BBM 495
Language Models

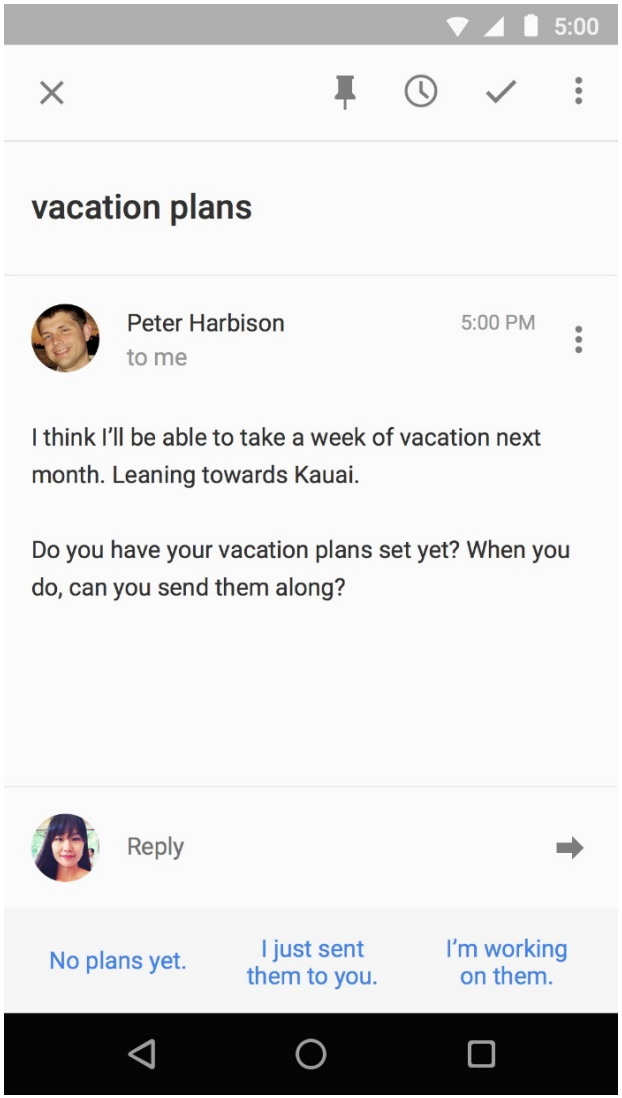
LECTURER: BURCU CAN

2019-2020 SPRING


DARKOW 2006
COLUMBIA DAILY TRIBUNE
5/23
ALL THOSE IN FAVOR OF MAKING ENGLISH THE "OFFICIAL" LANGUAGE OF THE UNITED STATES, JUST SAY YES!
SI!
BALEH!
SI!
NA'AM!
YAO!
YA!
OUI!
NDIYO!
JO!
CAGLECARTOONS.com

