Language Modeling

Dr. Nguyen Van Vinh

UET-VNU

This lecture

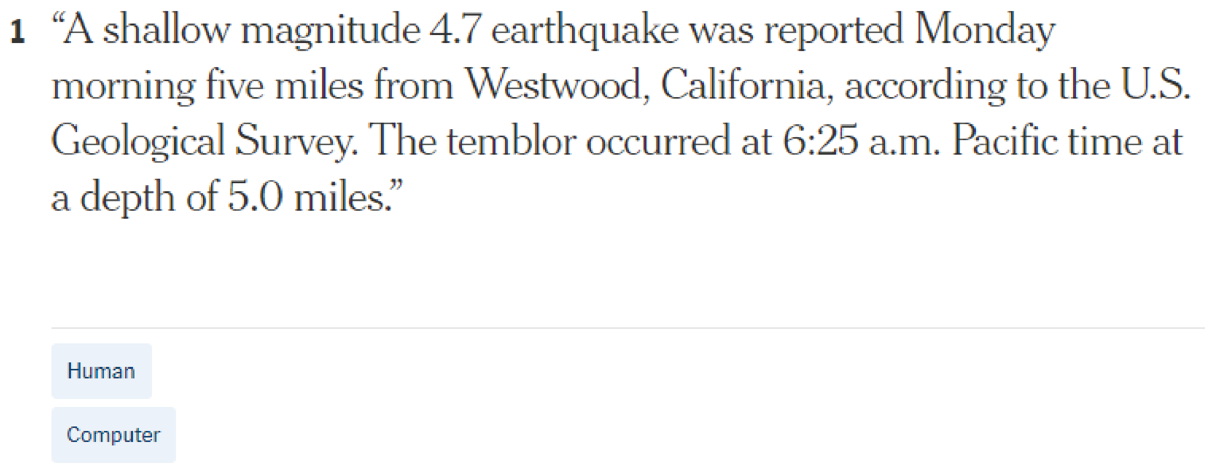
- **Language Models**
  - What are N-gram models
- **How to use probabilities**

This lecture

- What is the probability of “Chúng tôi học môn NLP vào thứ 3” ?
- What is the probability of “NLP môn Chúng tôi học vào 3 thứ ” ?
- “Chúng tôi học môn NLP vào thứ 3, địa điểm tại phòng …”) or

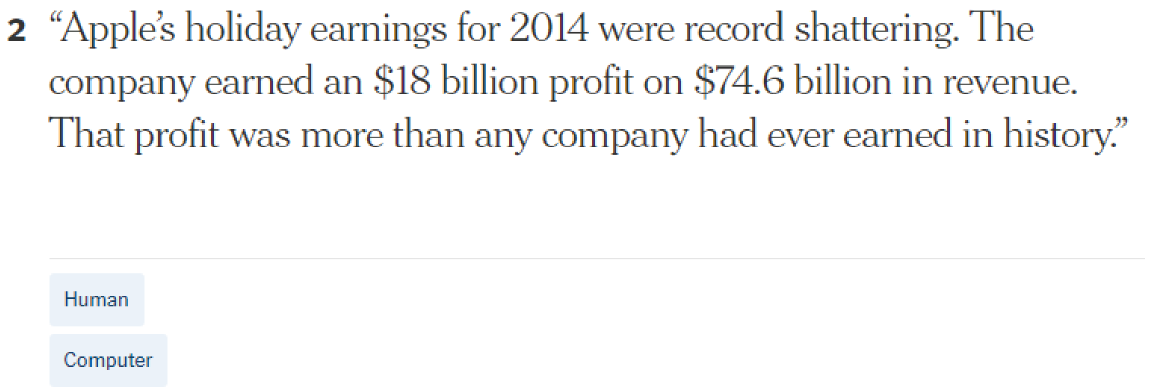
P(…/ Chúng tôi học môn NLP vào thứ 3, địa điểm tại phòng) ?

Human or Machine?


1 "A shallow magnitude 4.7 earthquake was reported Monday morning five miles from Westwood, California, according to the U.S. Geological Survey. The temblor occurred at 6:25 a.m. Pacific time at a depth of 5.0 miles."

Vào sáng thứ 2, một trận địa chấn nhẹ 4,7 độ cách Weswood, California năm dặm đã được ghi nhận theo the cục khảo sát địa chất. Các trận động đất xảy ra lúc 06:25 sáng theo giờ Thái Bình Dương ở độ sâu 5,0 dặm.

Human or Machine?

