CS 481

Introduction to NLP

January 31, 2023

Announcements / Reminders

- **Please follow the Week 03 TO DO List instructions**

- **Quiz #01: will be posted sometime before Tuesday lecture**

- **My office hours:**
  - **Tuesdays 11:30 AM - 01:30 PM in Stuart Building 217E or by appointment**

Plan for Today

- Other lexical resources
  - WordNet
- Probability and Bayes' Rule refresher
- N-grams and language models

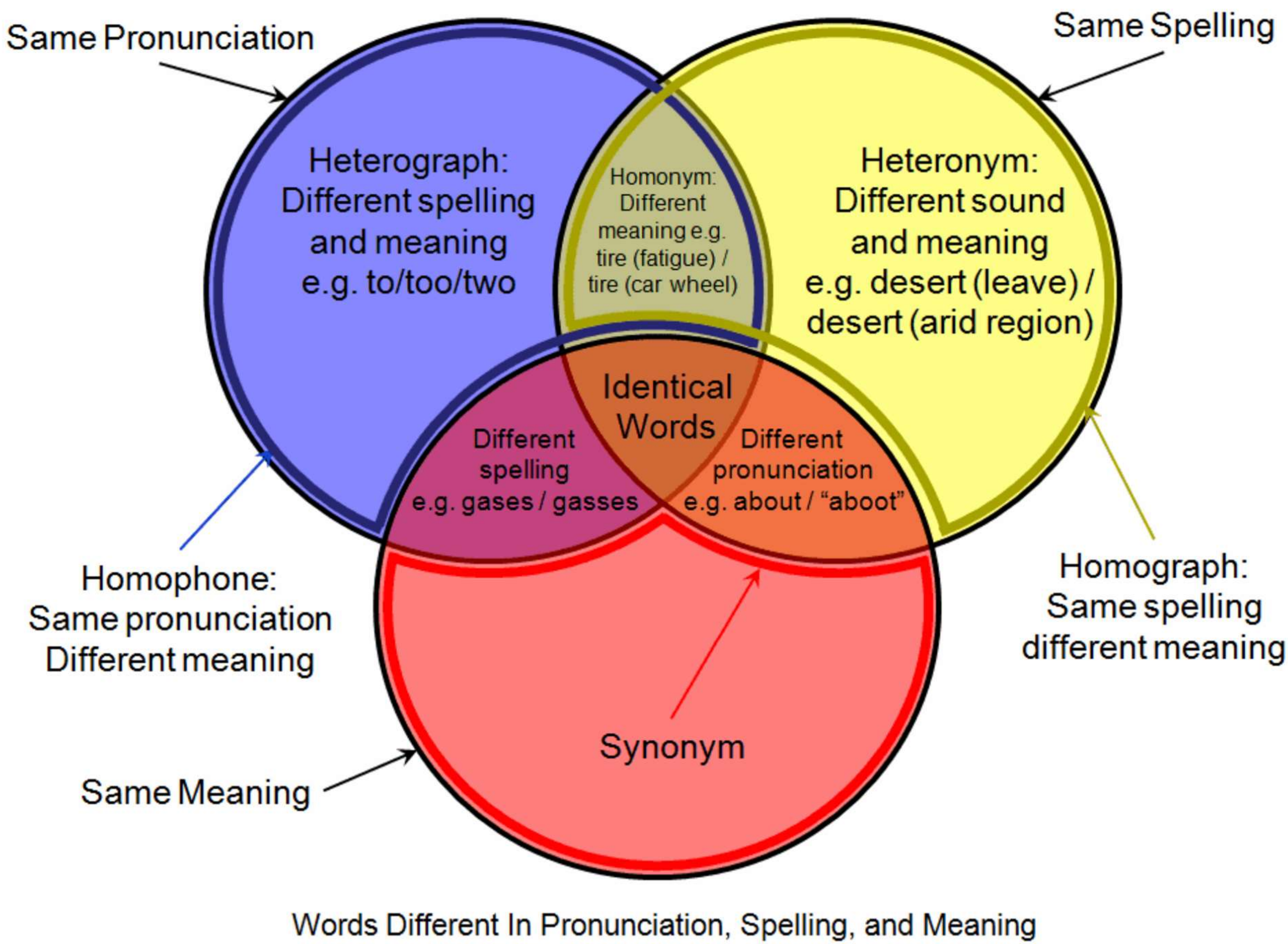
WordNet

WordNet® is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept.

Synsets are interlinked by means of conceptual-semantic and lexical relations.

Link: https://wordnet.princeton.edu/

Relationships Between Words



Source: https://owlcation.com/humanities/Lexical-Relations-Describing-Similarities-In-The-English-Language

Lexical Relationships

Lexical relationships are the connections established between one word and another:

- **Synonymy is the idea that some words have the same meaning as others**
  - *quick* **is similar to** *fast*
- **Antonymy is precisely the opposite of synonymy**
  - *good* **is the opposite** *bad*
- **Hyponymy is similar to the notion of embeddedness**
  - *Human* ← *Female* **(***Female* **is a more specific concept than** *Human***)**
- **Holonomy and Meronomy describe relationships between an object and its parts:**
  - *tree* **is a holonym of** *bark* **(***tree* **has bark)**
  - *bark* **is a meronym of** *tree* **(***bark* **is a part of** *tree***)**

Word Prediction

Words do not randomly appear in text.

The probability of a word appearing in a text is to a large degree related to the words that have appeared before it.

- **e. g.** *I'd like to make a collect. . .*
- *call* **is the most likely next word, but other words such as** *telephone*, *international*. **. . are also possible.**
- **other (very common) words are unlikely (e. g.** *dog*, *house*).

(Statistical) Language Model

- A (statistical) language model is a probability distribution over words or word sequences.
- In practice, a language model gives the probability of a certain word sequence being “valid”.
- Validity in this context does not need to mean grammatical validity at all.

Use lexical resources (corpora) to build LM.

Word Prediction

Words do not randomly appear in text.

The probability of a word appearing in a text is to a large degree related to the words that have appeared before it.

- **e. g.** *I'd like to make a collect. . .*
- *call* **is the most likely next word, but other words such as** *telephone*, *international*. . . **are also possible.**
- **other (very common) words are unlikely (e. g.** *dog*, *house*).

Word Prediction

- **Word prediction is very useful for applications such as:**
  - **speech recognition: it is possible to select between words that are hard for a speech recognizer to distinguish**
  - **augmentative communication for the disabled: speech generation systems can become more effective**
  - **spelling error detection:**
    - *They are leaving in about 15 minuets*.
    - *He is trying to fine out*.

- **Word prediction is also related to the problem of computing the probability of a sentence**

Words: Frequency and Rank

- **Frequency**: a the number of occurences of a word in the given document or corpus.
- **Rank**: position occupied by a word within a given document or a corpus. A word with the highest frequency will have the highest rank.

