THE LANGUAGE MODELING PROBLEM

Lecture adapted from
https://youtube.com/playlist?list=PLIQBy7xY8mbJONAWxZmZsHj0iqjMpKONi&si=TLh2Phyna3ZndoHD

The Language Modeling Problem

- One of the oldest problems studied in statistical NLP because very useful for many applications

The Language Modeling Problem

- We have some finite vacabulary, say V = {the, a , man, telescope, Beckham, two, …} (can be very large depending on the data)

- We have an (infinite) set of strings, $V^{\dagger}$ => set of all possible sentences in this language

- A sentence must have 0 or more words and each word must come from V, any sequence is possible.

The Language Modeling Problem

- Possible sentences :
  - The STOP
  - A STOP
  - The fan STOP
  - The fan saw Beckham STOP
  - The fan saw saw STOP
  - The the the STOP
  - STOP

The Language Modeling Problem

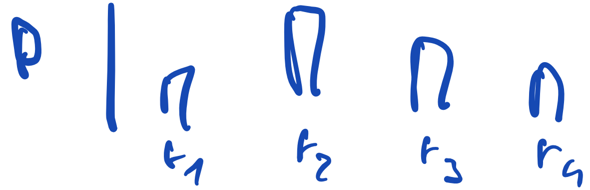
- We have a *training* s*ample* of example sentences in English

- Collection of sentences from the New York Times during the last ten years for example,
- or large sample of sentences from the web

- 90s => 20 million words/tokens
- 2000s => 1 billion
- Nowadays => couple trillion

The Language Modeling Problem

- Our task is to « learn » a probability distribution *p* over the sentences in our language.



- 2 conditions :
  - $p(x) \geq 0 \; \forall \, x \in V^{\dagger}$ => For any sentence x, the probability of that sentence must be greater or equal to 0
  - $\sum_{(x \in V^{\dagger})} p\,(x) = 1$ => If we sum over all of the probabilities of the sentences in the language we obtain 1, meaning *p* is a well-formed distribution.

The Language Modeling Problem

- p is essentially a function which returns the probability for a sequence in a given language.

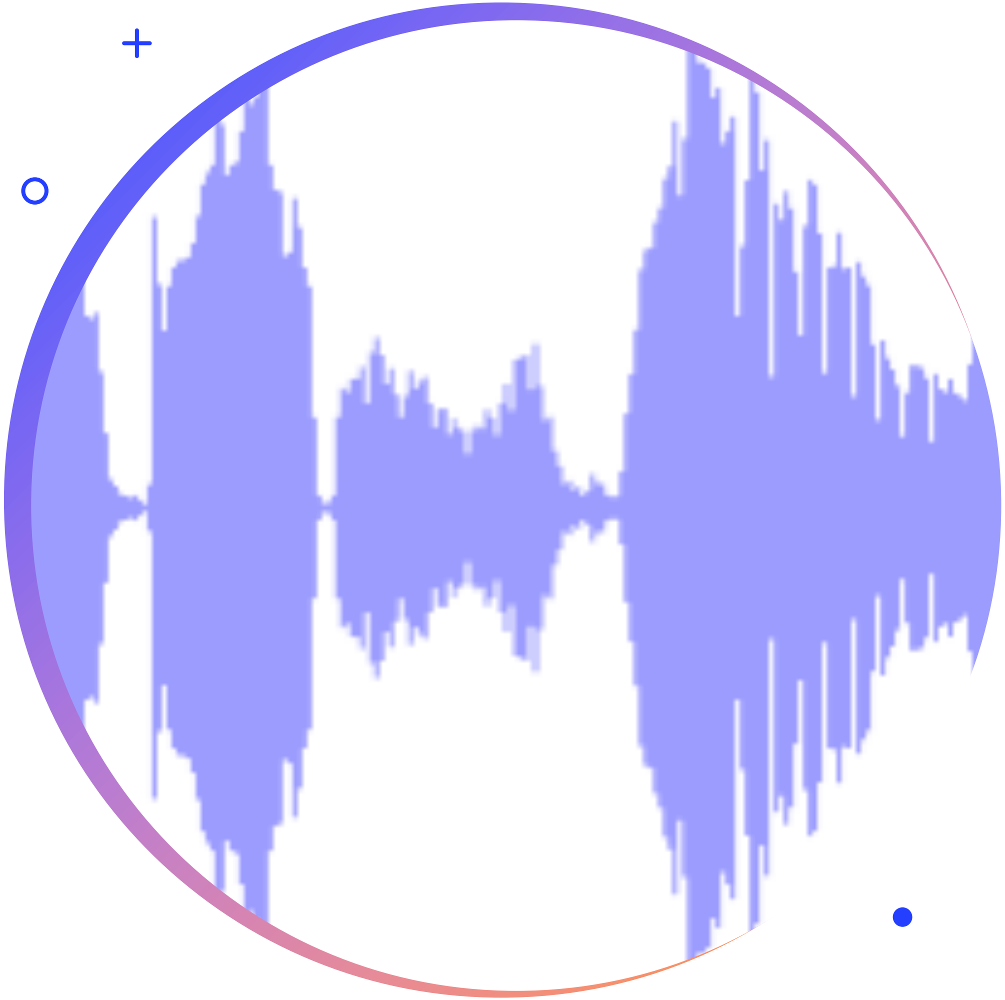
- P(the STOP) = 10^{-12}
- P(the fan STOP) = 10^{-8}
- P(the fan saw Beckham STOP) = 2 x 10^{-8}
- P(the fan saw saw) = 10^{-15}
- … assign a probability to every sequence in the language

