NLP 201: Natural Language Processing 1
Language Modeling II

Jeffrey Flanigan

October 26, 2021

University of California Santa Cruz
jmflanig@ucsc.edu

Plan for Today

- Evaluation of lanuage models
- Bayesian learning (MAP estimation)
- Zipf's law
- Interpolation and backoff methods
- Language models from probabilistic classifiers

What Is A Language Model?

- Probability distributions over sentences (i.e., word sequences)

  P(W) = P($w_1 w_2 w_3 w_4 \ldots w_k$)

- Can use them to generate strings

  P(w_k | $w_2 w_3 w_4 \ldots w_{k-1}$)

- Rank possible sentences
  - P("*Today is Tuesday*") > P("*Tuesday Today is*")
  - P("*Today is Tuesday*") > P("*Today is Atlanta*")

N-grams Models

- Unigram model: $P(w_1)P(w_2)P(w_3) \dots P(w_n)$
- Bigram model: $P(w_1)P(w_2|w_1)P(w_3|w_2) \dots P(w_n|w_{n-1})$
- Trigram model:

$P(w_1)P(w_2|w_1)P(w_3|w_2, w_1) \dots P(w_n|w_{n-1}w_{n-2})$

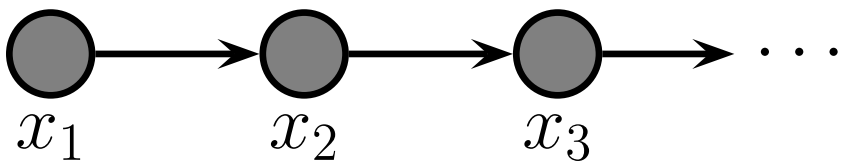
- N-gram model:

$P(w_1)P(w_2|w_1) \dots \ P(w_n|w_{n-1}w_{n-2} \dots w_{n-N})$

Example: Markov chains

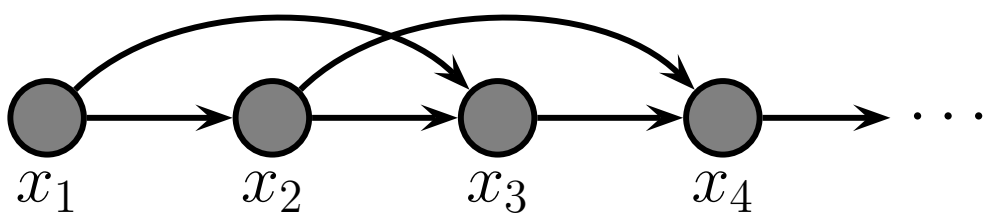
- First order Markov chain

$$p(x_{1:T}) = p(x_1) \prod_{t=2}^{T} p(x_t|x_{t-1})$$



- Second order Markov chain

$$p(x_{1:T}) = p(x_1, x_2) \prod_{t=3}^{T} p(x_t|x_{t-1}, x_{t-2})$$



Details: Variable Length

- Define always $x_n = STOP$, where STOP is a special symbol
- Then use a Markov process as before:

$$p(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n) = \prod_{i=1}^{n} p(X_i = x_i | X_{i-2} = x_{i-2}, X_{i-1} = x_{i-1})$$

- We now have probability distribution over all sequences
  - Intuition: at every step you have probability α_n to stop and $1 - \alpha_n$ to keep going

Maximum Likelihood Estimation

- For N-gram language models
  - $p(w_i|w_{i-1}, \dots, w_{i-n+1}) = \frac{c(w_i, w_{i-1}, \dots, w_{i-n+1})}{c(w_{i-1}, \dots, w_{i-n+1})}$