Simple Language Models

- **Probabilistic models of word sequence**
- **Simplest model:**
  - **every word may follow any other word**
  - **all words have equal probability**
- **More complex:**
  - **the probability of appearance of each word depends on its frequency in the corpus:**
    - *the* **appears 69 971 times in Brown corpus (7%)**
    - *rabbit* **appears 12 times (0.001%)**
- **But suppose we have the sentence:**
  - **Here comes the white. . .**

Probability Theory: Need to Know

- **What is an event A?**
- **What is the probability of event A occuring ($P(A)$)?**
- **What is a random variable X?**
- **What is the probability distribution for X?**
- **What is the probability density function for X?**
- **What are the expectation and variance of X?**

- **Check out https://seeing-theory.brown.edu/ for a refresher**

Probability Theory: Need to Know

- P(sure event) = 1 and
- P(impossible event) = 0
- **If** A, B **are exclusive events:** $P(A \vee B) = P(A) + P(B)$
- **If** A, B **are complementary events:** $P(A) + P(\neg A) = 1$
- **If** A, B **are arbitrary events:**

$$P(A \vee B) = P(A) + P(B) - P(A \wedge B)$$

- **If** $A \subseteq B$**, it is true that** $P(A) \leq P(B)$
- **If** $A_1, A_2, \ldots, A_n$ **are elementary events, then:**

$$\sum_{i=1}^{n} P(A_i) = 1$$

Prior (Unconditional) Probabilities

Degree of belief that some proposition A is true *in the absence of any other related information* is called unconditional or prior probability (or "prior" for short) $P(A)$.

Examples:

$$P(\text{isRaining})$$

$$P(\text{dieRoll} = 5)$$

$$P(\text{CS481FinalGrade} = \text{'A'})$$

$$P(\text{toothache})$$

Conditioning

Conditioning is a process of revising beliefs based on new evidence e:

- **start by taking all background information (prior probabilities) into account**
- **if new evidence e is acquired, a conditional probability of some proposition A given evidence e can be calculated (posterior probability):** $P(A \mid e)$

Posterior (Conditional) Probabilities

Typically, there is going to be some information, called evidence e, that affects our degree of belief about some proposition A being true. This allows us to also consider conditional or posterior probability (or "posterior" for short) $P(A \mid e)$.

Examples ($P(A \text{ given } e)$):

$$P(\text{isRaining} \mid \text{cloudy})$$

$$P(\text{CS481FinalGrade} = \text{'A'} \mid \text{CS481PA1Score} > 80)$$

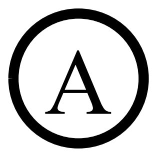
$$P(\text{cavity} \mid \text{toothache})$$

Evidence e

Evidence e **rules out possible worlds incompatible with** e**.**

Prior vs. Posterior Probabilities

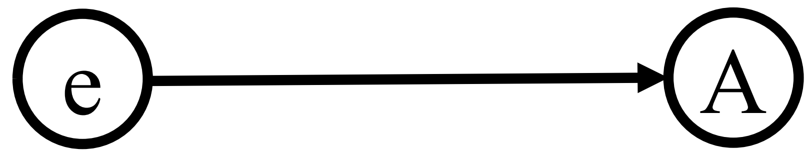
Prior Probability



$$P(A)$$

BTW: it is also $P(A \mid T)$

Posterior Probability



$$P(A \mid e)$$

Conditional Probability

$$P(A \mid B) = \frac{P(A \wedge B\ )}{P(B)}$$

where $\mathrm{P(B)} > 0$

Conditional Probability

$$P(A \mid evidence) = \frac{P(A \wedge evidence)}{P(evidence)}$$

where P(evidence) > 0

Conditional Probability (Product Rule)

$$\boldsymbol{P(A \wedge B) = P(A \mid B) * P(B)}$$

Conditional Probability (Product Rule)

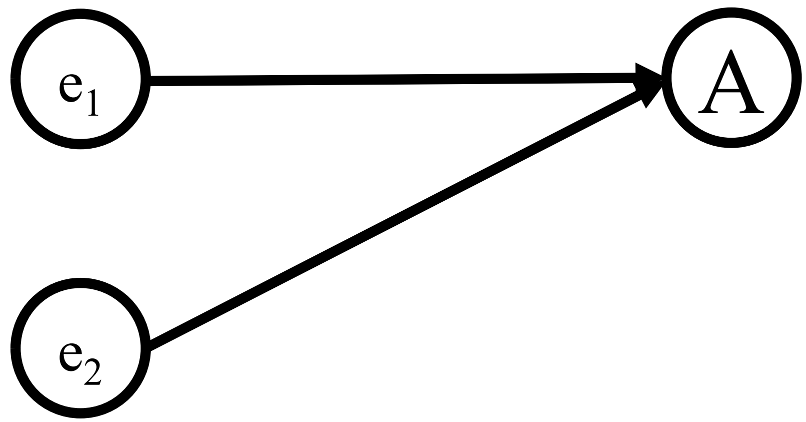
$$P(A \wedge evidence) \;=\; P(A \mid evidence) * \;P(evidence)$$

