LECTURE 4:
LANGUAGE MODELS
(INTRODUCTION TO PROBABILITY MODELS FOR NLP)

Mehmet Can Yavuz, PhD.

TODAY'S CLASS

Part 1: **Overview**

Part 2: **Review of Basic Probability**

Part 3: **Language Modeling with N-Grams**

Part 4: **Generating Text with Language Models**

Part 5: **Evaluating Language Models**

HMMM…

… what does it mean for a language model to "*define a probability distribution*"?

… *why* would we want to define probability distributions over languages?

… how can we construct a language model such that it *actually* defines a probability distribution?

… how do we know *how well* our model works?

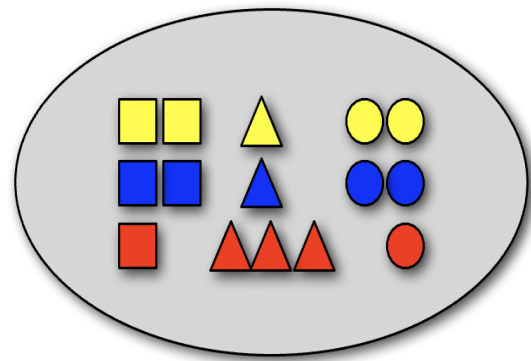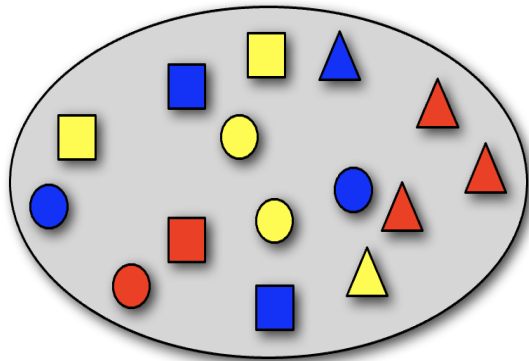You should be able to answer these questions after this lecture

TODAY'S KEY CONCEPTS

- N-gram language models
- Independence assumptions
- Getting from n-grams to a distribution over a language

  Relative frequency (maximum likelihood) estimation

  Smoothing
- Intrinsic evaluation: Perplexity,
- Extrinsic evaluation: Word error rate (WER)

PART 2: REVIEW OF BASIC PROBABILITY THEORY

SAMPLING WITH REPLACEMENT

$P(\blacksquare) = 2/15$ $P(\blacksquare) = 1/15$ $P(\blacksquare \text{ or } \blacktriangle) = 2/15$

$P(\text{blue}) = 5/15$ $P(\text{red}) = 5/15$ $P(\triangle | \text{red}) = 3/5$

$P(\text{blue} | \square) = 2/5$ $P(\square) = 5/15$

Pick a random shape, then put it back in the bag.

SAMPLING WITH REPLACEMENT

- Pick a random shape, then put it back in the bag. What sequence of shapes will you draw?
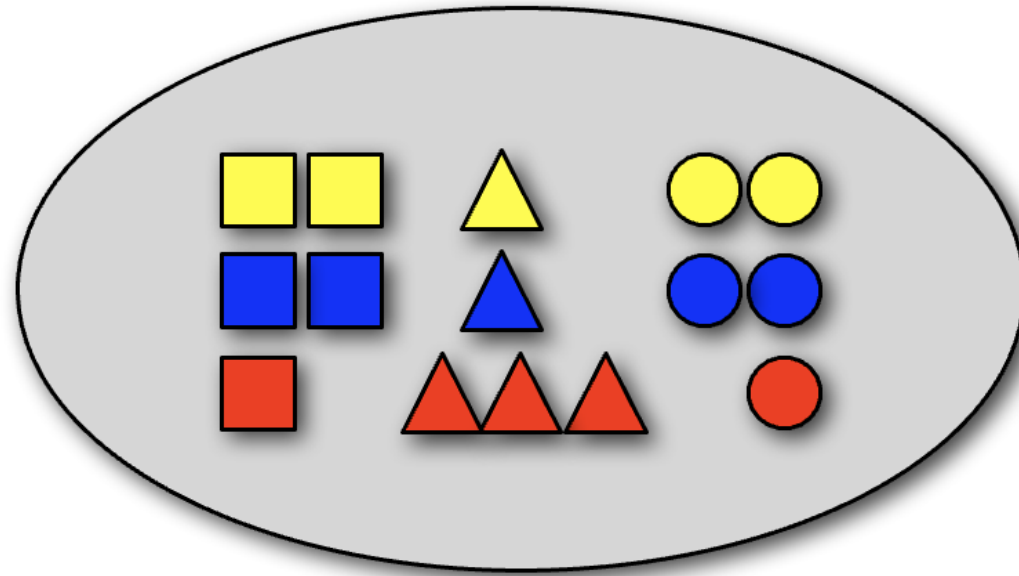
$P(\blacksquare) = 2/15$ $P(\blacksquare) = 1/15$