Practical Issues

- We do everything in the log space
  - Avoid underflow
  - Adding is faster than multiplying

$$\log(p_1 \times p_2) = \log(p_1) + \log(p_2)$$

How To Evaluate

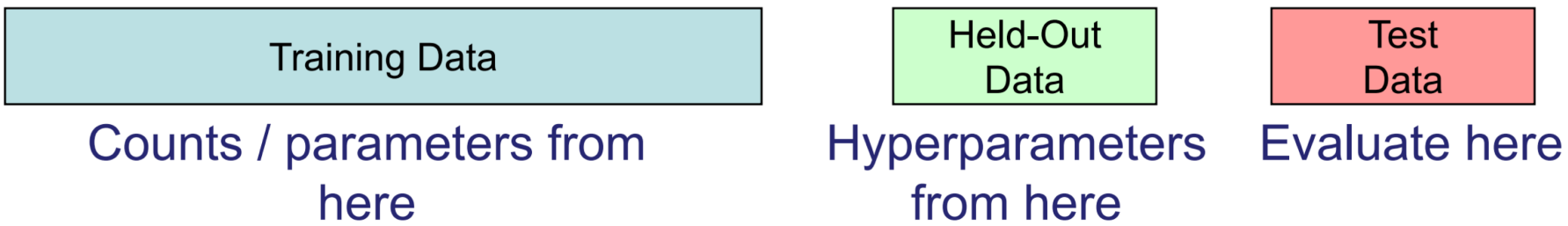
- Extrinsic: build a new language model, use it for some task (MT, ASR, etc.)
- Intrinsic: measure how good we are at modeling language

Intrinsic Evaluation

- Intuitively, language models should assign high probability to real language they have not seen before
  - Want to maximize likelihood on test, not training data
  - Models derived from counts / sufficient statistics require generalization parameters to be tuned on held-out data to stimulate test generalization
  - Set hyperparameters to maximize the likelihood of the held-out data (usually with grid search or EM)

Intrinsic Evaluation

- Intuitively, language models should assign high probability to real language they have not seen before



Evaluation: Perplexity

Intuitively, language models should assign high probability to real language they have not seen before.

For test data $\mathcal{D} = \{x_1, \ldots x_N\}$:

- Probability of $\mathcal{D}$ is $\prod_{i=1}^{N} p(x_i)$

Evaluation: Perplexity

Intuitively, language models should assign high probability to real language they have not seen before.

For test data $\mathcal{D} = \{x_1, \ldots x_N\}$:

- Probability of $\mathcal{D}$ is $\prod_{i=1}^{N} p(x_i)$
- Log-probability of $\mathcal{D}$ is $\sum_{i=1}^{m} \log_2 p(x_i)$

Evaluation: Perplexity

Intuitively, language models should assign high probability to real language they have not seen before.

For test data $\mathcal{D} = \{x_1, \ldots x_N\}$:

- Probability of $\mathcal{D}$ is $\prod_{i=1}^{N} p(x_i)$
- Log-probability of $\mathcal{D}$ is $\sum_{i=1}^{m} \log_2 p(x_i)$
- Average log-probability per word of $\mathcal{D}$ is

$$l = \frac{1}{M} \sum_{i=1}^{N} \log_2 p(\bar{\boldsymbol{x}}_i)$$

if $M = \sum_{i=1}^{N} |x_i|$ (total number of words in the corpus)

Evaluation: Perplexity

Intuitively, language models should assign high probability to real language they have not seen before.

For test data $\mathcal{D} = \{x_1, \ldots x_N\}$:

- Probability of $\mathcal{D}$ is $\prod_{i=1}^{N} p(x_i)$
- Log-probability of $\mathcal{D}$ is $\sum_{i=1}^{m} \log_2 p(x_i)$
- Average log-probability per word of $\mathcal{D}$ is

$$l = \frac{1}{M} \sum_{i=1}^{N} \log_2 p(\bar{\boldsymbol{x}}_i)$$

  if $M = \sum_{i=1}^{N} |x_i|$ (total number of words in the corpus)
- Perplexity on dataset $\mathcal{D}$ is 2^{-l}

Evaluation: Perplexity

Intuitively, language models should assign high probability to real language they have not seen before.

For test data $\mathcal{D} = \{x_1, \ldots x_N\}$:

- Probability of $\mathcal{D}$ is $\prod_{i=1}^{N} p(x_i)$
- Log-probability of $\mathcal{D}$ is $\sum_{i=1}^{m} \log_2 p(x_i)$
- Average log-probability per word of $\mathcal{D}$ is

$$l = \frac{1}{M} \sum_{i=1}^{N} \log_2 p(\bar{\boldsymbol{x}}_i)$$

  if $M = \sum_{i=1}^{N} |x_i|$ (total number of words in the corpus)
- Perplexity on dataset $\mathcal{D}$ is 2^{-l}

Understanding Perplexity

$$2^{-\frac{1}{M}\sum_{i=1}^{m}\log_2 p(x_i)}$$

It's a branching factor!

