Natural Language Processing - 2nd Semester (2024-2025)
1038141

1.7 - Language models

SAPIENZA
UNIVERSITÀ DI ROMA

Prof. Stefano Faralli
faralli@di.uniroma1.it

Prof. Iacopo Masi
masi@di.uniroma1.it

**credits are reported in the last slide

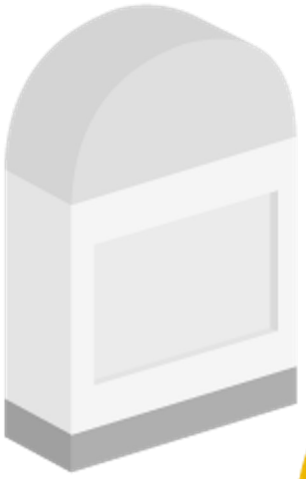
SAPIENZA
UNIVERSITÀ DI ROMA

1.7 - Language models

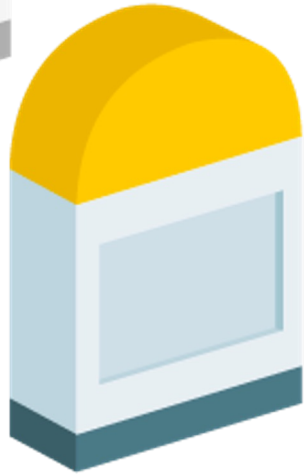
- Why Language models?, Applications
- Language Models, Probabilities
- Probabilities estimations, n-gram models
- Zipf's Law, Evaluation, Closed/Open vocabulary
- Intrinsic vs. Extrinsic evaluation, Perplexity,
- Smoothing
- Model Combination: interpolation, back-off
- Exercises
- Q&A

Milestones in NLP

today topic is mainly related to statistical/probabilistic models but is at the base of machine and deep learning.



