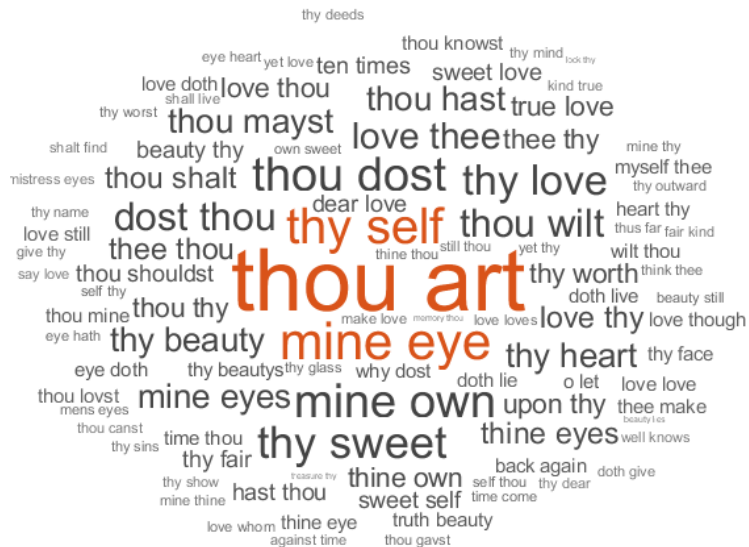


N-gram language models



my alarm clak did not
my alarm code soil rout
 circle raid hot
 shute risk riot
 clock visit not
 did must

wake me up this morning
wake me up thai moving
 taxi having
 this running
 tier morning
 loving

Prof Dr Marko Robnik-Šikonja

Contents

- language models
- n-grams (still used in the evaluation measures)

mostly based on Jurafsky & Martin, 3rd edition,
read Chapter 3.1 – 3.4

Probabilistic Language Models

- The goal: assign a probability to a sentence

- Machine Translation:

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

- Spell Correction

Why?

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

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

- Speech Recognition

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

- + Summarization, question-answering, etc., etc.!!

Probabilistic Language Modeling

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

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

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

- A better name would be: **the grammar model**
- But **language model** or **LM** is standard

How to statistically compute $P(W)$

- How to compute this joint probability:

$P(\text{its, water, is, so, transparent, that})$

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

Reminder: The Chain Rule

- Recall the definition of conditional probabilities

$$p(\mathbf{B} | \mathbf{A}) = \mathbf{P}(\mathbf{A}, \mathbf{B}) / \mathbf{P}(\mathbf{A}) \quad \text{Rewriting: } \mathbf{P}(\mathbf{A}, \mathbf{B}) = \mathbf{P}(\mathbf{A})\mathbf{P}(\mathbf{B} | \mathbf{A})$$

- 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

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

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

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

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?

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

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

Markov Assumption



Andrei Markov

- The memory is short
- First order Markov assumption

$P(\text{the | its water is so transparent that}) \gg P(\text{the | that})$

- The second order Markov assumption

$P(\text{the | its water is so transparent that}) \gg P(\text{the | transparent that})$

Using Markov assumption of order k

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

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

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

Simplest case: Unigram model

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

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:

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

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

Estimating bigram probabilities

- The Maximum Likelihood Estimate

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

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

An example

<s> I am Sam </s>

<s> Sam I am </s>

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

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

$$P(\mathbf{I} | \langle \mathbf{s} \rangle) = \frac{2}{3} = .67$$

$$P(\mathbf{Sam} | \langle \mathbf{s} \rangle) = \frac{1}{3} = .33$$

$$P(\mathbf{am} | \mathbf{I}) = \frac{2}{3} = .67$$

$$P(\langle \mathbf{s} \rangle | \mathbf{Sam}) = \frac{1}{2} = 0.5$$

$$P(\mathbf{Sam} | \mathbf{am}) = \frac{1}{2} = .5$$

$$P(\mathbf{do} | \mathbf{I}) = \frac{1}{3} = .33$$

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(s) which I had just put into the machine room on the fifth floor is (are) crashing.”
- N-gram models are better in English than in Slovene and many other languages. Why?

Example: Restaurant 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:

- Result:

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

	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

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

What kinds of knowledge bigram LM contains?

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

Practical Issues

- We do everything in log space instead of probability space
 - Avoids underflow
 - (also adding is faster than multiplying)

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

Google Book N-Grams, 2006

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 indispensable 40
- serve as the individual 234

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

Language Modeling Tools

- are ngram language models still useful?
- yes, e.g., in speech processing
- mostly replaced by neural LMs
- many variants of adapted neural LMs exist, e.g., word2vec, fastText, ELMo, BERT

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.
- Two types of evaluation
 - intrinsic (internal)
 - extrinsic (external, on a downstream task)

Extrinsic evaluation of N-gram models

- Best evaluation for comparing models A and B
 - Use 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.

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 _____

- 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

mushrooms 0.1

pepperoni 0.1

anchovies 0.01

....

fried rice 0.0001

....

and 1e-100

Perplexity

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

- Gives the highest $P(\text{sentence})$

Perplexity is the inverse probability of the test set, normalized by the number of words:

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

Chain rule:

$$PP(W) = \sqrt[N]{\prod_{i=1}^N \frac{1}{P(w_i | w_1 \dots w_{i-1})}}$$

For bigrams:

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

Minimizing perplexity is the same as maximizing probability

Perplexity example

- Let us suppose a sentence consisting of random digits
- What is the perplexity of this sentence according to a model that assign $P=1/10$ to each digit?

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

Lower perplexity = better model

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

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

The Shannon Visualization Method

- Choose a random bigram

(`<s>`, `w`) according to its probability

`<s>` I

I want

want to

to eat

eat Chinese

Chinese food

food `</s>`

- Now choose a random bigram
(`w`, `x`) according to its probability

- And so on until we choose `</s>`

- Then string the words together

I want to eat Chinese food

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 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)
- Quadrigrams are even 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

What is 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.

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 (outdated) generalization: Zeros!
 - Things that don't ever occur in the training set
 - But occur in the test set
- In practice, we use neural language models