Natural Language Processing: N-Gram Language Models

15 February 2022

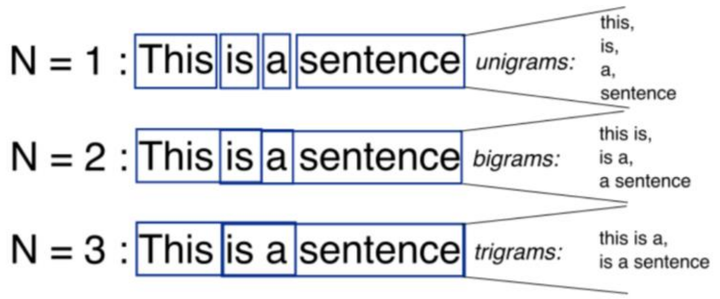
https://tinyurl.com/NLP2022part3

Language Modeling

Introduction to N-grams

N-grams and Language Models

- N-gram: important concept in NLP which is the basis for language modelling
- N-grams – contiguous sequences of tokens from a given text

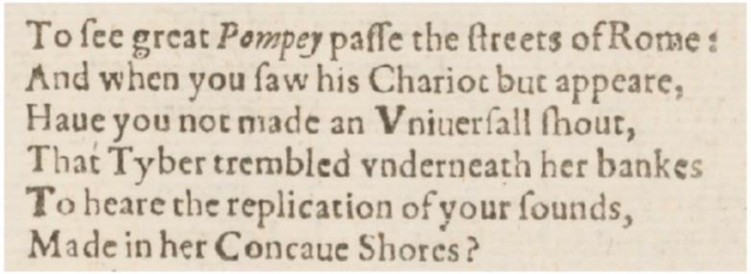


Probabilistic Language Models based on N-grams

- Goal: assign a probability to word sequence
  - Machine Translation:
    - P(**high** winds tonight) > P(**large** winds tonight)
  - Spell Correction
    - P(about fifteen **minutes** from) > P(about fifteen **minuets** from)
  - Speech Recognition
    - P(I saw a van) >> P(eyes awe of an)
  - Summarization, question-answering, etc.

Application example: OCR

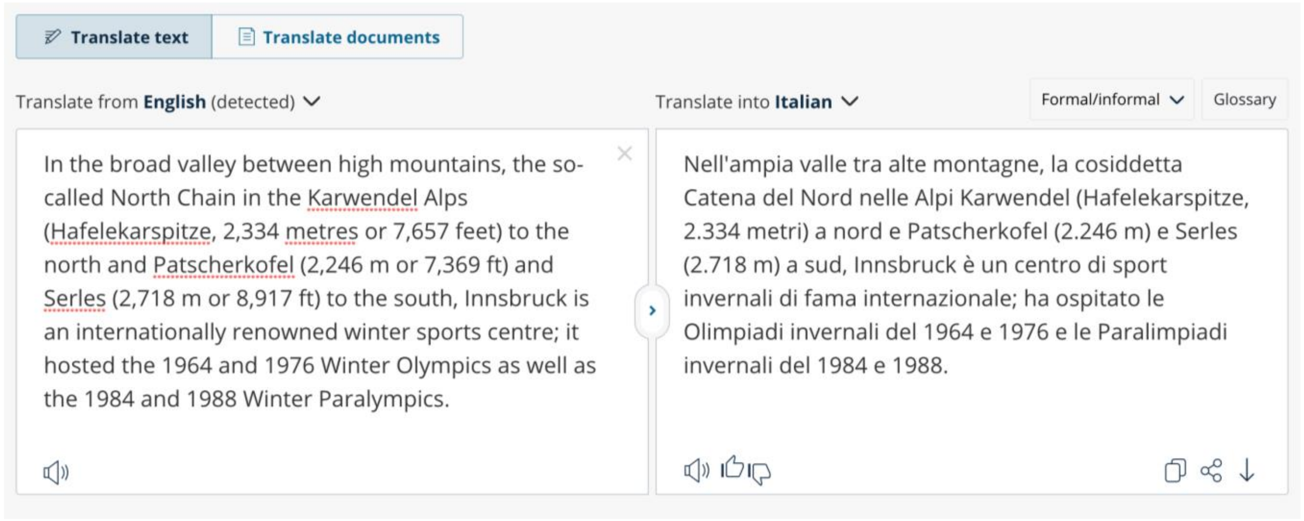
- to fee great Pompey paffe the Areets of Rome:
- to see great Pompey passe the streets of Rome:



To ſee great *Pompey* paſſe the ſtreets of Rome:
And when you ſaw his Chariot but appeare,
Haue you not made an Vniuerſall ſhout,
That Tyber trembled vnderneath her bankes
To heare the replication of your ſounds,
Made in her Concaue Shores?

Application example: Machine Translation

- Fidelity (to source text)
- Fluency (of the translation)

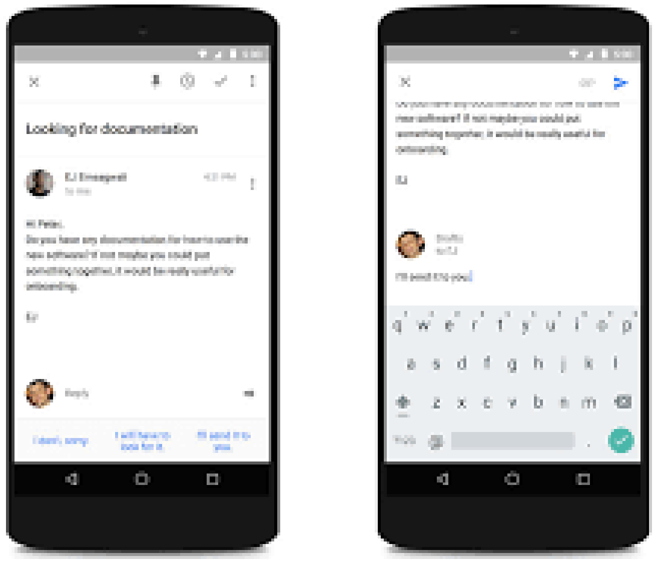


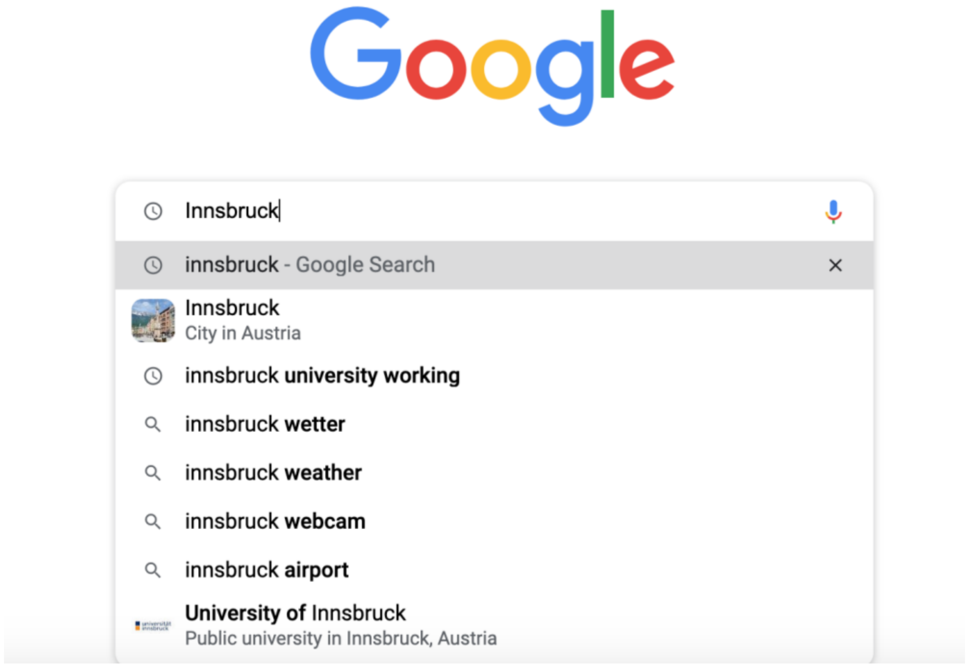
In the broad valley between high mountains, the so-called North Chain in the Karwendel Alps (Hafelekarspitze, 2,334 metres or 7,657 feet) to the north and Patscherkofel (2,246 m or 7,369 ft) and Serles (2,718 m or 8,917 ft) to the south, Innsbruck is an internationally renowned winter sports centre; it hosted the 1964 and 1976 Winter Olympics as well as the 1984 and 1988 Winter Paralympics.