The Language Modeling Problem

- We want to try and assign a high probability to likely sentences in English and low probability to unlikely sentences in English

Why would we want to do this ?!

- Language models are useful in many applications:
  - Speech recognition: language models are critical for modern speech recognizers (handwriting recognition also)
  - The estimation techniques used for this problem are useful for other NLP problems such as POS tagging or automatic translation.
  - Nowadays they are widely used for *generative* purposes



Language modeling for Speech Recognition

- Quick sketch :
    - Input => an acoustic recording
    - Then map this input to the words which are actually spoken

Language modeling for Speech Recognition

- Imagine the person says « recognize speech »

- In practice, there are actually many alternative sentences which could have been spoken :
    - « wreck a nice beach » 1
    - « wreck an ice peach » 1

- Similar sentences from an acoustic point of view

Language modeling for Speech Recognition

- A language model allows us to produce a probability for each sentence and estimate that « recognize speech » is more probable than other options.

- => Adds some very useful info to get rid of these kinds of confusions

A naive method for Language Modeling

- We have N sentences
- For any sentence or sequence $x_1 \dots x_n$,
- $\mathrm{C}(x_1 \ \dots \ x_n)$ is the # of times the sentence was seen in our training data.

- A naive estimate :

- $p(x_1 \dots x_n) = \frac{C(x_1 \dots x_n)}{N}$

A naive method for Language Modeling

- Has some deficiencies, although it’s a well-formed language model:

- Mainly it assigns proba 0 to any sentence not seen in our training sample...

- Cannot **generalize** to new sentences

Trigram Models

- Widely used statistical language model
- Build heavily on the idea of **Markov processes**…

Markov Processes

- Consider a sequence of random variables $X_1, \ldots, X_n$.
- Each random variable can take any value in a finite set V (vocab).
- We can assume the length n is fixed for now. (n=100 for ex.)
- We want to **model** the **joint probability**

$$P(X_1 = x_1, \ldots, X_n = x_n)$$

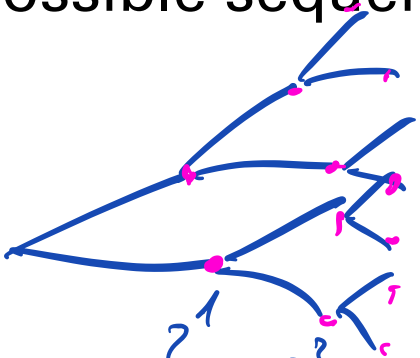
Markov Processes

- **Huge number** of possible values.