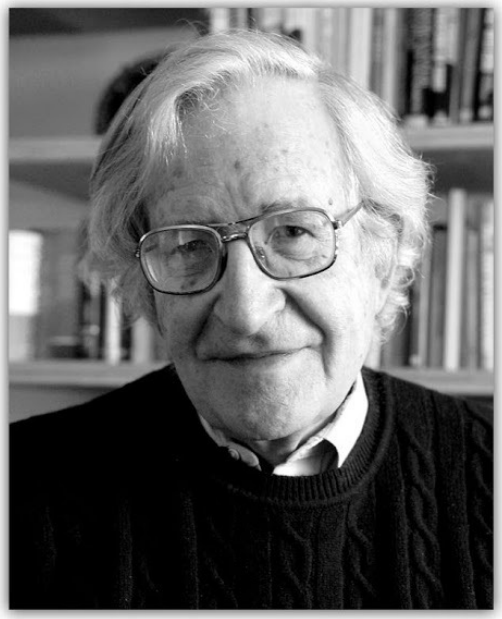
rule-based systems



statistical classical machine learning models

deep learning models

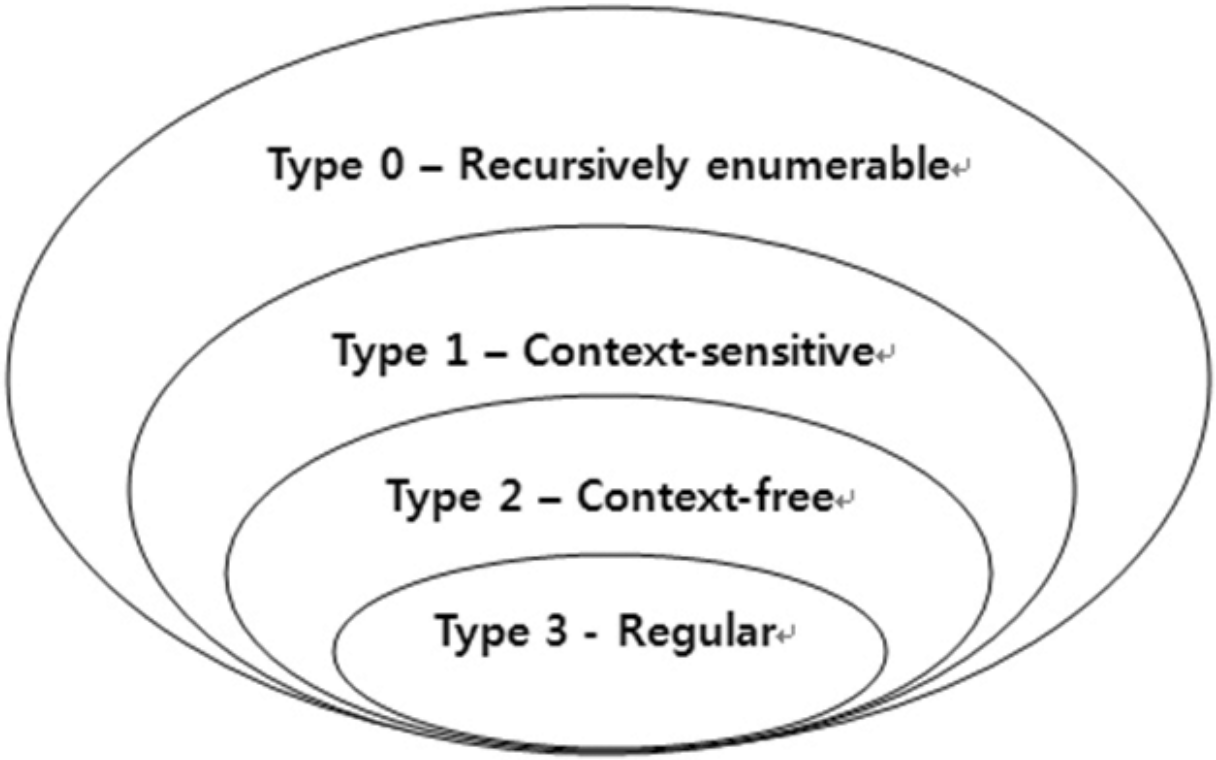
Languages



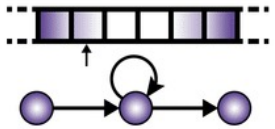
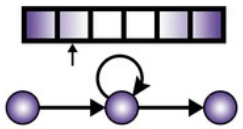
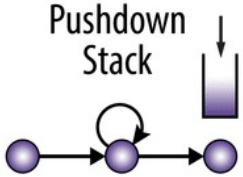
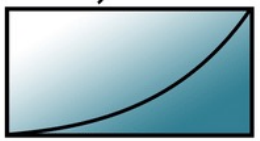
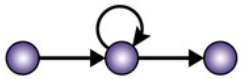
> **Language is a process of free creation; its laws and principles are fixed, but the manner in which the principles of generation are used is free and infinitely varied. Even the interpretation and use of words involves a process of free creation.**
>
> *~ Noam Chomsky*

https://devopedia.org/chomsky-hierarchy

Languages


Type 0 – Recursively enumerable
Type 1 – Context-sensitive
Type 2 – Context-free
Type 3 - Regular

Languages

Language	Automaton	Grammar	Recognition
Recursively Enumerable Languages	Turing Machine 	Unrestricted $Baa \rightarrow A$	Undecidable 
Context-Sensitive Languages	Linear Bounded 	Context Sensitive $A\,t \rightarrow aA$	Exponential? 
Context-Free Languages	Pushdown Stack 	Context Free $S \rightarrow gS\,c$	Polynomial 
Regular Languages	Finite-State Automaton 	Regular $A \rightarrow cA$	Linear 