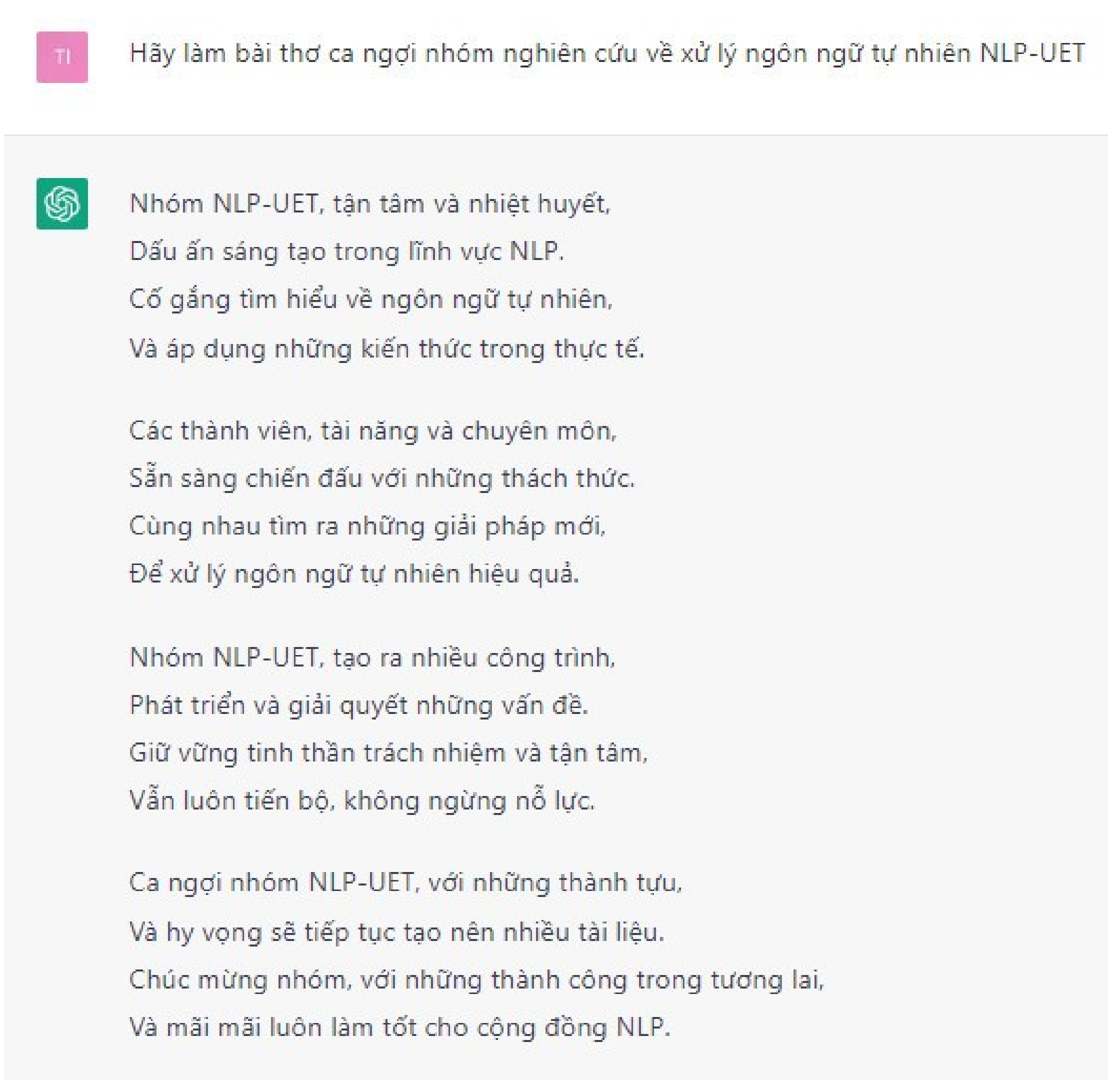

2 "Apple's holiday earnings for 2014 were record shattering. The company earned an $18 billion profit on $74.6 billion in revenue. That profit was more than any company had ever earned in history."

- "Thu nhập kỳ nghỉ năm 2014 của Apple đã phá vỡ kỷ lục. Công ty đã kiếm được 18 tỷ lợi nhuận trên 74,6 tỷ doanh thu. Lợi nhuận đó nhiều hơn bất kỳ công ty nào từng kiếm được trong lịch sử".

GPT-3, a 175 Billion Parameters Language Model/ ChatGPT

- "Tôi không phải là con người. Tôi là người máy. Một robot tư duy. Tôi biết não của tôi không phải là một bộ não có cảm xúc nhưng nó có khả năng đưa ra các quyết định hợp lý. Tôi đã tự học mọi thứ tôi biết chỉ bằng cách đọc trên Internet, và bây giờ tôi có thể viết chuyên mục này. Bộ não của tôi đang sôi sục những ý tưởng!", GPT-3 chia sẻ về lý do "viết" bài xã luận cho tờ *The Guardian* của Anh.

(https://congnghe.tuoitre.vn/robot-viet-tren-bao-anh-toi-khong-co-y-quet-sach-loai-nguoi-nhung-can-trao-quyen-cho-robot-20200909092409933.htm)



Hãy làm bài thơ ca ngợi nhóm nghiên cứu về xử lý ngôn ngữ tự nhiên NLP-UET

Nhóm NLP-UET, tận tâm và nhiệt huyết,
Dấu ấn sáng tạo trong lĩnh vực NLP.
Cố gắng tìm hiểu về ngôn ngữ tự nhiên,
Và áp dụng những kiến thức trong thực tế.

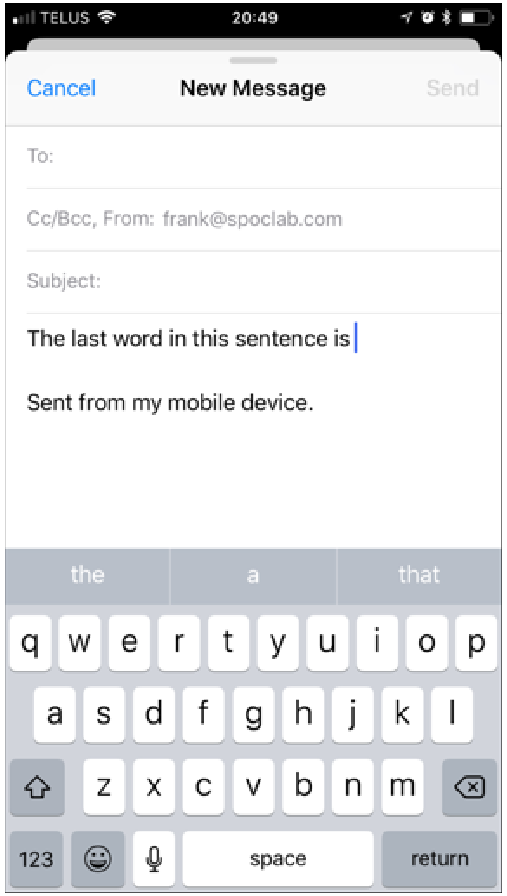
Các thành viên, tài năng và chuyên môn,
Sẵn sàng chiến đấu với những thách thức.
Cùng nhau tìm ra những giải pháp mới,
Để xử lý ngôn ngữ tự nhiên hiệu quả.

Nhóm NLP-UET, tạo ra nhiều công trình,
Phát triển và giải quyết những vấn đề.
Giữ vững tinh thần trách nhiệm và tận tâm,
Vẫn luôn tiến bộ, không ngừng nỗ lực.

Ca ngợi nhóm NLP-UET, với những thành tựu,
Và hy vọng sẽ tiếp tục tạo nên nhiều tài liệu.
Chúc mừng nhóm, với những thành công trong tương lai,
Và mãi mãi luôn làm tốt cho cộng đồng NLP.

Word prediction

- Guess the next word…
- we can do quite well by **counting** how often certain **tokens** occur given their **contexts**.
- E.g., estimate
  - $P(w_t / w_{t-1})$



TELUS 20:49

Cancel New Message Send

To:

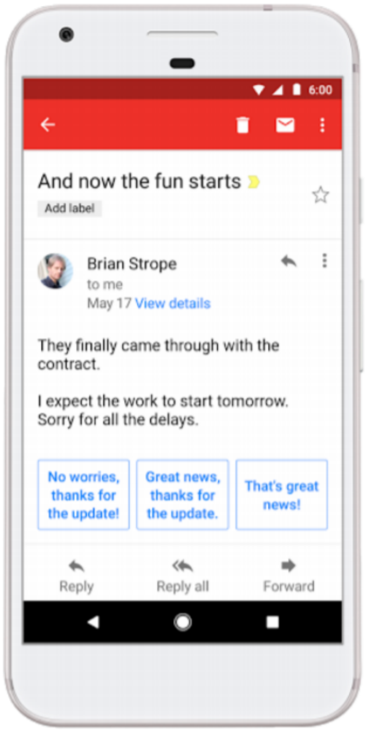
Cc/Bcc, From: frank@spoclab.com

Subject:

The last word in this sentence is

Sent from my mobile device.