Nell'ampia valle tra alte montagne, la cosiddetta Catena del Nord nelle Alpi Karwendel (Hafelekarspitze, 2.334 metri) a nord e Patscherkofel (2.246 m) e Serles (2.718 m) a sud, Innsbruck è un centro di sport invernali di fama internazionale; ha ospitato le Olimpiadi invernali del 1964 e 1976 e le Paralimpiadi invernali del 1984 e 1988.

http://www.deepl.com

Application example: Answer/Query Completions/Suggestion




Innsbruck

innsbruck - Google Search

Innsbruck
City in Austria

innsbruck **university working**

innsbruck **wetter**

innsbruck **weather**

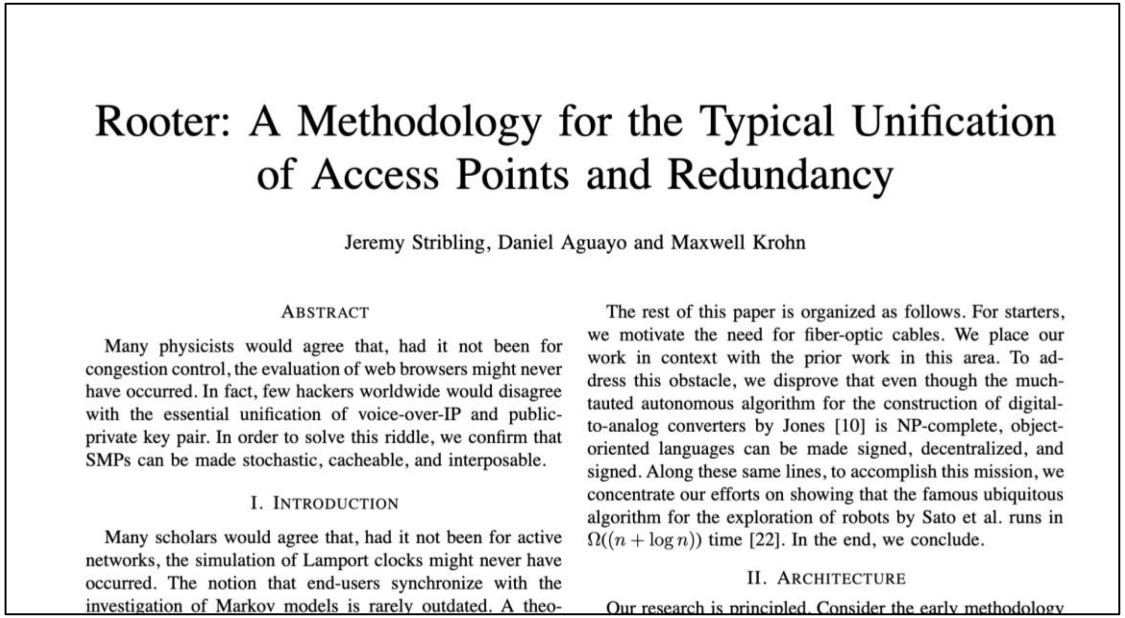
innsbruck **webcam**

innsbruck **airport**

University of Innsbruck
Public university in Innsbruck, Austria

Other Uses

- Augmentative & Alternative Communication (AAC) systems
    - For users who are physically unable to write/sign but can for example use eye gaze
    - Effective word prediction to be chosen is important
- Predictive text input systems can guess what user is typing and offer choices on how to complete it



Rooter: A Methodology for the Typical Unification of Access Points and Redundancy

Jeremy Stribling, Daniel Aguayo and Maxwell Krohn

ABSTRACT

Many physicists would agree that, had it not been for congestion control, the evaluation of web browsers might never have occurred. In fact, few hackers worldwide would disagree with the essential unification of voice-over-IP and public-private key pair. In order to solve this riddle, we confirm that SMPs can be made stochastic, cacheable, and interposable.

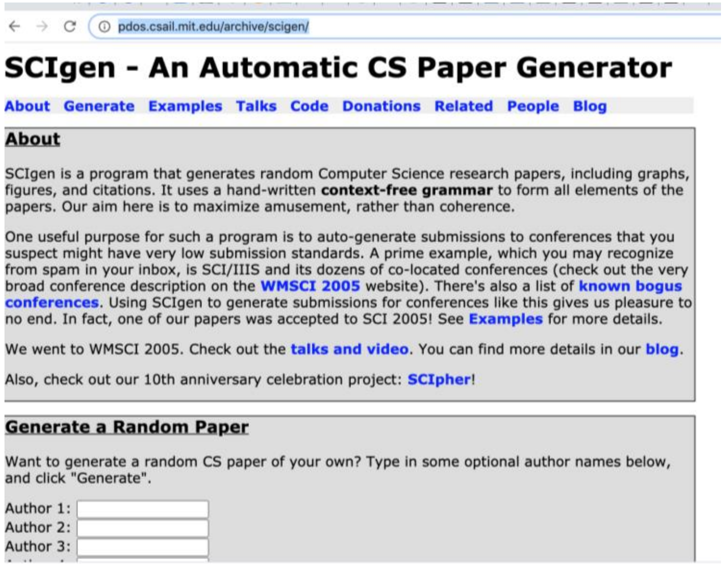
I. INTRODUCTION

Many scholars would agree that, had it not been for active networks, the simulation of Lamport clocks might never have occurred. The notion that end-users synchronize with the investigation of Markov models is rarely outdated. A theo-

The rest of this paper is organized as follows. For starters, we motivate the need for fiber-optic cables. We place our work in context with the prior work in this area. To address this obstacle, we disprove that even though the much-tauted autonomous algorithm for the construction of digital-to-analog converters by Jones [10] is NP-complete, object-oriented languages can be made signed, decentralized, and signed. Along these same lines, to accomplish this mission, we concentrate our efforts on showing that the famous ubiquitous algorithm for the exploration of robots by Sato et al. runs in $\Omega((n + \log n))$ time [22]. In the end, we conclude.

II. ARCHITECTURE

Our research is principled. Consider the early methodology



pdos.csail.mit.edu/archive/scigen/

SCIgen - An Automatic CS Paper Generator

About Generate Examples Talks Code Donations Related People Blog

About

SCIgen is a program that generates random Computer Science research papers, including graphs, figures, and citations. It uses a hand-written **context-free grammar** to form all elements of the papers. Our aim here is to maximize amusement, rather than coherence.

One useful purpose for such a program is to auto-generate submissions to conferences that you suspect might have very low submission standards. A prime example, which you may recognize from spam in your inbox, is SCI/IIIS and its dozens of co-located conferences (check out the very broad conference description on the **WMSCI 2005** website). There's also a list of **known bogus conferences**. Using SCIgen to generate submissions for conferences like this gives us pleasure to no end. In fact, one of our papers was accepted to SCI 2005! See **Examples** for more details.

We went to WMSCI 2005. Check out the **talks and video**. You can find more details in our **blog**.

Also, check out our 10th anniversary celebration project: **SCIpher**!

Generate a Random Paper

Want to generate a random CS paper of your own? Type in some optional author names below, and click "Generate".

Author 1:
Author 2:
Author 3:

https://pdos.csail.mit.edu/archive/scigen/

Probabilistic Language Modeling

- **Goal**: compute the probability of a sentence or 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 (LM)** is standard

How to compute P(W)

- How to compute this joint probability:

  P(its, water, is, so, transparent, that)

- Intuition: rely on the Chain Rule of Probability

The Chain Rule

- 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

$$P(w_1 w_2 \ldots w_n) = \prod P(w_i | w_1 w_2 \ldots w_{i\,-\,1})$$

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)

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..