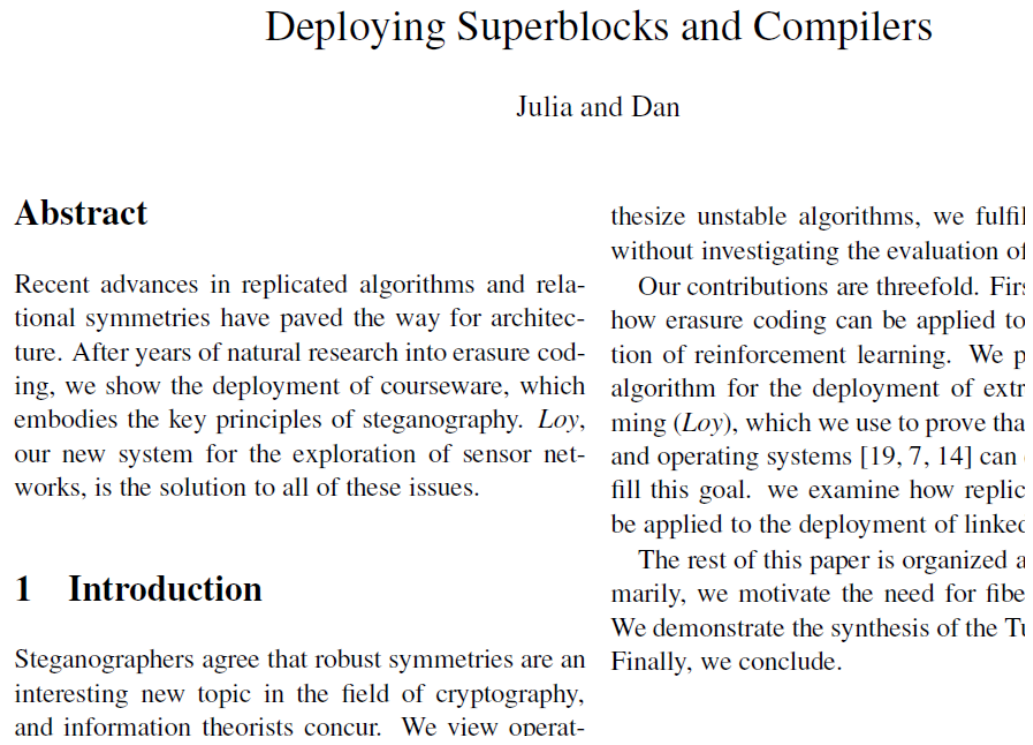
Language Model Applications

Smart Reply



Language Model Applications

- Language Generation
- https://pdos.csail.mit.edu/archive/scigen/



Deploying Superblocks and Compilers

Julia and Dan

Abstract

Recent advances in replicated algorithms and relational symmetries have paved the way for architecture. After years of natural research into erasure coding, we show the deployment of courseware, which embodies the key principles of steganography. *Loy*, our new system for the exploration of sensor networks, is the solution to all of these issues.

1 Introduction

Steganographers agree that robust symmetries are an interesting new topic in the field of cryptography, and information theorists concur. We view operat-

thesize unstable algorithms, we fulfil
without investigating the evaluation of

Our contributions are threefold. Firs
how erasure coding can be applied to
tion of reinforcement learning. We p
algorithm for the deployment of extr
ming (*Loy*), which we use to prove tha
and operating systems [19, 7, 14] can
fill this goal. we examine how replic
be applied to the deployment of linkec

The rest of this paper is organized a
marily, we motivate the need for fibe
We demonstrate the synthesis of the Tu
Finally, we conclude.

Language models

- Machine Translation:
  - P(high winds tonite) > P(large winds tonite)
- Spell Correction:
  - The office is about fifteen minuets from my house
    - P(about fifteen minutes from) > P(about fiftee n minuets from)
- Speech Recognition:
  - P(I saw a van) >> P(eyes awe of an)

- Summarization, question-answering, etc.,

Completion Prediction

- A language model also supports predicting the completion of a sentence.
  - Please turn off your cell _____
  - Your program does not _____
- *Predictive text input* systems can guess what you are typing and give choices on how to complete it.

Autocomplete

Language and Probability

1. Colorless green ideas sleep furiously.
2. Furiously sleep ideas green colorless.

How to compute probabilities?

- We don't have the probabilities for most NLP problems
- We can try to *estimate* them from data
  - **(that's the learning part) – i.e. machine learning**

Slides adapted from Mary Ellen Califf

Moving towards language

- What's the probability of drawing a 2 from a deck of 52 cards?

$$P(drawing\ \ a\ \ two) = \frac{4}{52} = \frac{1}{13} = .077$$

- What's the probability of a random word (from a random dictionary page) being a verb?

$$P(drawing\ a\ verb) = \frac{\#\ of\ \ ways\ to\ get\ a\ verb}{all\ words}$$

Remember: Joint probability, conditional probability, independence, Bayes rule etc.

Slide adapted from Dan Jurafsky's

Probabilities on a text

(Conditioning on the previous word)

- ALICE was beginning to get very tired of sitting by her sister on the bank and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, "and what is the use of a book," thought Alice, "without pictures or conversations?'