Prior vs. Posterior Probabilities

Prior Probability

$$P(A)$$

Posterior Probability



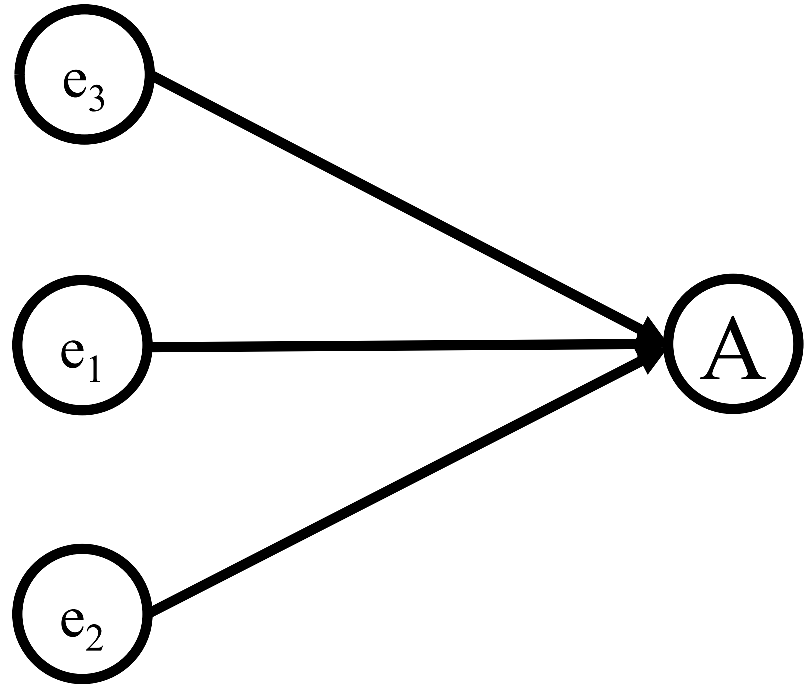
$$P(A \mid e_1 \wedge e_2)$$

Prior vs. Posterior Probabilities

Prior Probability

$$P(A)$$

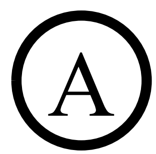
Posterior Probability



$$P(A \mid e_1 \wedge e_2 \wedge e_3)$$

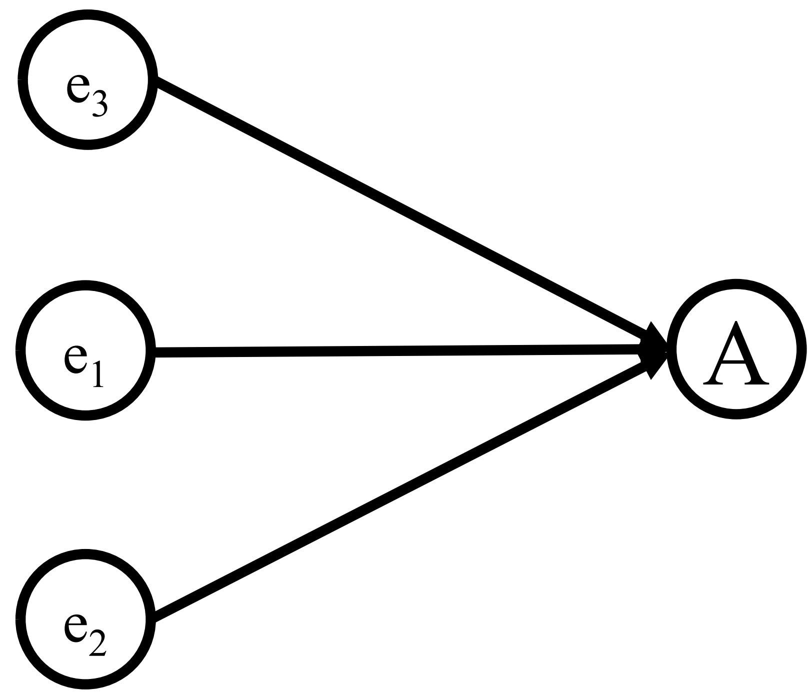
Prior vs. Posterior Probabilities

Prior Probability



$$P(A)$$

Posterior Probability



$$P(A \mid \text{parents}(A))$$

Marginal Probability

Marginal probability: the probability of an event occurring $\mathrm{P(A)}$.

It may be thought of as an unconditional probability.

It is not conditioned on another event.

Joint Probability

The probability of event A and event B occurring (or more than two events). It is the probability of the intersection of two or more events.

$$P(A \wedge B)$$

For any events $f_1, f_2, \ldots, f_n$:

$$P(f_1 \wedge f_2 \wedge \ldots \wedge f_n)$$

Chain Rule

Conditional probabilities can be used to decompose conjunctions using the chain rule. For any random events $f_1, f_2, \ldots, f_n$:

$$P(f_1 \wedge f_2 \wedge \ldots \wedge f_n) =$$

$$P(f_1) *$$

$$P(f_2 \mid f_1) *$$

$$P(f_3 \mid f_1 \wedge f_2) *$$

$$\ldots$$

$$P(f_n \mid f_1 \wedge \ldots \wedge f_{n-1}) =$$

$$= \prod_{i=1}^{n} P(f_i \mid f_1 \wedge \ldots \wedge f_{i-1})$$

Chain Rule

Conditional probabilities can be used to decompose conjunctions using the chain rule. For any random events $f_1, f_2, \ldots, f_n$:

$$P(f_1 = x_1 \wedge f_2 = x_2 \wedge \ldots \wedge f_n = x_n) =$$
$$P(f_1 = x_1) *$$
$$P(f_2 \mid f_1 = x_1) *$$
$$P(f_3 \mid f_1 = x_1 \wedge f_2 = x_2) *$$
$$\ldots$$
$$P(f_n = x_n \mid f_1 = x_1 \wedge \ldots \wedge f_{n-1} = x_{n\_1}) =$$
$$= \prod_{i=1}^{n} P(f_i = x_i \mid f_1 = x_1 \wedge \ldots \wedge f_{i-1} = x_{i\_1})$$

Bayes' Rule

$$P(A \mid B) = \frac{P(B \mid A) * P(A)}{P(B)}$$

Bayes' Rule

$$P(cause \mid effect) = \frac{P(effect \mid cause) * P(cause)}{P(effect)}$$

Bayes' Rule

$P(cause \mid effect)$ **diagnostic direction relation**

$$P(cause \mid effect) = \frac{P(effect \mid cause) * P(cause)}{P(effect)}$$

$P(effect \mid cause)$ **causal direction relation**

Bayes' Rule

$\boldsymbol{P(disease \mid symptoms)}$ **diagnostic direction relation**

$$P(disease \mid symptoms) = \frac{P(symptoms \mid disease) * P(disease)}{P(symptoms)}$$

$\boldsymbol{P(symptoms \mid disease)}$ **causal direction relation**

Bayes' Rule: Another Interpretation

Another way to think about Baye's rule: it allows us to update the hypothesis H in light of some new data/evidence e.

$$P(H \mid e) = \frac{P(e \mid H) * P(H)}{P(e)}$$

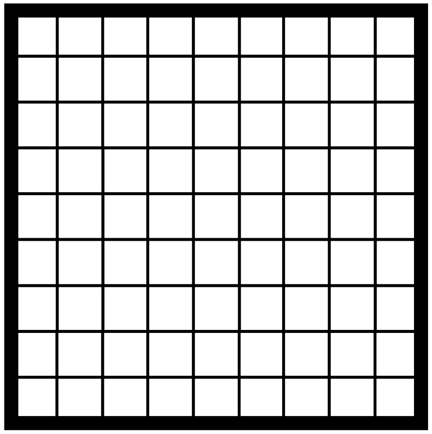
$$P(Hypothesis \mid evidence) = \frac{P(evidence \mid Hypothesis) * P(Hypothesis)}{P(evidence)}$$

where:

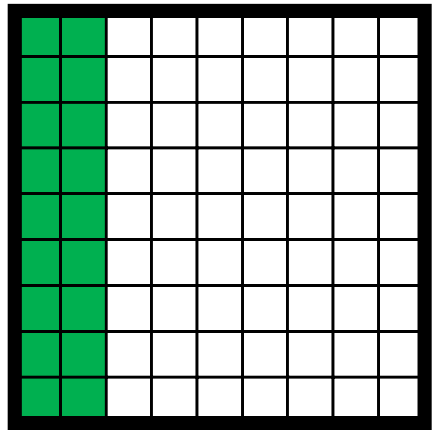
- P(H) - **probability of the** Hypothesis H **being true BEFORE we see new data/**evidence e **(prior probability)**
- P(H | e) - **probability of the** Hypothesis H **being true AFTER we see new data/**evidence e **(posterior probability)**
- P(e | H) - **probability of new data/**evidence e **being true under the** Hypothesis H **(likelihood)**
- P(e) - **probability of new data/**evidence e **being true under ANY hypothesis (normalizing constant)**

Bayes' Rule: Visual Interpretation

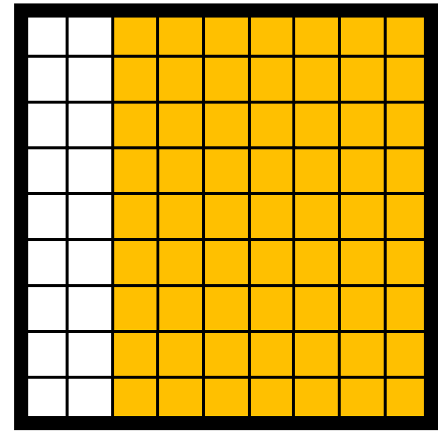
All possible cases



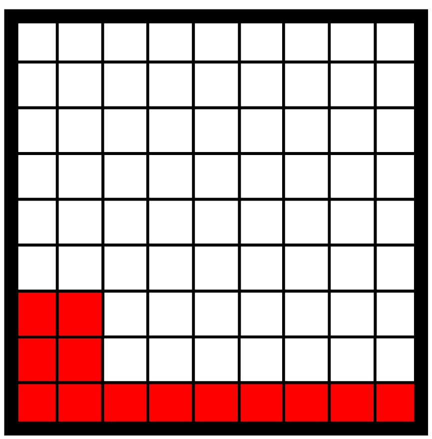
Cases where Hypothesis H **is true**
P(H)



