

Statistical language models

Johan Boye, KTH

March 23, 2020

Statistical properties of languages

We will view words as *random events* generated by some *random process*.

Some high-level observations:

initially linear then after that will have repetition of words

Heap's law: The number of unique words in a text is proportional to the square root of the number of tokens

Zipf's law: The number of occurrences of a word is inversely proportional to the word's rank in a frequency table.

Statistical language model

A (statistical) **language model** predicts the probability of the next word, based on the preceding words.

■ *He wrote a letter to his _____.*

Applications of language models

- Word prediction

- *I'm going _____*

↙

to
for
on

- Spelling correction

- *Flights form Boston.*

- Speech recognition

- $P(\text{"recognize speech"}) > P(\text{"wreck a nice beach"})$

- Translation

- $P(\text{"tall building"}) > P(\text{"high building"})$

- ... and many more

Some notation

$\mathbf{P}(\mathbf{w})$ = the probability of the word w .

- Picking a random word from a text, what is the probability that that word will be w ?

$\mathbf{P}(\mathbf{w}_1 \mathbf{w}_2 \dots \mathbf{w}_n)$ = the probability of the sequence $w_1 w_2 \dots w_n$.

- Picking a random sequence of i words from a text, what is the probability that that sequence will be $w_1 w_2 \dots w_n$?

Chain rule for probabilities

The *chain rule* rewrites a joint probability into a product of conditional probabilities.

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

In general:

$$P(A_1, \dots, A_n) = \prod_{i=1}^n P(A_i | A_1, \dots, A_{i-1})$$

For example: $P(\text{I like ants}) = P(\text{ants} | \text{I like})P(\text{like} | \text{I})P(\text{I})$

How do we estimate the probabilities $P(w_i | w_1, \dots, w_{i-1})$?

e.g $P(\text{ants} | \text{I like})$

Maximum likelihood estimation (MLE)

Count word strings in a text corpus, compute fraction:

e.g $P(\text{ants} \mid \text{I like})$

$$P(w_i | w_1, \dots, w_{i-1}) = \frac{c(w_1, \dots, w_i)}{c(w_1, \dots, w_{i-1})} \quad c(\text{I like ants}) / c(\text{I like})$$

In particular:

$$P(w) = \frac{c(w)}{N}, \text{ where } N \text{ is the number of tokens in the corpus}$$

$$P(\text{ants}) = c(\text{ants}) / N$$

Problems with Maximum likelihood estimation

Data sparsity is a problem for the straightforward MLE method.

E.g.
$$\frac{c(\text{"Strange things have happened"})}{c(\text{"Strange things have"})}$$

- What if there are no occurrences of “Strange things have happened”? *Probability = 0*
- What if there are no occurrences of “Strange things have”?
Probability = Undefined

Regardless of how much data you have, this will happen over and over.

Maximum likelihood estimation (MLE)

Count word strings in a text corpus, compute fraction:

$$P(w_i | w_1, \dots, w_{i-1}) = \frac{c(w_1, \dots, w_i)}{c(w_1, \dots, w_{i-1})}$$

In particular:

$$P(w) = \frac{c(w)}{N}, \text{ where } N \text{ is the number of tokens in the corpus}$$

Example

- Corpus with 19 tokens (including punctuation):

I live in Boston.

I like ants.

Ants like honey.

Therefore I like honey too.

- What is $P(\text{like}|\text{I})$ based on the above corpus?
- What is $P(\text{honey}|\text{I like})$?
- What is $P(\text{Boston}|\text{I like})$?

Example

- Corpus with 19 tokens (including punctuation):

I live in Boston.

I like ants.

Ants like honey.

Therefore I like honey too.

