Natural Language Processing

CS 3216/UG, AI 5203/PG

Week-5

Language models

Recap

- NLP
- Applications
- Regular expressions
- Tokenization
- Stemming
    - Porter Stemmer
- Lemmatization
- Normalization
- Stopwords
- Bag-of-Words
- TF-IDF
- NER
- POS tagging
- Semantics, Distributional semantics, Word2vec

Language Model

- **Classic definition-** Probability distribution over sequence of tokens
- Vocabulary V – a set of tokens!

- A language model **p** assigns each sequence of tokens $(x_1,\ldots\ldots\ldots x_L) \in V$, a probability (a number between 0 and 1),

$$P\,(x_1,\, x_2,\, x_3,\, \ldots\ldots\ldots\ldots,\, x_L)$$

- The probability intuitively tells us how "good" a sequence of tokens is.

Language Model

- Consider the probability of four strings in English.

- Here Vocabulary,

$$V = \{ate,\ ball,\ cheese,\ mouse,\ the\}$$

- $p(the,\ mouse,\ ate,\ the,\ cheese)=0.02$ ------> P_1
- $p(the,\ cheese,\ ate,\ the,\ mouse)=0.01$ -------> P_2
- $p(mouse\ ,the,\ the,\ cheese,\ ate)=0.0001$ -----> P_3

Clearly, $P1 > P2 > P3$

Language Model

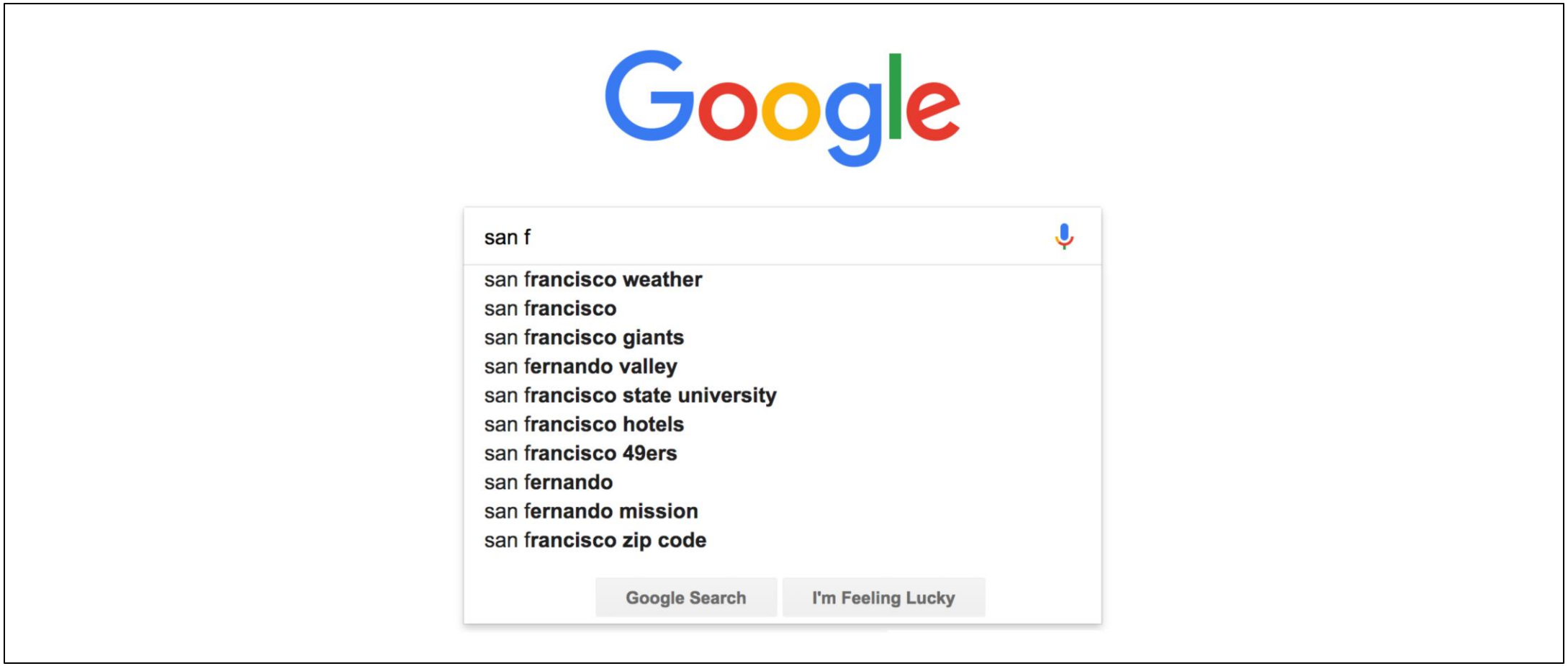
- LM assigned, *"mouse the the cheese ate"* a very low probability implicitly because it's ungrammatical (syntactic knowledge).