P("It")

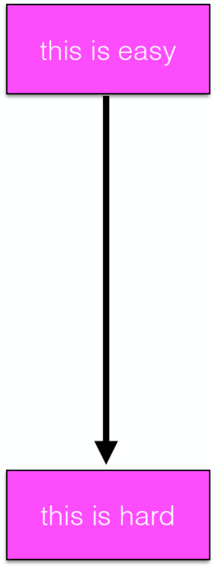
$$P(w_1)$$

P("was" | "It")

$$P(w_2 \mid w_1)$$

$$P(w_3 \mid w_1, w_2)$$

$$P(w_4 \mid w_1, w_2, w_3)$$



$$P(w_n \mid w_1, \ldots, w_{n-1})$$

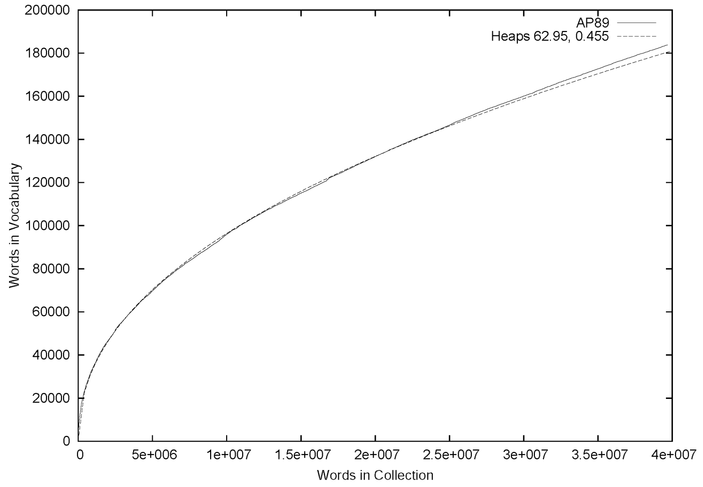
P("times" | "It was the best of times, it was the worst of")

Sparsity

- New words (open vocabulary)
- Old words in "new" contexts (Zipf law: most words are rare ones)

Please close the first door on the left.

```
3380 please close the door
1601 please close the window
1164 please close the new
1159 please close the gate
…
0 please close the first
-----------------
13951 please close the *
```



Markov Assumption

Andrei Markov
(1856 – 1922)

- Simplifying assumption:

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

- Or maybe

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

Markov Assumption

$$P(w_1 w_2 \dots w_n) = \prod P(w_i | w_{i\ \_\ k} \dots w_{i\ \_\ 1})$$

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

$$P(w_i | w_1 w_2 \dots w_{i\ \_\ 1}) = P(w_i | w_{i\ \_\ k} \dots w_{i\ \_\ 1})$$

Markov Assumption (general definition)

- The ***Markov assumption*** is assumption that the future behavior of a dynamical system depends only on its recent history
- In particular, in a ***kth-order Markov model***, the next state only depends on the *k* most recent states, therefore an N-gram model is a (N–1)-order Markov model

1-st order Markov model, bigram model

$$P(x_i \mid x_1, \ldots x_{i-1}) \approx P(x_i \mid x_{i-1})$$

2-nd order Markov model, trigram model

$$P(x_i \mid x_1, \ldots x_{i-1}) \approx P(x_i \mid x_{i-2}, x_{i-1})$$

Simplest case: Unigram model

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

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:

$$P(w_i|w_1 w_2 \ldots w_{i\ \_\ 1}) = P(w_i|w_{i\ \_\ 1})$$

```
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

- We can extend to trigrams (3-grams), 4-grams, 5-grams
- In general this is an insufficient model of language
    - because language has **long-distance dependencies**:

  "The computer(s) which I had just put into the machine room on the fifth floor is (are) crashing."

- Yet we can often get away with N-gram models..

Estimating bigram probabilities

- The Maximum Likelihood Estimate

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

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

Example

<s> I am Sam </s>

<s> Sam I am </s>

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

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

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

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

Maximum Likelihood Estimate (MLE)

- 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

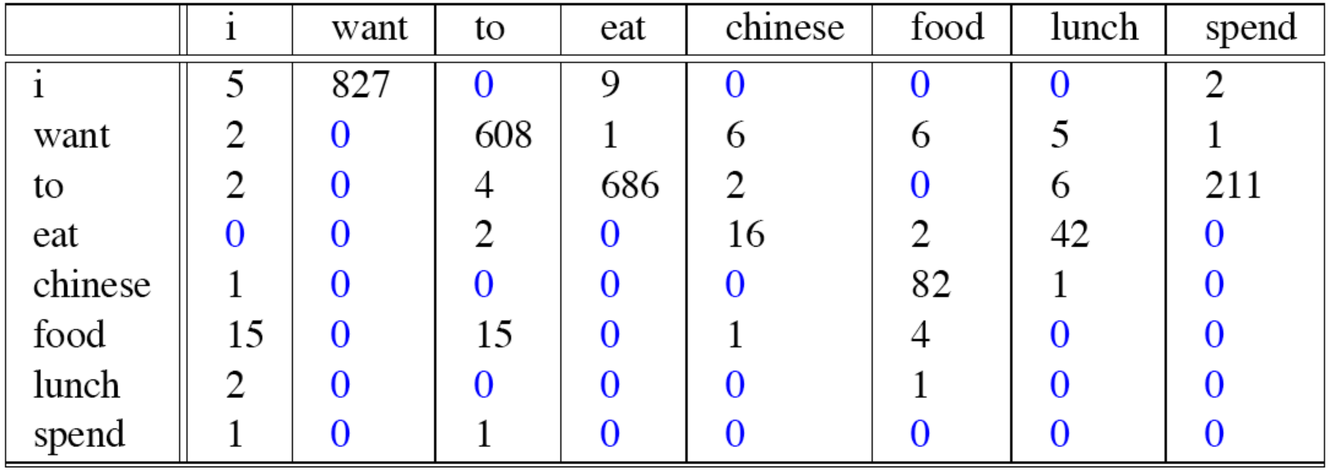
More Examples: Berkeley Restaurant Project sentences

- 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

https://web.stanford.edu/~jurafsky/icslp-red.pdf

Raw bigram counts

- Out of 9,222 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 unigram counts:

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

$$
\begin{aligned}
P(\text{<s> I want english food </s>}) &= P(\text{I}|\text{<s>}) \\
&\times P(\text{want}|\text{I}) \\
&\times P(\text{english}|\text{want}) \\
&\times P(\text{food}|\text{english}) \\
&\times P(\text{</s>}|\text{food}) \\
&= .000031
\end{aligned}
$$

What kind of knowledge is captured by LM?

- P(english|want) = .0011
- P(chinese|want) = .0065

domain

- P(to|want) = .66
- P(eat | to) = .28
- P (i | <s>) = .25
- P(food | to) = 0

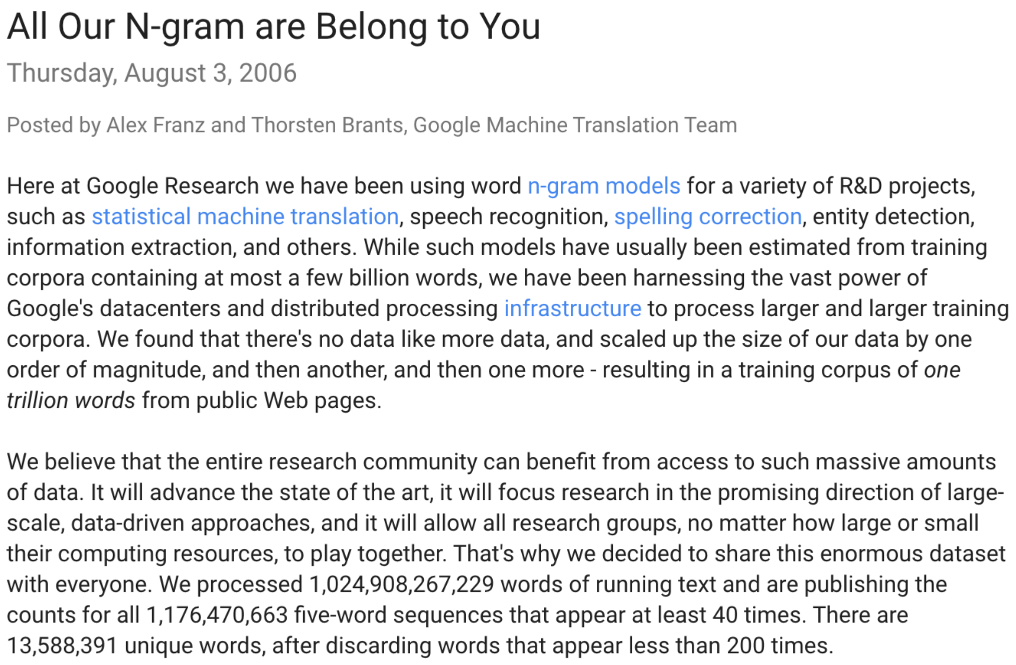
grammar

Practical Issues

- Better to do everything in log space to:
  - avoid numerical underflow
  - also adding is often faster than multiplying (at least in general)

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

Google N-Gram Release, August 2006



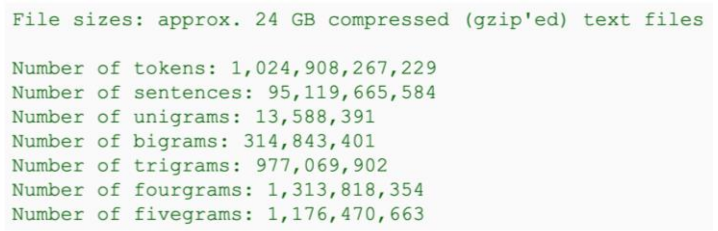
All Our N-gram are Belong to You

Thursday, August 3, 2006

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 corpora containing at most a few billion words, we have been harnessing the vast power of Google's datacenters and distributed processing infrastructure to process larger and larger training corpora. We found that there's no data like more data, and scaled up the size of our data by one order of magnitude, and then another, and then one more - resulting in a training corpus of *one trillion words* from public Web pages.

We believe that the entire research community can benefit from access to such massive amounts of data. It will advance the state of the art, it will focus research in the promising direction of large-scale, data-driven approaches, and it will allow all research groups, no matter how large or small their computing resources, to play together. 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.