$P(\text{blue}) = 5/15$ $P(\text{red}) = 5/15$

$P(\text{blue} \mid \square) = 2/5$ $P(\square) = 5/15$

NOW LET'S LOOK AT NATURAL LANGUAGE

Text as a bag of words

Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, 'and what is the use of a book,' thought Alice 'without pictures or conversation?'

$P(\text{of}) = 3/66$ $P(\text{to}) = 2/66$ $P(\text{,}) = 4/66$

$P(\text{Alice}) = 2/66$ $P(\text{her}) = 2/66$ $P(\text{'}) = 4/66$

$P(\text{was}) = 2/66$ $P(\text{sister}) = 2/66$

SAMPLING WITH REPLACEMENT

A sampled sequence of words

beginning by, very Alice but was and?
reading no tired of to into sitting
sister the, bank, and thought of without
her nothing: having conversations Alice
once do or on she it get the book her had
peeped was conversation it pictures or
sister in, 'what is the use had twice of
a book''pictures or' to

$P(\text{of}) = 3/66$ $P(\text{to}) = 2/66$ $P(\text{,}) = 4/66$

$P(\text{Alice}) = 2/66$ $P(\text{her}) = 2/66$ $P(\text{'}) = 4/66$

$P(\text{was}) = 2/66$ $P(\text{sister}) = 2/66$

In this model, $P(\text{English sentence}) = P(\text{word salad})$

PROBABILITY THEORY: TERMINOLOGY

Trial (aka "experiment")

- Picking a shape, predicting a word

Sample space Ω:

- The **set of all possible outcomes**
- (all shapes; all words in *Alice in Wonderland*)

Event $\omega \subseteq \Omega$:

- An **actual outcome** (a subset of Ω) (predicting '*the*', picking a triangle)

Random variable $X: \Omega \to T$

- A function from the sample space (often the identity function) Provides a 'measurement of interest' from a trial/experiment (Did we pick 'Alice'/a noun/a word starting with "x"/…?
- How often does the word 'Alice' occur? How many words occur in each sentence?)

WHAT IS A PROBABILITY DISTRIBUTION?

- $P(\omega)$ defines a **distribution** over Ω iff

1) *Every* event ω has a probability $P(\omega)$ between 0 and 1:

$$0 \leq P(w \subseteq \Omega) \leq 1$$

2) The *null* event $\emptyset$ has probability $P(\emptyset) = 0$:

$$P(\emptyset) = 0$$

3) And the probability of all *disjoint* events sums to 1.

$$\sum_{\omega_i \subseteq \Omega} P(\omega_i) = 1 \ \text{ if } \forall j \neq i : \omega_i \cap \omega_j = \emptyset$$
$$\text{and } \bigcup_i \omega_i = \Omega$$

DISCRETE PROBABILITY DISTRIBUTIONS: SINGLE TRIALS

'**Discrete'**: a fixed (often finite) number of outcomes

Bernoulli distribution (Two possible outcomes (*head, tail*) Defined by the probability of success (= *head*/yes)
- The probability of *head* is p. The probability of *tail* is $1-p$.