- The LM should assign ***the mouse ate the cheese*** higher probability than ***the cheese ate the mouse*** implicitly because of,

  - world knowledge: both sentences are the same syntactically,

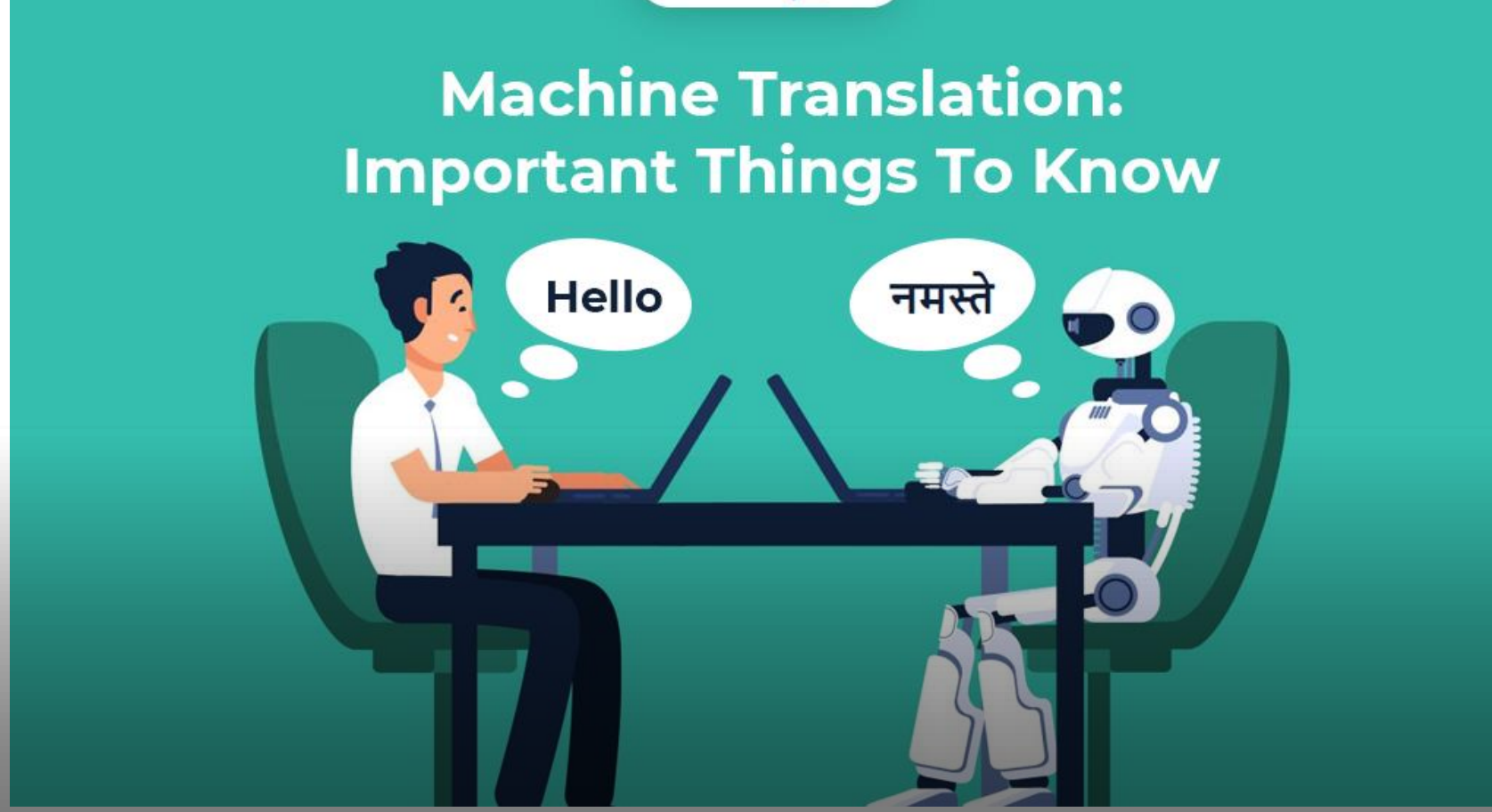
  - they differ in semantic plausibility