- Assign probability of 1 to the test data $\Rightarrow$ perplexity $= 1$
- Assign probability of $\frac{1}{|V|}$ to every word $\Rightarrow$ perplexity $= |V|$
- Assign probability of 0 to *anything* $\Rightarrow$ perplexity $= \infty$
    - This motivates a stricter constraint than we had before:
        - For any $x \in V^*$, $p(x) > 0$

How about Unseen Words/Phrases

- Example: Shakespeare corpus consists of N=884,647 word tokens and a vocabulary of V=29,066 word types
- Only 30,000 word types occurred
  - Words not in the training data ⇒ **0** probability
- Only 0.04% of all possible bigrams occurred

Smoothing

- When estimating a language model, we're relying on the data we've observed in a training corpus.
- Training data is a small (and biased) sample of the creativity of language.

$$\prod_{i}^{n} P(w_i \mid w_{i-1}) \times P(\text{STOP} \mid w_n)$$

- As in Naive Bayes, $P(w_i) = 0$ causes P(w) = 0. (Perplexity?)

Smoothing in NB

- One solution: add a little probability mass to every element.

maximum likelihood estimate

$$P(x_i \mid y) = \frac{n_{i,y}}{n_y}$$

$n_{i,y}$ = count of word i in class y
n_y = number of words in y
V = size of vocabulary

smoothed estimates

$$P(x_i \mid y) = \frac{n_{i,y} + \alpha}{n_y + V\alpha}$$

same α for all x_i

$$P(x_i \mid y) = \frac{n_{i,y} + \alpha_i}{n_y + \sum_{j=1}^{V} \alpha_j}$$

possibly different α for each x_i

Additive smoothing

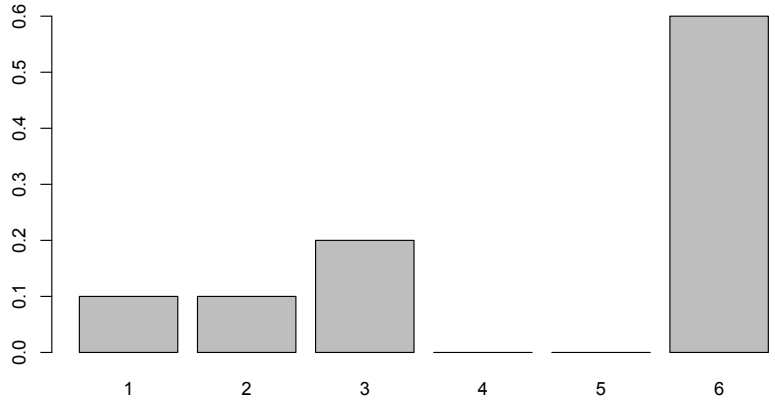
Laplace smoothing:
α = 1

$$P(w_i) = \frac{c(w_i) + \alpha}{N + V\alpha}$$

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

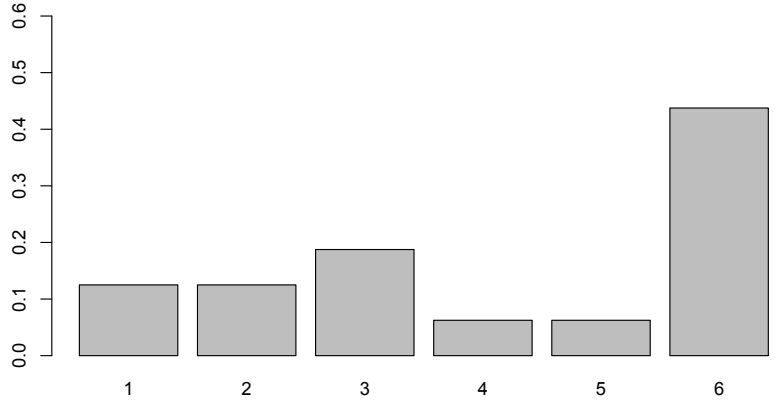
Smoothing

MLE



Smoothing is the re-allocation of probability mass

smoothing with α =1



Types vs tokens

Tokens are the tokens in the document or sentence.
Types are the unique tokens.