Categorical distribution (N possible outcomes $c_1 \ldots c_N$)
- The probability of category/outcome c_i is p_i ($0 \leq p_i \leq 1$; $\sum_i p_i = 1$).
- e.g. the probability of getting a six when rolling a die once
- e.g. the probability of the next word (picked among a vocabulary of N words) (NB: Most of the distributions we will see in this class are categorical.
- Some people call them multinomial distributions, but those refer to *sequences*
- of trials, e.g. the probability of getting five sixes when rolling a die ten times)

JOINT AND CONDITIONAL PROBABILITY

- The **conditional probability** of X given Y, $P(X | Y)$, is defined in terms of the **probability** of Y, $P(Y)$, and the **joint probability** of X and Y, $P(X, Y)$:

$$P(X | Y) = \frac{P(X,Y)}{P(Y)}$$

- What is the probability that we get a blue shape if we pick a square?
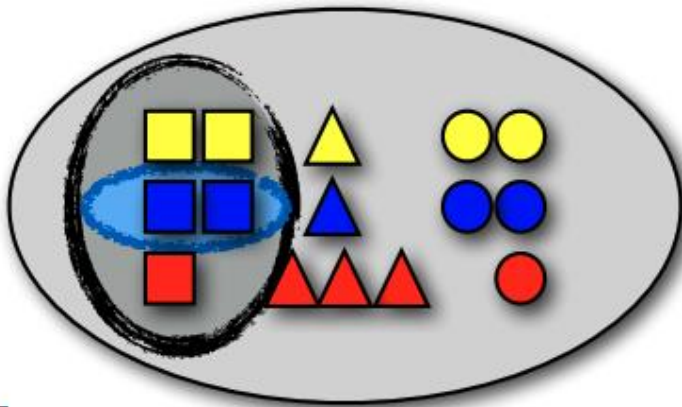
$$P(\text{blue} | \blacksquare) = 2/5$$

THE CHAIN RULE

The **joint probability** $P(X,Y)$ can also be expressed in terms of the **conditional probability** $P(X|Y)$

$$P(X, Y) = P(X|Y)P(Y)$$

Generalizing this to N joint events (or random variables) leads to the so-called **chain rule**:

$$
\begin{aligned}
P(X_1, X_2, \ldots, X_n) &= P(X_1)P(X_2|X_1)P(X_3|X_2, X_1)\ldots.P(X_n|X_1, \ldots X_{n-1}) \\
&= P(X_1)\prod_{i=2}^{n} P(X_i|X_1 \ldots X_{i-1})
\end{aligned}
$$

INDEPENDENCE

- Two events or random variables X and Y are **independent** if

$$P(X,Y) = P(X)\, P(Y)$$

- If X and Y are independent, then $P(X \mid Y) = P(X)$:

$$
\begin{aligned}
P(X|Y) &= \frac{P(X,Y)}{P(Y)} \\
&= \frac{P(X)P(Y)}{P(Y)} \quad (X, Y \text{ independent}) \\
&= P(X)
\end{aligned}
$$

PROBABILITY MODELS

Building a probability model consists of two steps:

- Defining the model
- Estimating the model's parameters (= training/learning)

Probability models (almost) always make independence *assumptions.*

- Even though X and Y are not *actually* independent, our model may treat them as independent.

This can drastically reduce the number of parameters to estimate.

- Models without independence assumptions have (way)
  - too many parameters to estimate reliably from the data we have
- But since independence assumptions are often incorrect, those models are often incorrect as well:
  - they assign probability mass to events that cannot occur

PART 3: LANGUAGE MODELING WITH N-GRAM

LANGUAGE MODELING WITH N-GRAMS

A **language model** over a vocabulary V assigns probabilities to strings drawn from V*.

How do we compute the **probability of a string**

$w^{(1)} \ldots w^{(i)}$?

Recall the **chain rule**:

$P(w^{(1)} \ldots w^{(i)}) = P(w^{(1)}) \cdot P(w^{(2)} \mid w^{(1)}) \cdot \ldots \cdot P(w^{(i)} \mid w^{(i-1)}, \ldots, w^{(1)})$

An **n-gram** language model assumes each word $w^{(i)}$ depends only on the **last n-1 words** $w^{(i-1)}, \ldots, w^{(i-(n+1))}$

$P_{ngram}(w^{(1)} \ldots w^{(i)}) = P(w^{(1)}) \cdot P(w^{(2)} \mid w^{(1)}) \cdot \ldots \cdot P(w^{(i)} \mid w^{(i-z1)}, \ldots, w^{(i-(n+1))})$

N-GRAM MODELS

N-gram models *assume* each word (event) depends only on the previous n−1 words (events):

$$\textbf{Unigram model: } P(w^{(1)} \dots w^{(N)}) = \prod_{i=1}^{N} P(w^{(i)})$$

$$\textbf{Bigram model: } P(w^{(1)} \dots w^{(N)}) = \prod_{i=1}^{N} P(w^{(i)} \mid w^{(i-1)})$$

$$\textbf{Trigram model: } P(w^{(1)} \dots w^{(N)}) = \prod_{i=1}^{N} P(w^{(i)} \mid w^{(i-1)}, w^{(i-2)})$$

NB: Independence assumptions where the n-th event in a sequence depends only on the last n-1 events are called Markov assumptions (of order n−1).

