


n-Gram Language Models

Context Sensitive Spelling Correction

The office is about fifteen minuets from my house

Context Sensitive Spelling Correction

The office is about fifteen minuets from my house


min·u·et  *noun* \,mɪn-yə-'wet\

: a slow, graceful dance that was popular in the 17th and 18th centuries

: the music for a minuet

Context Sensitive Spelling Correction

The office is about fifteen minuets from my house

min·u·et  *noun* \,min-yə-'wet\

: a slow, graceful dance that was popular in the 17th and 18th centuries

: the music for a minuet

Use a Language Model

$P(\text{about fifteen } \mathbf{minutes} \text{ from}) > P(\text{about fifteen } \mathbf{minuets} \text{ from})$

Speech Recognition

- $P(\text{I saw a van}) \gg P(\text{eyes awe of an})$

Probabilistic Language Models: Applications

Speech Recognition

- $P(\text{I saw a van}) \gg P(\text{eyes awe of an})$

Machine Translation

Which sentence is more plausible in the target language?

- $P(\text{high winds}) > P(\text{large winds})$

Probabilistic Language Models: Applications

Speech Recognition

- $P(\text{I saw a van}) \gg P(\text{eyes awe of an})$

Machine Translation

Which sentence is more plausible in the target language?

- $P(\text{high winds}) > P(\text{large winds})$

Other Applications

- Context Sensitive Spelling Correction
- Natural Language Generation
- ...

- Language model also supports predicting the completion of a sentence.
 - ▶ Please turn off your cell ...
 - ▶ Your program does not ...

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

- **Goal:** Compute the probability of a sentence or sequence of words:

$$P(W) = P(w_1, w_2, w_3, \dots, w_n)$$

- **Goal:** Compute the probability of a sentence or sequence of words:

$$P(W) = P(w_1, w_2, w_3, \dots, w_n)$$

- **Related Task:** probability of an upcoming word:

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

- **Goal:** Compute the probability of a sentence or sequence of words:

$$P(W) = P(w_1, w_2, w_3, \dots, w_n)$$

- **Related Task:** probability of an upcoming word:

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

- A model that computes either of these is called a **language model**

Computing $P(W)$

How to compute the joint probability

$P(\text{about, fifteen, minutes, from})$

Computing $P(W)$

How to compute the joint probability

$P(\text{about, fifteen, minutes, from})$

Basic Idea

Rely on the Chain Rule of Probability

Conditional Probabilities

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

Conditional Probabilities

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

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

The Chain Rule

Conditional Probabilities

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

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

More Variables

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

The Chain Rule

Conditional Probabilities

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

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

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

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

Probability of words in sentences

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

$P(\text{"about fifteen minutes from"}) =$

Probability of words in sentences

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

$P(\text{"about fifteen minutes from"}) =$

$P(\text{about}) \times P(\text{fifteen} \mid \text{about}) \times P(\text{minutes} \mid \text{about fifteen}) \times P(\text{from} \mid \text{about fifteen minutes})$

Estimating These Probability Values

Count and divide

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

The counts can be estimated from a large corpus of text.

Estimating These Probability Values

Count and divide

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

What is the problem

We may never see enough data for estimating these

Markov Assumption

Simplifying Assumption: Use only the previous word

$P(\text{office} \mid \text{about fifteen minutes from}) \approx P(\text{office} \mid \text{from})$

Markov Assumption

Simplifying Assumption: Use only the previous word

$P(\text{office} \mid \text{about fifteen minutes from}) \approx P(\text{office} \mid \text{from})$

Or the couple previous words

$P(\text{office} \mid \text{about fifteen minutes from}) \approx P(\text{office} \mid \text{minutes from})$

We can hope to get many more occurrences of these shorter phrases in a large corpus, which will help us better estimate the probabilities.

Markov Assumption

More Formally: kth order Markov Model

Chain Rule:

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

Markov Assumption

More Formally: kth order Markov Model

Chain Rule:

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

Using Markov Assumption: only k previous words

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

Markov Assumption

More Formally: k th order Markov Model

Chain Rule:

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

Using Markov Assumption: only k previous words

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

We approximate each component in the product

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

P(office | about fifteen minutes from)

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

P(office | about fifteen minutes from)

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

- Unigram: $P(\text{office})$
- Bigram: $P(\text{office} | \text{from})$
- Trigram: $P(\text{office} | \text{minutes from})$

Unigram model: no context is used

P(office | about fifteen minutes from)

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

- Unigram: $P(\text{office})$
- Bigram: $P(\text{office} | \text{from})$
- Trigram: $P(\text{office} | \text{minutes from})$

Markov model and Language Model

N-Gram Models

P(office | about fifteen minutes from)

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

- Unigram: $P(\text{office})$
- Bigram: $P(\text{office} | \text{from})$
- Trigram: $P(\text{office} | \text{minutes from})$

Markov model and Language Model

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

- We can extend to trigrams, 4-grams, 5-grams
- In general, an insufficient model of language:

- We can extend to trigrams, 4-grams, 5-grams
- In general, an insufficient model of language:
language has long-distance dependencies:
“The computer which I had just put into the machine room on the fifth floor **crashed**.”