Example sentence:

the cat sat on the mat .

Tokens: (the, cat, sat, on, the, mat, .)
Types: { the, cat, sat, on, mat, . }

Sparsity

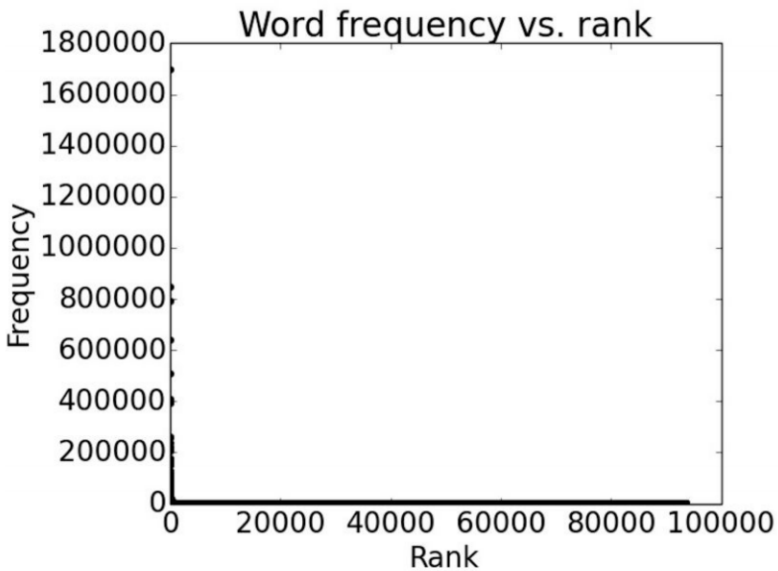
- Sparse data due to Zipf's Law
- Example: the frequency of different words in a large text corpus

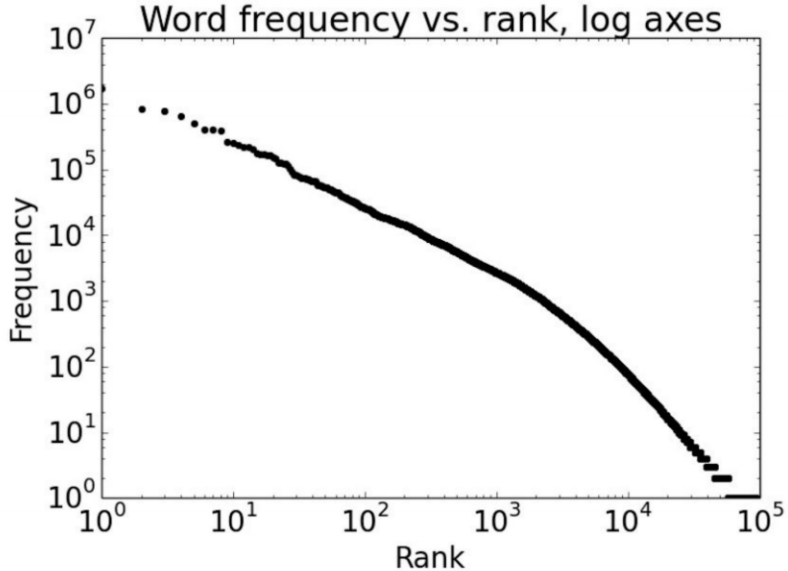
any word	
Frequency	Token
1,698,599	the
849,256	of
793,731	to
640,257	and
508,560	in
407,638	that
400,467	is
394,778	a
263,040	I

nouns	
Frequency	Token
124,598	European
104,325	Mr
92,195	Commission
66,781	President
62,867	Parliament
57,804	Union
53,683	report
53,547	Council
45,842	States

Sparsity

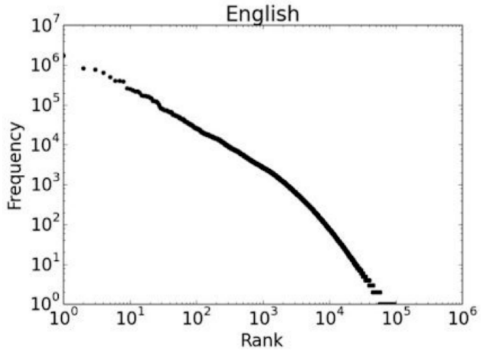
- Order words by frequency. What is the frequency of nth ranked word?