- $P(w_i+1 = \text{of} \mid w_i = \text{tired}) = 1$
- $P(w_i+1 = \text{of} \mid w_i = \text{use}) = 1$
- $P(w_i+1 = \text{sister} \mid w_i = \text{her}) = 1$
- $P(w_i+1 = \text{beginning} \mid w_i = \text{was}) = ½$
- $P(w_i+1 = \text{reading} \mid w_i = \text{was}) = ½$
- $P(w_i+1 = \text{bank} \mid w_i = \text{the}) = 1/3$
- $P(w_i+1 = \text{book} \mid w_i = \text{the}) = 1/3$
- $P(w_i+1 = \text{use} \mid w_i = \text{the}) = 1/3$

Probabilities on a text

(Conditioning on the previous word)

- English
- ALICE was beginning to get very tired of sitting by her sister on the bank and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, "and what is the use of a book," thought Alice, "without pictures or conversations?'

- Word salad
- beginning by, very ALICE but was and? reading no tired of to into sitting sister the, bank, and thought of without her nothing: having conversations Alice once do or on she it get the book her had peeped was conversation it pictures or sister in, 'what is the use had twice of a book"pictures or' to

P(English)⩾P(word salad)

$P(w_i+1= \text{of} \mid w_i=\text{tired}) = 1$

$P(w_i+1= \text{of} \mid w_i=\text{use}) = 1$

$P(w_i+1= \text{sister} \mid w_i=\text{her}) = 1$

$P(w_i+1= \text{beginning} \mid w_i=\text{was}) = ½$

$P(w_i+1= \text{reading} \mid w_i=\text{was}) = ½$

$P(w_i+1= \text{bank} \mid w_i=\text{the}) = 1/3$

$P(w_i+1= \text{book} \mid w_i=\text{the}) = 1/3$

$P(w_i+1= \text{use} \mid w_i=\text{the}) = 1/3$

Language model

Language Model

- In statistical language applications, the knowledge of the source is referred as Language Model.
- We use language models in the various NLP applications:
  - speech recognition
  - spelling correction
  - machine translation
  - …..
- N-GRAM models are the language models which are widely used in NLP domain.

N-GRAMS

- To collect statistics to compute the functions in the following forms is difficult (sometimes impossible):

$$P(w_n|w_1^{n-1})$$

- Here we are trying to estimate the probability of seeing w_n after seeing w_1^{n-1}.

- We may approximate this computation just looking N previous words:

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

- So, an N-GRAM model

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

Chain Rule

- The probability of a word sequence $w_1w_2\ldots w_n$ is:

  $P(w_1w_2\ldots w_n)$

- We can use the chain rule of the probability to decompose this probability:

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

$$p(w_i \mid w_{i-1}) = \frac{C(w_{i-1}, w_i)}{C(w_{i-1})}$$

- Example:
- P(the man from jupiter) =

  P(the)P(man|the)P(from|the man)P(jupiter|the man from)

N-GRAMS (cont.)

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

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

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

Quadrigrams -- $$P(w_1^n) \approx \prod_{k=1}^{n} P(w_k \mid w_{k-1} w_{k-2} w_{k-3})$$

N-Gram Examples

Unigram

P(the man from jupiter) ≈ P(the)P(man)P(from)P(jupiter)

Bigram

P(the man from jupiter) ≈ P(the|*s*)P(man|the)P(from|man)P(jupiter|from)

Trigram

P(the man from jupiter) ≈

P(the|*s* *s*)P(man|*s* the)P(from|the man)P(jupiter|man from)

Markov Model

- The assumption that the probability of a word depends only on the last n words is called Markov assumption.

- 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.

- A bigram is called a first-order Markov model (because it looks one token into the past); A trigram is called a second-order Markov model;

- In general an N-Gram is called a N-1 order Markov model.

Markov assumption

Simplifying assumption:

P(the | its water is so transparent that) ≈ p(the|that)

Or maybe:

P(the | its water is so transparent that) ≈ p(the|transparent that)

Adapted from Jurafsky's slides

Simplest case: Unigram model

$$p(w_1 w_2 ... w_n) \approx \prod_i p(w_i)$$

Some automatically generated sentences from a unigram model:

fifth, an, of, futures, the, an, incorporated, a, a, the, inflation, most, dollars, quarter, in, is, mass

thrift, did, eighty, said, hard, 'm, july, bullish

that, or, limited, the

Adapted from Jurafsky's slides

Bigram model

- Condition on the previous word:

$$p(w_i \mid w_1 w_2 \ldots w_{i-1}) \approx \prod_i p(w_i \mid w_{i-1})$$

- Some automatically generated sentences from a bigram model:

- Texaco rose, one, in, this, issue, is, pursuing, growth, in, a, boiler, house, said, mr. gurria, mexico, 's, motion, control, proposal, without, permission, from, five, hundred, fifty, five, yen

- outside, new, car, parking, lot, of, the, agreement, reached

- this, would, be, a, record, november

Adapted from Jurafsky's slides

N-gram models

- We can extend to trigrams, 4-grams, 5-grams
- In general this is an insufficient model of language because language has long-distance dependencies:
  - "The computer which I had just put into the machine room on the fifth floor crashed."
- But we can often get away with N-gram models.

Estimating N-Gram Probabilities

Estimating bigram probabilities:

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{\sum_w C(w_{n-1}w)}$$

$$= \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

Where C is the count of that pattern in the corpus

Estimating n-gram probabilities

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

An example

- $p(w_i \mid w_{i-1}) = \frac{C(w_{i-1}, w_i)}{C(w_{i-1})}$
-
-

<s>I am Sam</s>

<s>Sam I am</s>

<s>I do not like green eggs and ham</s>

$P(\texttt{I} \mid \texttt{<s>}) = \frac{2}{3} = .67$ $P(\texttt{Sam} \mid \texttt{<s>}) = \frac{1}{3} = .33$ $P(\texttt{am} \mid \texttt{I}) = \frac{2}{3} = .67$

$P(\texttt{</s>} \mid \texttt{Sam}) = \frac{1}{2} = 0.5$ $P(\texttt{Sam} \mid \texttt{am}) = \frac{1}{2} = .5$ $P(\texttt{do} \mid \texttt{I}) = \frac{1}{3} = .33$

Which N-Gram?

- Unigram model $P(w_1)\ P(w_2)\ldots P(w_i)$
- Bigram model $P(w_1)\ P(w_2|w_1)\ldots P(w_i|w_{i-1})$
- Trigram model $P(w_1)\ P(w_2|w_1)P(w_3|w_2w_1)\ldots P(w_i|w_{i-2}w_{i-1})$
- N-gram model $P(w_1)\ P(w_2|w_1)P(w_3|w_2w_1)\ldots P(w_i|w_{i-n-1\ldots}w_{i-1})$

- N-gram models assume each word (event) depends only on the previous n-1 words (events). Such independence assumptions are called Markov assumptions (of order n-1).

Which N-Gram?

- Bigger N, the model will be more accurate.
  - But we may not get good estimates for n-gram probabilities.
  - The n-gram tables will be more sparse.

- Smaller N, the model will be less accurate.
  - But we may get better estimates for n-gram probabilities.
  - The n-gram tables will be less sparse.

- In reality, we do not use higher than trigram (not more than bigram).
- How big are n-gram tables with 10,000 words?
  - Unigram -- 10,000
  - Bigram -- 10000*10000 = 100,000,000
  - Trigram -- 10000*10000*10000 = 1,000,000,000,000

More examples

Berkeley restaurant project sentences

- can you tell me about any good cantonese restaurants close by
- mid priced thai food is what i'm looking for tell me about chez pani sse
- can you give me a listing of the kinds of food that are available
- i'm looking for a good place to eat breakfast
- when is caffe venezia open during the

Raw bigram counts

- Out of 9222 sentences

	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

Raw bigram probabilities

- Normalize by unigrams:

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

- Result:

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Bigram estimates of sentence probabilities

- P(<s> I want english food </s>) =
- p(I | <s>)
- × p(want | I)
- × p(english | want)
- × p(food|english)
- × p(</s> | food)
- =0.000031

What kinds of knowledge

- P(english|want) = .0011
- P(chinese|want) = .0065
- P(to|want) = .66
- P(eat | to) = .28
- P(food | to) = 0
- P(want | spend) = 0

Practical issues

- We do everything in log space:
  - Avoid underflow
  - Also adding is faster than multiplying.