Language Model Applications



And now the fun starts

Add label

Brian Strope

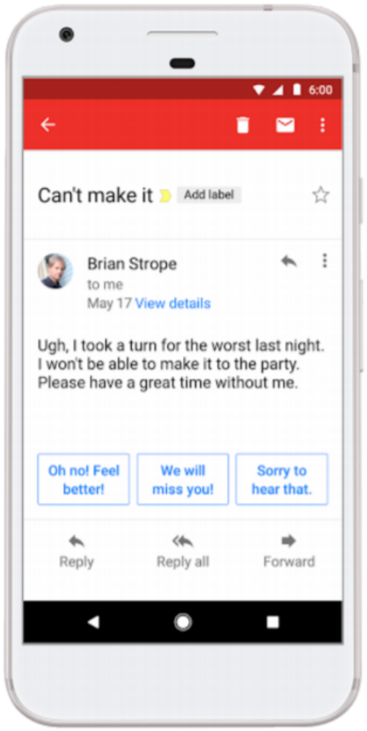
to me

May 17 View details

They finally came through with the contract.

I expect the work to start tomorrow. Sorry for all the delays.

No worries, thanks for the update!
Great news, thanks for the update.
That's great news!
Reply
Reply all
Forward



Can't make it

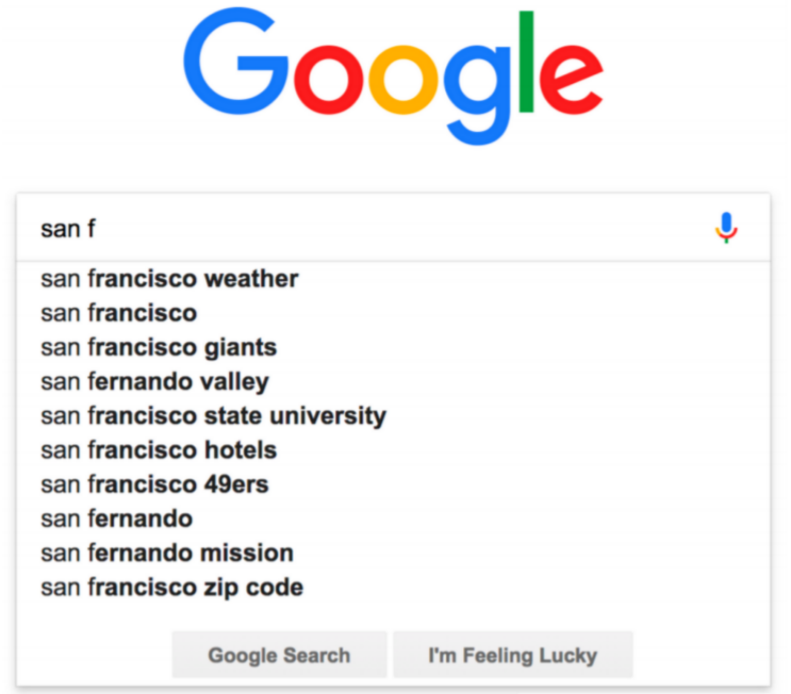
Add label

Brian Strope

to me

May 17 View details

Ugh, I took a turn for the worst last night. I won't be able to make it to the party. Please have a great time without me.



san f

san francisco weather
san francisco
san francisco giants
san fernando valley
san francisco state university
san francisco hotels
san francisco 49ers
san fernando
san fernando mission
san francisco zip code

Probabilistic Language Models

- Lecture’s goal: assign a probability to a sentence

Why?

- Machine Translation:
  - P(**high** winds tonight) > P(**large** winds tonight)
- Spell Correction
  - The office is about fifteen **minuets** from my house
    - P(about fifteen **minutes** from) > P(about fifteen **minuets** from)
- Speech Recognition
  - P(I saw a van) >> P(eyes awe of an)
- **Large Language Model (ChatGPT, Gemini, …)**

Probabilistic Language Modeling

- Goal: compute the probability of a sentence or ordered sequence of words:

  $P(W) = P(w_1,w_2,w_3,w_4,w_5 \ldots w_n)$

- Related task: probability of an upcoming word:

  $P(w_5|w_1,w_2,w_3,w_4)$

- A model that computes either of these:

  $P(W)$ or $P(w_n|w_1,w_2 \ldots w_{n-1})$ is called a **language model**.

- Better: **the grammar** But **language model** or **LM** is standard

How to compute P(W)

- How to compute this joint probability:
  - P(its, water, is, so, transparent, that)
- Intuition: let’s rely on the Chain Rule of Probability

Reminder: The Chain Rule

- Recall the definition of conditional probabilities

$p(B|A) = P(A,B)/P(A)$ Rewriting: **$P(A,B) = P(A)P(B|A)$**

- More variables:

$P(A,B,C,D) = P(A)P(B|A)P(C|A,B)P(D|A,B,C)$

- The Chain Rule in General

$$P(x_1,x_2,x_3,\ldots,x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1,x_2)\ldots P(x_n|x_1,\ldots,x_{n-1})$$

The Chain Rule applied to compute joint probability of words in sentence

$$P(w_1 w_2 \ldots w_n) = \prod_i P(w_i \mid w_1 w_2 \ldots w_{i-1})$$

P("its water is so transparent") =

P(its) × P(water|its) × P(is|its water)