HOW MANY PARAMETERS DO N-GRAM MODELS HAVE?

- Given a vocabulary V of $|V|$ word types:

- **Unigram model:** $|V|$ parameters

- (one distribution $P(w^{(i)})$ with $|V|$ outcomes [each $w \in V$ is one outcome])
- **Bigram model:** $|V|^2$ parameters
- ($|V|$ distributions $P(w^{(i)} | w^{(i-1)})$, one distribution for each $w \in V$ with $|V|$ outcomes each [each $w \in V$ is one outcome])
- **Trigram model:** $|V|^3$ parameters ($|V|^2$ distributions $P(w^{(i)} | w^{(i-1)}, w^{(i-2)})$, one per bigram w'w", with $|V|$ outcomes each [each $w \in V$ is one outcome])

so, for $|V| = 10^4$:

10^4 parameters

10^8 parameters

10^{12} parameters

A BIGRAM MODEL FOR ALICE

Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, 'and what is the use of a book,' thought Alice 'without pictures or conversation?'

$P(w^{(i)} = \textbf{of} \mid w^{(i-1)} = \textbf{tired}) = 1$

$P(w^{(i)} = \textbf{of} \mid w^{(i-1)} = \textbf{use}) = 1$

$P(w^{(i)} = \textbf{sister} \mid w^{(i-1)} = \textbf{her}) = 1$

$P(w^{(i)} = \textbf{beginning} \mid w^{(i-1)} = \textbf{was}) = 1/2$

$P(w^{(i)} = \textbf{reading} \mid w^{(i-1)} = \textbf{was}) = 1/2$

$P(w^{(i)} = \textbf{bank} \mid w^{(i-1)} = \textbf{the}) = 1/3$

$P(w^{(i)} = \textbf{book} \mid w^{(i-1)} = \textbf{the}) = 1/3$

$P(w^{(i)} = \textbf{use} \mid w^{(i-1)} = \textbf{the}) = 1/3$

USING A BIGRAM MODEL FOR ALICE

English

Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, 'and what is the use of a book,' thought Alice 'without pictures or conversation?'

Word Salad

beginning by, very Alice but was and? reading no tired of to into sitting sister the, bank, and thought of without her nothing: having conversations Alice once do or on she it get the book her had peeped was conversation it pictures or sister in, 'what is the use had twice of a book''pictures or' to

> *Now, P(English) ≫ P(word salad)*

$P(w^{(i)} = \textbf{of} \mid w^{(i-1)} = \textbf{tired}) = 1$

$P(w^{(i)} = \textbf{of} \mid w^{(i-1)} = \textbf{use}) = 1$

$P(w^{(i)} = \textbf{sister} \mid w^{(i-1)} = \textbf{her}) = 1$

$P(w^{(i)} = \textbf{beginning} \mid w^{(i-1)} = \textbf{was}) = 1/2$

$P(w^{(i)} = \textbf{reading} \mid w^{(i-1)} = \textbf{was}) = 1/2$

$P(w^{(i)} = \textbf{bank} \mid w^{(i-1)} = \textbf{the}) = 1/3$

$P(w^{(i)} = \textbf{book} \mid w^{(i-1)} = \textbf{the}) = 1/3$

$P(w^{(i)} = \textbf{use} \mid w^{(i-1)} = \textbf{the}) = 1/3$

FROM N-GRAM PROBABILITIES TO LANGUAGE MODELS

- Recall: a language $L \subseteq V^*$ is a (possibly infinite) set of strings over a (finite) vocabulary V.