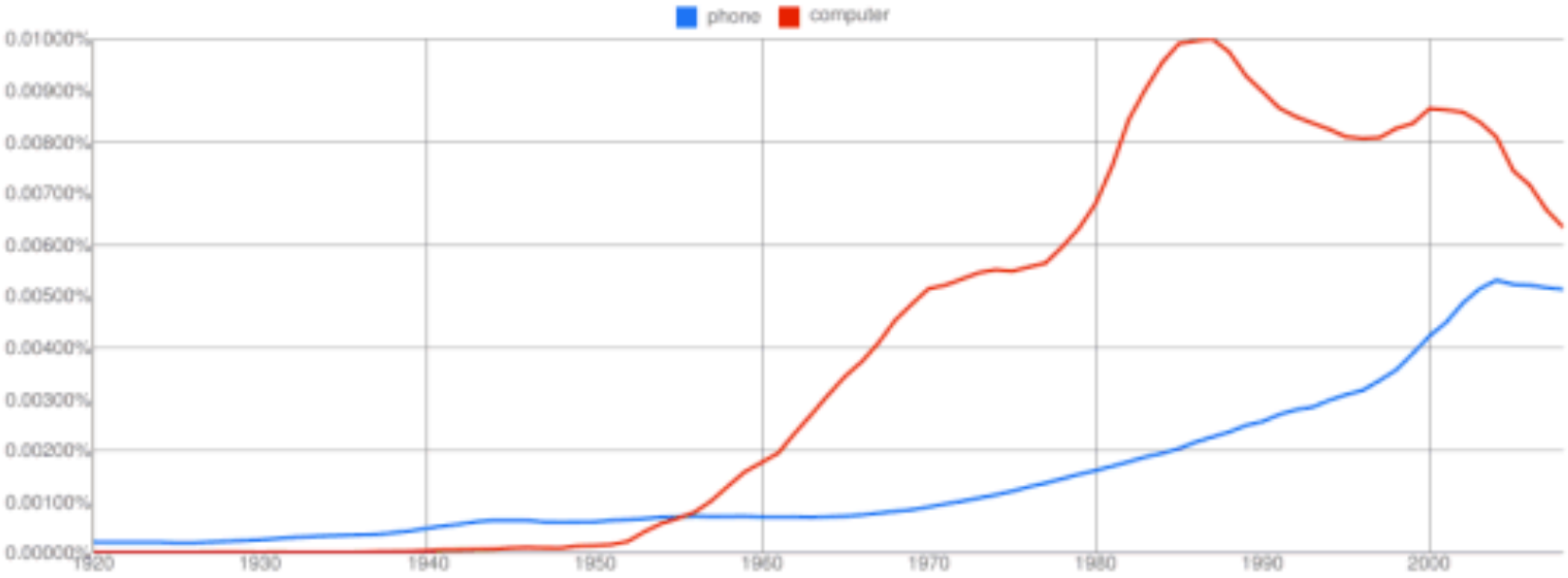
$$\log(p_1 \times p_2 \times p_3 \times p_4) = \log p_1 + \log p_2 + \log p_3 + \log p_4$$