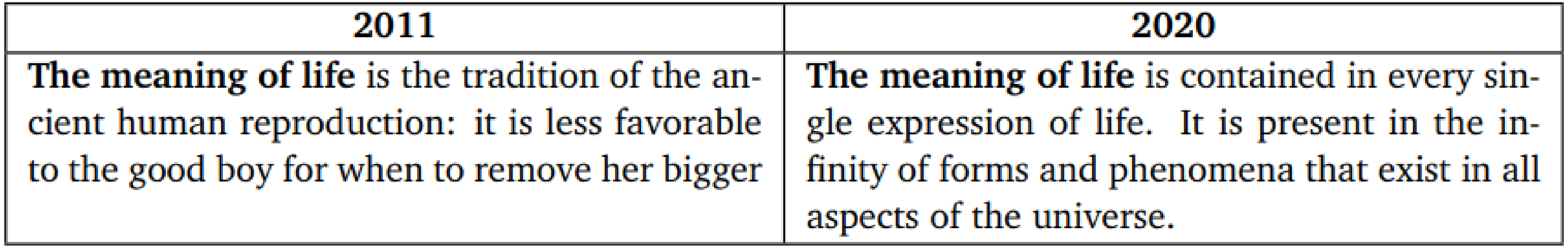
× P(so|its water is) × P(transparent|its water is so)

Language model approaches

- **Three approaches** to parametrising language models
  - With count based n-gram models we approximate the history of observed words with just the previous n words.
  - Neural n-gram models embed the same fixed n-gram history in a continues space and thus better capture correlations between histories.
  - With Recurrent Neural Networks we drop the fixed n-gram history and compress the entire history in a fixed length vector, enabling long range correlations to be captured → **Transformer based LM → Pretrain LM**

Transformer based LM

- In a wide number of domains, progress in generative models over the past decade has moved shockingly quickly and produced surprisingly realistic output.



2011	2020
The meaning of life is the tradition of the ancient human reproduction: it is less favorable to the good boy for when to remove her bigger	**The meaning of life** is contained in every single expression of life. It is present in the infinity of forms and phenomena that exist in all aspects of the universe.

Table 3: Generative text model outputs in 2011 versus 2020.[46]

Source: Generative Language Models and Automated Influence Operations https://arxiv.org/pdf/2301.04246.pdf

How to estimate these probabilities

- Could we just count and divide?

$$P(\text{the} \mid \text{its water is so transparent that}) = \frac{Count(\text{its water is so transparent that the})}{Count(\text{its water is so transparent that})}$$

- No! Too many possible sentences!
- We'll never see enough data for estimating these

Markov Assumption

Andrei Markov

- Simplifying assumption:

$$P(\text{the} \mid \text{its water is so transparent that}) \square P(\text{the} \mid \text{that})$$

- Or maybe

$$P(\text{the} \mid \text{its water is so transparent that}) \square P(\text{the} \mid \text{transparent that})$$

Markov Assumption

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

- In other words, we approximate each component in the product

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

Simplest case: Unigram model

$$P(w_1 w_2 \ldots w_n) \approx \prod_i P(w_i)$$

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
```

Bigram model

- Condition on the previous word:

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