$P(w^{(i)} \mid w^{(i-1)})$ defines a distribution over the words in V:

$$\forall w \in V : \left[\sum_{w' \in V} P\left(w^{(i)} = w' \mid w^{(i-1)} = w\right) \right] = 1$$

- By multiplying this distribution N times, we get

one distribution over all strings of the same length N (V^N):

Prob. of one N-word string: $P(w_1 \ldots w_N) = \prod_{i=1\ldots N} P\left(w^{(i)} = w_i \mid w^{(i-1)} = w_{i-1}\right)$

Prob. of all N-word strings $P(V^N) = \sum_{w,w' \in V} \left[\prod_{i=1\ldots N} P\left(w^{(i)} = w \mid w^{(i-1)} = w'\right) \right] = 1$

- But instead of N separate distributions…

  …we want one distribution over strings of *any* length

FROM N-GRAM PROBABILITIES TO LANGUAGE MODELS

We have just seen how to use n-gram probabilities to define *one* distribution $P(V^N)$ *for each string length* N.

But a language model $P(L)=P(V^*)$ should define one distribution $P(V^*)$ that sums to one over *all* strings in $L \subseteq V^*$, *regardless of their length*:
$$P(L) = P(V^1) + P(V^2) + P(V^3) + ... P(V^n) + ... = 1$$

Solution:
Add an End-of-Sentence (EOS) token to V
Assume a) that each string ends in EOS and b) that EOS can only appear at the end of a string.

FROM N-GRAM PROBABILITIES TO LANGUAGE MODELS WITH EOS

Think of a language model as a **stochastic process:**
— At each time step, randomly pick one more word.
— **Stop** generating more words when the word you pick is a special end-of-sentence (EOS) token.

To be able to pick the EOS token, we have to **modify our training data** so that each sentence ends in EOS.

• This means our vocabulary is now $V^{EOS} = V \cup \{EOS\}$

We then get an actual language model,

i.e. a distribution over strings of *any* length

- Technically, this is only true because P(EOS | …) will be high enough that we are always guaranteed to stop after having generated a finite number of words
- A leaky or inconsistent language model would have P(L) < 1. That could happen if EOS had a very small probability (but doesn't really happen in practice).
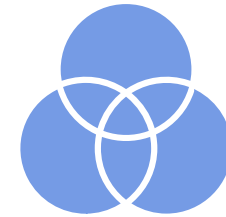
WHY DO WE WANT *ONE* DISTRIBUTION OVER L?

*Why do we care about having **one probability distribution for all lengths?***
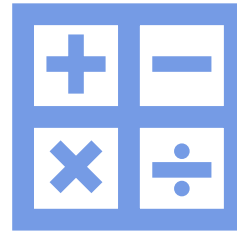
This allows us to *compare the probabilities* of strings of different lengths, because they're computed by the same distribution.

This allows us to *generate* strings of arbitrary length with one model.

PARAMETER ESTIMATION
OR: WHERE DO WE GET THE PROBABILITIES FROM?

LEARNING (ESTIMATING) A LANGUAGE MODEL

Where do we get the parameters of our model (its actual probabilities) from?

- $P(w^{(i)} = \text{'the'} \mid w^{(i-1)} = \text{'on'}) = \text{???}$

We need (a large amount of) text as training data to estimate the parameters of a language model.

The most basic parameter estimation technique: relative frequency estimation (frequency = counts)

- $P(w^{(i)} = \text{'the'} \mid w^{(i-1)} = \text{'on'}) = C(\text{'on the'}) / C(\text{'on'})$

Also called Maximum Likelihood Estimation (MLE)

- $C(\text{'on the'})$ [or f('on the') for frequency]:
- How often does 'on the' appear in the training data?
- NB: $C(\text{'on'}) = \sum_{w \hat{\imath} V} C(\text{'on' } w)$

HANDLING UNKNOWN WORDS: UNK

Training:
— Define a fixed vocabulary V such that all words in V appear at least n times in the training data

 (e.g. all words that occur at least 5 times in the training corpus, or the most common 10,000 words in training)

— Add a new token UNK to V, and replace all other words in the corpus that are not in V by this token UNK

— Estimate the model on this modified training corpus.

Testing (when computing the probability of a string): Replace any words not in the vocabulary by UNK

Refinements: Use different UNK tokens for different types of words (numbers, capitalized words, lower-case words, etc.)

WHAT ABOUT THE BEGINNING OF THE SENTENCE?

In a trigram model
$$P(w^{(1)}w^{(2)}w^{(3)}) = P(w^{(1)})P(w^{(2)}|w^{(1)})P(w^{(3)}|w^{(2)},w^{(1)})$$

only the third term $P(w^{(3)}|w^{(2)},w^{(1)})$ is an actual trigram probability. What about $P(w^{(1)})$ and $P(w^{(2)}|w^{(1)})$?