Where do you see language models? Google Search system



Spell correction

- The office is about fifteen minuets from my house
  - P(about fifteen minutes from) > P(about fifteen minuets from)



Machine Translation

P(high winds tonite) > P(large winds tonite)

Speech recognition

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

Types of Language Models

- Probabilistic language models (PLMs)
- Neural language models (NLMs)

Probabilistic Language Modeling

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

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

- Related task: probability of an upcoming word:

$$P(w_5 \mid w_1, w_2, w_3, w_4)$$

- A model that computes either of these:

$$P(W) \text{ or } P(w_n \mid w_1, w_2 \ldots\ldots w_{n-1})$$ is called a language model.

Credits: Slide adapted from [2]

How to compute P(W)

- How to compute this joint probability:
  - $P(its, water, is, so, transparent, that)$
- Intuition: let's rely on the Chain Rule of Probability

Credits: Slide adapted from [2]

Reminder: The Chain Rule

- Recall the definition of conditional probabilities

$$P(B \mid A) = P(A,B) / P(A)$$

Rewriting: $P(A,B) = P(A)P(B \mid A)$

- More variables:

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

- The Chain Rule in General

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

Credits: Slide adapted from [2]

Chain Rule

To Compute joint probability of words in a sentence

"its water is so transparent"

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

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)

Credits: Slide adapted from [2]

How to estimate these probabilities

Could we just count and divide?

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

Why Not?

No!!!! Too many possible sentences!

We will never see enough data to estimate these

Credits: Slide adapted from [2]

Markov Assumption :

- Simplifying assumption:

$$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{that})$$

- Or maybe

$$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{transparent that})$$

Credits: Slide adapted from [2]

Markov Assumption

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

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

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

Credits: Slide adapted from [2]