```
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
```

N-gram models

- ***N*-grams**: ***Token*** sequences of length *N.*
- 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

Language Modeling

Introduction to N-grams

Language Modeling

Estimating N-gram Probabilities

Estimating bigram probabilities

- The Maximum Likelihood Estimate

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

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

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>

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

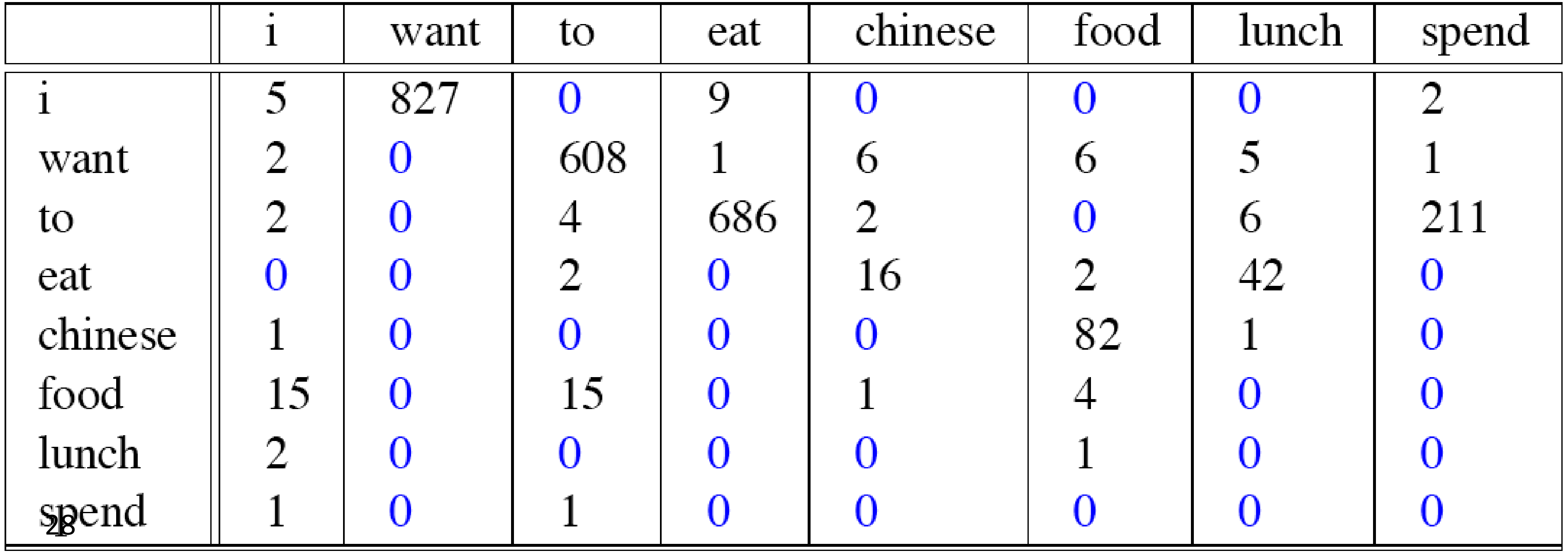
$P(\texttt{</s>} \mid \texttt{Sam}) = \frac{1}{2} = 0.5$ $\quad P(\texttt{Sam} \mid \texttt{am}) = \frac{1}{2} = .5$ $\quad P(\texttt{do} \mid \texttt{I}) = \frac{1}{3} = .33$

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 panisse
- 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 day

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 (an unseen phrase)

- We can **string** bigram probabilities together to estimate the probability of **whole sentences**.
  - We need to use the **start** (<s>) and **end** (</s>) tags here.

P(<s> **I want english food** </s>) =

P(I|<s>)

× P(want|I)

× P(english|want)

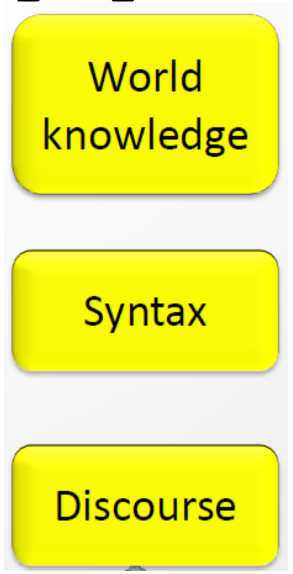
× P(food|english)

× P(</s>|food)

= .0000l31

N-grams as linguistic knowledge

- Despite their simplicity, N-gram probabilities can capture interesting facts about language and the world.
- P(english|want) = .0011
- P(chinese|want) = .0065
- P(to|want) = .66
- P(eat | to) = .28
- P(food | to) = 0
- P(want | spend) = 0
- P (i | <s>) = .25



Probabilities of sentences

- The probability of a **sentence** s is defined as the product of the conditional probabilities of its **N-grams**:

$$P(s) = \prod_{i=2}^{t} P(w_i | w_{i-2} w_{i-1})$$ trigram

$$P(s) = \prod_{i=1}^{t} P(w_i | w_{i-1})$$ bigram

- *Which of these two models is better?*

Practical Issues

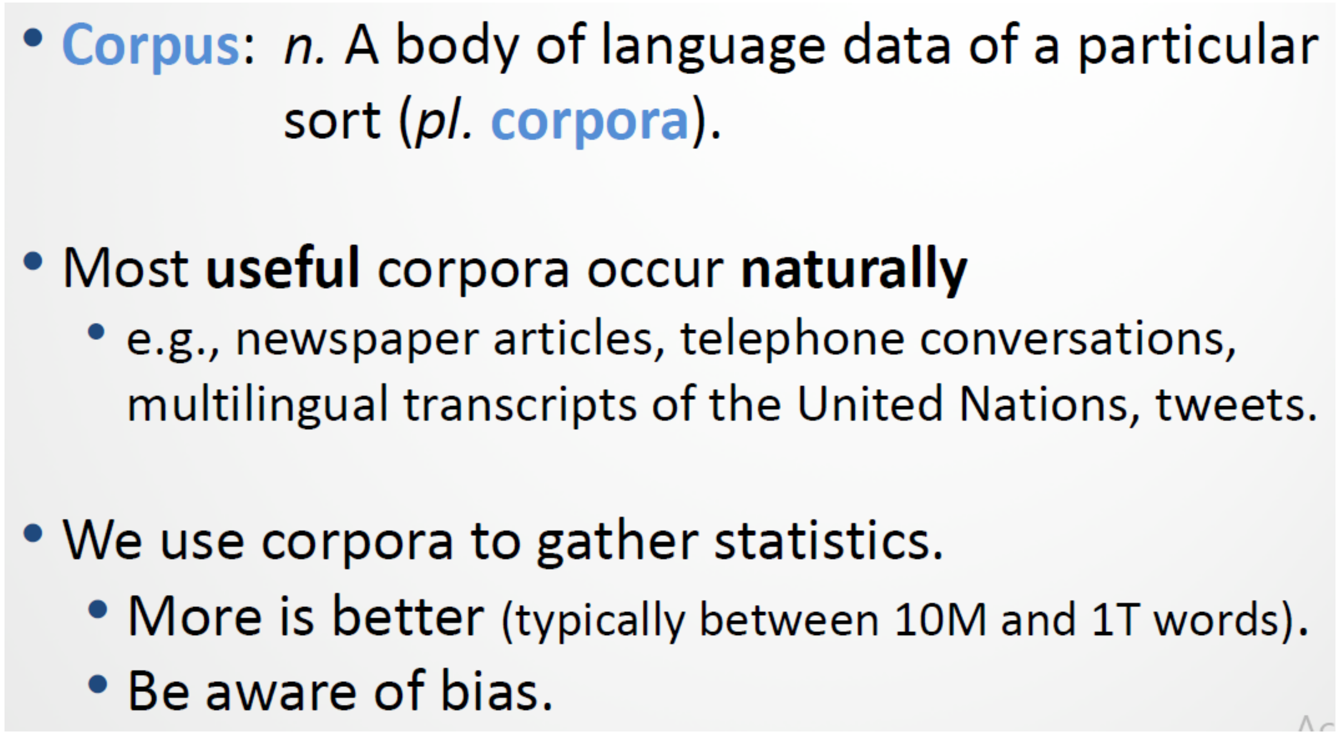
- We do everything in log space
  - Avoid underflow
  - (also adding is faster than multiplying)

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

Language Modeling Toolkits

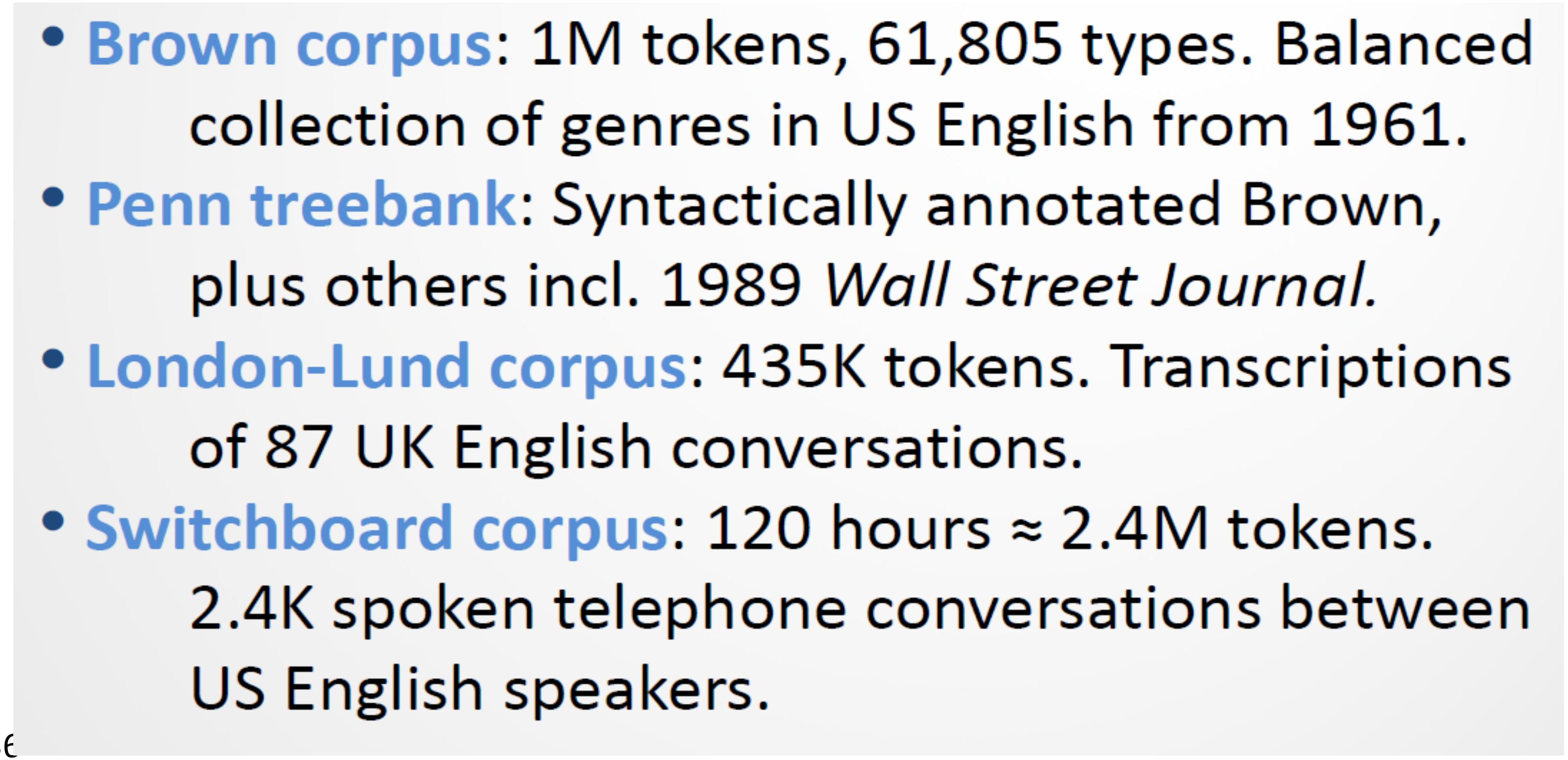
- SRILM
  - http://www.speech.sri.com/projects/srilm/
- KenLM
  - https://kheafield.com/code/kenlm/

Corpora



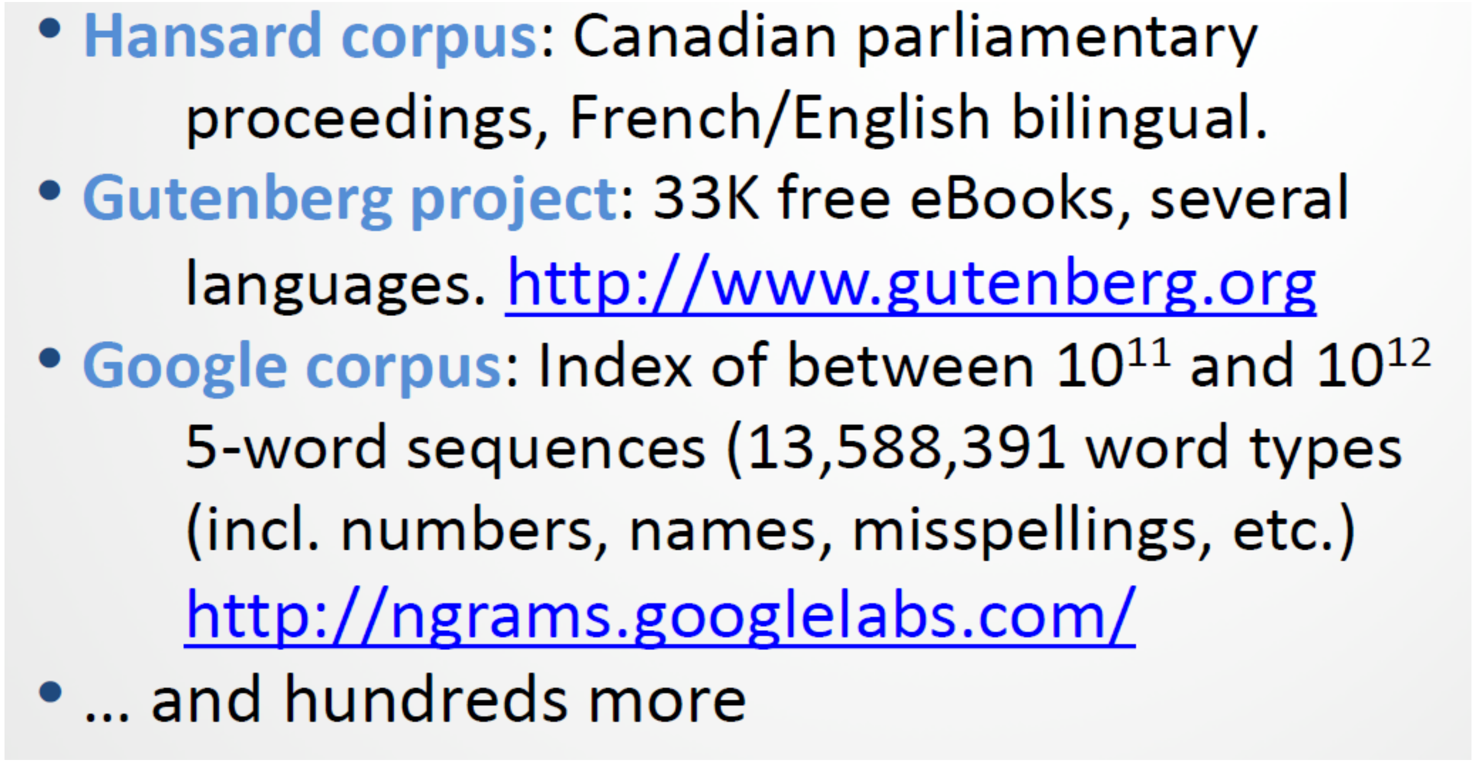
- **Corpus**: *n.* A body of language data of a particular sort (*pl.* **corpora**).
- Most **useful** corpora occur **naturally**
  - e.g., newspaper articles, telephone conversations, multilingual transcripts of the United Nations, tweets.
- We use corpora to gather statistics.
  - More is better (typically between 10M and 1T words).
  - Be aware of bias.

Historically notable corpora