Formal Grammars

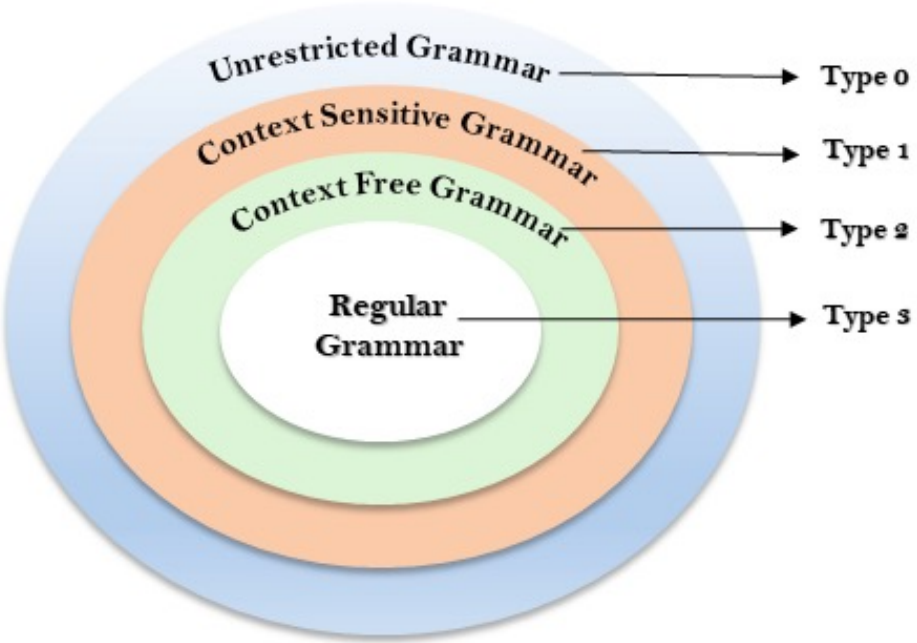
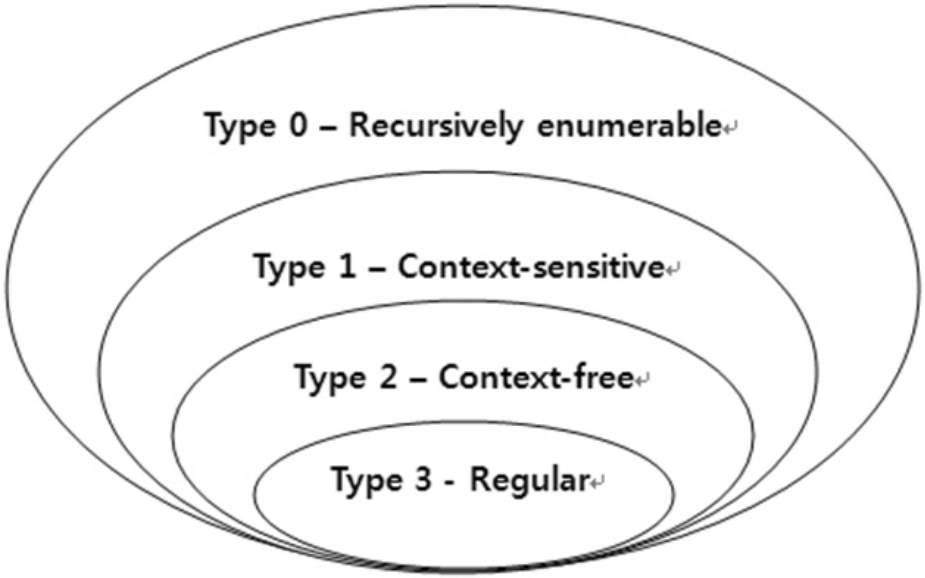


Fig: Chomsky Hierarchy



Language Models

definition:

Language models are **probabilistic distributions** over sequences of words.

Applications of Language Models

- Speech recognition
  - "I ate a cherry" is a more likely sentence than "Eye eight uh Jerry"
- OCR & Handwriting recognition
  - More probable sentences are more likely correct readings
- Machine translation
  - More likely sentences are probably better translations
- Generation
  - More likely sentences are probably better NL generations
- Context sensitive spelling correction
  - "Their are problems wit this sentence."

Why probabilistic distributions?

- Formal grammars (e.g., **regular**, context free) give a **hard "binary"** model of the legal sentences in a language
- For NLP, a ***probabilistic*** model of a language that gives a probability that a string is a member of a language is more useful

Language Models

definition:

Language models are probabilistic distributions over sequences of words.

example:

- P("and nothing but the truth") ≈ 0.001
- P("and supercalifragilistichespiralidoso but the truth") ≈ 0

… or better: *close* to 0 (we will talk about handling zeros later)

Language Models

definition:

Language models are probabilistic distributions over sequences of words.

example:

- P("and nothing but the truth") ≈ 0.001
- P("and supercalifragilistichespiraldoso but the truth") ≈ 0

… or better: *close* to 0 (we will talk about handling zeros later)

Language Models

definition:

Language models are probabilistic distributions over sequences of words.

example:

- P("and nothing but the truth") ≈ 0.001
- P("and supercalifragilistichespiraldoso but the truth") ≈ 0

… or better: *close* to 0 (we will talk about handling zeros later)

Language Models

definition:

Language models are probabilistic distributions over sequences of words.