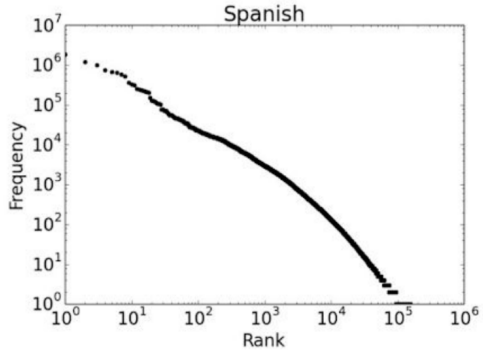


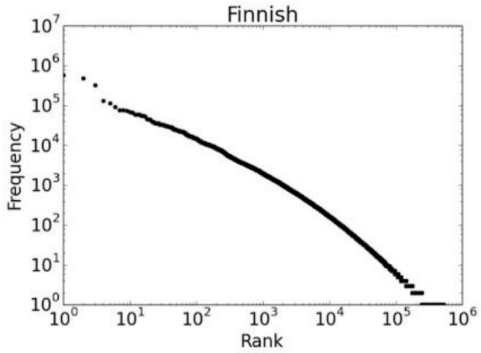


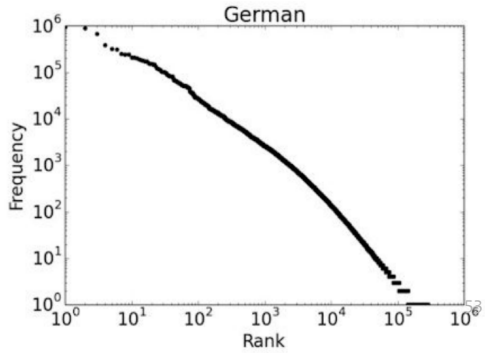
Sparsity

- Regardless of how large our corpus is, there will be a lot of infrequent words
- This means we need to find clever ways to estimate probabilities for things we have rarely or never seen









Other ways to improve sparsity of n-gram models

- Interpolation
- Backoff methods
  - Large number of methods: Witten-bell, Katz, absolute discounting, etc. See the book Foundations of Statistical Natural Language Processing for an overview.
  - Modified Kneser-Ney is the state of the art
  - "Stupid backoff" (not my choice of terms!) is faster, and almost as good

Interpolation

- As ngram order rises, we have the potential for higher precision but also higher variability in our estimates.
- A linear interpolation of any two language models p and q (with $\lambda \in [0,1]$) is also a valid language model.

$$\lambda p + (1 - \lambda) q$$

p = the web

q = political speeches

Interpolation

- We can use this fact to make higher-order language models more robust.

$$P(w_i \mid w_{i-2}, w_{i-1}) = \lambda_1 P(w_i \mid w_{i-2}, w_{i-1}) + \lambda_2 P(w_i \mid w_{i-1}) + \lambda_3 P(w_i)$$

$$\lambda_1 + \lambda_2 + \lambda_3 = 1$$

Interpolation

- How do we pick the best values of λ?
  - Grid search over development corpus
  - Expectation-Maximization algorithm (treat as missing parameters to be estimated to maximize the probability of the data we see).

“Stupid backoff”

if full sequence observed

$$S(w_i \mid w_{i-k+1}, \ldots, w_{i-1}) = \frac{c(w_{i-k+1}, \ldots, w_i)}{c(w_{i-k+1}, \ldots, w_{i-1})}$$

No discounting here, just back off to lower order ngram if the higher order is not observed.

