalization: Zeros!
    - Things that don't ever occur in the training set
      - But occur in the test set
    - Out-of-Vocabulary (OOV)

Zeros

- Training set:
  … denied the allegations
  … denied the reports
  … denied the claims
  … denied the request

  P("offer" | denied the) = 0

- Test set
  … denied the offer
  … denied the loan

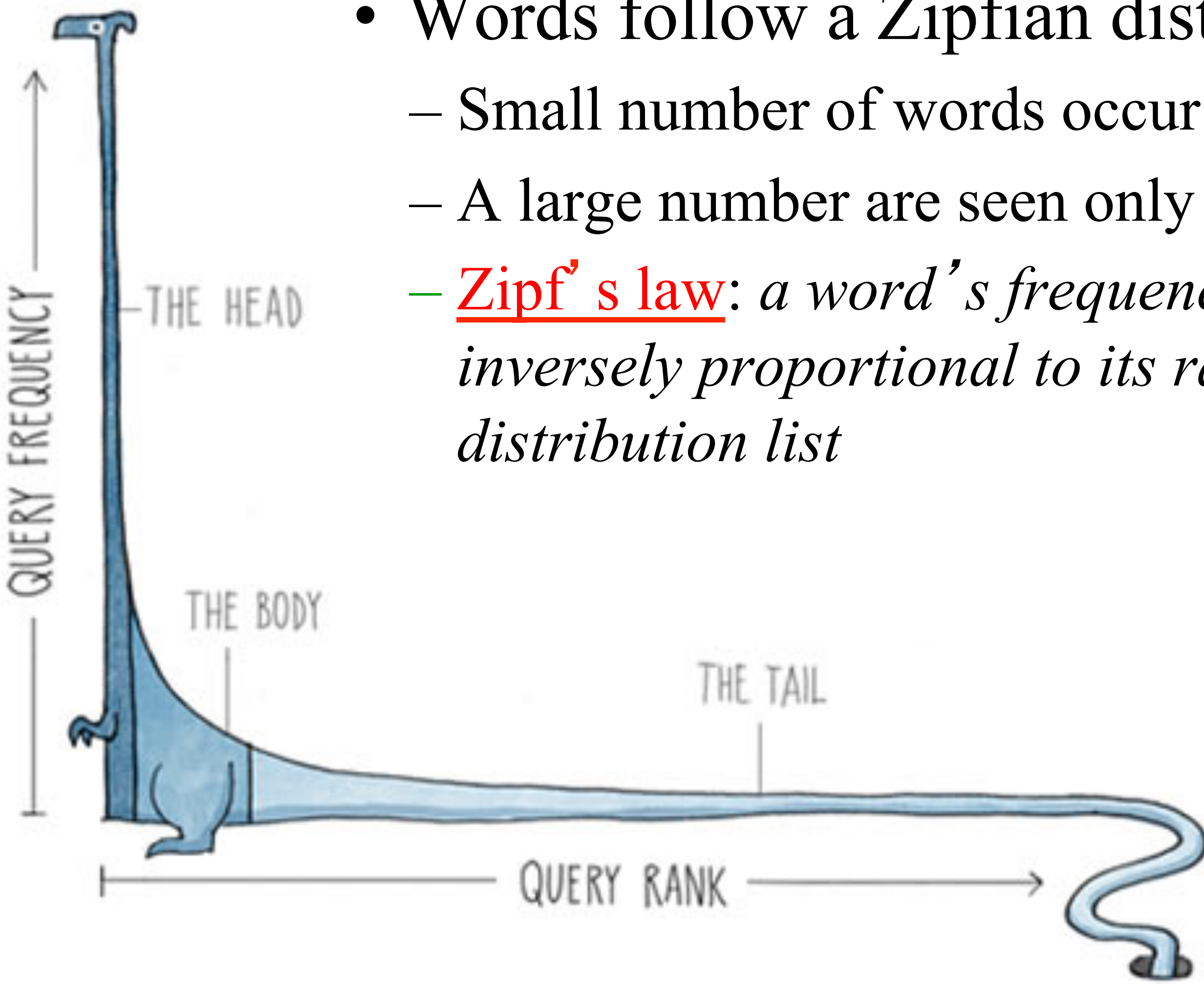
Zero probability bigrams

- Bigrams with zero probability
  - mean that we will assign 0 probability to the test set!
- And hence we cannot compute perplexity (can’t divide by 0)!

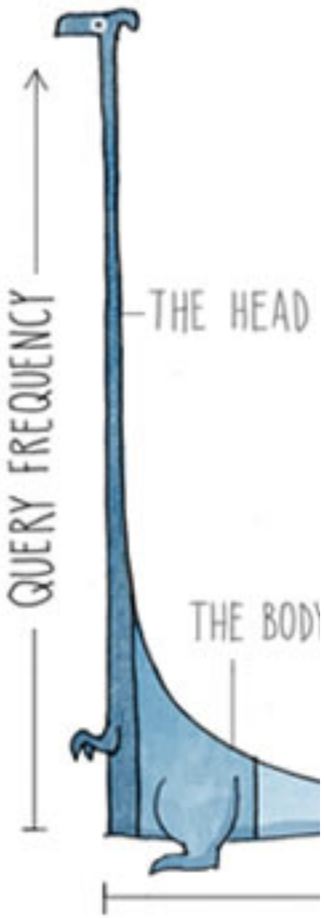
- What to do ?