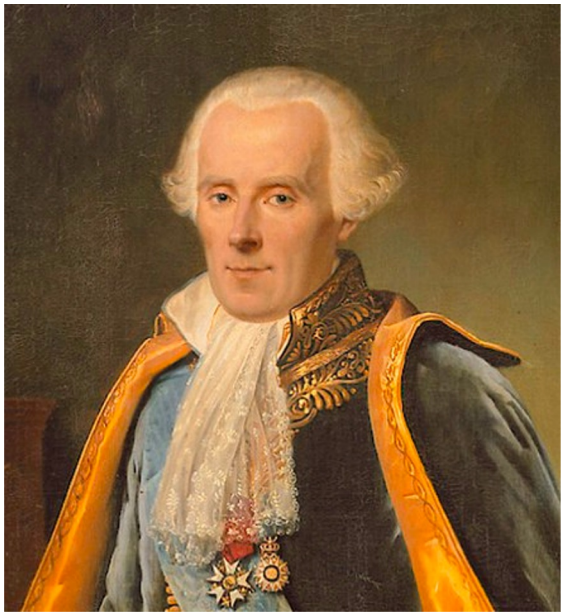
example:

- P("and nothing but the truth") ≈ 0.001
- P("and supercalifragilistichespiralidoso but the truth") ≈ 0

… or better: *close* to 0 (we will talk about handling zeros later)

Probability: basics

"*Probability theory is nothing but common sense reduced to calculation*"
– Pierre Laplace, 1812



https://en.wikiquote.org/wiki/Pierre-Simon_Laplace

Probability: basics

definition

State space or **Sample space** is a finite (or countably infinite) set S.

definition

A **discrete random variable** X can take any value from S.

definition

An **event** is an instance of a random variable **X=x** s.t. **x** $\subseteq S$
An event is a subset of S.

Probability: basics

definition

The **probability of the event** X = x is denoted by **p**(X = x) or simply **p**(x)

p is called the **probability mass function** and must satisfy two conditions:

$$0 \leq p(x) \leq 1 \quad \forall x \in S$$

$$\sum_{x \in S} p(x) = 1$$

Probability: basics

example

given a 5-sided dice

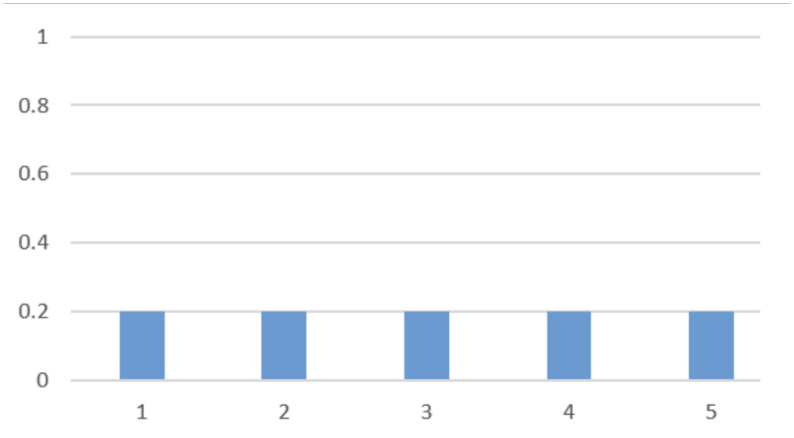
S={1,2,3,4,5}

X={x} s.t. x ∈ S the random variable representing the experiment throwing a dice

p(X={1}) = p(X={2}) = p(X={3}) = p(X={4}) = p(X={5}) = 0.2

p(1) = p(2) = p(3) = p(4) = p(5) = 0.2

p(1)+p(2)+p(3)+p(4)+p(5) =1.0



uniform probability distribution: each side has the same probability in the experiment defined by X

Probability: basics

- X can take values as subset of S

Probability of the union of two events:

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

· **Joint probability of two events** occurring together:

$$P(A, B) = probability\ of\ the\ joint\ event\ A \cap B$$

Probability: basics

- X can take values as subset of S

Probability of the union of two events:

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

· **Joint probability of two events** occurring together:

$$P(A, B) = probability\ of\ the\ joint\ event\ A \cap B$$

Example:
S = {1,2,3,4,5,6}
p({x})= ⅙ , x ∈S

A= the number is even, A={2,4,6}
B= the number is prime, B={2,3,5}
P(A U B) = P(A) + P(B) - P(A∩ B) = 3/6 + 3/6 - ⅙ =⅚ (probability of side even or prime)
P(A , B) = P(A ∩ B) = P({2,4,6}∩{2,3,5}) =P({2})= ⅙ (probability if side even and prime)

Probability: basics

definition

Conditional probability,

$$P(A|B) = \frac{P(A,B)}{P(B)}$$

is the probability of event A occurring, given that event B occurs.