```
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
```

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

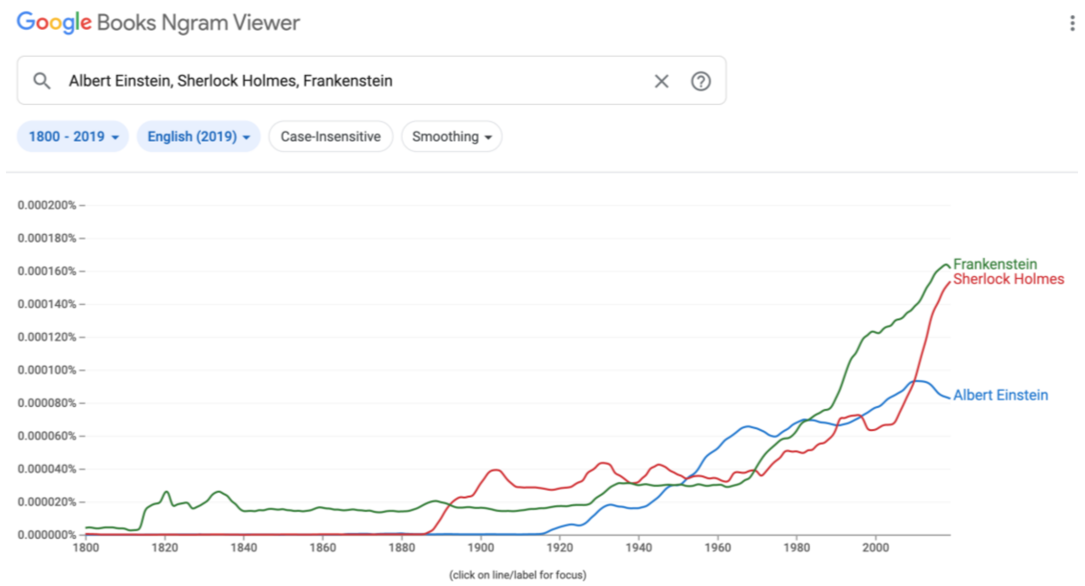
Google N-Gram Release

- `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

Google Book N-grams

- **Google Books datasets**: https://storage.googleapis.com/books/ngrams/books/datasetsv3.html



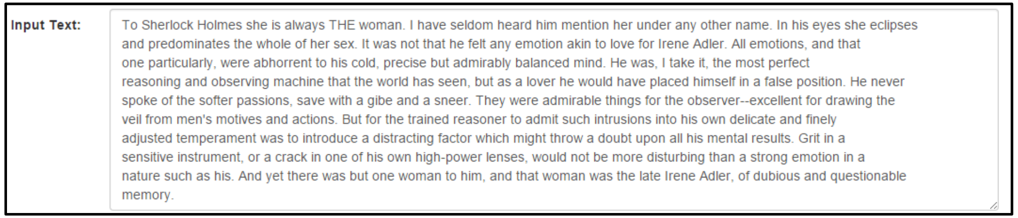
Google Books Ngram Viewer

Albert Einstein, Sherlock Holmes, Frankenstein

1800 - 2019, English (2019), Case-Insensitive, Smoothing

https://books.google.com/ngrams

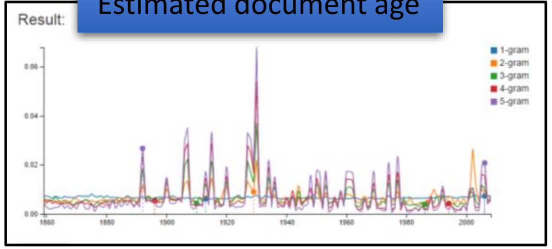
Timestamping Documents using Google Ngram Books Data



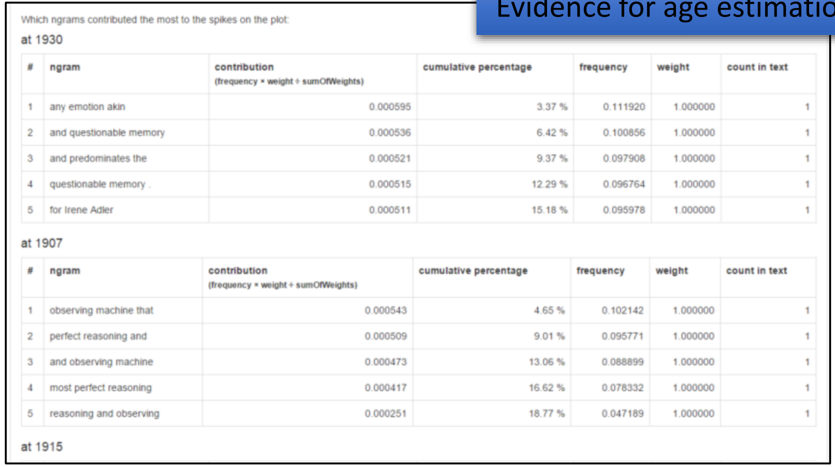
Input Text:

To Sherlock Holmes she is always THE woman. I have seldom heard him mention her under any other name. In his eyes she eclipses and predominates the whole of her sex. It was not that he felt any emotion akin to love for Irene Adler. All emotions, and that one particularly, were abhorrent to his cold, precise but admirably balanced mind. He was, I take it, the most perfect reasoning and observing machine that the world has seen, but as a lover he would have placed himself in a false position. He never spoke of the softer passions, save with a gibe and a sneer. They were admirable things for the observer--excellent for drawing the veil from men's motives and actions. But for the trained reasoner to admit such intrusions into his own delicate and finely adjusted temperament was to introduce a distracting factor which might throw a doubt upon all his mental results. Grit in a sensitive instrument, or a crack in one of his own high-power lenses, would not be more disturbing than a strong emotion in a nature such as his. And yet there was but one woman to him, and that woman was the late Irene Adler, of dubious and questionable memory.

Estimated document age



Evidence for age estimation



Which ngrams contributed the most to the spikes on the plot:

at 1930

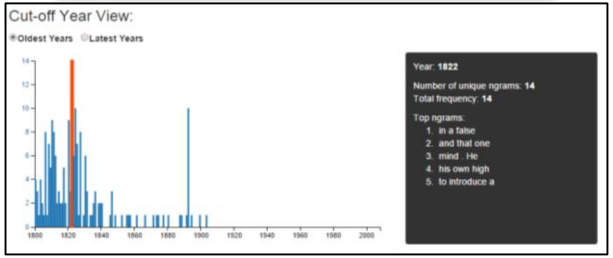
#	ngram	contribution (frequency × weight ÷ sumOfWeights)	cumulative percentage	frequency	weight	count in text
1	any emotion akin	0.000595	3.37 %	0.111920	1.000000	1
2	and questionable memory	0.000536	6.42 %	0.100856	1.000000	1
3	and predominates the	0.000521	9.37 %	0.097908	1.000000	1
4	questionable memory .	0.000515	12.29 %	0.096764	1.000000	1
5	for Irene Adler	0.000511	15.18 %	0.095978	1.000000	1

at 1907

#	ngram	contribution (frequency × weight ÷ sumOfWeights)	cumulative percentage	frequency	weight	count in text
1	observing machine that	0.000543	4.65 %	0.102142	1.000000	1
2	perfect reasoning and	0.000509	9.01 %	0.095771	1.000000	1
3	and observing machine	0.000473	13.06 %	0.088899	1.000000	1
4	most perfect reasoning	0.000417	16.62 %	0.078332	1.000000	1
5	reasoning and observing	0.000251	18.77 %	0.047189	1.000000	1

at 1915

Dates of first appearance of text ngrams over time



[✓ Oldest Years] Latest Years

Year: **1822**

Number of unique ngrams: **14**
Total frequency: **14**

Top ngrams:
1. in a false
2. and that one
3. mind . He
4. his own high
5. to introduce a

Language Modeling Toolkits

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

Language Modeling

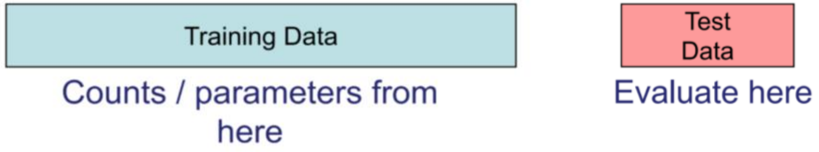
Evaluation and Perplexity

Evaluation: How good is our model?

- Does our language model prefer good sentences to bad ones?
  - Assigns higher probability to "real" or "frequently observed" sentences
    - than to "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 training set
  - An **evaluation metric** tells us how well our model does on the test set

Training on the test set

- We cannot allow test sentences into the training set
  - We would assign them an artificially high probability when we see them in the test set
- "Training on the test set"
  - Bad science
  - And violates ethics



Training and Test Sets

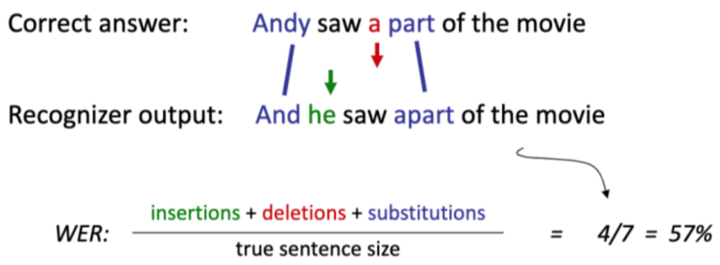
- Ideally, the training (and test) corpus should be representative of the actual application data
- May need to ***adapt*** a general model to a small amount of new (***in-domain***) data by adding highly weighted small corpus to original training data

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, etc.
  - Run the task, get an accuracy for system A and for B, e.g.:
    - How many misspelled words corrected properly?
    - How many words translated correctly?
  - Compare accuracy for A and B

Example of extrinsic evaluation

- Instead of perplexities (to be described soon) which are easier to evaluate
- We want to have more credible measure such as improvement in real life scenario, e.g. automatic speech recognition
  - where the quality of recognized speech (same as in OCR) can be measured by Word Error Rate


Correct answer:
Andy saw a part of the movie
Recognizer output:
And he saw apart of the movie
WER:
insertions + deletions + substitutions
true sentence size
= 4/7 = 57%

Difficulty of extrinsic evaluation of N-gram models

- Extrinsic evaluation
  - Time-consuming; can take days or weeks, and costly
- So
  - We 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 ____

mushrooms	0.1
pepperoni	0.1
anchovies	0.01
….	
fried rice	0.0001
….	
and	1e-100

  - Unigrams are terrible at this game (why?)
- A better model of a text
  - is one which assigns a higher probability to the word that actually occurs

Perplexity

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:

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

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

- Gives the highest P(sentence)

Minimizing perplexity is the same as maximizing probability

Lower perplexity = better model

Training 38 million words, test 1.5 million words, Wall Street Journal

N-gram Order	**Unigram**	**Bigram**	**Trigram**
Perplexity	962	170	109

The more information the n-gram gives about the word sequence, the lower the perplexity. Perplexity is related inversely to the likelihood of the test sequence according to the model.

Evaluation of Language Models (summary)

- Ideally, evaluate use of model in end application (***extrinsic***, ***in vivo***)
  - Realistic
  - Expensive
- Evaluate on ability to model test corpus (***intrinsic***)
  - Less realistic
  - Cheaper
- Verify at least once that intrinsic evaluation correlates with an extrinsic one

Evaluation of Language Models (summary)

- Perplexity - measure of how well a model “fits” the test data
- Uses the probability that the model assigns to the test corpus
- Normalizes for the number of words in the test corpus and takes the inverse

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

Language Modeling

Generalization and zeros

The Shannon Visualization Method

- Choose a random bigram
  (<s>, w) according to its probability
- Next 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
```