- $|V|^n$ possible sequences in our example

$=$

$len(r)^n$



2^1 2^2 2^3 ......

$2 \times 2 \times 2 \times 2 \times 2 \ldots$

First-Order Markov Process

- Going to use the **chain rule** to **decompose** this joint proba

- Remember: P(A,B) = P(A) x P(B|A)
- And therefore P(A,B,C) = P(A) x P(B|A) x P(C|A,B) $\times P(D \mid A,B,C)$

D

- So :

$$P(X_1 = x_1, \ldots, X_n = x_n)$$
$$=$$
$$P(X_1 = x_1) \prod_{i=2}^{n} P(X_i = x_i \mid X_1 = x_1, \ldots, X_{i-1} = x_{i-1})$$

$P(x_1, x_2, x_3) = P(x_1)$
$\times P(x_2 \mid x_1)$
$\times P(x_3 \mid x_1, x_2)$
$\ldots$

$P(X_2 = x_2 \mid X_1 = x_1) \times P(X_3 = x_3 \mid X_1 = x_1, X_2 = x_2)$

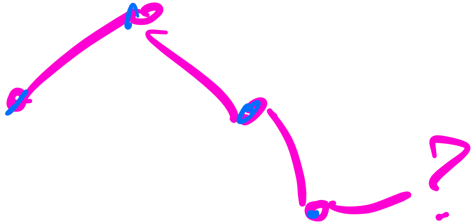
First-Order Markov Process

- The **1st order** Markov assumption states that

$$P(A, B, C, D, E) = P(A) \times P(B \mid A) \times P(C \mid A, B) \times P(D \mid A, B, C) \times P(E \mid A, B, C, D)$$

$$\prod_{i=2}^{n} P(X_i = x_i \mid X_1 = x_1, \ldots, X_{i-1} = x_{i-1})$$

Can be **simplified** as:



$$\prod_{i=2}^{n} P(X_i = x_i \mid X_{i-1} = x_{i-1})$$

$$P(A, B, C, D) \propto P(A) \times P(B \mid A) \times P(C \mid B)$$

$\times P(D \mid C)$

First-Order Markov Process

$$P(X_1 = x_1, \ldots, X_n = x_n)$$

(exact equality)

$$P(X_1 = x_1) \prod_{i=2}^{n} P(X_i = x_i \mid X_1 = x_1, \ldots, X_{i-1} = x_{i-1})$$

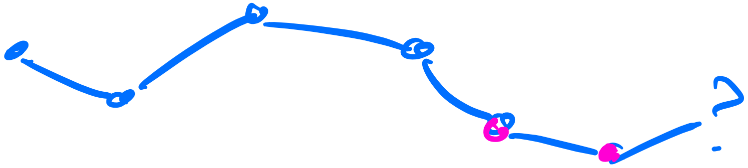
(Markov assumption)

$$P(X_1 = x_1) \prod_{i=2}^{n} P(X_i = x_i \mid X_{i-1} = x_{i-1})$$

First-Order Markov Process

- **Huge assumption** to state that the probabilty of a word here is **only conditioned** on the previous word…

Second-Order Markov Processes



- Very similar model :

$$P(X_1 = x_1, \dots, X_n = x_n)$$

$$= P(X_1 = x_1)\ P(X_2 = x_2 | X_1 = x_1) \prod_{i=3}^{n} P(X_i = x_i |\ X_{i-2} = x_{i-2}, X_{i-1} = x_{i-1})$$

- Condition on **previous 2 elements** vs. only the previous element

Second-Order Markov Processes

- We would also like to make the length of a sentence be a random variable: not all sentences will have 100 words…

- So we can define X_n to always be equal to STOP where STOP is a special symbol.

