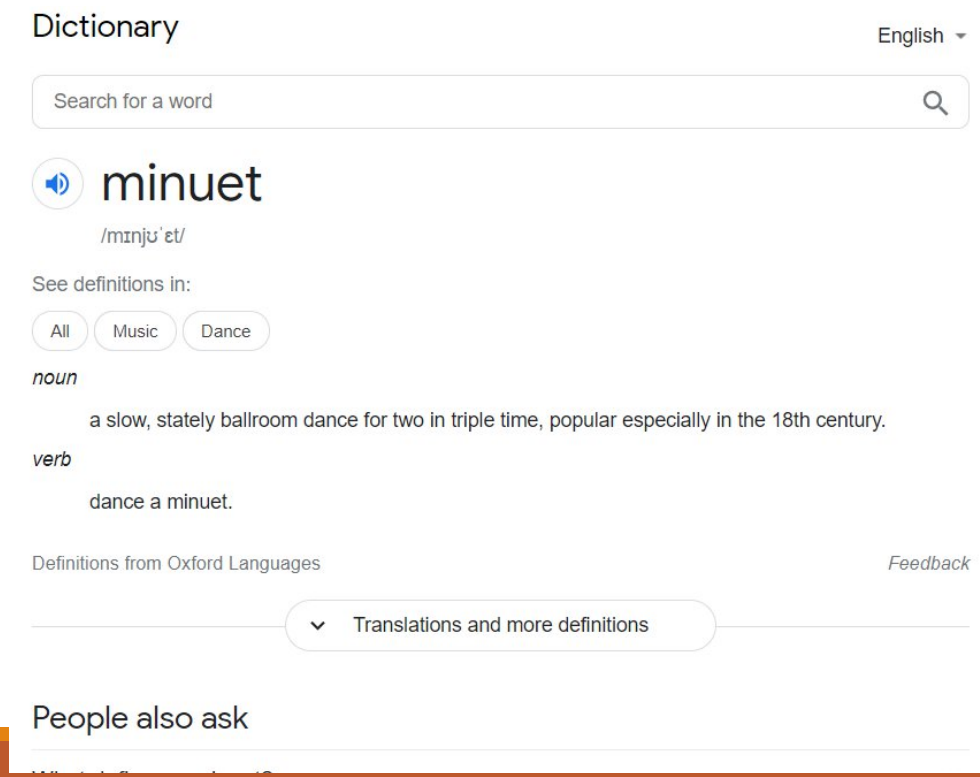
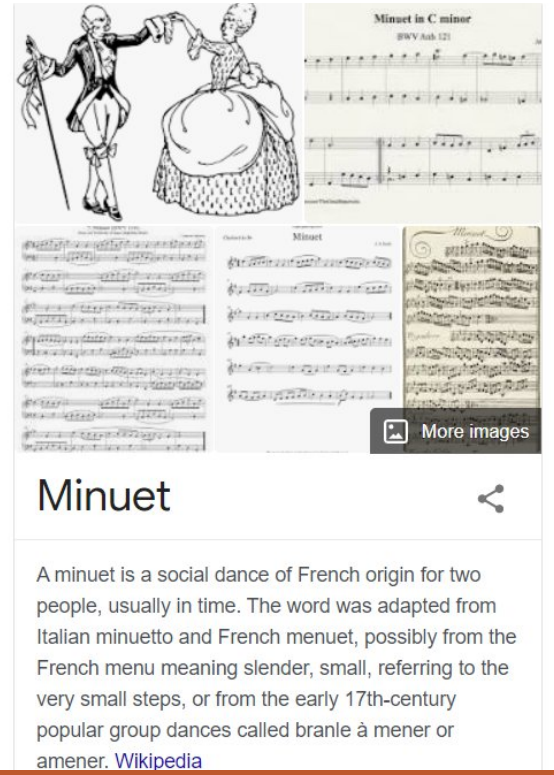
Language Modelling

Context Sensitive Spelling Correction

Consider the sentence:-

The office is about fifteen *minuets* from my house.