Smoothing

- Words follow a Zipfian distribution
  - Small number of words occur very frequently
  - A large number are seen only once
  - Zipf's law: *a word's frequency is approximately inversely proportional to its rank in the word distribution list*



Smoothing

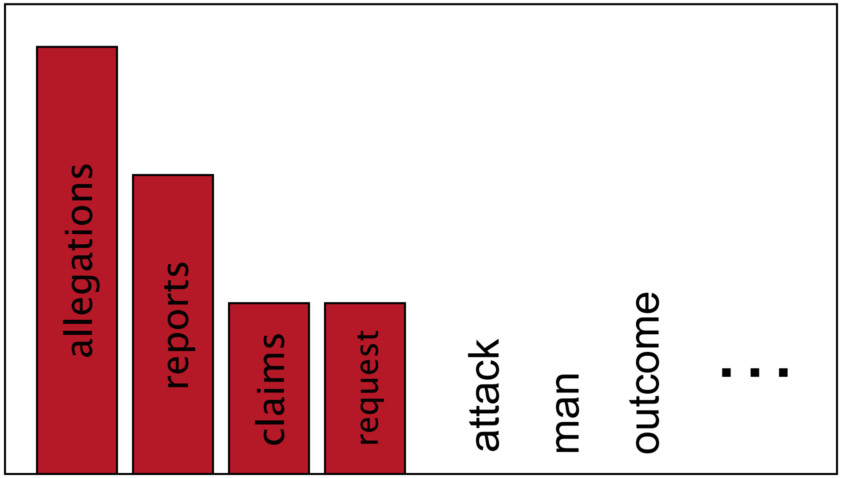


- Words follow a Zipfian distribution
  - Small number of words occur very frequently
  - A large number are seen only once
  - Zipf's law: *a word's frequency is approximately inversely proportional to its rank in the word distribution list*
- Zero probabilities on one bigram cause a zero probability on the entire sentence
- So….how do we estimate the likelihood of unseen n-grams?

The intuition of smoothing (from Dan Klein)

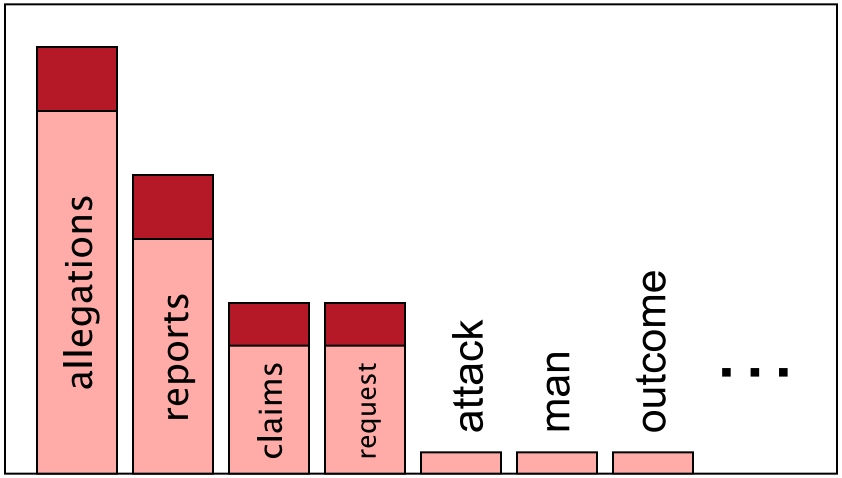
- When we have sparse statistics:

P(w | denied the)
3 allegations
2 reports
1 claims
1 request
7 total



- Steal probability mass to generalize better

P(w | denied the)
2.5 allegations
1.5 reports
0.5 claims
0.5 request
2 other
7 total



Add-one estimation

- Also called Laplace smoothing
- Pretend we saw each word one more time than we did
- Just add one to all the counts!

- MLE estimate:

$$P_{MLE}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

- Add-1 estimate:

$$P_{Add-1}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i) + 1}{c(w_{i-1}) + V}$$

Maximum Likelihood Estimates

- The maximum likelihood estimate
  - of some parameter of a model M from a training set T
  - maximizes the likelihood of the training set T given the model M
- Suppose the word "bagel" occurs 400 times in a corpus of a million words
- What is the probability that a random word from some other text will be "bagel"?
- MLE estimate is 400/1,000,000 = .0004
- This may be a bad estimate for some other corpus
  - But it is the **estimate** that makes it **most likely** that "bagel" will occur 400 times in a million word corpus.