Example:
S = {1,2,3,4,5,6}
p({x})= ⅙ , x ∈S

A= the number is even, A={2,4,6}
B= the number is prime, B={2,3,5}
P(A U B) = P(A) + P(B) - P(A∩ B) = 3/6 + 3/6 - ⅙ =⅚ (probability of side even or prime)
P(A , B) = P(A ∩ B) = P({2,4,6}∩{2,3,5}) =P({2})= ⅙ (probability if side even and prime)
P(A | B) = P(A, B) / P(B) = ⅙ / ½ (probability of side even given that is prime)

Probability: basics

definition

A and B are two **independent** events if and only if:

$$P(A \cap B) = P(A)P(B), \text{or}$$
$$\text{equivalently, } P(A|B) = P(A).$$

definition

two events A and B are **conditionally independent given an event C with P(C)>0** if:

$$P(A \cap B|C) = P(A|C)P(B|C)$$

Completion Prediction

- A language model also supports predicting the completion of a sentence
  - Please turn off your cell _____
  - Your program does not ______

- *Predictive text input* systems can guess what you are typing and give choices on how to complete it

Completion Prediction

- The basic idea underlying the statistical approach to word prediction is to use the probabilities of SEQUENCES OF WORDS to choose the most likely next word / correction of spelling error
- i.e., to compute

$$P(w_n \mid w_1 , \ldots, w_{n-1})$$

- For all words **w**, and predict as next word the one for which this (conditional) probability is highest

Language models

are probability distributions of sequences of words:

$$P(w_1, \ldots, w_n)$$

are helpful to compute:

$$P(W_n \mid W_1, \ldots, W_{n-1})$$

Language models

are probability distributions of sequences of words:

$$P(w_1, \ldots, w_n)$$

are helpful to compute:

$$P(W_n \mid W_1, \ldots, W_{n-1})$$

Language models and relative frequency estimations

how to estimate?

$$P(w_1, \ldots, w_n) = \frac{C(w_1, \ldots, w_n)}{N}$$

On a VERY LARGE corpus

We count the number of words N

We count how many times the sequence occurs : $C(w_1, \ldots, w_n)$

$$P(w_n \mid w_1, \ldots, w_{n-1}) = \frac{C(w_1, \ldots, w_{n-1}, w_n)}{C(w_1, \ldots, w_{n-1})}$$

Language models and relative frequency estimations

how to estimate?

$$P(w_1, \ldots, w_n) = \frac{C(w_1, \ldots, w_n)}{N}$$

On a VERY LARGE corpus

We count the number of words N

We count how many times the sequence occurs : $C(w_1, \ldots, w_n)$

$$P(w_n \mid w_1, \ldots, w_{n-1}) = \frac{C(w_1, \ldots, w_{n-1}, w_n)}{C(w_1, \ldots, w_{n-1})}$$

this way of estimating the probabilities is called **relative frequency estimate.**

pros of relative frequency estimate:

- The relative frequency estimate is a **Maximum Likelihood Estimate** (MLE) (Given a model, the **MLE** yields the maximum possible probability that can be calculated using the given data for estimation).

cons of relative frequency estimate:

- We need a VERY LARGE corpus for good estimations;
- Computationally infeasible;

Language models and relative frequency estimations

how to estimate?

$$P(w_1, \ldots, w_n) = \frac{C(w_1, \ldots, w_n)}{N}$$

On a VERY LARGE corpus

We count the number of words N

We count how many times the sequence occurs : $C(w_1, \ldots, w_n)$

$$P(w_n \mid w_1, \ldots, w_{n-1}) = \frac{C(w_1, \ldots, w_{n-1}, w_n)}{C(w_1, \ldots, w_{n-1})}$$

this way of estimating the probabilities is called **relative frequency estimate.**

pros of relative frequency estimate:

- The relative frequency estimate is a Maximum Likelihood Estimate (MLE) (Given a model, the MLE yields the maximum possible probability that can be calculated using the given data for estimation).

cons of relative frequency estimate:

- We need a VERY LARGE corpus for good estimations;
- Computationally infeasible;

We need another approach!!

Language models and relative frequency estimations

how to estimate?

$P(w_1, \ldots, w_n) =$

we apply the **chain rule:**
https://en.wikipedia.org/wiki/Chain_rule_(probability)

Language models and relative frequency estimations

how to estimate?