Probabilistic Language Models & Application

BASIC GOAL:- Assign probability to a sentence

- **Spelling Correction**

P(about fifteen minutes from) > P(about fifteen minuets from)

- **Speech Recognition**

P(I saw a van) >> P(eyes awe of an)

- **Machine Translation**

Which sentence is more plausible in the target language?

P (high winds tonight) > P (large winds tonight)

- **Completion Prediction**

Please turn of your cell….

Example:- Predictive Text

Predictive text input systems can guess what you are typing and present choices for completion.

Probabilistic Language Modeling

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

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

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

How to compute P(W)

How to compute this joint probability:

- P(about, fifteen, minutes, from)

Intuition: let's rely on the Chain Rule of Probability

The Chain Rule: General

The definition of conditional probabilities

$$P(B|A) = \frac{P(A,B)}{P(A)}$$

Rewriting we get, $P(A,B)=P(B|A)\ P(A)$

More variables:

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

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

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: Probability of words in sentences

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

P("about fifteen minutes from") =

P(about) x P(fifteen | about) x P(minutes | about fifteen) x P(from | about fifteen minutes)

How to estimate these probabilities

Could we just count and divide?

$$P(\text{office} \mid \text{about fifteen minutes from}) = \frac{\textit{Count}(\text{about fifteen minutes from office})}{\textit{Count}(\text{about fifteen minutes from})}$$

Issue:-

Too many possible sentences!

We'll never see enough data for estimating these

Markov Assumption

Andrei Markov

Simplifying Assumption: Use only the previous word

P(office | about fifteen minutes from) ≈ P(office | from)

Or the couple previous words

P(office | about fifteen minutes from) ≈ P(office | minutes from)

Markov Assumption

More Formally: kth order Markov Model

Chain Rule:

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

Using Markov Assumption: only k previous words

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

N-Gram Models

P(office | about fifteen minutes from)

An N-gram model uses only $N-1$ words of prior context.

- Unigram: P(office)
- Bigram: P(office | from)
- Trigram: P(office | minutes from)

Markov model and Language Model

An N-gram model is an $N-1$-order Markov Model

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**."

In most of the application, we can often get away with N-gram models.

Estimating N-grams probabilities

The Maximum Likelihood Estimate (MLE)

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

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

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>

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

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

Raw bigram counts (absolute measure)

Out of 9222 Restaurant 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 (relative measure)

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

Computing Sentence probabilities

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

P(I|<s>) × P(want|I) × P(english|want) x P(food|english) × P(</s>|food)

= .000031

What kinds of knowledge does n-gram represent?

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

Practical Issues

We do everything in log space

- Avoid underflow: multiplying extremely small numbers
- Adding is faster than multiplying

$$p_1 \cdot p_2 \cdot p_3 \cdot p_4 \propto \log p_1 + \log p_2 + \log p_3 + \log p_4$$

- Handling Zeroes using Smoothing

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 having large corpus of text**.

We test the model’s performance on data we haven’t seen.

- A **test set** is an unseen dataset that is different from our training set, totally unused.
- An **evaluation metric** tells us how well our model does on the test set.

Extrinsic evaluation of N-gram models

Best evaluation for comparing models A and B

- Put each model in a task
  - spelling corrector, speech recognizer, machine translation 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 evaluation of N-gram models

Extrinsic evaluation

- Time-consuming; can take days or weeks

So instead

- 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.

Intrinsic Evaluation: Perplexity [Idea come from 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?)

A better model

- is one which assigns a higher probability to the word that actually occurs

- Perplexity is an evaluation metric for N-gram models.
- It is the weighted average number of choices a random variable can make, i.e. the number of possible next words that can follow a given word.

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:

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

Minimizing perplexity is the same as maximizing probability

Perplexity Example

Let's suppose a sentence consisting of N random digits

What is the perplexity of this sentence according to a model that assign P=1/10 to each digit?

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

Lower perplexity = better model

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

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

Unigram perplexity=962!!!

The model is as confused on test data as if it had to choose uniformly and independently among 962 possibilities for each word.