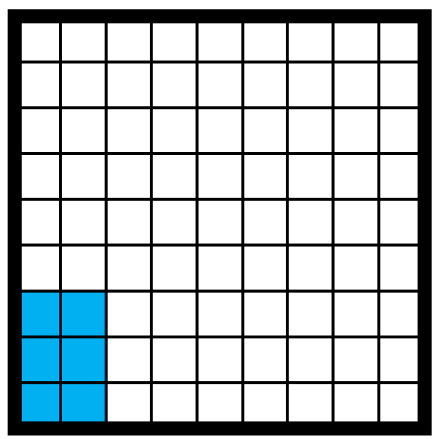
Cases where Hypothesis H **is false**
P(¬H)



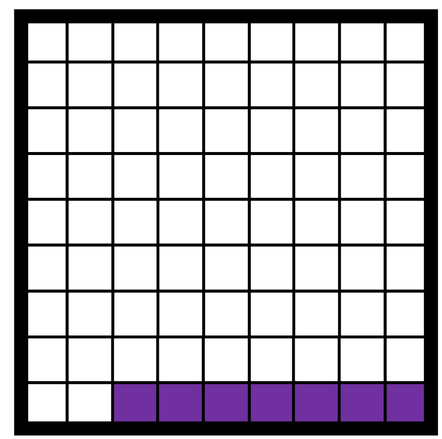
Cases where evidence e **is true**
P(e)



Cases where evidence e **is true given** Hypothesis H **true** P(e | H)



Cases where evidence e **is true given** Hypothesis H **false** P(e | ¬H)



Bayes' Rule: Visual Interpretation

Bayes' Rule:

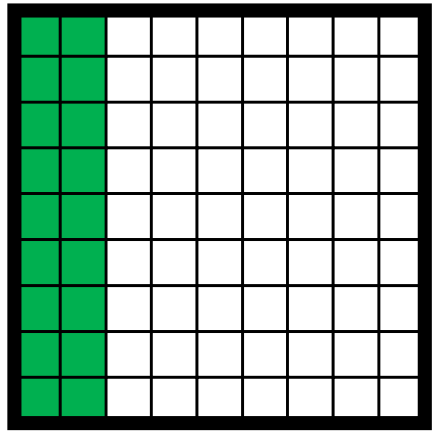
$$P(H \mid e) = \frac{P(e \mid H) * P(H)}{P(e)}$$

$$P(H \mid e) = \frac{P(e \mid H) * P(H)}{P(e)}$$

$$P(H \mid e) = \frac{P(e \mid H) * P(H)}{P(H) * P(e \mid H) + P(\neg H) * P(e \mid \neg H)}$$

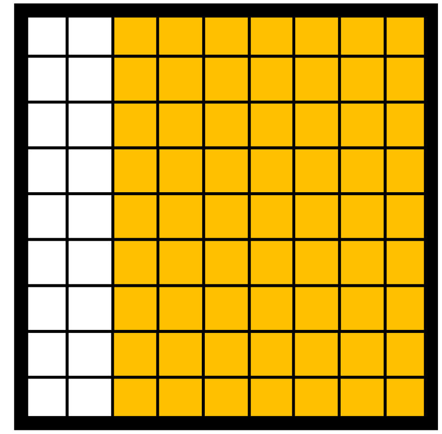
Cases where Hypothesis H is true

$P(H)$



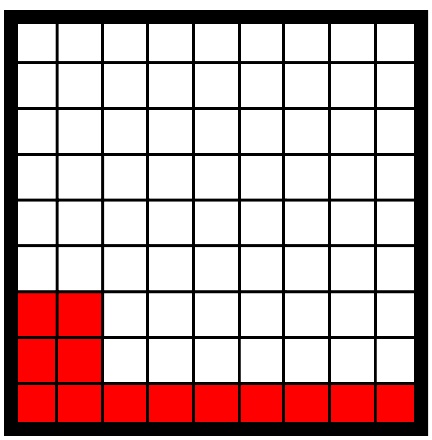
Cases where Hypothesis H is false

$P(\neg H)$

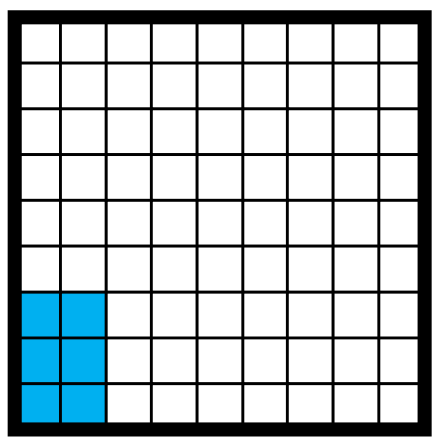


Cases where evidence e is true

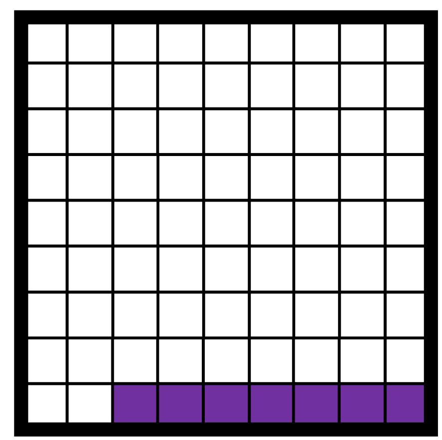
$P(e)$



Cases where evidence e is true given Hypothesis H true $P(e \mid H)$



Cases where evidence e is true given Hypothesis H false $P(e \mid \neg H)$



Bayes' Rule

$$P(cause \mid effect) = \frac{P(effect \mid cause) * P(cause)}{P(effect)}$$

Problem: a single card is drawn from a standard deck of cards. What is the probability that we drew a queen if we know that a face card (J, Q, K) was drawn?

$$P(queen \mid face) = \frac{P(face \mid queen) * P(queen)}{P(face)}$$

$$P(queen \mid face) = \frac{1 * 4 / 52}{12 / 52} = \frac{1}{3}$$

Simple Language Models

- **Probabilistic models of word sequence**
- **Simplest model:**
  - **every word may follow any other word**
  - **all words have equal probability**
- **More complex:**
  - **the probability of appearance of each word depends on its frequency in the corpus:**
    - *the* **appears 69 971 times in Brown corpus (7%)**
    - *rabbit* **appears 12 times (0.001%)**
- **But suppose we have the sentence:**
  - **Here comes the white. . .**

The Idea

- Examine short sequences of words
- How likely is each sequence?

What is an N-Gram

An N-gram is a subsequence of N items from a given sequence.

- **unigram: n-gram of size 1**
- **bigram (or Digram): n-gram of size 2**
- **trigram: n-gram of size 3**

Item:

- **phonemes**
- **syllables**
- **letters**
- **words**
- **anything else depending on the application.**

Bigrams and Trigrams: Examples

Bigram:

- "# the", "*the dog*", "dog smelled", "*smelled like*", "*like a*", "*a skunk*"

Trigram:

- "*# the dog*", "*the dog smelled*", "*dog smelled like*", "*smelled like a*", "*like a skunk*" **and** "*a skunk #*".