Cheap to calculate; works almost as well as KN when there is a lot of data

otherwise

$$= \lambda S(w_i \mid w_{i-k+2}, \ldots, w_{i-1})$$

Brants et al. (2007), “Large Language Models in Machine Translation”

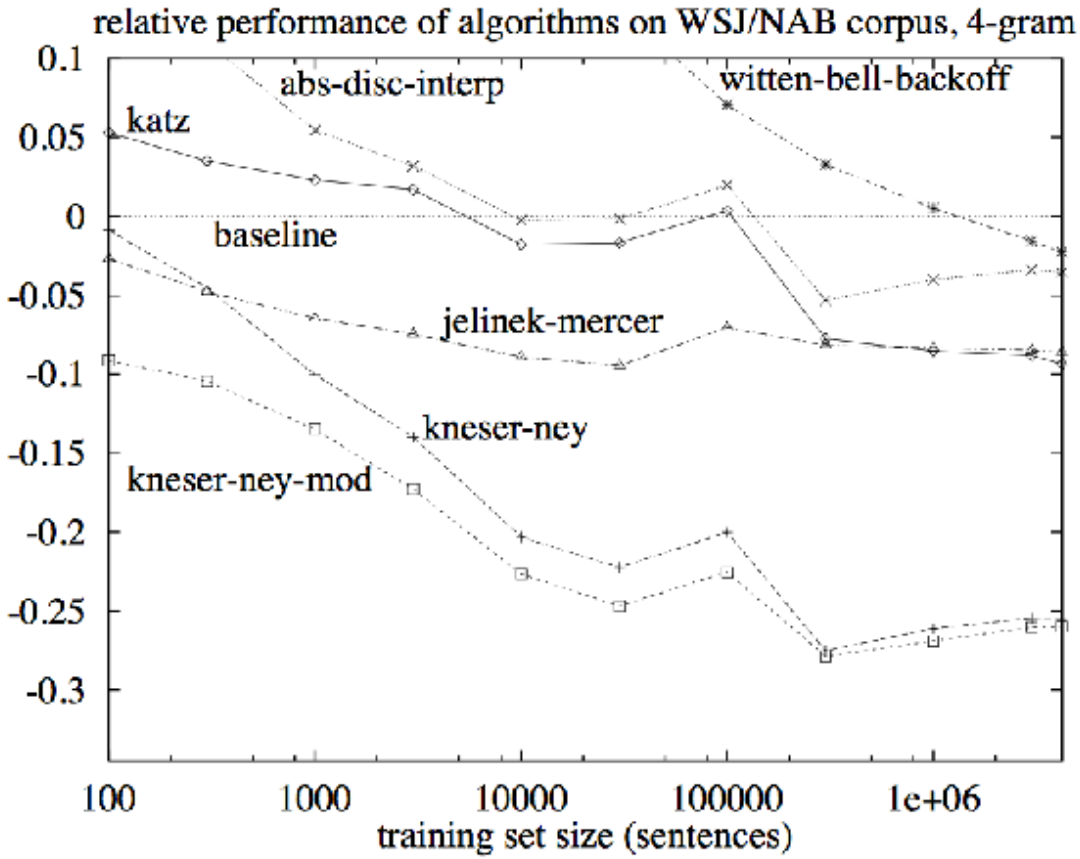
Kneser-Ney smoothing

- Intuition: When backing off to a lower-order ngram, maybe the overall ngram frequency is not our best guess.

  I can't see without my reading _____

  P("Francisco") > P("glasses")

- *Francisco* is more frequent, but shows up in fewer unique bigrams ("San Francisco") — so we shouldn't expect it in new contexts; *glasses*, however, does show up in many different bigrams



Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Center for Research in Computing Technology, Harvard University, 1998.