Simplest case: Unigram model

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

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
```

Credits: Slide adapted from [2]

Bigram model

- Condition on the previous word:

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

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

Credits: Slide adapted from [2]

Estimating bigram probabilities

The Maximum Likelihood Estimate

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

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

Credits: Slide adapted from [2]

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>

Credits: Slide adapted from [2]

Solution?

Solution

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

<s> I am Sam </s>

<s> Sam I am </s>

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

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

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

Credits: Slide adapted from [2]

Raw bigram counts

- Total- 9333 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

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

Credits: Slide adapted from [2]

Bigram estimates of sentence probabilities

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

P(I|<s>)

× P(want|I)

× P(english|want)

× P(food|english)

× P(</s>|food)

= .000031

Credits: Slide adapted from [2]

What kinds of knowledge?

- 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

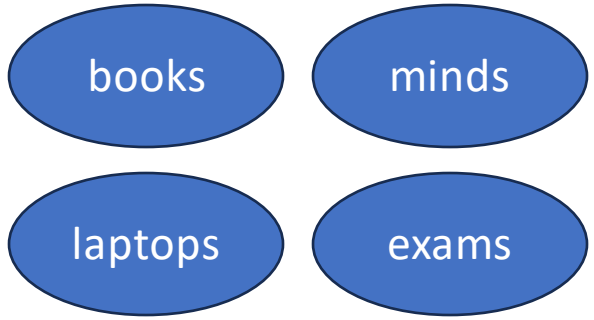
- We do everything in log space
  - Avoid underflow
  - (also adding is faster than multiplying)

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

Summary- Language Modeling Task

Language Modeling is the task of predicting what word comes next

the students opened their _____



- More formally: given a sequence of words , compute the probability distribution of the next word
- A system that does this is called a Language Model

Credits: Slide adapted from [2]

Summary: N-gram Language Models

the students opened their _____

- Question: How to learn a Language Model?
- Answer (pre- Deep Learning): learn an n-gram Language Model!

  - Definition: An n-gram is a chunk of n consecutive words.
  - ***unigrams***: "the", "students", "opened", "their"
  - ***bigrams***: "the students", "students opened", "opened their"
  - ***trigrams***: "the students opened", "students opened their"
  - ***four-grams***: "the students opened their"

- Idea: *Collect statistics about how frequent different n-grams are and use these to predict next word.*

Credits: Slide adapted from [2]