If this bothers, you:

Add n–1 **beginning-of-sentence** (BOS) symbols to each sentence for an n–gram model:

$$BOS_1\ BOS_2\ Alice\ was\ \dots$$

Now the unigram and bigram probabilities involve only BOS symbols.

SUMMARY: ESTIMATING A BIGRAM MODEL WITH BOS (<S>), EOS (</S>) AND UNK USING MLE

1. Replace all words not in V in the training corpus with UNK

2. Bracket each sentence by special start and end symbols:

`<s> Alice was beginning to get very tired …` `</s>`

3. Define the Vocabulary V' = all tokens in modified training corpus (all common words, UNK, <s>, </s>)

4. Count the frequency of each bigram….

C(<s> Alice) = 1, C(Alice was) = 1, …

5. …. and normalize these frequencies to get probabilities: $P(was \mid Alice) = \sum_{w_i \in V'} \dfrac{C(Alice\ was)}{C(Alice\ w_i)}$

LECTURE 4, PART 4: GENERATING TEXT WITH LANGUAGE MODELS

HOW DO WE USE LANGUAGE MODELS?

- Independently of any application, we could use a language model as a random sentence *generator*

(we sample sentences according to their language model probability)

NB: There are very few real world use cases where you want to generate language randomly but understanding how to do this and what happens when you do so will allow us to do more interesting things later.

- We can use a language model as a sentence *ranker*. Systems for applications such as machine translation, speech recognition, spell-checking, generation, etc. often produce many candidate sentences as output.
- We prefer output sentences S_{out} that have a higher language model probability. We can use a language model $P(S_{out})$ to score and rank these different candidate output sentences, e.g. as follows:

$$\text{argmax}_{Sout} \, P(S_{out} \mid \text{Input}) = \text{argmax}_{Sout} \, P(\text{Input} \mid S_{out})P(S_{out})$$

GENERATING FROM A DISTRIBUTION

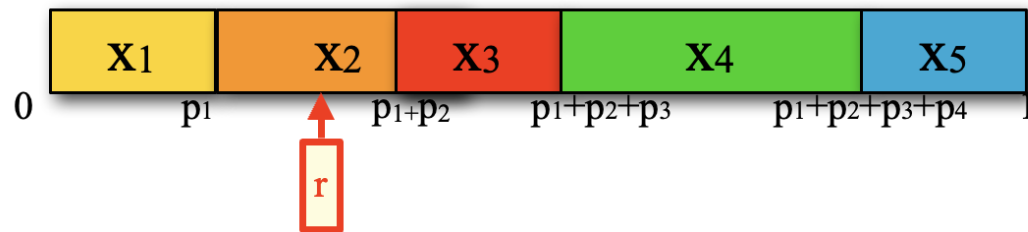How do you generate text from an *n*-gram model?

That is, how do you sample from a distribution $P(X|Y=y)$?

Assume X has N possible outcomes (values): $\{x_1, \ldots, x_N\}$

- and $P(X=x_i|Y=y) = p_i$
- Divide the interval [0,1] into N smaller intervals according to the probabilities of the outcomes

Generate a random number r between 0 and 1.

Return the x_1 whose interval the number is in.

GENERATING THE WALL STREET JOURNAL

unigram: Months the my and issue of year foreign new exchange's september were recession exchange new endorsed a acquire to six executives

bigram: 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

trigram: 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

GENERATING SHAKESPEARE

Unigram	• To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have • Every enter now severally so, let • Hill he late speaks; or! a more to leg less first you enter • Are where exeunt and sighs have rise excellency took of.. Sleep knave we. near; vile like
Bigram	• What means, sir. I confess she? then all sorts, he is trim, captain. • Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow. • What we, hath got so she that I rest and sent to scold and nature bankrupt, nor the first gentleman? • Enter Menenius, if it so many good direction found'st thou art a strong upon command of fear not a liberal largess given away, Falstaff! Exeunt
Trigram	• Sweet prince, Falstaff shall die. Harry of Monmouth's grave. • This shall forbid it should be branded, if renown made it empty. • Indeed the duke; and had a very good friend. • Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done.
Quadrigram	• King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in; • Will you not tell me who I am? • It cannot be but so. • Indeed the short and the long. Marry, 'tis a noble Lepidus.