- Basically, if STOP is at position i, then this marks the end of the sentence and $i = n$

Trigram Language Model

- Given these concepts we can define a trigram language model, which consists of :

- A finite set V

- A **parameter** $\boldsymbol{q(w|u,v)}$ for each trigram u, v, w such that $w \epsilon V \cup \{STOP\}$ and $u, v \epsilon \boxed{V} \cup \{*\}$ (special start symbols)

$p(A) \to p(A | *, *)$

$p(B | ...$

$p(D \mid *, *)$

Trigram Language Model Formal Definition

- For any sentence made up of tokens $x_1, \dots, x_n$
  - where $x_i \in V$ for $i = 1, \dots, n-1$
  - and $x_n = \text{STOP}$
- The probability of the sentence under the trigram language model is

$$p(x_1, x_2, \dots, x_n) = \prod_{i=1}^{n} q(x_i | x_{i-2}, x_{i-1})$$

- Where we define $x_{-1} = x_0 = *$

An example to make things clearer

- Sentence : « * * The dog barks STOP »
  - $$p(** \; the \; dog \; barks \; STOP) = q(the|*,*) \times q(dog|*,the) \times q(barks|the,dog) \times q(STOP|dog,barks)$$
- Product of terms to get the proba of the sentence under this type of language model
- We're treating sentences as being generated by a second order Markov process, where each word generated is **dependent purely on the 2 previous words.**

Trigram Language Model

- Advantages :
  - Simple, easy and cheap
  - useful for many applications
  - availablity of statistics over the internet
  - well understood math
- Disadvantages:
  - Language: they **do not capture non-local dependencies**

Estimating the parameters

- So we need to estimate $q(w_i|w_{i-2}, w_{i-1})$
- Remember, if we have two dependent events :

$$p(A,B) = p(A) \times \boxed{p(B|A)}$$

$p(A)$ $\qquad$ $p(A)$

- Which is equivalent to

$$\boxed{p(B \mid A) = \frac{p(A,B)}{p(A)}}$$

- Which can be generalized to 3 events

$$p(C|A,B) = \frac{\boxed{p(A,B,C)}}{\boxed{p(A,B)}}$$

Estimating the parameters

- A natural estimate is therefore :

$$q(w_i|w_{i-2}, w_{i-1}) = \frac{Count(w_{i-2}, w_{i-1}, w_i)}{Count(w_{i-2}, w_{i-1})}$$

- So for example :

$$q(laughs|the, dog) = \frac{Count(the, dog, laughs)}{Count(the, dog)}$$

Estimating the parameters from a toy corpus

- An example corpus:

*

1. the cat saw the mouse.
2. the cat heard a mouse.

STOP

3. the mouse heard.
4. a mouse saw.
5. a cat saw.
6. a cat heard the mouse.

$$q(\text{the} \mid *) = \frac{\text{Count}(*, \text{the})}{\text{Count}(*)} = 1$$

$$q(\text{saw} \mid \text{cat}) = \frac{C(\text{cat}, \text{saw})}{C(\text{cat})} = \frac{1}{2} = 0{,}5$$

=> Using the corpus, give the parameter estimates for :

- a bigram language model
- a trigram language model

$$q(\text{the} \mid *) \times q(\text{cat} \mid \text{the}) \ldots \propto p(S)$$

Estimating the parameters

Bigram	Count	Unigram	Count	Relative frequency
* the	3	*	6	3/6
the cat	2	the	5	2/5
cat saw	2	cat	4	2/4
saw the	1	saw	3	1/3
the mouse	2	the	5	2/5
mouse STOP	3	mouse	5	3/5
cat heard	2	cat	4	2/4
heard a	1	heard	3	1/3
a mouse	2	a	4	2/4
...	...	...	...	...

Estimating the parameters

Trigrams	Count	Bigram	Count	Relative frequency
* * the	3	**	6	3/6
* the cat	2	* the	3	2/3
the cat saw	1	the cat	2	1/2
cat saw the	1	cat saw	2	1/2
saw the mouse	1	saw the	1	1
the mouse STOP	2	the mouse	3	2/3
the cat heard	1	the cat	2	1/2
cat heard a	1	cat heard	2	1/2
heard a mouse	1	heard a	1	1
a mouse STOP	1	a mouse	2	1/2
...	...	...	...	...

$$q(\text{the} \mid \text{cat}, \text{saw}) = 0{,}5$$

$$q(\text{a} \mid \text{cat}, \text{saw}) = 0{,}25$$

0,5 0,25

the a ...