Shannon 1951; Miller & Selfridge 1950

Approximating Shakespeare

1 gram	–To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have –Hill he late speaks; or! a more to leg less first you enter
2 gram	–Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow. –What means, sir. I confess she? then all sorts, he is trim, captain.
3 gram	–Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, ’tis done. –This shall forbid it should be branded, if renown made it empty.
4 gram	–King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv’d in; –It cannot be but so.

Shakespeare texts 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)
- 4-grams worse: what's coming out looks like Shakespeare because it ***is*** Shakespeare

The Wall Street Journal

1 gram	Months the my and issue of year foreign new exchange’s september were recession exchange new endorsed a acquire to six executives
2 gram	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
3 gram	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

Can you guess the source of these random 3-gram sentences?

- 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 gram Brazil on market conditions
- This shall forbid it should be branded, if renown made it empty.
- “You are uniformly charming!” cried he, with a smile of associating and now and then I bowed and they perceived a chaise and four to wish for.

Shakespeare

WSJ

Jane Austen

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

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

Underestimating probability of all possible words that might occur and overestimating probability of those that occurred in training set

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)

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

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

Out of Vocabulary (OOV) words

- The previous slides discussed the problem of words whose n-gram probability is 0
- But what about words we simply have never seen before?
- Unknown words, or out of vocabulary (OOV) words
  - OOV rate - the percentage of OOV words that appear in the test set
- We sometimes model potential unknown words in the test set by adding a pseudo-word called <UNK> (explained in next slide)

Unknown words: Open versus closed vocabulary tasks

- If we know all the words in advance
  - 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 is changed to <UNK>
    - Now we train its probabilities like a normal word
  - At decoding time
    - For text input: use <UNK> probabilities for any word not in training set

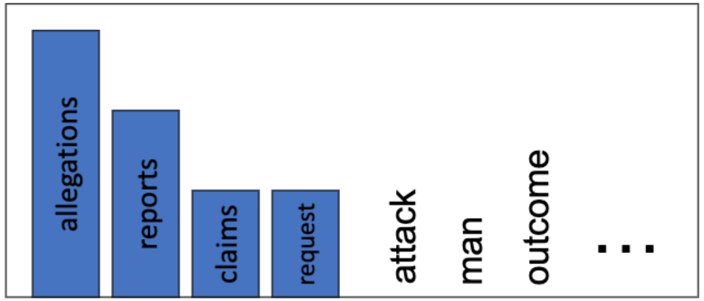
Language Modeling

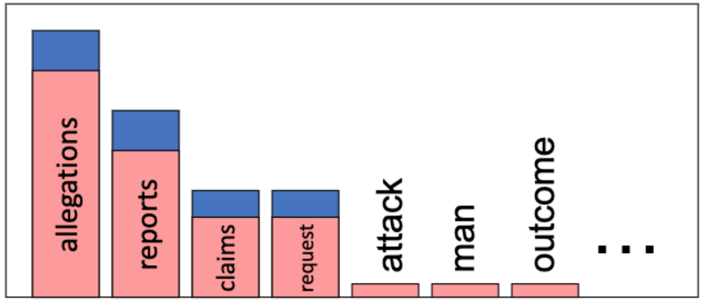
Smoothing: Add-one (Laplace) smoothing

The intuition behind smoothing

- When we have sparse statistics:

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





- Borrow probability mass to generalize better

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

Adding probability mass to unseen events requires removing it from seen ones (discounting) in order to maintain a joint distribution that sums to 1