Bayesian Nonparametrics and Kneser-Ney Smoothing

- There is a Bayesian interpretation of K.N. smoothing using Pitman-Yor processes (from Bayesian nonparametrics) `https://www.stats.ox.ac.uk/~teh/research/compling/hpylm.pdf`
- Bayesian nonparametrics were very popular before the deep learning revolution in NLP
- They give more control over the form of the learned distributions (can bias the model towards power-law distributions)

Beyond N-Gram Language Models

- We can use *any* multiclass classifier over vocabulary V that returns probabilities to build a language model (Naive Bayes, logistic regression, neural networks, etc)

$$p(w_1, \ldots w_m) = \prod_{i=1}^{m} p(w_i | w_1, \ldots, w_{i-1})$$

  Words w_i are in the set V

- Training examples are words w_i given the context $w_1 \ldots w_{i-1}$

(Our executive director Adwait Ratnaparkhi's PhD thesis was on this)

The Cutting Edge for LMs

- Deep learning models for sequence-to-sequence tasks (like translation, summarization) are often **conditional language models**, which predict output sequence $y_1 \dots y_m$ given an input sequence $x_1 \dots x_n$ by maximizing

$$p(y_1 \dots y_m | x_1 \dots x_n)$$

- Language modeling is one of the three NLP supertasks.
- Recent work using LMs for zero-shot and few-shot learning: GPT-3 paper
- LMs can be used to learn contextualized word representations

More on all of this later in the course

More Resources

- Google n-gram
- https://research.googleblog.com/2006/08/all-our-n-gram-are-belong-to-you.html

```
File sizes: approx. 24 GB compressed (gzip'ed) text files

Number of tokens: 1,024,908,267,229
Number of sentences: 95,119,665,584
Number of unigrams: 13,588,391
Number of bigrams: 314,843,401
Number of trigrams: 977,069,902
Number of fourgrams: 1,313,818,354
Number of fivegrams: 1,176,470,663
```

More Resources

- Google n-gram viewer

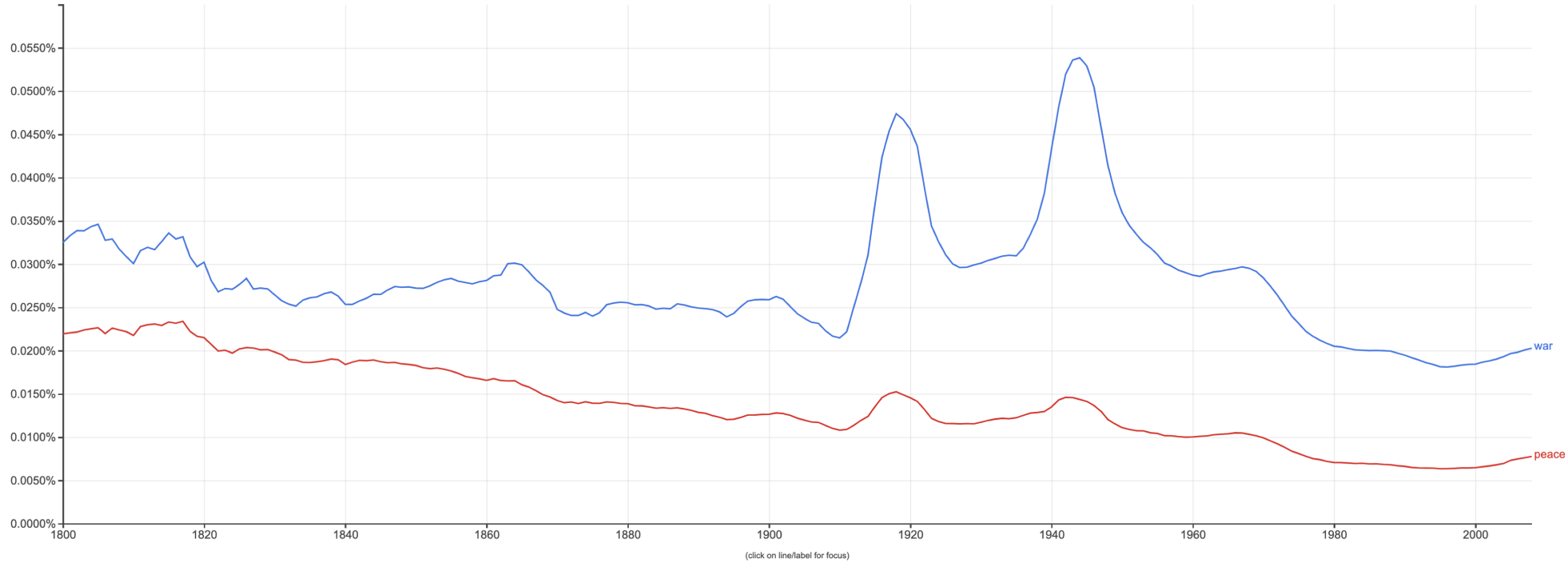
https://books.google.com/ngrams/

Data: http://storage.googleapis.com/books/ngrams/books/datasetsv2.html

Google Books Ngram Viewer

Graph these comma-separated phrases: war,peace case-insensitive

between 1800 and 2008 from the corpus English with smoothing of 3. Search lots of books