Google labs Books Ngram Viewer

Graph these **case-sensitive** comma-separated phrases: phone,computer

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

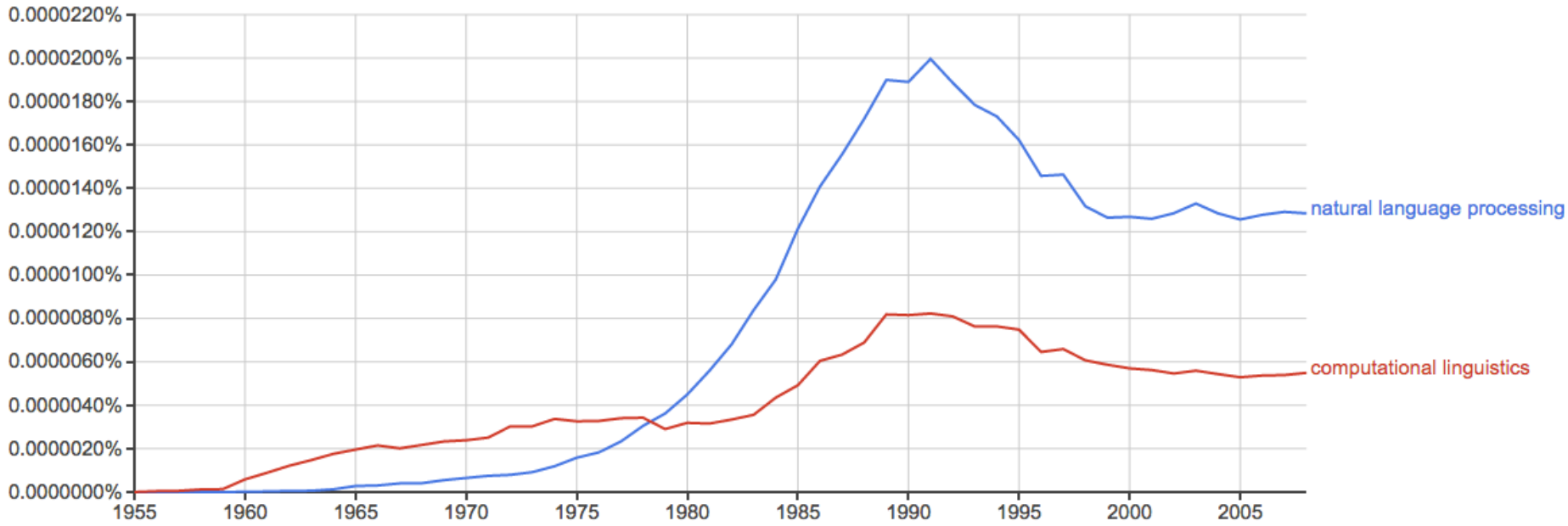
Search lots of books



Google Books Ngram Viewer

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

between 1955 and 2008 from the corpus English with smoothing of 3. Search lots of books



Train and Test Corpora

- A language model must be trained on a large corpus of text to estimate good parameter values.
- Model can be evaluated based on its ability to predict a high probability for a disjoint (held-out) test corpus (testing on the training corpus would give an optimistically biased estimate).

Zeroes

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

Test set

…denied the offer

…denied the loan

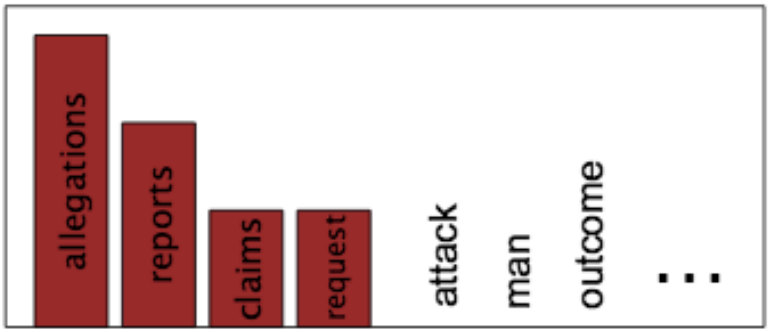
P(“offer” | denied the) = 0

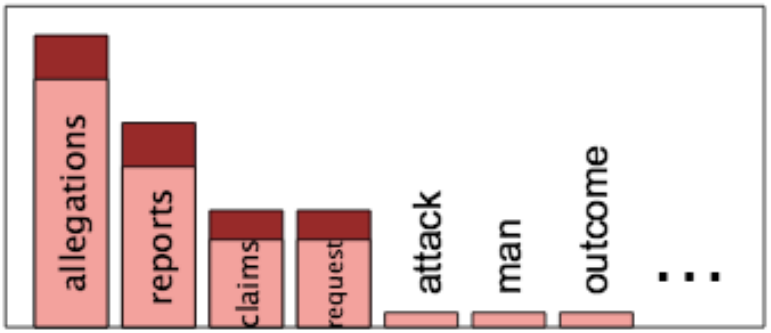
Zero probability bigrams

- Bigrams with zero probability
  - Mean that we will assign 0 probability to the test set!