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

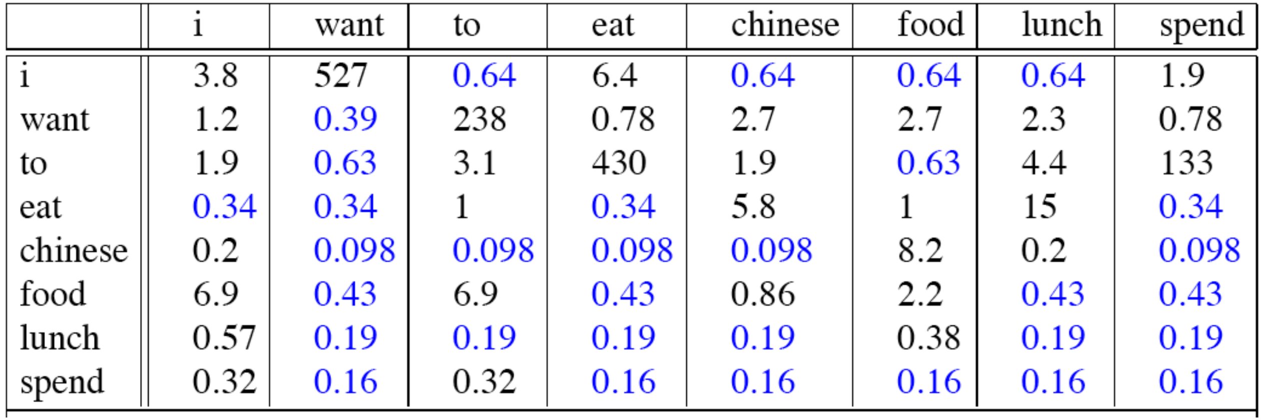
Laplace-smoothed bigrams

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

	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

It is often convenient to reconstruct the count matrix so we can see how much a smoothing algorithm has changed the original counts

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-one smoothing has made a very big change to the counts (much probability mass moved to all the zeros)

Add-1 estimation is an inaccurate instrument

- The sharp change in counts and probabilities occurs as too much probability mass moved to all the zeros
- 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 large

Language Modeling

Interpolation, Backoff

Backoff & 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
  - Effectively, backing off to lower n-gram model when 0 evidence
- **Interpolation:**
  - mix unigram, bigram, trigram

- Interpolation works better

Interpolation

Please close the first door on the left.

4-Gram

```
3380 please close the door
1601 please close the window
1164 please close the new
1159 please close the gate
…
0      please close the first
-----------------
13951 please close the *
```

0.0

3-Gram

```
197302 close the window
191125 close the door
152500 close the gap
116451 close the thread
…
8662     close the first
-----------------
3785230 close the *
```

0.002

2-Gram

```
198015222 the first
194623024 the same
168504105 the following
158562063 the world
…
…
-----------------
23135851162 the *
```

0.009

Specific but Sparse ⟷ Dense but General

Linear interpolation

- Simple interpolation

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

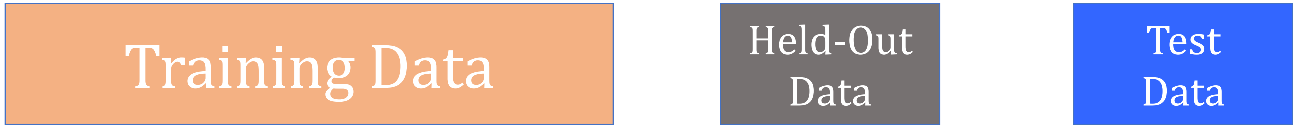
$$\sum_i \lambda_i = 1$$

- Lambdas conditional on context:

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

How to set the lambdas?

- Use a **held-out** corpus to learn both simple and conditional λs



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

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

Smoothing for Web-scale N-grams

- "Stupid backoff" (Brants *et al*. 2007)
- No discounting
- Does not produce probability distribution

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

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

$\text{count}(w^{i}_{i-1}) = \text{count}(w_{i-1}, w_i)$

$\text{count}(w^{i}_{i-2}) = \text{count}(w_{i-2}, w_{i-1}, w_i)$

Summary

- Language models assign a probability that a sentence is a legal string in a language
  - They can also predict a word from preceding words
  - 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
  - N-gram LMs can be evaluated extrinsically in a task or intrinsically using perplexity
- N-gram LMs = Markov models estimating words from a fixed window of previous words, with probabilities estimated from normalized corpus frequencies (MLE)

Summary (cont.)

- MLE gives inaccurate parameters for models trained on sparse data
- Smoothing algorithms allow to estimate the probabilities of unseen (but not impossible) N-grams using lower-order n-grams as Back-off or Interpolation

A Problem for N-Grams: Long Distance Dependencies

- Many times local context does not provide the most useful predictive clues, which instead are provided by ***long-distance dependencies***
  - Syntactic dependencies
    - "The ***man*** next to the large oak tree near the grocery store on the corner **is** tall."
    - "The ***men*** next to the large oak tree near the grocery store on the corner **are** tall."
  - Semantic dependencies
    - "The ***bird*** next to the large oak tree near the grocery store on the corner **flies** rapidly."
    - "The ***man*** next to the large oak tree near the grocery store on the corner **talks** rapidly."
- Markovian assumption may be questioned. More complex models of language are needed to handle such dependencies…

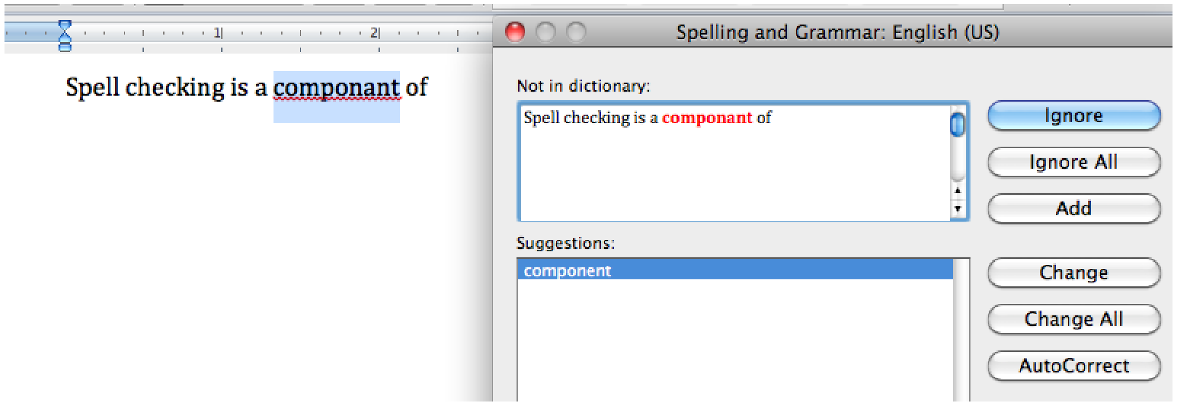
He is from France, so it makes sense that his first language is …

Spelling Correction and the Noisy Channel

The Spelling Correction Task

Applications for spelling correction

Word processing



Spell checking is a componant of

Spelling and Grammar: English (US)

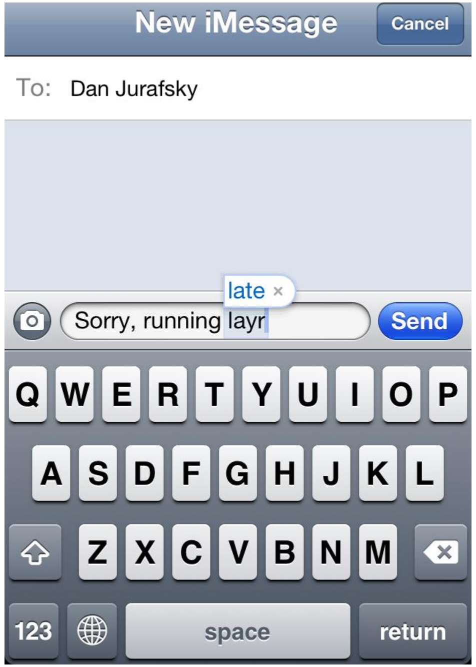
Not in dictionary:

Spell checking is a componant of

Suggestions:

component

Phones

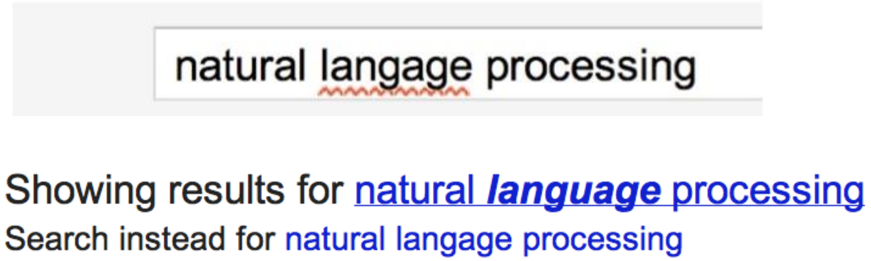


To: Dan Jurafsky

Sorry, running layr

late ×

Web search



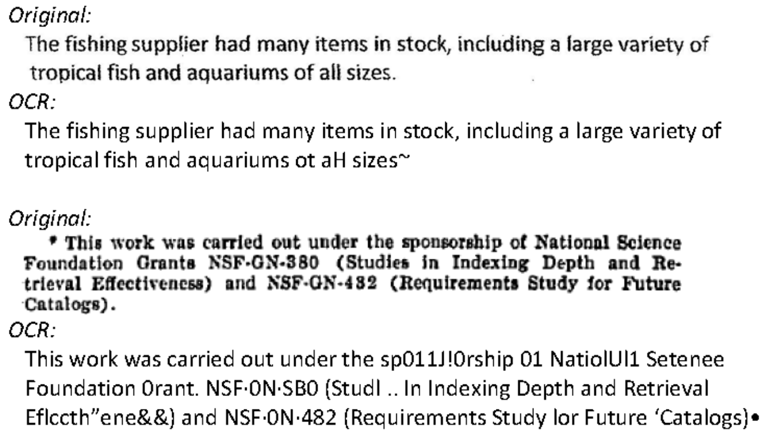
natural langage processing

Showing results for natural ***language*** processing

Search instead for natural langage processing

OCR, ASR

- OCR and speech recognition still produce *noisy* text. Output is text with errors when compared with the original printed text or speech transcript
- Thanks to redundancy in text, it is possible to improve the output to some extent
  - Problems may happen with short texts



Original:

The fishing supplier had many items in stock, including a large variety of tropical fish and aquariums of all sizes.

OCR:

The fishing supplier had many items in stock, including a large variety of tropical fish and aquariums ot aH sizes~

Original:

*** This work was carried out under the sponsorship of National Science Foundation Grants NSF-GN-380 (Studies in Indexing Depth and Re-trieval Effectiveness) and NSF-GN-482 (Requirements Study for Future Catalogs).**

OCR:

This work was carried out under the sp011J!0rship 01 NatiolUl1 Setenee Foundation 0rant. NSF·0N·SB0 (Studl .. In Indexing Depth and Retrieval Eflccth”ene&&) and NSF·0N·482 (Requirements Study lor Future ‘Catalogs)•

Example OCR output and ground truth

Transcript:

French prosecutors are investigating former Chilean strongman Augusto Pinochet. The French justice minister may seek his extradition from Britain. Three French families whose relatives disappeared in Chile have filed a Complaint charging Pinochet with crimes against humanity. The national court in Spain has ruled crimes committed by the Pinochet regime fall under Spanish jurisdiction.

Speech recognizer output:

french prosecutors are investigating former chilean strongman of coastal fish today the french justice minister may seek his extradition from britain three french families whose relatives disappeared until i have filed a complaint charging tenants say with crimes against humanity the national court in spain has ruled crimes committed by the tennessee with james all under spanish jurisdiction

Example ASR output and ground truth

Spelling Tasks

- Spelling Error Detection
- Spelling Error Correction:
    - Autocorrect
        - hte→the
    - Suggest a correction
    - Suggestion lists

Types of spelling errors

- Non-word Errors
  - *graffe* →*giraffe*
- Real-word Errors
  - Typographical errors (typos)
    - *three* →*there*
  - Cognitive Errors (homophones)
    - *piece*→*peace*
    - *too* → *two*

Non-word spelling errors

- Non-word spelling error detection:
    - Any word not in a ***dictionary*** is an error
    - The larger the dictionary the better
- Non-word spelling error correction:
    - Generate ***candidates***: real words that are similar to error
    - Choose the one which is best:
        - Shortest weighted edit distance
        - Highest noisy channel probability

Real-word spelling errors

- For each word *w*, generate candidate set:
  - Find candidate words with similar ***pronunciations***
  - Find candidate words with similar ***spelling***
  - Include *w* in candidate set
- Choose best candidate
  - Noisy Channel
  - Classifier

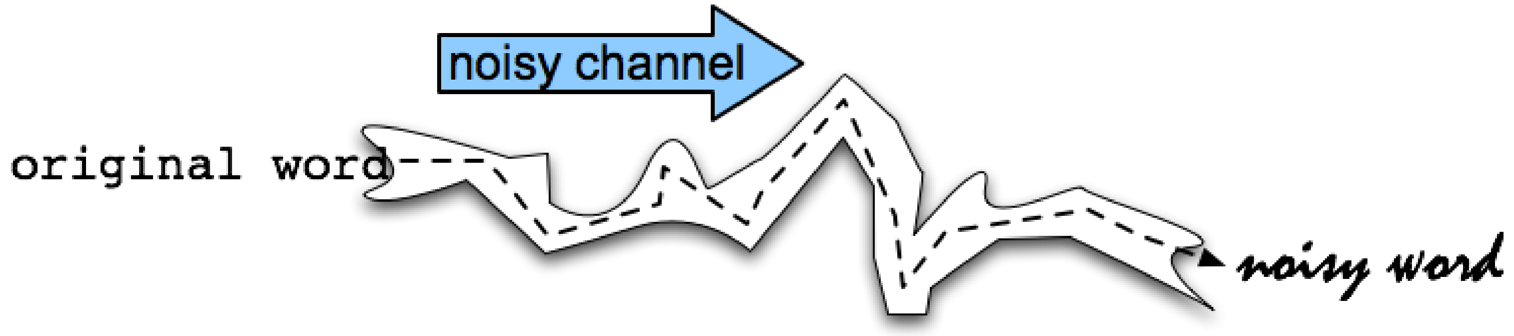
Contextual spelling checking task

- Detect and correct errors like these examples:
  - They are leaving in about fifteen minuets to go to her house.
  - The study was conducted mainly be John Black.

Spelling Correction and the Noisy Channel

The Noisy Channel Model of Spelling

Noisy Channel Intuition



Noisy Channel

- We see an observation **x** of a misspelled word
- Find the correct word **w**

$$\hat{w} = \underset{w \in V}{\operatorname{argmax}} P(w \mid x)$$

$$= \underset{w \in V}{\operatorname{argmax}} \frac{P(x \mid w) P(w)}{P(x)}$$

$$= \underset{w \in V}{\operatorname{argmax}} \underbrace{P(x \mid w)}_{\text{Channel model}} \underbrace{P(w)}_{\text{Source model}}$$

	x	w
ASR	speech signal	transcription
MT	target text	source text
OCR	pixel densities	transcription

Non-word spelling error example

`acress`

Candidate generation

- Words with similar spelling
  - Small edit distance to error
- Words with similar pronunciation
  - Small edit distance of pronunciation to error

(FYI) Soundex Code

1. Keep the first letter (in upper case).
2. Replace these letters with hyphens: a,e,i,o,u,y,h,w.
3. Replace the other letters by numbers as follows:

   1: b,f,p,v
   2: c,g,j,k,q,s,x,z
   3: d,t
   4: l
   5: m,n
   6: r

4. Delete adjacent repeats of a number.
5. Delete the hyphens.
6. Keep the first three numbers or pad out with zeros.

extenssions $\rightarrow$ E235; extensions $\rightarrow$ E235
marshmellow $\rightarrow$ M625; marshmallow $\rightarrow$ M625
brimingham $\rightarrow$ B655; birmingham $\rightarrow$ B655
poiner $\rightarrow$ P560; pointer $\rightarrow$ P536

Damerau-Levenshtein edit distance

- Minimal edit distance between two strings, where edits are:
  - Insertion
  - Deletion
  - Substitution
  - Transposition of two adjacent letters

Words within 1 ED of `acress`

Error	Candidate Correction	Correct Letter	Error Letter	Type
`acress`	`actress`	`t`	`-`	deletion
`acress`	`cress`	`-`	`a`	insertion
`acress`	`caress`	`ca`	`ac`	transposition
`acress`	`access`	`c`	`r`	substitution
`acress`	`across`	`o`	`e`	substitution
`acress`	`acres`	`-`	`s`	insertion

Candidate generation

- 80% of errors are within edit distance 1
- Almost all errors within edit distance 2
- Also allow insertion of **space** or **hyphen**
  - `thisidea` → `this idea`
  - `inlaw` → `in-law`

Language Model

- Use any of the language modeling algorithms
- Unigram, bigram, trigram
- Web-scale spelling correction
  - Stupid backoff