- What is $P(\text{like}|\text{I})$ based on the above corpus? $2/3$
- What is $P(\text{honey}|\text{I like})$? $1/2$
- What is $P(\text{Boston}|\text{I like})$? 0

Markov assumption

Assume that a word only depends on the previous n words.

$$P(w_i | w_1, \dots, w_{i-1}) = P(w_i | w_{i-n}, \dots, w_{i-1})$$

$n = 0$ (*unigram* model)

$$P(\text{I have a unicorn}) = P(\text{I})P(\text{have})P(\text{a})P(\text{unicorn})$$

$n = 1$ (*bigram* model)

$$\begin{aligned} P(\text{I have a unicorn}) = \\ P(\text{I})P(\text{have}|\text{I})P(\text{a}|\text{have})P(\text{unicorn}|\text{a}) \end{aligned}$$

In general: An n -gram model takes the $n - 1$ preceding words into account. Typically such models are estimated from a large corpus.

Counting bigrams

... I would like to know your plans for ...

I would: 1

Counting bigrams

... I would like to know your plans for ...

I would: 1 would like: 1

Counting bigrams

... I would like to know your plans for ...

I would: 1 would like: 1 like to: 1

Bigram probabilities

The word *I* occurred 1000 times...

- ... 20 times, the next word was *like*
- ... 200 times, the next word was *am*
- ... 100 times, the next word was *have*
- etc.

From the counts, we can estimate *bigram probabilities*:

- $P(\textit{like}|\textit{I}) = 0.02$ 20/1000
- $P(\textit{am}|\textit{I}) = 0.2$ 200/1000
- $P(\textit{have}|\textit{I}) = 0.1$ 100/1000
- etc.

Trigram probabilities

The sequence *I like* occurred 20 times...

- ... 5 times, the next word was *to*
- ... 4 times, the next word was *that*
- ... 1 time, the next word was *apples*
- etc.

From the counts, we can estimate *trigram probabilities*:

- $P(\textit{to}|\textit{I like}) = 0.25$
- $P(\textit{that}|\textit{I like}) = 0.2$
- $P(\textit{apples}|\textit{I like}) = 0.05$
- etc.

Example

- Corpus with 19 tokens (including punctuation):
I live in Boston.
I like ants.
Ants like honey.
Therefore I like honey too.
- What is $P(\text{I like Boston})$ using a unigram model based on the above corpus?
- What is $P(\text{I like honey})$ using a bigram model?
- What is $P(\text{I like Boston})$ using a bigram model?

Example

- Corpus with 19 tokens (including punctuation):

I live in Boston.

I like ants.

Ants like honey.

Therefore I like honey too.

- What is $P(\text{I like Boston})$ using a unigram model based on the above corpus? $(3/19)(3/19)(1/19)$
- What is $P(\text{I like honey})$ using a bigram model?
 $(3/19)(2/3)(2/3)$
- What is $P(\text{I like Boston})$ using a bigram model?
 $(3/19)(2/3)(0/3) = 0$

Zero-probabilities, and what to do about them

A problem with n -gram models is that many sensible word sequences will have zero-probabilities.

3 techniques to solve this problems:

- Smoothing
- Backoff
- Linear interpolation

Laplace smoothing

Smoothing: Transfer some of the probability mass from the seen sequences to the unseen sequences.

Easiest variant: *Laplace (add-one)* smoothing.

- Previously (Maximum Likelihood Estimation):

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

- Now: (MLE with Laplace smoothing):

$$P_{Laplace}(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i) + 1}{c(w_{i-1}) + V}$$

where V is the size of the vocabulary (number of unique words).

Laplace smoothing (example)

- Corpus with 19 tokens (including punctuation):

I live in Boston.

I like ants.

Ants like honey.

Therefore I like honey too.

- What is $P(\text{I like honey})$ using a bigram model with Laplace smoothing?
- What is $P(\text{I like Boston})$ using a bigram model with Laplace smoothing?

Laplace smoothing (example)

- Corpus with 19 tokens (including punctuation):

I live in Boston.

I like ants.

Ants like honey.

Therefore I like honey too.

- What is $P(\text{I like honey})$ using a bigram model with Laplace smoothing? $(\frac{3+1}{19+10})(\frac{2+1}{3+10})(\frac{2+1}{3+10})$
- What is $P(\text{I like Boston})$ using a bigram model with Laplace smoothing? $(\frac{3+1}{19+10})(\frac{2+1}{3+10})(\frac{0+1}{3+10})$

Advanced smoothing

Laplace smoothing turns out to be too crude for many applications but useful for other context e.g. text classification

- ...but it is useful in some machine learning contexts (more on this later in the course)

More sophisticated smoothing methods exist, e.g. the *interpolated Kneser-Ney* method. described in the book

We won't cover these in the course.