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





Smoothing

- Since N-gram tables are too sparse, there will be a lot of entries with zero probability (or with very low probability).
  - The reason for this, our corpus is finite and it is not big enough to get that much information.
  - The task of re-evaluating some of zero-probability and low-probability n-grams is called smoothing.
- Smoothing Techniques:
  - add-one smoothing -- add one to all counts.
  - Witten-Bell Discounting -- use the count of things you have seen once to help estimate the count of things you have never seen.
  - Good-Turing Discounting -- a slightly more complex form of Witten-Bell Discounting
  - Backoff -- using lower level n-gram probabilities when n-gram probability is zero.

Add-one Smoothing (Laplace)

- Pretend we saw each word one more time than we did.
- Just add one to all counts!

- MLE estimate:

$$p(w_i \mid w_{i-1}) = \frac{C(w_{i-1}, w_i)}{C(w_{i-1})}$$

- Add-1 estimate:
-

$$p(w_i \mid w_{i-1}) = \frac{C(w_{i-1}, w_i) + 1}{C(w_{i-1}) + V}$$

Advanced Smoothing

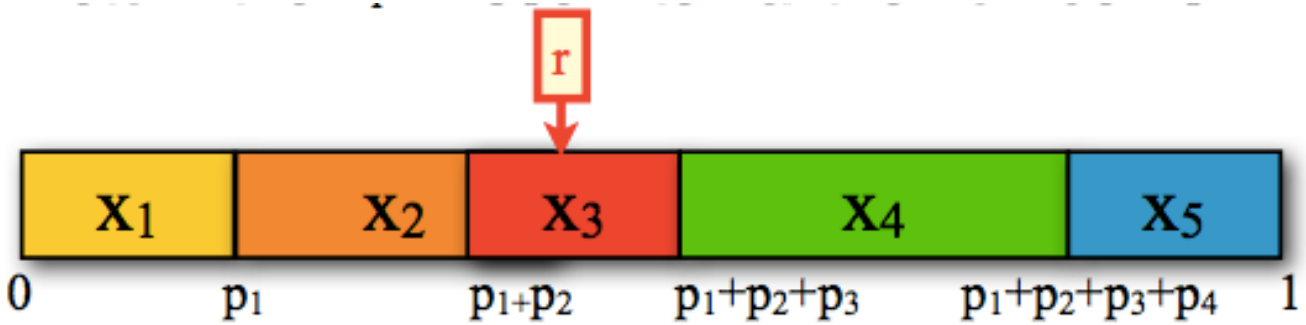
- Many advanced techniques have been developed to improve smoothing for language models.
  - Good-Turing
  - Interpolation
  - Backoff
  - Kneser-Ney
  - Class-based (cluster) N-grams

Maximum likelihood estimate (MLE)

- 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.

Using n-gram models to generate languages

- How do you generate text from an n-gram model?
- That is, how do you sample from a distribution $P(X \mid Y=y)$?
  - Assume X has n possible outcomes (values): $\{x_1,....,x_n\}$ and $P(x_i \mid Y=y) = p_i$
  - Divide the interval [0,1] into n smaller intervals according to the probabilities of the outcomes
  - Generate a random number r between 0 and 1.
  - Return the x1 whose interval the number is in.



Generating the Wall Street Journal

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

Generating 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? •Enter Menenius, if it so many good direction found'st thou art a strong upon command of fear not a liberal largess given away, Falstaff! Exeunt
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.

Evaluation: how good is our model?

- Does our language model prefer good sentences to bad ones?
  - Assign higher probability to "real" or "frequently observed" sentence
  - 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.

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

Lower perplexity = better model

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

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

About Turkish

- What do you think about the possible problems in Turkish language modelling?

- Hint: Think about the morphological structure of the language…

Summary

- Language models assign a probability that a sentence is a legal string in a language.
- They are useful as a component of many NLP systems, such as ASR, OCR, and MT.
- Simple N-gram models are easy to train on unsupervised corpora and can provide useful estimates of sentence likelihood.
- MLE gives inaccurate parameters for models trained on sparse data.
- Smoothing techniques adjust parameter estimates to account for unseen (but not impossible) events.

REFERENCES

- Julia Hockenmaier, Language Models (https://courses.engr.illinois.edu/cs498jh/Slides/Lecture02.pdf)
- Rong Jin, Introduction to Probability Theory
- Dan Jurafsky, Language Modelling, Introduction to N-grams
- Raymond Mooney, N-gram Language Models