For any reasonable value of n , a n -gram language model cannot completely model a natural language.

- We can extend to trigrams, 4-grams, 5-grams
- In general, an insufficient model of language:
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 applications, we can get away with N-gram models

How to estimate the n-gram probabilities from a large text corpus?

Estimating N-grams probabilities

Maximum Likelihood Estimate

Value that makes the observed data the “most probable”

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

Estimating N-grams probabilities

Maximum Likelihood Estimate

Value that makes the observed data the “most probable”

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

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

An Example

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

<s>I am here </s>

<s>who am I </s>

<s>I would like to know </s>

An Example

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

<s>I am here </s>

<s>who am I </s>

<s>I would like to know </s>

Estimating bigrams

$P(I | <s>) =$

$P(</s> | \text{here}) =$

$P(\text{would} | I) =$

$P(\text{here} | \text{am}) =$

$P(\text{know} | \text{like}) =$

An Example

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

<s>I am here </s>

<s>who am I </s>

<s>I would like to know </s>

Estimating bigrams

$$P(I | <s>) = 2/3$$

$$P(</s> | \text{here}) = 1$$

$$P(\text{would} | I) = 1/3$$

$$P(\text{here} | \text{am}) = 1/2$$

$$P(\text{know} | \text{like}) = 0$$

**So, given a corpus, we can easily estimate such probabilities.
The larger the corpus, the better will be the probability estimates.**

Bigram counts from 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

Some frequent bigrams:

I want

want to

to eat

eat lunch

to spend

Computing bigram probabilities

Normlize by unigrams

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

To compute the bigram probabilities, we also need the frequencies of each unigram.

$$P(\text{want} | i) = c(i \text{ want}) / c(i) = 827 / 2533 = 0.33$$

Computing bigram probabilities

Normlize by unigrams

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

Bigram Probabilities

	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

Now, we know how to estimate bigram probabilities from a given text corpus.

Next, how to estimate the probability of a sentence / phrase?

$$P(<s> \text{ I want english food } </s>)$$

$$= P(I \mid <s>) \times P(\text{want} \mid I) \times P(\text{english} \mid \text{want}) \times P(\text{food} \mid \text{english}) \times P(</s> \mid \text{food})$$

Computing Sentence Probabilities

$P(< s > \text{ I want english food } < / s >)$

$$\begin{aligned} &= P(I \mid < s >) \times P(\text{want} \mid I) \times P(\text{english} \mid \text{want}) \times P(\text{food} \mid \text{english}) \times P(< / s > \mid \text{food}) \\ &= 0.000031 \end{aligned}$$

Using the chain rule of probabilities, with a first order Markov assumption.

Now that we know how to compute the probability of a sentence / phrase, we can do many practical tasks:

Query completion

Predicting the next word as one types

Deciding which translation is more ‘natural’

Context-sensitive spelling correction

What knowledge does n -gram represent?

- $P(\text{english}|\text{want}) = .0011$
- $P(\text{chinese}|\text{want}) = .0065$
- $P(\text{to}|\text{want}) = .66$
- $P(\text{eat} | \text{to}) = .28$
- $P(\text{food} | \text{to}) = 0$
- $P(\text{want} | \text{spend}) = 0$
- $P(i | <s>) = .25$

For the given corpus (food), Chinese is more desirable than English

Usually, a verb comes after 'to', not a noun

Two verbs, e.g., 'want' and 'spend' do not come consecutively

Many sentences start with 'I'

Multiplication of many probability values can lead to underflow (can lead to zero values for small probabilities)

Everything in log space

- Avoids underflow

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

Handling zeros

Use smoothing

Suppose a sentence contains a bigram that never occurs in the corpus (from which the language model is learned)

SRILM

<http://www.speech.sri.com/projects/srilm/>

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

Example from the 4-gram data

serve as the inspector 66
serve as the inspiration 1390
serve as the installation 136
serve as the institute 187
serve as the institution 279
serve as the institutional 461

Google books Ngram Data

Google books Ngram Viewer

Graph these comma-separated phrases: ☐ case-insensitive

between and from the corpus with smoothing of [Search lots of books](#)