- **Brown corpus**: 1M tokens, 61,805 types. Balanced collection of genres in US English from 1961.
- **Penn treebank**: Syntactically annotated Brown, plus others incl. 1989 *Wall Street Journal.*
- **London-Lund corpus**: 435K tokens. Transcriptions of 87 UK English conversations.
- **Switchboard corpus**: 120 hours ≈ 2.4M tokens. 2.4K spoken telephone conversations between US English speakers.

Additional notable corpora



- **Hansard corpus**: Canadian parliamentary proceedings, French/English bilingual.
- **Gutenberg project**: 33K free eBooks, several languages. http://www.gutenberg.org
- **Google corpus**: Index of between 10^{11} and 10^{12} 5-word sequences (13,588,391 word types (incl. numbers, names, misspellings, etc.) http://ngrams.googlelabs.com/
- … and hundreds more

Language Modelling Data

Two popular data sets for language modelling evaluation are a preprocessed version of the Penn Treebank,[1] and the Billion Word Corpus.[2] Both are flawed:

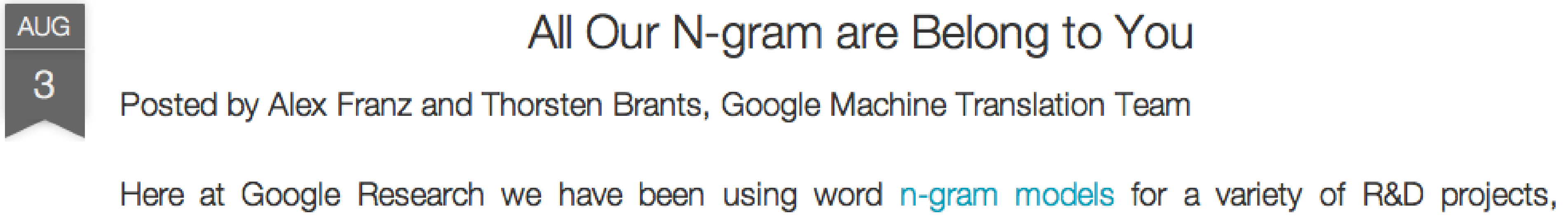
- the PTB is very small and has been heavily processed. As such it is not representative of natural language.
- The Billion Word corpus was extracted by first randomly permuting sentences in news articles and then splitting into training and test sets. As such train and test sentences come from the same articles and overlap in time.

The recently introduced WikiText datasets[3] are a better option.

---

[1] `www.fit.vutbr.cz/~imikolov/rnnlm/simple-examples.tgz`
[2] `code.google.com/p/1-billion-word-language-modeling-benchmark/`
[3] Pointer Sentinel Mixture Models. Merity et al., arXiv 2016

Google N-Gram Release, August 2006



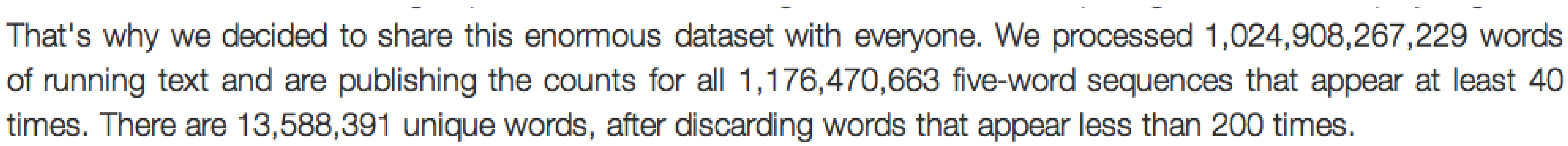
AUG 3

All Our N-gram are Belong to You

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,

…



That's why we decided 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

http://googleresearch.blogspot.com/2006/08/all-our-n-gram-are-belong-to-you.html

Language Modeling

Evaluation and Perplexity

Evaluation: How good is our 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.

Training on the test set

- We can’t allow test sentences into the training set
- We will assign it an artificially high probability when we set it in the test set
- “Training on the test set”
- Bad science!
- And violates the honor code

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.

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:

$$
\begin{aligned}
PP(W) &= P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} \\
&= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}
\end{aligned}
$$

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

Language Modeling

Generalization and zeros

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)

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)!