Problems with simple MLE estimate: Zeros

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

- Test set
  … denied the offer
  … denied the loan

Zero probability n-grams

- P(offer | denied the) = 0
- The test set will be assigned a probability 0
- And the perplexity can't be computed

Smoothing

- Smoothing is the task of adjusting the maximum likelihood estimate of probabilities to produce more accurate probabilities.

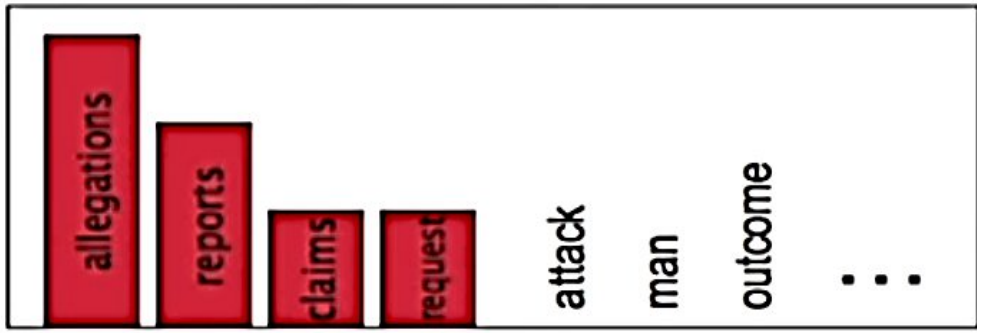
- The name comes from the fact that these techniques tend to make distributions more uniform, by adjusting low probabilities such as zero probabilities upward, and high probabilities downward.

- Smoothing not only prevents zero probabilities, attempts to improves the accuracy of the model as a whole.

Smoothing

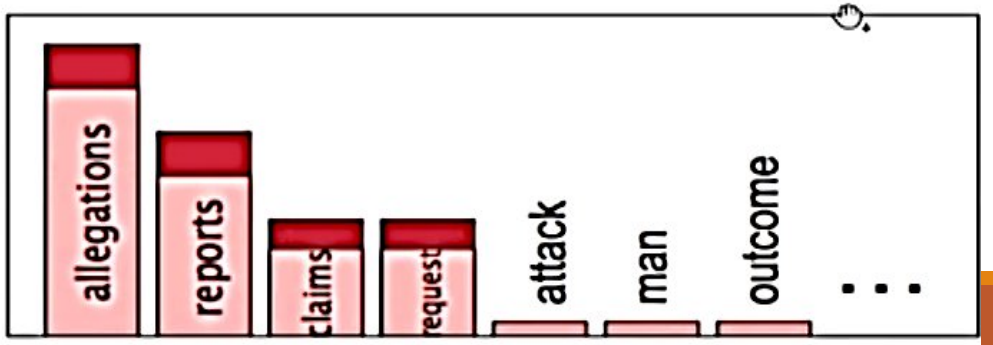
With sparse statistics

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



Laplace Smoothing (Add-one estimate)

- Pretend as if we saw each word (N-gram) one more time that we actually did
- Just add one to all the counts!
- MLE estimate for bigram: $P_{MLE}(w_i|w_{i-1}) = \frac{c(w_{i-1},w_i)}{c(w_{i-1})}$
- Add-1 estimate: $P_{Add-1}(w_i|w_{i-1}) = \frac{c(w_{i-1},w_i)+1}{c(w_{i-1})+V}$

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

For Laplace Smoothing, value of k=1

Effective bigram count ($c^*(w_{n-1}w_n)$)

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

Example

Humpty Dumpty sat on a wall .
Humpty Dumpty had a great fall .

V: the size of the vocabulary

$$P(w_k \mid w_{k-1}) = \frac{C(w_{k-1}w_k)+1}{C(w_{k-1})+V} \qquad P(\text{sat} \mid \text{Dumpty}) = \frac{1+1}{2+10}$$

Comparing with Restaurant Corpus: Smoothed bigram count

	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	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

Berkeley Restaurant Corpus

- V=1446
- Normalize by unigrams:

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

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