SHAKESPEARE AS CORPUS

The Shakespeare corpus has N=884647-word tokens for a vocabulary of V=29066-word types

Shakespeare used 300,000 bigram types out of V^2= 844 million possible bigram types.
- 99.96% of possible bigrams don't occur in this corpus.

Corollary: A relative frequency estimate based on this corpus assigns non-zero probability to only 0.04% of possible bigrams
  - That percentage is even lower for trigrams, 4-grams, etc.
  - 4-grams *look* like Shakespeare because they *are* Shakespeare!

THE UNK TOKEN

What would happen if we used an UNK token on a corpus the size of Shakespeare's?

1. If we set the frequency threshold for which words to replace too high, a very large fraction of tokens become UNK.

2. Even with a low threshold, UNK will have a very high probability, because in such a small corpus, many words appear only once.

3. But we would still only observe a small fraction of possible bigrams (or trigrams, quadrigrams, etc.)
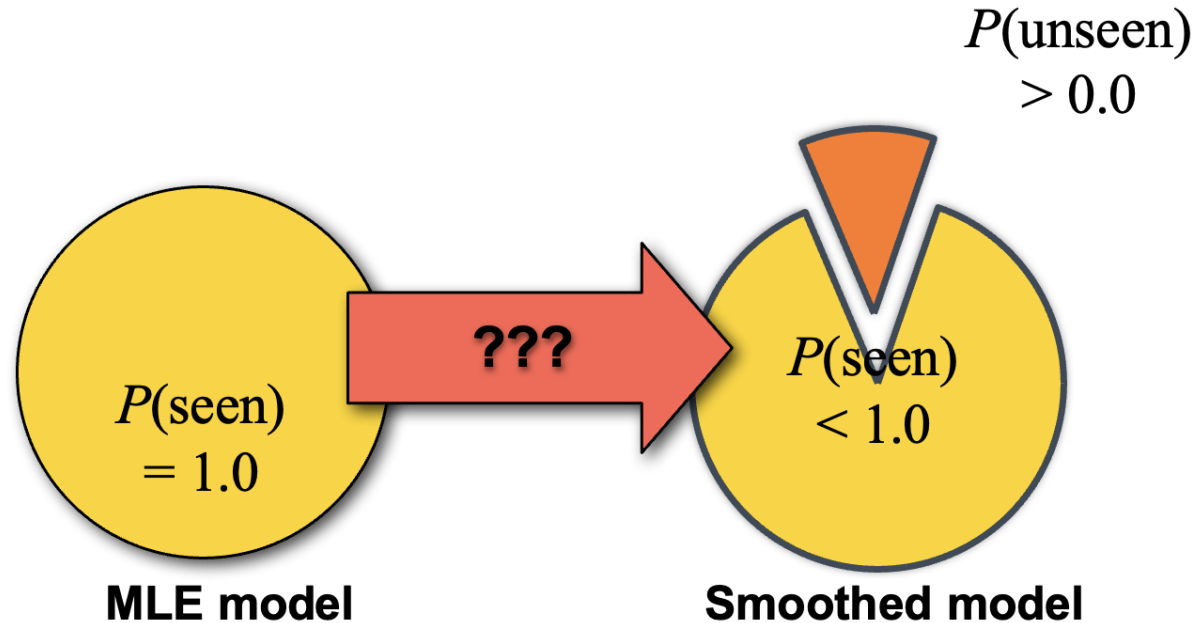
MLE DOESN'T CAPTURE UNSEEN EVENTS

We estimated a model on 884K word tokens, but:

Only 30,000-word types occur in the training data Any word that does not occur in the training data has zero probability!

Only 0.04% of all possible bigrams (for 30K word types) occur in the training data

Any bigram that does not occur in the training data has *zero* probability (even if we have seen both words in the bigram by themselves)

HOW CAN YOU ASSIGN NON-ZERO PROBABILITY TO UNSEEN EVENTS?

$P(\text{unseen}) > 0.0$

???

$P(\text{seen}) = 1.0$

$P(\text{seen}) < 1.0$

MLE model

Smoothed model

We won't talk much about smoothing this year.

- We have to "smooth" our distributions to assign some probability mass to unseen events

SMOOTHING METHODS

Add-one smoothing:

- Hallucinate counts that didn't occur in the data

Linear interpolation:

- $P_2(w \mid w', w'') = \lambda P_1(w \mid w', w'') + (1 - \lambda)P_2(w \mid w')$
- Interpolate n-gram model with (n–1)-gram model.