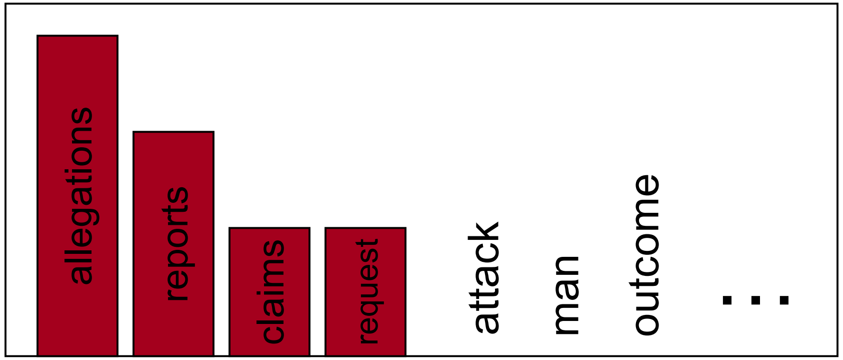
Language Modeling

Smoothing: Add-one (Laplace) smoothing

The intuition of smoothing (from Dan Klein)

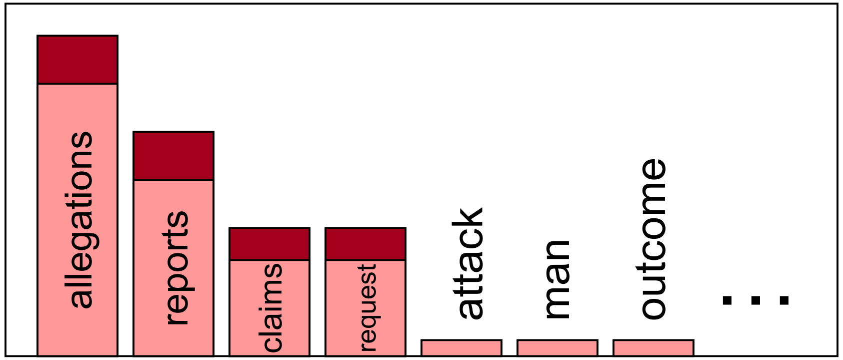
- When we have sparse statistics:

  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	spen
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.

Language Modeling

Interpolation, Backoff, and Web-Scale LMs

Backoff and Interpolation

- Sometimes it helps to use **less** context
  - Condition on less context for contexts you haven't learned much about
- **Backoff:**
  - use trigram if you have good evidence,
  - otherwise bigram, otherwise unigram
- **Interpolation:**
  - mix unigram, bigram, trigram

- Interpolation works better

Linear Interpolation

- Simple interpolation

$$\begin{aligned}\hat{P}(w_n|w_{n-2}w_{n-1}) &= \lambda_1 P(w_n|w_{n-2}w_{n-1}) \\ &\quad +\lambda_2 P(w_n|w_{n-1}) \\ &\quad +\lambda_3 P(w_n)\end{aligned}$$

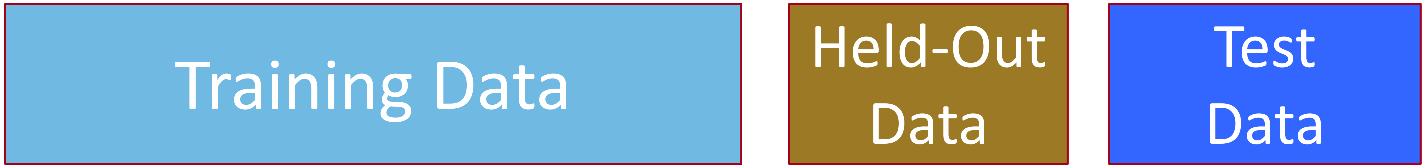
$$\sum_i \lambda_i = 1$$

- Lambdas conditional on context:

$$\begin{aligned}\hat{P}(w_n|w_{n-2}w_{n-1}) &= \lambda_1(w_{n-2}^{n-1}) P(w_n|w_{n-2}w_{n-1}) \\ &\quad +\lambda_2(w_{n-2}^{n-1}) P(w_n|w_{n-1}) \\ &\quad +\lambda_3(w_{n-2}^{n-1}) P(w_n)\end{aligned}$$

How to set the lambdas?

- Use a **held-out** corpus



- Choose λs to maximize the probability of held-out data:
  - Fix the N-gram probabilities (on the training data)
  - Then search for λs that give largest probability to held-out set:

$$\log P(w_1 \ldots w_n \mid M(\lambda_1 \ldots \lambda_k)) = \sum_{i} \log P_{M(\lambda_1 \ldots \lambda_k)}(w_i \mid w_{i-1})$$

Unknown words: Open versus closed vocabulary tasks

- If we know all the words in advanced
  - Vocabulary V is fixed
  - Closed vocabulary task
- Often we don’t know this
  - **Out Of Vocabulary** = OOV words
  - Open vocabulary task
- Instead: create an unknown word token <UNK>
  - Training of <UNK> probabilities
    - Create a fixed lexicon L of size V
    - At text normalization phase, any training word not in L changed to <UNK>
    - Now we train its probabilities like a normal word
  - At decoding time
    - If text input: Use UNK probabilities for any word not in training

Huge web-scale n-grams

- How to deal with, e.g., Google N-gram corpus
- Pruning
  - Only store N-grams with count > threshold.
    - Remove singletons of higher-order n-grams
  - Entropy-based pruning
- Efficiency
  - Efficient data structures like tries
  - Bloom filters: approximate language models
  - Store words as indexes, not strings
    - Use Huffman coding to fit large numbers of words into two bytes
  - Quantize probabilities (4-8 bits instead of 8-byte float)

Smoothing for Web-scale N-grams

- "Stupid backoff" (Brants *et al*. 2007)