The Idea

- Examine short sequences of words
- How likely is each sequence?
- Use Markov assumption / property

Chain Rule

Conditional probabilities can be used to decompose conjunctions using the chain rule. For random variables $\boldsymbol{f_1, f_2, \ldots, f_n}$:

$P(f_1 \wedge f_2 \wedge \ldots \wedge f_n) =$

$P(f_1)$ *

$P(f_2 \mid f_1)$ *

$P(f_3 \mid f_1 \wedge f_2)$ *

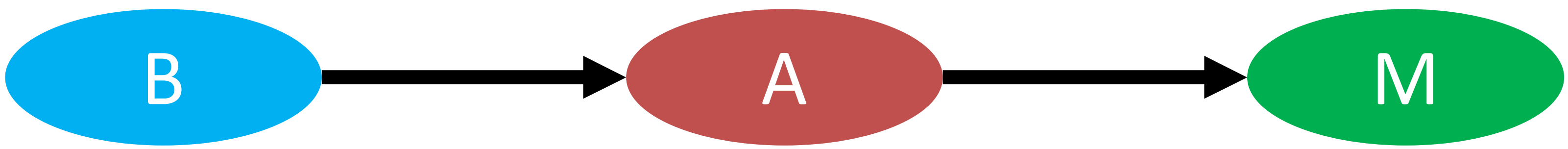
$\ldots$

$P(f_n \mid f_1 \wedge \ldots \wedge f_{n-1}) =$

$= \prod_{i=1}^{n} P(f_i \mid Parents(f_i))$ ← Enabled by conditional independence

Conditional Independence

Causal Chain:



$$P(M \mid A, B) = \frac{P(A, B, M)}{P(A, B)} = \frac{P(B) * P(A \mid B) * P(M \mid A)}{P(B) * P(A \mid B)} = P(M \mid A)$$

B and M are CONDITIONALLY independent given A.

If A is given, what "happened before" does not directly influence M.

The Idea: Markov Assumption

- **The probability of the appearance of a word depends on the words that have appeared before it.**

$$P(\text{rabbit} \mid \text{Just then the white})$$

- **Impossible to calculate this probability from a corpus. The exact word sequence would have to appear in the corpus.**
- **Markov simplifying assumption: we approximate the probability of a word given all the previous words with the probability given only the previous word.**

$$P(\text{rabbit} \mid \text{Just then the white}) \approx P(\text{rabbit} \mid \text{white})$$

Probabilistic Language Models

- **Goal: assign a probability to a sentence**
- **What for:**
  - **Machine Translation:**
    - P(*high winds tonight*) > P(*large winds tonight*)
  - **Spelling correction:**
    - **The office is about fifteen minuets from my house**
      - P(*about fifteen minutes from*) > P(*about fifteen minuets from*)
  - **Speech Recognition**
    - P(*I saw a van*) >> P(*eyes awe of an*)
  - **Summarization, question-answering, etc.**

Probabilistic Language Models

- **Task A: compute the joint probability of a sentence or sequence of words**

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

- **Task B: compute conditional probability of an upcoming word**

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

- **A model that can compute either**

$$P(W) \quad \textbf{or} \quad P(w_n \mid w_1, w_2, \ldots, w_{n-1})$$

is called a language model (LM).

$w_1, w_2, w_3, w_4, w_5, \ldots, w_n$ - words

Language Models Task A: Example

How can we compute this joint probability:

P("*computers are useless*")

Language Models Task A: Example

How about?

$$P(\text{“}\textit{computers are useless}\text{”}) = \frac{\text{count(“}\textit{computers are useless}\text{”)}}{\text{count(all English sentences ever spoken)}}$$

Language Models Task A: Example

How about?

$$P(\text{“}\textit{computers are useless}\text{”}) = \frac{\text{count(“}\textit{computers are useless}\text{”)}}{\text{count(all English sentences ever spoken)}}$$

Not really. We don’t have those counts or they are difficult to get.

Chain Rule

Conditional probabilities can be used to decompose conjunctions using the chain rule. For any words $w_1, w_2, \ldots, w_n$:

$$P(w_1 = x_1 \wedge w_2 = x_2 \wedge \ldots \wedge w_n = x_n) =$$

$$P(w_1 = x_1) *$$

$$P(w_2 \mid w_1 = x_1) *$$

$$P(w_3 \mid w_1 = x_1 \wedge w_2 = x_2) *$$

$$\ldots$$

$$P(w_n = x_n \mid w_1 = x_1 \wedge \ldots \wedge w_{n-1} = x_{n\_1}) =$$

$$= \prod_{i=1}^{n} P(w_i = x_i \mid w_1 = x_1 \wedge \ldots \wedge w_{i-1} = x_{i\_1})$$

Chain Rule

Conditional probabilities can be used to decompose conjunctions using the chain rule. For any words: $f_1, f_2, \ldots, f_n$ **:**

$P(f_1 = \text{"computers"} \wedge f_2 = \text{"are"} \wedge f_3 = \text{"useless"}) =$
$P(f_1 = \text{"computers"})$ *
$P(f_2 \mid f_1 = computers) *$
$P(f_3 \mid f_1 = \text{"computers"} \wedge f_2 = \text{"are"}) *$
...
$P(f_n = x_n \mid f_1 = x_1 \wedge \ldots \wedge f_{n-1} = x_{n\_1}) =$
$= \prod_{i=1}^{n} P(f_i = x_i \mid f_1 = x_1 \wedge \ldots \wedge f_{i-1} = x_{i\_1})$

Language Models Task A: Example

Now, let's try:

$$P(w_3 = \text{“}useless\text{”} \mid w_1 = \text{“}computers\text{”} \wedge w_2 = \text{“}are\text{”}) =$$

$$= P(\text{“}useless\text{”} \mid \text{“}computers\ are\text{”}) =$$

$$= P(useless \mid computers\ are) =$$

$$= \frac{count(computers\ are\ useless)}{\sum_{x \in V} count(computers\ are\ x)} =$$

$$= \frac{count(computers\ are\ useless)}{count(computers\ are)}$$

where: V **- vocabulary**

Not really. Too many possibilities and/or not enough data in corpora.