n-gram Language Models

- First we make a Markov assumption: $x^{(t+1)}$ depends only on the preceding *n*-1 words

$$P(\boldsymbol{x}^{(t+1)}|\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(1)}) = P(\boldsymbol{x}^{(t+1)}|\overbrace{\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)}}^{n\text{-1 words}}) \quad \text{(assumption)}$$

prob of a n-gram

prob of a (n-1)-gram

$$= \frac{P(\boldsymbol{x}^{(t+1)},\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})}{P(\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})} \quad \text{(definition of conditional prob)}$$

- **Question:** How do we get these *n*-gram and (*n*-1)-gram probabilities?
- **Answer:** By counting them in some large corpus of text!

$$\approx \frac{\text{count}(\boldsymbol{x}^{(t+1)},\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})}{\text{count}(\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})} \quad \text{(statistical approximation)}$$

n-gram Language Models: Example

Suppose we are learning a 4-gram Language Model.

~~*as the proctor started the clock, the*~~ *students opened their* _____

discard

condition on this

$$P(\boldsymbol{w}|\text{students opened their}) = \frac{\text{count}(\text{students opened their } \boldsymbol{w})}{\text{count}(\text{students opened their})}$$

For example, suppose that in the corpus:

- "students opened their" occurred 1000 times
- "students opened their books" occurred 400 times
  - → P(books | students opened their) = 0.4
- "students opened their exams" occurred 100 times
  - → P(exams | students opened their) = 0.1

Should we have discarded the "proctor" context?

Sparsity Problems with n-gram Language Models

Sparsity Problem 1

Problem: What if *"students opened their w"* never occurred in data? Then w has probability 0!

(Partial) Solution: Add small δ to the count for every $w \in V$. This is called *smoothing*.

$$P(\boldsymbol{w}|\text{students opened their}) = \frac{\text{count}(\text{students opened their } \boldsymbol{w})}{\text{count}(\text{students opened their})}$$

Sparsity Problem 2

Problem: What if *"students opened their"* never occurred in data? Then we can't calculate probability for *any* w!

(Partial) Solution: Just condition on *"opened their"* instead. This is called *backoff*.

Note: Increasing n makes sparsity problems *worse.* Typically, we can't have n bigger than 5.

Storage Problems with n-gram Language Models

Storage: Need to store count for all *n*-grams you saw in the corpus.

$$P(\boldsymbol{w}|\text{students opened their}) = \frac{\text{count}(\text{students opened their } \boldsymbol{w})}{\text{count}(\text{students opened their})}$$