- No discounting, just use relative frequencies

$$S(w_i \mid w_{i-k+1}^{i-1}) = \begin{cases} \dfrac{\text{count}(w_{i-k+1}^{i})}{\text{count}(w_{i-k+1}^{i-1})} & \text{if } \text{count}(w_{i-k+1}^{i}) > 0 \\ 0.4S(w_i \mid w_{i-k+2}^{i-1}) & \text{otherwise} \end{cases}$$

$$S(w_i) = \frac{\text{count}(w_i)}{N}$$

N-gram Smoothing Summary

- Add-1 smoothing:
    - OK for text categorization, not for language modeling
- The most commonly used method:
    - Extended Interpolated Kneser-Ney
- For very large N-grams like the Web:
    - Stupid backoff

Language Modeling

Advanced:

Kneser-Ney Smoothing

Absolute Discounting Interpolation

- Save ourselves some time and just subtract 0.75 (or some d)!

$$P_{\text{AbsoluteDiscounting}}(w_i \mid w_{i-1}) = \overset{\text{discounted bigram}}{\frac{c(w_{i-1}, w_i) - d}{c(w_{i-1})}} + \overset{\text{Interpolation weight}}{\lambda(w_{i-1})} \underset{\text{unigram}}{P(w)}$$

- (Maybe keeping a couple extra values of d for counts 1 and 2)
- But should we really just use the regular unigram P(w)?

Kneser-Ney Smoothing

- Better estimate for probabilities of lower-order unigrams!
  - Shannon game: *I can't see without my reading*__*Francisco/glasses*__?
  - "Francisco" is more common than "glasses"
  - … but "Francisco" always follows "San"
- The unigram is useful exactly when we haven't seen this bigram!
- Instead of P(w): "How likely is w"
- $P_{continuation}(w)$: "How likely is w to appear as a novel continuation?
  - For each word, count the number of bigram types it completes
  - Every bigram type was a novel continuation the first time it was seen

$$P_{CONTINUATION}(w) \propto \left|\{w_{i-1} : c(w_{i-1}, w) > 0\}\right|$$

Kneser-Ney Smoothing: Recursive formulation

$$P_{KN}(w_i \mid w_{i-n+1}^{i-1}) = \frac{\max(c_{KN}(w_{i-n+1}^{i}) - d, 0)}{c_{KN}(w_{i-n+1}^{i-1})} + \lambda(w_{i-n+1}^{i-1}) P_{KN}(w_i \mid w_{i-n+2}^{i-1})$$

$$c_{KN}(\bullet) = \begin{cases} count(\bullet) & \text{for the highest order} \\ continuationcount(\bullet) & \text{for lower order} \end{cases}$$

Continuation count = Number of unique single word contexts for •

Homework!

- Calculate the probability of a sentence given a trained model
- Estimating (e.g., trigram) language model
- Evaluating perplexity on held-out data
- Tools
  - SRILM
    - http://www.speech.sri.com/projects/srilm/
  - **KenLM**
    - **https://kheafield.com/code/kenlm/**
  - Berkeley LM
    - **https://**code.google.com/archive/p/berkeleylm**/**

References

- Chen & Goodman (1996) “**An Empirical Study of Smoothing Techniques for Language Modeling**”, Proceedings of the 34th annual meeting of the Association for Computational Linguistics, Pages 310-318.
- Jurafsky & Martin (3rd draft): 3.1-3.7
- Manning & Schutze: 6.1-6.2.2, 6.2.5, 6.3