The Idea: Markov Assumption

- **The probability of the appearance of a word depends on the words that have appeared before it.**

$$P(\text{useless} \mid \text{computers are})$$

- **Impossible to calculate this probability from a corpus. The exact word sequence would have to appear in the corpus.**
- **Markov simplifying assumption: we approximate the probability of a word given all the previous words with the probability given only the previous word.**

$$P(\text{useless} \mid \text{computers are}) \approx P(\text{useless} \mid \text{are})$$

Chain Rule /w N-gram Approximation

Conditional probabilities can be used to decompose conjunctions using the chain rule. For any words $w_1, w_2, \ldots, w_n$:

$$P(w_1 = x_1 \wedge w_2 = x_2 \wedge \ldots \wedge w_n = x_n) =$$
$$P(w_1 = x_1) *$$
$$P(w_2 \mid w_1 = x_1) *$$
$$P(w_3 \mid w_1 = x_1 \wedge w_2 = x_2) *$$
$$\ldots$$
$$P(w_n \mid w_1 \wedge w_2 \wedge \ldots \wedge w_{n-1}) =$$
$$\approx \prod_{i=1}^{n} P(w_i \mid w_1 \wedge w_2 \wedge \ldots \wedge w_{n-N+1})$$

Chain Rule

Conditional probabilities can be used to decompose conjunctions using the chain rule. For any words $w_1, w_2, \ldots, w_n$:

$$P(w_1 = x_1 \wedge w_2 = x_2 \wedge \ldots \wedge w_n = x_n) =$$

$$P(w_1 = x_1) *$$

$$P(w_2 \mid w_1 = x_1) *$$

$$P(w_3 \mid w_1 = x_1 \wedge w_2 = x_2) *$$

$$\ldots$$

$$P(w_n = x_n \mid w_1 = x_1 \wedge \ldots \wedge w_{n-1} = x_{n\_1}) =$$

$$= \prod_{i=1}^{n} P(w_i = x_i \mid w_1 = x_1 \wedge \ldots \wedge w_{i-1} = x_{i\_1})$$

How do we get (estimate) these in practice?

Probabilities and Their Estimates

- **Probability of a single word (token) occuring**

$$P(word) \approx \frac{count(word)}{count(all\,words\,/\,tokens)}$$

- **Probability of a sequence of words (tokens) occurring (where w_i - ith word / token)**

By chain rule:

$$P(w_1 = x_1 \wedge w_2 = x_2 \wedge \ldots \wedge w_n = x_n) = \prod_{i=1}^{n} P(w_i = x_i \,|\, w_1 = x_1 \wedge \ldots \wedge w_{i-1} = x_{i\,\_\,1})$$

$$P(first, second, .., \quad nth) = \prod_{i=1}^{n} P(ith \,|\, all\,words\,preceding\,ith)$$

By Markov assumption:

$$P(next \,|\, all\,words\,preceding\,next) \approx P(next \,|\, N - gram\,preceding\,next)$$

Probabilities and Their Estimates

Probability of a sequence of words (tokens), an N-gram ($[\boldsymbol{last\ K\ words\ before\ next}], \boldsymbol{next}$), occurring:

$$\boldsymbol{P(next \mid last\ K\ words\ before\ next)} =$$

$$= \frac{\boldsymbol{count([last\ K\ words\ before\ next]\ next)}}{\sum_{x \in V} \boldsymbol{count([last\ K\ words\ before\ next]\ x)}} =$$

$$= \frac{\boldsymbol{count([last\ K\ words\ preceding\ next]\ next)}}{\boldsymbol{count([last\ K\ words\ preceding\ next])}}$$

Probabilities and Their Estimates

Probability of a sequence of words (tokens), an N-gram ($[\boldsymbol{prefix}], \boldsymbol{next}$), occurring:

$$\boldsymbol{P}(\boldsymbol{next} \mid [\boldsymbol{prefix}]\ \boldsymbol{next}) = \frac{\boldsymbol{count}([\boldsymbol{prefix}]\ \boldsymbol{next})}{\sum_{\boldsymbol{x} \in \boldsymbol{V}} \boldsymbol{count}([\boldsymbol{prefix}]\ \boldsymbol{x})} =$$

$$= \frac{\boldsymbol{count}([\boldsymbol{prefix}]\ \boldsymbol{next})}{\boldsymbol{count}(\boldsymbol{prefix})} =$$

$$= \frac{\boldsymbol{observed\ frequency\ of\ a}\ ([\boldsymbol{prefix}] + \boldsymbol{next})\ \boldsymbol{sentence}}{\boldsymbol{observed\ frequency\ of\ a\ prefix}} =$$

$$= \boldsymbol{relative\ frequency\ of}([\boldsymbol{prefix}] + \boldsymbol{next})$$

Unigram Language Model

Zeroth-order Markov Assumption:

$$P(w_1, w_2, \ldots, w_n) \approx \prod_{i=1}^{n} P(w_i)$$

Bigram Language Model

First-order Markov Assumption:

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

General Maximum Likelihood Estimation (MLE) of an 2-gram (bigram):

$$P(w_N \mid w_{N-1}) = \frac{count(w_{N-1}, w_N)}{count(w_{N-1})}$$

where:

w_i - **ith word / token**

Trigram Language Model

Second-order Markov Assumption:

$$P(w_i \mid w_1, w_2, \ldots, w_{i-1}) \approx P(\boldsymbol{w_i} \mid \boldsymbol{w_{i-2}}, \boldsymbol{w_{i-1}})$$

General Maximum Likelihood Estimation (MLE) of an 2-gram (bigram):

$$P(\boldsymbol{w_N} \mid \boldsymbol{w_{N-2}}, \boldsymbol{w_{N-1}}) = \frac{\boldsymbol{count}(\boldsymbol{w_{N-2}}, \boldsymbol{w_{N-1}}, \boldsymbol{w_N})}{\boldsymbol{count}(\boldsymbol{w_{N-2}}, \boldsymbol{w_{N-1}})}$$

where:

w_i **- ith word / token**

N-gram Language Models

General Maximum Likelihood Estimation (MLE) of an N-gram:

$$P(w_N \mid w_{N-K+1}, w_{N-K+2}, \ldots, w_{N-1}) = \frac{count(w_{N-K+1}, w_{N-K+2}, \ldots, w_{N-1}, w_N)}{count(w_{N-K+1}, w_{N-K+2}, \ldots, w_{N-1})}$$

where:

w_i - **ith word / token**

In MLE, the resulting parameter set maximizes the likelihood of the training set T given the model M (i.e., P(T | M)).

N-gram Language Models

- **We can extend to 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."

- **But in many cases N-gram models suffice**

Example: A Simple Corpus

Consider the following corpus (three sentences):

I am Sam

Sam I am

I do not like green eggs and ham

Example: A Simple Corpus

Let's add sentence start / end tokens:

<s> *I am Sam* </s>

<s> *Sam I am* </s>

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

This will help us create "starts with" and "ends with" bigrams.

Example: A Simple Corpus

Let's add sentence start / end tokens:

<s> *I am Sam* </s>

<s> *Sam I am* </s>

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

Let's list all <u>unique</u> unigrams first:

I, am, Sam, do, not, like, green, eggs, and, ham

And let's count their occurences:

	I	am	Sam	do	not	like	green	eggs	and	ham
c(word)	3	2	2	1	1	1	1	1	1	1

Example: A Simple Corpus

Given our corpus:

<s> *I* am Sam </s>

<s> *Sam I am* </s>

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

and word token frequency counts:

	I	am	Sam	do	not	like	green	eggs	and	ham
c(word)	3	2	2	1	1	1	1	1	1	1

Bigram probability estimate:

$$P(Sam \mid am) = P(w_N \mid w_{N-1}) = \frac{count(w_{N-1}, w_N)}{count(w_{N-1})} = \frac{count(am, Sam)}{count(am)} = \frac{1}{2}$$

Example: A Simple Corpus

Given our corpus:

<s> *I am* *Sam* </s>

<s> *Sam* *I am* </s>

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

and word token frequency counts:

	I	am	Sam	do	not	like	green	eggs	and	ham
c(word)	3	2	2	1	1	1	1	1	1	1

Bigram probability estimate:

$$P(am \mid I) = P(w_N \mid w_{N-1}) = \frac{count(w_{N-1}, w_N)}{count(w_{N-1})} = \frac{count(I, am)}{count(I)} = \frac{2}{3}$$

Example: A Simple Corpus

Given our corpus:

<s> *I am Sam* </s>

<s> *Sam I am* </s>

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

and word token frequency counts:

	I	am	Sam	do	not	like	green	eggs	and	ham
c(word)	3	2	2	1	1	1	1	1	1	1

Bigram probability estimate:

$$P(do \mid I) = P(w_N \mid w_{N-1}) = \frac{count(w_{N-1}, w_N)}{count(w_{N-1})} = \frac{count(I, do)}{count(I)} = \frac{1}{3}$$

Example: A Simple Corpus

Given our corpus:

<s> I am Sam </s>

<s> Sam I am </s>

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

and word token frequency counts:

	I	am	Sam	do	not	like	green	eggs	and	ham
c(word)	3	2	2	1	1	1	1	1	1	1

Bigram probability estimate:

$$P(I \mid <s>) = P(w_N \mid w_{N-1}) = \frac{count(w_{N-1}, w_N)}{count(w_{N-1})} = \frac{count(<s>, I)}{count(<s>)} = \frac{2}{3}$$

Example: A Simple Corpus

Given our corpus:

</s> I am Sam </s>

</s> Sam I am </s>

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

and word token frequency counts:

	I	am	Sam	do	not	like	green	eggs	and	ham
c(word)	3	2	2	1	1	1	1	1	1	1

Bigram probability estimate:

$$P(</s> \mid Sam) = P(w_N \mid w_{N-1}) = \frac{count(I, do)}{count(I)} = \frac{count(Sam, </s>)}{count(Sam)} = \frac{1}{2}$$

Example: A More Complex Corpus

The Berkeley Restaurant Project (BeRP) was a testbed for a speech recognition system developed by the International Computer Science Institute (ICSI) in Berkeley, CA, USA, in the 1990's.

The BeRP system was designed to be an automated consultant whose domain of knowledge was restaurants in the city of Berkeley. The system served as a testbed for several research projects, including robust feature extraction, neural-net based phonetic likelihood estimation, automatic induction of multiple pronunciation lexicons, foreign accent detection and modeling, advanced language models, and lip-reading.

Example: A More Complex Corpus

Selected sentences from BeRP:

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

Example: A More Complex Corpus

Selected sentences from BeRP (9332 sentences, V = 1446 words):

<s> can you tell me about any good cantonese restaurants close by </s>
<s> mid priced thai food is what i'm looking for </s>
<s> tell me about chez panisse </s>
<s> can you give me a listing of the kinds of food that are available </s>
<s> i'm looking for a good place to eat breakfast </s>
<s> when is caffe venezia open during the day </s>

with "sentence start" and "sentence end" tokens.

BeRP Selected Bigrams: Counts

BeRP bigram **counts** for selected words/tokens:

	i	want	to	eat	chinese	food	lunch	spend
i	5	$\boxed{827}$	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

Recall the formula for MLE of a bigram:

$$P(w_N \mid w_{N-1}) = \frac{count(w_{N-1}, w_N)}{count(w_{N-1})} \leftarrow \boxed{827} = \text{count}(\textit{i}, \textit{want})$$

BeRP Selected Bigrams: Counts

BeRP bigram counts for selected words/tokens:

	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

Recall the formula for MLE of a bigram:

$$P(w_N \mid w_{N-1}) = \frac{count(w_{N-1}, w_N)}{count(w_{N-1})} \leftarrow \boxed{2}$$

= count(*want*, *i*)

BeRP Selected Bigrams: Counts

BeRP bigram counts for selected words/tokens:

	i	want	to	eat	chinese	food	lunch	spend
i	$\boxed{5}$	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

Recall the formula for MLE of a bigram:

$$P(w_N \mid w_{N-1}) = \frac{count(w_{N-1}, w_N)}{count(w_{N-1})} \leftarrow \boxed{5} = \text{count}(i, i)$$

Corpus Bigram Counts: Normalizing

Let's try to turn counts into probabilities:

	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

Unigram counts (within the corpus) for all the words above:

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

Corpus Bigram Counts: Normalizing

Let’s try to turn counts into probabilities:

	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

Recall the formula for MLE of a bigram:

$$P(w_N \mid w_{N-1}) = \frac{count(w_{N-1}, w_N)}{count(w_{N-1})}$$

Corpus Bigram Counts: Normalizing

Normalizing (using (N-1)-gram counts):

	i	want	to	eat	chinese	food	lunch	spend
i	5/2533	827/2533	0/2533	9/2533	0/2533	0/2533	0/2533	2/2533
want	2/927	0/927	608/927	1/927	6/927	6/927	5/927	1/927
to	2/2417	0/2417	4/2417	686/2417	2/2417	0/2417	6/2417	211/2417
eat	0/746	0/746	2/746	0/746	16/746	2/746	42/746	0/746
chinese	1/153	0/153	0/153	0/153	0/153	82/153	1/153	0/153
food	15/1093	0/1093	15/1093	0/1093	1/1093	4/1093	0/1093	0/1093
lunch	2/341	0/341	0/341	0/341	0/341	1/341	0/341	0/341
spend	1/278	0/278	1/278	0/278	0/278	0/278	0/278	0/278

Unigram counts (within the corpus) for all the words above:

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

Corpus Bigram Counts: Normalizing

Normalizing:

	i	want	to	eat	chinese	food	lunch	spend
i	5/2533	827/2533	0/2533	9/2533	0/2533	0/2533	0/2533	2/2533
want	2/927	0/927	608/927	1/927	6/927	6/927	5/927	1/927
to	2/2417	0/2417	4/2417	686/2417	2/2417	0/2417	6/2417	211/2417
eat	0/746	0/746	2/746	0/746	16/746	2/746	42/746	0/746
chinese	1/153	0/153	0/153	0/153	0/153	82/153	1/153	0/153
food	15/1093	0/1093	15/1093	0/1093	1/1093	4/1093	0/1093	0/1093
lunch	2/341	0/341	0/341	0/341	0/341	1/341	0/341	0/341
spend	1/278	0/278	1/278	0/278	0/278	0/278	0/278	0/278

Unigram counts:

	i	want
c(word)	2533	927

$$P(w_N \mid w_{N-1}) = \frac{count(w_{N-1}, w_N)}{count(w_{N-1})}$$

5 / 2533

Corpus Bigram Counts: Normalizing

Normalizing:

	i	want	to	eat	chinese	food	lunch	spend
i	5/2533	827/2533	0/2533	9/2533	0/2533	0/2533	0/2533	2/2533
want	2/927	0/927	608/927	1/927	6/927	6/927	5/927	1/927
to	2/2417	0/2417	4/2417	686/2417	2/2417	0/2417	6/2417	211/2417
eat	0/746	0/746	2/746	0/746	16/746	2/746	42/746	0/746
chinese	1/153	0/153	0/153	0/153	0/153	82/153	1/153	0/153
food	15/1093	0/1093	15/1093	0/1093	1/1093	4/1093	0/1093	0/1093
lunch	2/341	0/341	0/341	0/341	0/341	1/341	0/341	0/341
spend	1/278	0/278	1/278	0/278	0/278	0/278	0/278	0/278