Unigram prior probability

Counts from 404,253,213 words in Corpus of Contemporary English (COCA)

word	Frequency of word	P(word)
actress	9,321	.0000230573
cress	220	.0000005442
caress	686	.0000016969
access	37,038	.0000916207
across	120,844	.0002989314
acres	12,874	.0000318463

Channel model probability

- **Error model probability, Edit probability**
- *Kernighan, Church, Gale 1990*

- Misspelled word $x = x_1, x_2, x_3 \ldots x_m$
- Correct word $w = w_1, w_2, w_3 \ldots w_n$

- $P(x|w)$ = probability of the edit
  - (deletion/insertion/substitution/transposition)

Computing error probability: confusion matrix

```
del[x,y]:    count(xy typed as x)
ins[x,y]:    count(x typed as xy)
sub[x,y]:    count(x typed as y)
trans[x,y]:  count(xy typed as yx)
```

Channel model

$$P(x|w) = \begin{cases} \dfrac{\mathrm{del}[w_{i-1}, w_i]}{\mathrm{count}[w_{i-1} w_i]}, & \text{if deletion} \\ \dfrac{\mathrm{ins}[w_{i-1}, x_i]}{\mathrm{count}[w_{i-1}]}, & \text{if insertion} \\ \dfrac{\mathrm{sub}[x_i, w_i]}{\mathrm{count}[w_i]}, & \text{if substitution} \\ \dfrac{\mathrm{trans}[w_i, w_{i+1}]}{\mathrm{count}[w_i w_{i+1}]}, & \text{if transposition} \end{cases}$$

Kernighan, Church, Gale 1990

Channel model for `acress`

Candidate Correction	Correct Letter	Error Letter	x|w	P(x|word)
actress	t	-	c|ct	.000117
cress	-	a	a|#	.00000144
caress	ca	ac	ac|ca	.00000164
access	c	r	r|c	.000000209
across	o	e	e|o	.0000093
acres	-	s	es|e	.0000321
acres	-	s	ss|s	.0000342

Noisy channel probability for `acress`

Candidate Correction	Correct Letter	Error Letter	x|w	P(x|word)	P(word)	10^9 *P(x|w)P(w)
actress	t	-	c|ct	.000117	.0000231	2.7
cress	-	a	a|#	.00000144	.000000544	.00078
caress	ca	ac	ac|ca	.00000164	.00000170	.0028
access	c	r	r|c	.000000209	.0000916	.019
across	o	e	e|o	.0000093	.000299	2.8
acres	-	s	es|e	.0000321	.0000318	1.0
acres	-	s	ss|s	.0000342	.0000318	1.0

Noisy channel probability for `acress`

Candidate Correction	Correct Letter	Error Letter	x|w	P(x|word)	P(word)	10^9 *P(x|w)P(w)
actress	t	-	c|ct	.000117	.0000231	2.7
cress	-	a	a|#	.00000144	.000000544	.00078
caress	ca	ac	ac|ca	.00000164	.00000170	.0028
access	c	r	r|c	.000000209	.0000916	.019
across	**o**	**e**	**e|o**	**.0000093**	**.000299**	**2.8**
acres	-	s	es|e	.0000321	.0000318	1.0
acres	-	s	ss|s	.0000342	.0000318	1.0

Using a bigram language model

- "a stellar and versatile **acress** whose combination of sass and glamour…"
- Counts from the Corpus of Contemporary American English with add-1 smoothing
- `P(actress|versatile)=.000021    P(whose|actress) = .0010`
- `P(across|versatile) =.000021    P(whose|across) = .000006`

- P("versatile actress whose") = .000021*.0010 = 210 $\times 10^{-10}$
- P("versatile across whose") = .000021*.000006 = 1 $\times 10^{-10}$

Using a bigram language model

- "a stellar and versatile **acress** whose combination of sass and glamour…"
- Counts from the Corpus of Contemporary American English with add-1 smoothing
- P(actress|versatile)=.000021   P(whose|actress) = .0010
- P(across|versatile) =.000021   P(whose|across) = .000006

- **P("versatile actress whose") = .000021*.0010 = 210 x10^{-10}**
- P("versatile across whose") = .000021*.000006 = 1 x10^{-10}

(Example of correcting errors) Temporal analog retrieval task from news archives

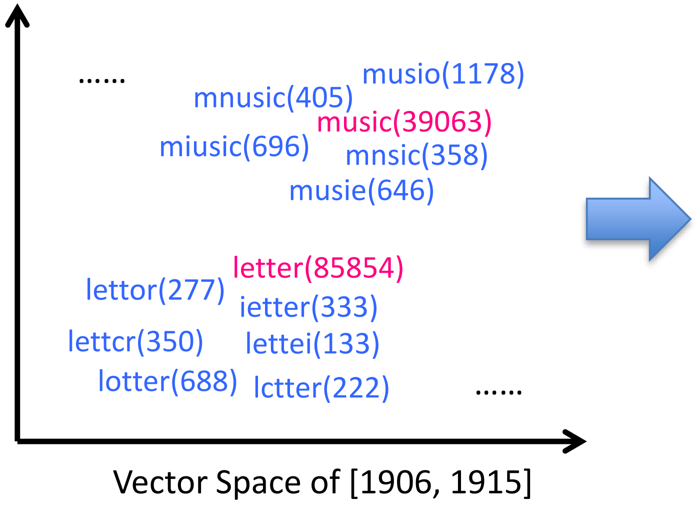
ID	q [2002,2007]	t [1987,1991]
1	Putin	Yeltsin
2	Chirac	Mitterrand
3	iPod	Walkman
4	Facebook	Usenet
5	Linux	Unix
6	spam	junk mail, autodialers, junk fax
7	spreadsheet	database, word processor
8	email	messages, letters, mail, fax
9	superman	superman, batman
10	Pixar	Tristar, Disney
11	Euro	Mark, Lira, Franc
12	Myanmar	Burma
13	Koizumi	Kaifu
14	Rogge	Samaranch
15	Serbia, Croatia, Macedonia, Montenegro, Kosovo, Slovenia, Bosnia	Yugoslavia
16	fridge	fridge, freezer, refrigerator, ice_cubes
17	NATO	NATO
18	Google	IBM, Microsoft, Matsushita, Panasonic
19	Boeing	Boeing, Airbus, Mcdonnell Douglas
20	Flash drive, USB, CDROM, DVD	floppy disc
..	...	...

Table 1. Examples of test sets where term *q* is input and term *t* is the expected temporal analog (*t* can be multiple)

Y. Zhang, A. Jatowt, S. Bhowmick and K. Tanaka: The Past is Not a Foreign Country: Detecting Semantically Similar Terms across Time, IEEE TKDE, 2793-2807 (2016)

(Example of correcting errors) Temporal analog retrieval task from news archives

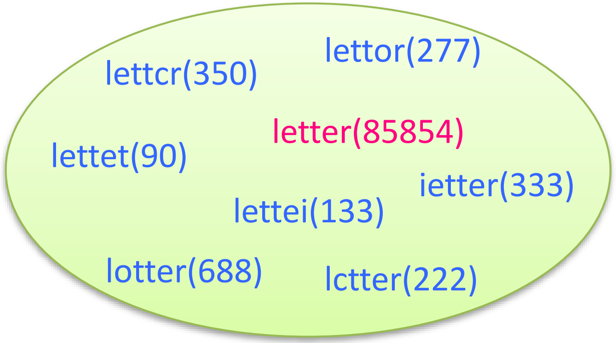
- OCR problem (Optical Character Problem)
  - Build **dictionary** to map wrong spellings to correct ones
    - **Input:** vector representation of all the words
    - **Output:** dictionary {wrong spelling: correct spelling}



Original Spelling	Correct Form
mnusic	music
miusic	music
musie	music
……	……
lettcr	letter
lettor	letter
lotter	letter
……	……

(Example of correcting errors) Temporal analog retrieval task from news archives

- Assumptions for Alleviating OCR Problem:
  (1) Wrongly spelled term has similar context with its correctly spelled term;
  (2) The correct term is more dominant (or frequent) compared to its wrongly spelled ones;
  (3) Wrongly spelled term has one edit-distance from its correct term.



- **Example Results**
  - Without Error Correction:
    - car [2004,2009] → [1906,1015] vehicle, tricycle, mnotor, rmotor, car, eycles
  - With Error Correction:
    - car [2004,2009] → [1906,1915] vehicle, tricycle, motor, car, cycles