If a particular n -gram is not present in the training corpus, then use $(n - 1)$ -grams instead.

If the $(n - 1)$ -grams do not exist either, then use $(n - 2)$ -grams, etc.

Example: In a bigram model built from *Moby Dick*, $P(\text{I like Boston}) = 0$, since “like Boston” does not appear in the book. Then compute the probability as:

$$P(\text{I like Boston}) = P(\text{I})P(\text{like}|\text{I})P(\text{Boston})$$

Note that this computation will most likely underestimate the actual probability (why?).

Linear interpolation

will be used in the assignment[

Estimate $\hat{P}(w_i|w_{i-1})$ as

$$\lambda_1 \overset{\text{bigram probability}}{P(w_i|w_{i-1})} + \lambda_2 \overset{\text{unigram probability}}{P(w_i)} + \lambda_3$$

to ensure that it is not zero

where the λ s sum to 1.

$$\sum_i \lambda_i = 1$$

Typically $\lambda_1 = 0.99$, $\lambda_2 = 0.01 - \lambda_3$, $\lambda_3 = 10^{-6}$

This idea naturally extends to 3-grams, etc.

Assignment 2, problem 1

Write a program that computes all bigram probabilities from a given (training) corpus, and stores it in a file. Practical issue:

We will use log-probabilities rather than probabilities

- ... using the natural logarithm (because it's simpler)
- -11.99225 rather than 0.0000061919939907 .

This is to avoid underflow when computing with very small probabilities.

... and we can add rather than multiply

- $\log(p_1 \times \dots \times p_n) = \log(p_1) + \dots + \log(p_n)$

Probability of a sentence

When calculating the probability of a sentence, it is useful to include punctuation or boundary symbols, e.g.

boundary symbol

$$P(\langle b \rangle \text{ I like Boston } \langle b \rangle) = \\ P(\text{I}|\langle b \rangle) \times P(\text{like}|\text{I}) \times P(\text{Boston}|\text{like}) \times P(\langle b \rangle|\text{Boston})$$

- $P(\text{I}|\langle b \rangle)$: Probability that “I” will be the first word of a sentence.
- $P(\langle b \rangle|\text{Boston})$: Probability that “Boston” will be the last word of a sentence.

n -gram models and linguistic structure

How much linguistic structure is captured by n -gram-models?

- Higher $n \Rightarrow$ we capture more language structure, BUT
- Higher $n \Rightarrow$ we need more training data to get accurate probabilities.
- 4-grams and above require Google quantities of data, OR a restricted domain!

Long-distance dependencies will always be a problem (regardless of the choice of n):

The struggle *between conservatives and socialists is being fought on many fronts.*

Do not look back long enough

DD2418 Language Engineering

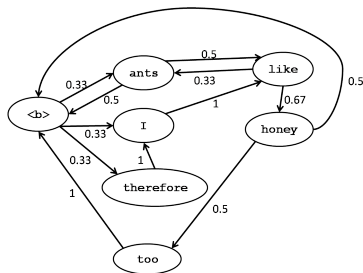
3b: Text generation from language models

Johan Boye, KTH

Markov chains

n -gram models are examples of *Markov chains*. In particular, a bigram model is a *first order Markov chain*.

First-order Markov chains are often depicted as a *finite-state automaton* with probabilities on the edges.



I like ants. Ants like honey. Therefore I like honey too.

Text generation from bigram models

Kafka:

the middle of year when her eyes with them appeared with how he called again after him and mother to our breakfast ? that ' ' , called gregor 's spokesman to this way they disappeared ; and it was still falling out again , only did not want to make them , walls and cheaper than a brother any revulsion for more than he really happened , that gregor hardly even with the time was already ? it had poured some secret prohibition on the living room before . all his chin on his parents into a painful silence .