(click on line/label for focus)

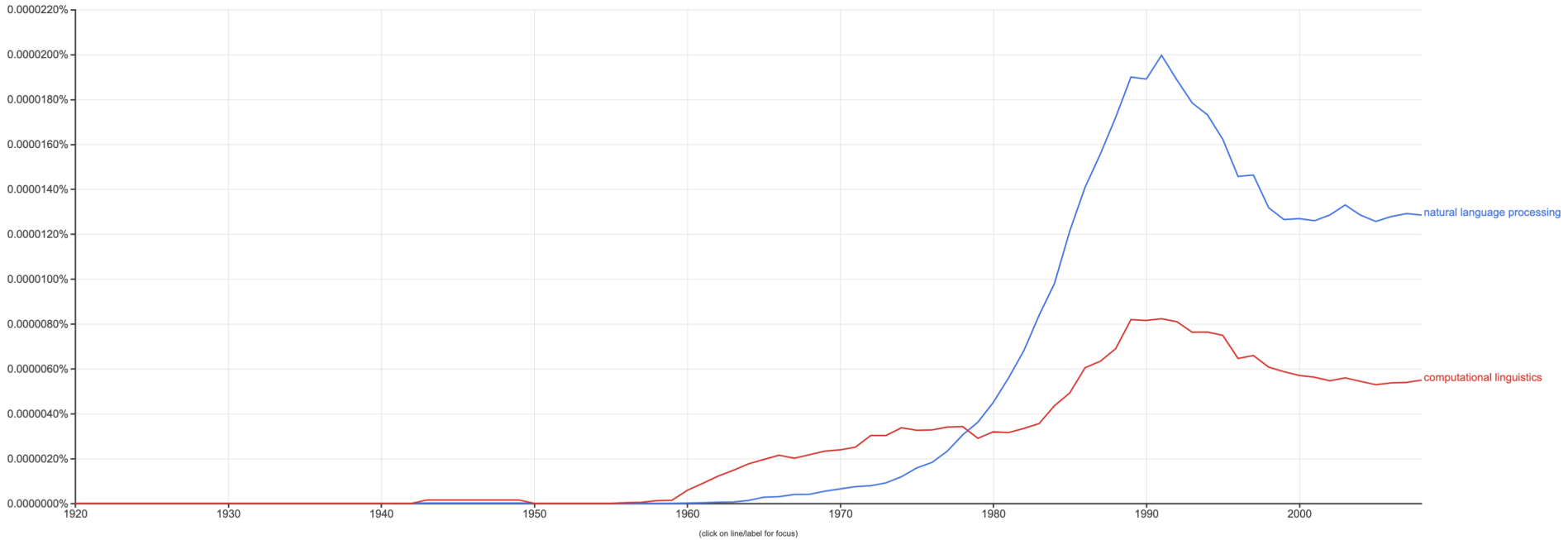
Search in Google Books:

1800 - 1817	1818 - 1942	1943 - 1956	1957 - 1978	1979 - 2008	war	English
1800 - 1812	1813 - 1825	1826 - 1875	1876 - 1969	1970 - 2008	peace	English

Google Books Ngram Viewer

Graph these comma-separated phrases: natural language processing,computational linguistics case-insensitive

between 1920 and 2008 from the corpus English with smoothing of 3. Search lots of books



More Examples

- http://nbviewer.jupyter.org/gist/yoavg/d76121dfde2618422139
- 10-gram character-level LM

```
First Citizen: Nay, then, that was hers, It speaks against
your other service: But since the youth of the circumstance
be spoken: Your uncle and one Baptista's daughter.

SEBASTIAN: Do I stand till the break off.
```