Berkeley Restaurant Corpus: Laplace smoothed bigram counts

	i	want	to	eat	chinese	food	lunch	spend
i	6	828	1	10	1	1	1	3
want	3	1	609	2	7	7	6	2
to	3	1	5	687	3	1	7	212
eat	1	1	3	1	17	3	43	1
chinese	2	1	1	1	1	83	2	1
food	16	1	16	1	2	5	1	1
lunch	3	1	1	1	1	2	1	1
spend	2	1	2	1	1	1	1	1

Laplace-smoothed bigrams

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

	i	want	to	eat	chinese	food	lunch	spend
i	0.0015	0.21	0.00025	0.0025	0.00025	0.00025	0.00025	0.00075
want	0.0013	0.00042	0.26	0.00084	0.0029	0.0029	0.0025	0.00084
to	0.00078	0.00026	0.0013	0.18	0.00078	0.00026	0.0018	0.055
eat	0.00046	0.00046	0.0014	0.00046	0.0078	0.0014	0.02	0.00046
chinese	0.0012	0.00062	0.00062	0.00062	0.00062	0.052	0.0012	0.00062
food	0.0063	0.00039	0.0063	0.00039	0.00079	0.002	0.00039	0.00039
lunch	0.0017	0.00056	0.00056	0.00056	0.00056	0.0011	0.00056	0.00056
spend	0.0012	0.00058	0.0012	0.00058	0.00058	0.00058	0.00058	0.00058

Reconstituted counts

$$c^*(w_{n-1}w_n) = \frac{[C(w_{n-1}w_n)+1] \times C(w_{n-1})}{C(w_{n-1})+V}$$

	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

Compare with raw bigram counts

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

Add-1 estimation is a blunt instrument

- So add-1 isn't used for N-grams:
  - We'll see better methods
- But add-1 is used to smooth other NLP models
  - For text classification
  - In domains where the number of zeros isn't so huge.

Practical Issues

- We do everything in log space
  - Avoid underflow
  - (also adding is faster than multiplying)

$$p_1 \times p_2 \times p_3 \times p_4 = \exp(\log p_1 + \log p_2 + \log p_3 + \log p_4)$$

ARPA format

$$\begin{array}{llll} \text{unigram:} & \log p^*(w_i) & w_i & \log \alpha(w_i) \\ \text{bigram:} & \log p^*(w_i|w_{i-1}) & w_{i-1}w_i & \log \alpha(w_{i-1}w_i) \\ \text{trigram:} & \log p^*(w_i|w_{i-2},w_{i-1}) & w_{i-2}w_{i-1}w_i & \end{array}$$

- Each n-gram definition starts with a probability value stored as $\log_{10}$ followed by a sequence of words describing the actual n-gram. In all sections excepts the last one, this is followed by a back-off weight which is also stored as $\log_{10}$.

- http://www1.icsi.berkeley.edu/Speech/docs/HTKBook3.2/node213_mn.html

ARPA format

```
\data\
ngram 1=1447
ngram 2=9420
ngram 3=5201

\1-grams:
-0.8679678   </s>
-99          <s>                          -1.068532
-4.743076    chow-fun                     -0.1943932
-4.266155    fries                        -0.5432462
-3.175167    thursday                     -0.7510199
-1.776296    want                         -1.04292
...

\2-grams:
-0.6077676   <s>    i                     -0.6257131
-0.4861297   i      want                  0.0425899
-2.832415    to     drink                 -0.06423882
-0.5469525   to     eat                   -0.008193135
-0.09403705  today </s>
...

\3-grams:
-2.579416    <s>    i         prefer
-1.148009    <s>    about     fifteen
-0.4120701   to     go        to
-0.3735807   me     a         list
-0.260361    at     jupiter   </s>
-0.260361    a      malaysian restaurant
...
\end\
```

Language Modeling Toolkits

- SRILM
- CMU-Cambridge LM Toolkit

Google N-Gram Release

All Our N-gram are Belong to You

By Peter Norvig - 8/03/2006 11:26:00 AM

Posted by Alex Franz and Thorsten Brants, Google Machine Translation Team

Here at Google Research we have been using word n-gram models for a variety of R&D projects, such as statistical machine translation, speech recognition, spelling correction, entity detection, information extraction, and others. While such models have usually been estimated from training

to share this enormous dataset with everyone. We processed 1,024,908,267,229 words of running text and are publishing the counts for all 1,176,470,663 five-word sequences that appear at least 40 times. There are 13,588,391 unique words, after discarding words that appear less than 200 times.

Google N-Gram Release

- serve as the incoming 92
- serve as the incubator 99
- serve as the independent 794
- serve as the index 223
- serve as the indication 72
- serve as the indicator 120
- serve as the indicators 45
- serve as the indispensable 111
- serve as the indispensible 40
- serve as the individual 234

Next Time

- We finish off Language Modeling
- Read J+M Chap 5 (intro up to 5.5)
- Assignment #2 due March 10 midnight