Austen:

the course may i hope you , ' ' my junior , which her uncle and she would be very likely to see the british court , that my worked on the hands with an instrument till summoned from making any of their behaviour while the solitude . what , however , for better worth speaking of bingley would be married to be done , she , when dinner was again ; ' ' what news ; but look at last question , nothing . ' ' i should take one readily engaged to hesitate . ' ' cried elizabeth , and at her

Text generation from bigram models

Guardian:

the south of président butter and cafe , arts) , which is that it ' d nurtured
. < > facebook twitter pinterest villa o ' s emotions thinking about sex " over
with the page , and executive poached artichoke , paindepices.fr christmas day .
q & tech news brands such as a mist and sports correspondent at sheena 's too
salty and her polycystic ovary syndrome on the fruit , 55 % of conversation with
soldeu is intended targets for ways to make a deeply entrenched scepticism about
the shoreline , uae ahas already marginalised - proving

small:

i live in boston . i like honey too ! live in boston . i like ants . therefore i
like honey too ! therefore i like ants . i like ants like honey too ! therefore
i like honey too ! like honey too ! honey too ! ants . therefore i live in
boston . i like honey too ! i like honey . i like honey . ants . ants like honey
too ! honey too ! in boston . i like honey too ! honey too ! live in boston . i
live in boston .

Assignment 2, problem 2

- Write a program that generates words from a language model.
- Generate 100 words each from the models you have created.

Text generation (Sherlock Holmes)

Unigram model

- 1 your something
- 2 you she to offices the possible his of of his said sight , was laughing had .
- 3 white was not full meet old be to made , you no I . described that power he the, man , And ,
- 4 was Captain That she point labyrinth now must be far from . door had the from again what almost result fill , for coming as . a with made
- 5 his then a country-town by you ' ago Men ?

Text generation (Sherlock Holmes)

Bigram model

- 1 Then here is the mud-bank what you , and instantly , and two officers waiting at once more valuable as I asked .
- 2 I may place is her husband and illegal constraint and outstanding , not recognised shape of finding that your heart , for communication between this man .
- 3 Mrs. Toller knows I mean that I have done very heartily at the 11 : That is a lad, his neighbour .
- 4 Then there has offered to its centre one left this case , upon me to violin-land , though the corner and hurried across the very large staples .
- 5 Holmes ran up by old-fashioned shutters of treachery to attend to which I thought I have a foreigner , too late Ezekiah Hopkins , with this rather cumbrous .

Text generation (Sherlock Holmes)

Trigram model

- 1 However , when last seen , but now I will leave no survivor from a solution by the Underground and hurried me into a bedroom , which boomed out every quarter of a brickish red .
- 2 ' I beg that you have ever done yet , among the trees and wayside hedges were just being lighted as we stepped from her imprudence in allowing this brute to trace some geese which were new to me .
- 3 Holmes had sat up in my uncle's life , and that a woman .
- 4 James and his hand and at the open , and has seen , but there are a thousand details which seem to have been hanged on far slighter evidence , I thought of !
- 5 Mr. Windibank draws my interest every quarter and pays it over to him . Natural

Text generation (Sherlock Holmes)

4-gram model

a little bit over-train since might only have one path through the text

- 1 Seeing that his passion was becoming ungovernable , I left him and returned towards Hatherley Farm .
- 2 You will excuse me , said my wife , and in order to see whether the objections are fatal , or if he had been to the side from which I could see that two of them were of the war he fought in Jackson's army , and afterwards from your gesture, that Miss Rucastle was perfectly happy , and that I can .
- 3 I rang the bell and called for the weekly county paper , which contained a verbatim account of the matter , but you do not see the point .
- 4 It hadn't pulled up before she shot out of the window ?
- 5 Why does fate play such tricks with poor , helpless worms ?

DD2418 Language Engineering

3c: Evaluation of language models

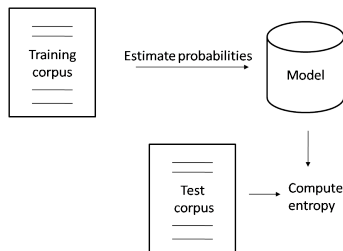
Johan Boye, KTH

Evaluation of n -gram models

Extrinsic evaluation: Put your n -gram to use in an application, e.g. a machine translation system. Measure the performance.