Increasing *n* or increasing corpus increases model size!

n-gram Language Models in practice

- You can build a simple trigram Language Model over a 1.7 million word corpus (Reuters) in a few seconds on your laptop*

Business and financial news

today the _____

get probability distribution

company	0.153
bank	0.153
price	0.077
italian	0.039
emirate	0.039
…	

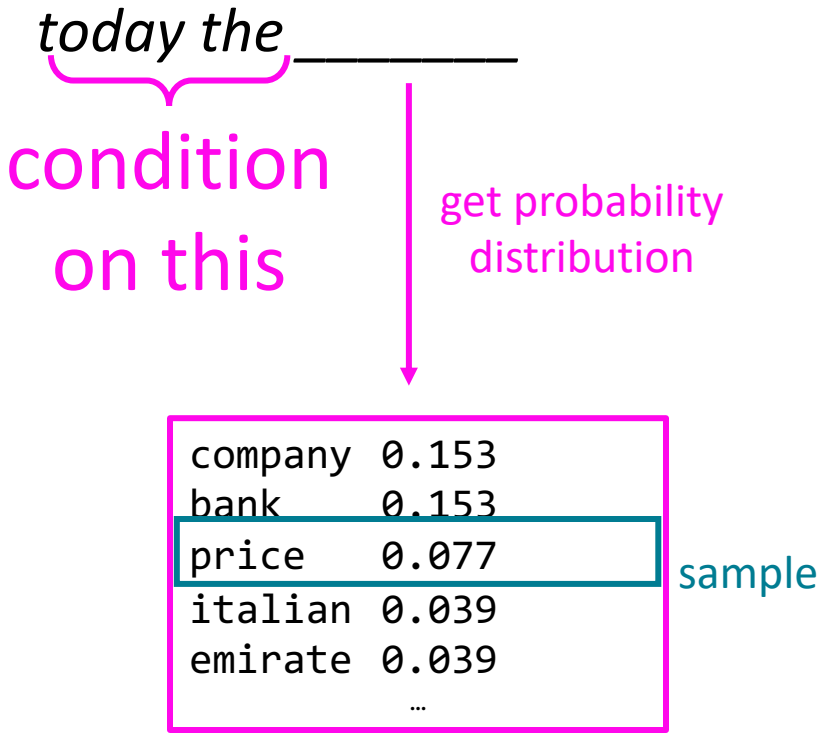
Sparsity problem: not much granularity in the probability distribution

Otherwise, seems reasonable!

*** Try for yourself:** https://nlpforhackers.io/language-models/

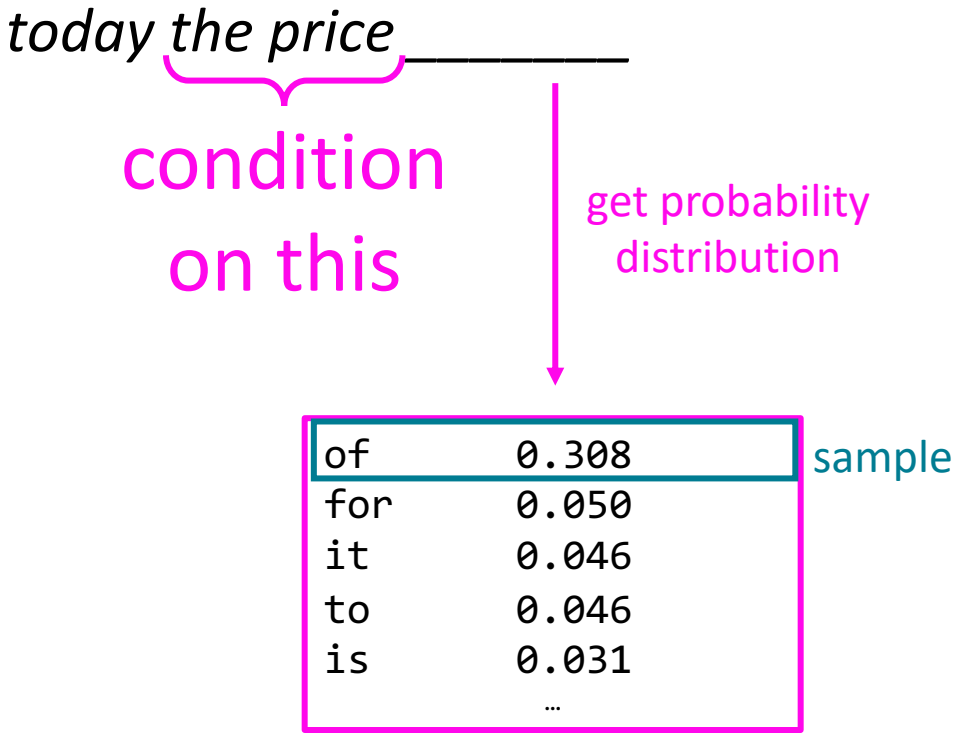
Generating text with a n-gram Language Model

You can also use a Language Model to generate text



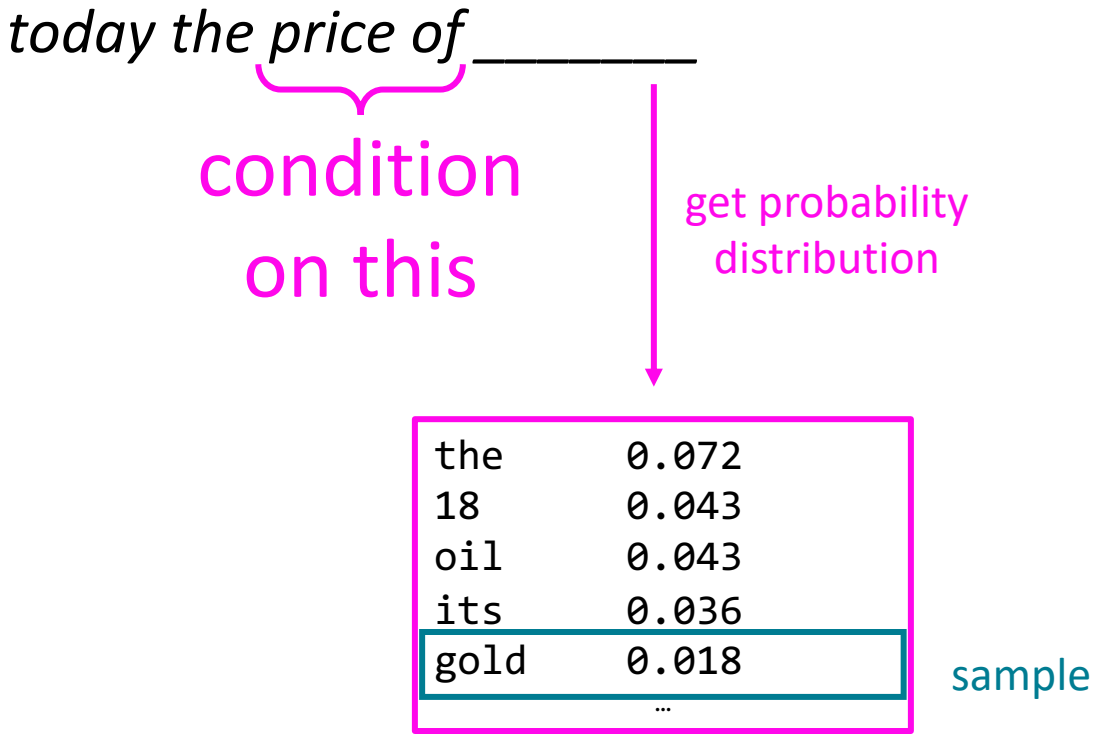
Generating text with a n-gram Language Model

You can also use a Language Model to generate text



Generating text with a n-gram Language Model

You can also use a Language Model to generate text



N-gram models

- 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."

- But we can often get away with N-gram models

Evaluating Language Models

- The standard evaluation metric for Language Models is perplexity.

$$\text{perplexity} = \prod_{t=1}^{T} \left(\frac{1}{P_{\text{LM}}(\boldsymbol{x}^{(t+1)} | \ \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)})} \right)^{1/T}$$

Normalized by number of words

Inverse probability of corpus, according to Language Model

- This is equal to the exponential of the cross-entropy loss $J(\theta)$:

$$= \prod_{t=1}^{T} \left(\frac{1}{\hat{\boldsymbol{y}}^{(t)}_{\boldsymbol{x}_{t+1}}} \right)^{1/T} = \exp\left(\frac{1}{T} \sum_{t=1}^{T} -\log \hat{\boldsymbol{y}}^{(t)}_{\boldsymbol{x}_{t+1}} \right) = \exp(J(\theta))$$

Lower perplexity is better!

References

[1] https://stanford-cs324.github.io/winter2022/lectures/introduction/

[2] https://web.stanford.edu/~jurafsky/slp3/slides/LM_4.pdf

Acknowledgments

- These slides were adapted from the book SPEECH and LANGUAGE PROCESSING: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition
- Practical Natural Language Processing (A Comprehensive Guide to Building Real-World NLP Systems) O'reilly and some modifications from presentations and resources found in the WEB by several scholars.

Reference materials

- https://vlanc-lab.github.io/mu-nlp-course/

- Lecture notes

- (A) Speech and Language Processing by Daniel Jurafsky and James H. Martin
- (B) Natural Language Processing with Python. (updated edition based on Python 3 and NLTK 3) Steven Bird et al. O'Reilly Media