Absolute Discounting: Subtract constant count from frequent events and add it to rare events

- **Kneser-Ney**: AD with modified unigram probabilities

Good-Turing: Use probability of rare events to estimate probability of unseen events

ADD-ONE (LAPLACE) SMOOTHING

A really simple way to do smoothing:
- Increment the actual observed count of every *possible*
- event (e.g. bigram) by a hallucinated count of 1
- (or by a hallucinated count of some k with 0 < k < 1).

Shakespeare bigram model (roughly):
- 0.88 million actual bigram counts
- + 844.xx million hallucinated bigram counts

Oops. Now almost *none* of the counts in our model come from actual data. We're back to word salad.
- K needs to be really small. But it turns out that that still doesn't work very well.

LECTURE 4, PART 5: EVALUATING LANGUAGE MODELS

INTRINSIC VS EXTRINSIC EVALUATION

How do we know whether one language model is better than another?

There are two ways to evaluate models:

- intrinsic evaluation measures how well the model captures what it is supposed to capture (e.g. probabilities)
- extrinsic (task-based) evaluation measures how useful the model is in a particular task.

Both cases require an evaluation metric that allows us to measure and compare the performance of different models.

INTRINSIC EVALUATION OF LANGUAGE MODELS: PERPLEXITY

INTRINSIC EVALUATION OF LANGUAGE MODELS: PERPLEXITY

Define an evaluation metric (scoring function).	We will want to measure how similar the predictions of the model are to real text.
Train the model on a 'seen' training set	Perhaps: tune some parameters based on held-out data (disjoint from the training data, meant to emulate unseen data)
Test the model on an unseen test set	(usually from the same source (e.g. WSJ) as the training data) Test data must be disjoint from training and held-out data Compare models by their scores (more on this in the next lecture).

PERPLEXITY

- The perplexity of a language models is defined as the inverse ($\frac{1}{P(\ldots)}$) of the probability of the test set, normalized ($\sqrt[N]{\ldots}$) by the # of tokens (N) in the test set.

- If a LM assigns probability $P(w_1, \ldots, w_N)$ to a test corpus $w_1\ldots w_N$, the LM's perplexity, $\underline{PP}(w_1\ldots w_N)$, is

$$PP(w_1...w_N) \quad = \quad \sqrt[N]{\frac{1}{P(w_1...w_N)}}$$

- A LM with **lower** perplexity is **better** because it assigns a higher probability to the unseen test corpus.

LM$_1$ and LM$_2$'s perplexity can only be compared if they use the same vocabulary
— Trigram models have lower perplexity than bigram models;
— Bigram models have lower perplexity than unigram models, etc.

PRACTICAL ISSUES: USE LOGARITHMS!

- Since language model probabilities are very small, multiplying them together often yields to underflow.

- It is often better to use logarithms instead, so replace

$$PP(w_1...w_N) =_{def} \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-1},...,w_{i-n+1})}}$$

with

$$PP(w_1...w_N) =_{def} \exp\left(-\frac{1}{N}\sum_{i=1}^{N} \log P(w_i|w_{i-1},...,w_{i-n+1})\right)$$

EXTRINSIC (TASK-BASED) EVALUATION OF LMS: WORD ERROR RATE

INTRINSIC VS. EXTRINSIC EVALUATION

Perplexity tells us which LM assigns a higher probability to unseen text

This doesn't necessarily tell us which LM is better for our task (i.e. is better at scoring candidate sentences)

Task-based evaluation:

- - Train model A, plug it into your system for performing task T
- - Evaluate performance of system A *on task T*.
- - Train model B, plug it in, evaluate system B on same task T.
- - Compare scores of system A and system B on task T.

WORD ERROR RATE (WER)

- Originally developed for speech recognition.

- How much does the *predicted* sequence of words differ from the *actual* sequence of words
- in the correct transcript?

$$WER = \frac{Insertions + Deletions + Substitutions}{Actual\ words\ in\ transcript}$$

Insertions: "eat lunch" $\rightarrow$ "eat **a** lunch"
Deletions: "see **a** movie" $\rightarrow$ "see movie"
Substitutions: "drink **ice** tea"$\rightarrow$ "drink **nice** tea"