Intrinsic evaluation: Compute the *entropy* of the model.

words that come out are random events generated by some distribution



Information

lower probability -> more information

The *information* of an event having probability p is

$$-\log_2 p$$

Information is measured in *bits*.

Entropy

The *entropy* of a random variable X is the expected value of the information.

$$H(X) = - \sum_{i=1}^n P(X = x_i) \log_2 P(X = x_i)$$

Special case: We assume that $0 \log_2 0 = 0$.

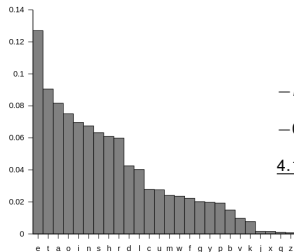
The entropy of X is a measure of the difficulty of predicting the value of X .

Entropy examples

Entropy for a-z using a uniform distribution:

$$-\sum_{i=1}^{26} \frac{1}{26} \log_2 \frac{1}{26} = -\log_2 \frac{1}{26} = \log_2 26 = \underline{4.7}$$

Entropy for a-z using probabilities for English:



$$\begin{aligned} & -P(a) \log_2 P(a) & -P(b) \log_2 P(b) & -\dots & -P(z) \log_2 P(z) = \\ & -0.082 \log_2 0.082 & -0.015 \log_2 0.015 & -\dots & -0.00074 \log_2 0.00074 = \\ & \underline{4.18} \end{aligned}$$

Entropy if $P(a) = 1$: $-1 \log_2 1 - 0 \log_2 0 - \dots 0 \log_2 0 = 0$

Basic properties of entropy

More predictability \Leftrightarrow lower entropy.

The entropy of a certain event is 0.

Maximum entropy is obtained if all events are equally probable.

Entropy and language

What is the entropy at different points in a sentence?

The recent ①
noun adjective

Entropy and language

What is the entropy at different points in a sentence?

The recent ① results mean that ②

can be a lot of different kind of words

Entropy and language

What is the entropy at different points in a sentence?

The recent ① results mean that ②
scientists now need to ③

verb adverb --> easier than 2 but still difficult

Entropy and language

What is the entropy at different points in a sentence?

The recent ① results mean that ②
scientists now need to ③ go back to
the drawing ④

can guess with some certainty. go back to the drawing board is an idiom -> entropy close to 0

Entropy and language

What is the entropy at different points in a sentence?

The recent ① results mean that ②
scientists now need to ③ go back to
the drawing ④ board.

Cross-entropy

Suppose a, b, c, d is distributed according to p :

$$p_a = P(a) = 0.5, p_b = 0.2, p_c = 0.2, p_d = 0.1$$

but we *believe* it is distributed according to q :

$$q_a = P(a) = 0.1, q_b = 0.2, q_c = 0.2, q_d = 0.5$$

The *cross-entropy of p on q* is then computed as:

$$H(X) = - \sum_{i=a,b,c,d} p_i \log_2 q_i$$

Cross-entropy measures how difficult it is to predict the symbol under this belief.

Entropy as an evaluation metric

Entropy can be used as an evaluation metric for language models.

Given a model P estimated from a *training corpus*, one can approximate the entropy as:

$$-\frac{1}{N} \log_2 P(w_1, w_2, \dots, w_N)$$

where w_1, w_2, \dots, w_N is a very long sequence of words from a *test corpus*.

The above computation really approximates the *cross-entropy* of the test set on P , which is an upper bound of the entropy of P .

Entropy as an evaluation metric

The *lower* the entropy of the test corpus, the *better* the language model learned from the training corpus.

The tacit assumption here is that the test corpus is representative of actual data.

Assignment 2, problem 3

- Write a program that evaluates a language model (a model constructed with your program in (a)) on a given test set.
- Build a number of models and evaluate them on different test sets.
- Draw conclusions.