Unigram counts:

	i	want
c(word)	2533	927

$$P(w_N \mid w_{N-1}) = \frac{count(w_{N-1}, w_N)}{count(w_{N-1})}$$

2 / 927

Bigram Probability Estimates

Bigram probability estimates after normalizing:

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.022	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	0.0014	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 Probability Estimates

Bigram probability estimates after normalizing:

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.022	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	0.0014	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

Let's introduce a couple more probabilities:

P(*i* | *<s>*) = 0.25

P(*food* | *english*) = 0.5

P(*english* | *want*) = 0.0011

P(*</s>* | *food*) = 0.68

Bigram Probability Estimates

Bigram probability estimates after normalizing:

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.022	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	0.0014	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

Now we are ready to calculate probability of some sentence:

$$P(first, second, .., \quad nth) = \prod_{i=1}^{n} P(ith \mid all\ words\ preceding\ ith)$$

Bigram Probability Estimates

Bigram probability estimates after normalizing:

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.022	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	0.0014	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

Now we are ready to calculate probability of some sentence *S1* ("*I want English food*"):

$$\boldsymbol{P}(\boldsymbol{S1}) = \boldsymbol{P}(<\mathrm{s}>, \mathrm{I}, \mathrm{want}, \mathrm{english}, \mathrm{food}, </\mathrm{s}>) = \prod_{i=1}^{n} \boldsymbol{P}(\boldsymbol{ith} \mid \boldsymbol{all\ words\ preceding\ ith})$$

Bigram Probability Estimates

Bigram probability estimates after normalizing:

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	**0.33**	0	0.0036	0	0	0	0.00079
want	0.022	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	0.0014	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

Now we are ready to calculate probability of some sentence *S1* ("*I want English food*"):

$$P(S1) = P(i \mid <s>) * P(want \mid i) * P(english \mid want) * P(food \mid english) * P(</s> \mid food)$$
$$= 0.25 * 0.33 * 0.0011 * 0.5 * 0.68 \approx 0.000031$$

Bigram Probability Estimates

Bigram probability estimates after normalizing:

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.022	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	0.0014	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

Let's calculate probability of another sentence *S2* ("*I want Chinese food*"):

$$\boldsymbol{P}(\boldsymbol{S2}) = \boldsymbol{P}(< \mathrm{s} >, \mathrm{I}, \mathrm{want}, \mathbf{chinese}, \mathrm{food}, </\mathrm{s} >) = \prod_{i=1}^{n} \boldsymbol{P}(\boldsymbol{ith} \mid \boldsymbol{all\ words\ preceding\ ith})$$

Bigram Probability Estimates

Bigram probability estimates after normalizing:

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	**0.33**	0	0.0036	0	0	0	0.00079
want	0.022	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	0.0014	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

Let's calculate probability of another sentence *S2* ("*I want Chinese food*"):

$$\boldsymbol{P(S2) = P(i \mid <s>) * P(want \mid i) * P(chinese \mid want) * P(food \mid chinese) * P(</s> \mid food)}$$
$$\boldsymbol{= 0.25 * 0.33 * 0.0065 * 0.52 * 0.68 \approx 0.00019}$$

Bigram Probability Estimates

Bigram probability estimates after normalizing:

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.022	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	0.0014	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

Which sentence is more likely: "*I want English food*" or "*I want Chinese food*"?

$$\boldsymbol{P(S1) = P(I\ want\ English\ food)\ \approx 0.000031}$$
$$\boldsymbol{P(S2) = P(I\ want\ Chinese\ food)\ \approx 0.00019}$$

In Practice: Calculations

- **Perform calculations in log space**

  log(P1 * P2 * P3 * P4) = log P1 + log P2 + log P3 + log P4

  - **adding faster than multiplying**
  - **avoids underflow**

How Good Is Your Model?

- **Does our language model prefer good sentences to bad ones?**
  - **Assigns higher probability to “real” or “frequently observed” sentences (as opposed to ”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: unseen dataset that is different from training set**
  - **an evaluation metric tells us how well our model does on the test set.**

Extrinsic Evaluation of N-gram Models

- Best evaluation for comparing models A and B
  - apply each model to a specific task (spelling corrector, speech recognizer, machine translation)
  - Run the task, get accuracy for both A and B
    - How many misspelled words corrected properly?
    - How many words translated correctly?
  - Compare accuracy for A and B

Extrinsic Evaluation of N-gram Models

- Extrinsic evaluation
  - Time-consuming; can take days or weeks
  - Bad approximation
    - unless the test data looks just like the training data
    - generally only useful in pilot experiments
  - But is helpful to do

- Alternatives:
  - use intrinsic evaluation: perplexity

Intrinsic Evaluation: Perplexity

- **The best language model is the one that best predicts an unseen test set**
  - **Gives the highest P(*sentence*)**
- **Perplexity is the inverse probability of the test set, normalized by the number of words N:**

$$PP(W) = P(w_1 w_2 \dots w_N)^{-\frac{1}{N}}$$

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \dots w_N)}}$$

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

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

Minimizing perplexity is the same as maximizing probability

In Practice: Sparse Matrix

BeRP bigram counts for selected words/tokens:

	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

Note: This is a sparse matrix (lots of zeros)!

In Practice: Zeros

Training set:

… denied the allegations
… denied the reports
… denied the claims
… denied the request

P(*offer* | *denied the*) = 0

Test set:

… denied the offer
… denied the loan

In Practice: Smoothing

- Smoothing removes zero-probabilities
- Smoothing assigns probabilities to unseen events

- There are many smoothing algorithms:
  - simple: Laplace / Add One
  - “stupid backoff”
  - advanced: Extended Interpolated Kneser-Nay

Laplace / Add One Smoothing

Pretend you saw everything +1 times:

	i	want	to	eat	chinese	food	lunch	spend
i	5+1	827+1	0+1	9+1	0+1	0+1	0+1	2+1
want	2+1	0+1	608+1	1+1	6+1	6+1	5+1	1+1
to	2+1	0+1	4+1	686+1	2+1	0+1	6+1	211+1
eat	0+1	0+1	2+1	0+1	16+1	2+1	42+1	0+1
chinese	1+1	0+1	0+1	0+1	0+1	82+1	1+1	0+1
food	15+1	0+1	15+1	0+1	1+1	4+1	0+1	0+1
lunch	2+1	0+1	0+1	0+1	0+1	1+1	0+1	0+1
spend	1+1	0+1	1+1	0+1	0+1	0+1	0+1	0+1

Updated unigram probability:

$$P_{addOne}(w_N) = \frac{count(w_N) + 1}{count(all\ words/tokens) + number\ of\ unique\ words/tokens\ V}$$

In Practice: Out-Of-Vocabulary Words

- If we know all the words in advance, so the vocabulary V is fixed
  - closed vocabulary task
- In practice often we don’t know the entire vocabulary
  - Out Of Vocabulary = OOV words
  - open vocabulary task

- Potential solution: create an unknown word token <UNK>
  - Training of <UNK> probabilities
    - Create a fixed lexicon L of size V
    - At text normalization phase, any training word not in L changed to <UNK>
    - Now we train its probabilities like a normal word
  - At decoding time
    - If text input: Use <UNK> probabilities for any word not in training