$P(w_1, \ldots, w_n) =$

$P(w_1)P(w_2|w_1)\ P(w_3|w_1w_2)\ldots P(w_n|w_1w_2\ldots w_{n-1})=$

$$\prod_{k=1}^{n} P(w_k \mid w_1^{k-1})$$

we apply the **chain rule:**
https://en.wikipedia.org/wiki/Chain_rule_(probability)

This was just a rewriting exercise

Language models and relative frequency estimations

how to estimate?

$P(w_1 ,\ldots, w_n) =$

$P(w_1)P(w_2|w_1)\ P(w_3|w_1w_2)\ldots P(w_n|w_1w_2\ldots w_{n-1})=$

$$\prod_{k=1}^{n} P(w_k \mid w_1^{k-1})$$

we apply the **chain rule:**
https://en.wikipedia.org/wiki/Chain_rule_(probability)

and make the Markov assumption:

A model **M** has the Markov property if the next state of the model depends on a limited number of previous states.

Andrey Andreyevich Markov
1856-1922

n-grams models

n-grams models

- The intuition of the **N-gram** model is that instead of computing the probability of a word given its entire history, we assume the Markov property and **we will approximate the history by just the last few N-1 words**.

$$P(W_n \mid W_1 \ldots W_{n-1}) \approx P(W_n \mid W_{n-N+1} \ldots W_{n-1})$$

$$P(W_1 \ldots W_n) \approx \prod_{k=1}^{n} P(W_k \mid W_{k-N+1} \ldots W_{k-1})$$

Bigrams (N=2) models

$$P(W_n \mid W_1 \ldots W_{n-1}) \approx P(W_n \mid W_{n-1})$$

$$P(W_1 \ldots W_n) \approx \prod_{k=1}^{n} P(W_k \mid W_{k-1})$$

Trigrams (N=3) models

$$P(W_n \mid W_1 \ldots W_{n-1}) \approx P(W_n \mid W_{n-2} W_{n-1})$$

$$P(W_1 \ldots W_n) \approx \prod_{k=1}^{n} P(W_k \mid W_{k-2} W_{k-1})$$

larger N vs. smaller N

- A larger **N** implies:
  - more information about the context of the specific instance…
  - greater **discrimination**…
  - … but suffers more from data sparseness!
- A smaller **N** implies:
  - less precision…
  - … but more instances in training data and more **reliable** statistical estimates

Summary of N-gram language models formulas:

- Word sequences $w_1^n = w_1 \ldots w_n$

- Chain rule of probability

$$P(w_1^n) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1^2)\ldots P(w_n \mid w_1^{n-1}) = \prod_{k=1}^{n} P(w_k \mid w_1^{k-1})$$

- Bigram approximation

$$P(w_1^n) = \prod_{k=1}^{n} P(w_k \mid w_{k-1})$$

- N-gram approximation

$$P(w_1^n) = \prod_{k=1}^{n} P(w_k \mid w_{k-N+1}^{k-1})$$

Estimation of N-Gram probabilities

- N-gram conditional probabilities can be estimated from raw text based on the ***relative frequency*** of word sequences

- Bigram $$P(w_n \mid w_{n-1}) = \frac{C(w_{n-1}w_n)}{\sum_w C(w_{n-1}w)} = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

- N-gram $$P(w_n \mid w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n-1}w_n)}{C(w_{n-N+1}^{n-1})}$$

Exercise: The Berkeley Restaurant Project (BERP) corpus

- BERP is a speech-based restaurant consultant corpus: https://www1.icsi.berkeley.edu/Speech/berp.html

- The corpus contains user queries – examples:
  - I'm looking for Cantonese food
  - I'd like to eat dinner someplace nearby
  - Tell me about Chez Panisse
  - I'm looking for a good place to eat breakfast

Exercise: The Berkeley Restaurant Project (BERP) corpus

- Given a corpus like BERP, we can compute the probability of a sentence like "I want Chinese food"

- To have a consistent probabilistic model, append a unique start (**<s>**) and end (**</s>**) symbol to every sentence and treat these as additional words

- "I want Chinese food" becomes "<s> I want Chinese food </s>"

- Making the bigram assumption and using the chain rule, the probability can be approximated as follows:
  - P(<s> I want Chinese food </s>) ~

    P(I|<s>) P(want|I) P(Chinese|want)P(food|Chinese) P(</s>|food)

Exercise: The Berkeley Restaurant Project (BERP) corpus

count the unigrams:

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

Exercise: The Berkeley Restaurant Project (BERP) corpus

count the unigrams:

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

remember to include <s> and </s>

count the bigrams:

	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

Exercise: The Berkeley Restaurant Project (BERP) corpus

count the unigrams:

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

count the bigrams:

	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

- Divide bigram counts by prefix unigram counts to get probabilities

Exercise: The Berkeley Restaurant Project (BERP) corpus

- Divide bigram counts by prefix unigram counts to get probabilities

	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

Exercise: The Berkeley Restaurant Project (BERP) corpus

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

P(I | <s>) P(want | I) P(English | want) P(food | English) P(</s> | food) =

.25 x .33 x .0011 x .5 x .68 = .000031

- "I want Chinese food"

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

P(I |<s>) P(want | I) P(Chinese | want) P(food | Chinese) P(</s> | food) =

.25 x .33 x .0065 x .52 x .68 = .00019

Kinds of knowledge

- As crude as they are, N-gram probabilities capture a range of interesting facts about language.

- 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

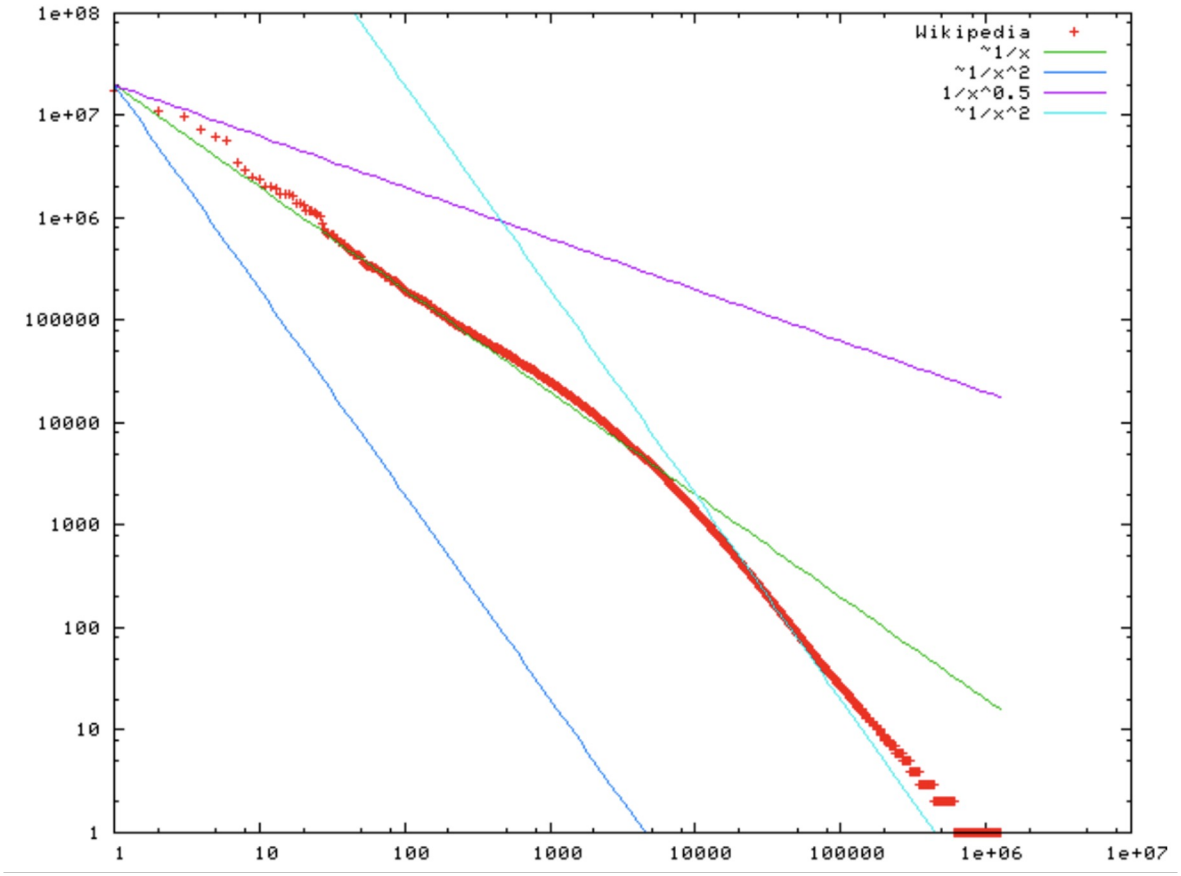
World knowledge

Syntax

Discourse

Zipf's Law: the long tail

The frequency of any word is inversely proportional to its rank in the frequency table with exponential decay · Word frequencies (Y) vs. Word ranks (X) in Wikipedia:



- Small number of words with high frequency
- Large number of words with low frequency

Evaluating language models

- How do we know if our models are any good?
- And in particular, how do we know if one model is better than another?
- Ideally, we would a good model to
  - prefer *fluent* sentences, namely grammatical / frequently observed ones
  - over *bad* sentences, that is, ungrammatical or rarely seen ones

Evaluating language models

- How do we know if our models are any good?
- And in particular, how do we know if one model is better than another?
- Ideally, we would a good model to
  - prefer *fluent* sentences, namely grammatical / frequently observed ones
  - over *bad* sentences, that is, ungrammatical or rarely seen ones

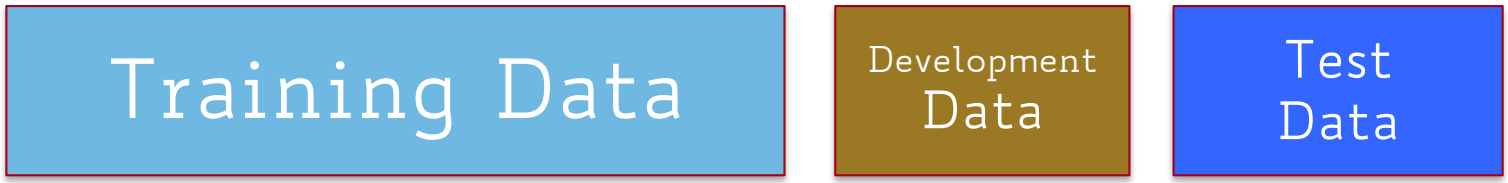
- KEY INTUITION: given two probabilistic models, the better model is the one that has a tighter fit to the test data, or predicts the details of the test data better

Evaluating language models

- How do we know if our models are any good?
- And in particular, how do we know if one model is better than another?
- Ideally, we would a good model to
  - prefer *fluent* sentences, namely grammatical / frequently observed ones
  - over *bad* sentences, that is, ungrammatical or rarely seen ones

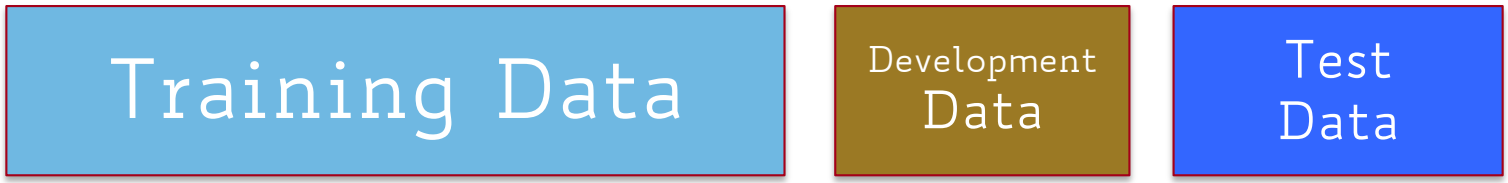
- KEY INTUITION: given two probabilistic models, the better model is the one that has a tighter fit to the test data, or predicts the details of the test data better
- We can measure better prediction by looking at the probability the model assigns to the test data; *the better model will assign a higher probability to the test data*

Evaluating language models: the standard setting



- N-gram models are good examples of statistical models that are "trained" from existing data
- We set aside a dataset called training set (or training corpus)
- We learn the model from the training dataset
- We apply it to a test set (test corpus), containing fresh data (no intersection with training set)
- If we need to tune parameters, we can use a development set (again, no intersection with other datasets)
- Typical division of data: 80% training, 10% development, 10% test

Evaluating language models: the standard setting



- We can generate sentences using the models and have a look at their quality
- Quality will differ if we use unigrams, bigrams, trigrams, quadrigrams, etc

Closed vocabulary vs. Open vocabulary

- **Closed vocabulary assumption:** we have a fixed vocabulary (lexicon) and all words in the test set will be in our lexicon

- **Open vocabulary assumption:** we model unknown words Out-Of-Vocabulary words. OOV words in the test set are replaced with a pseudoword called **<UNK>.**
  We train the probabilities in three steps:
  - Choose a vocabulary;
  - Convert any OOV word in the training set to **<UNK>;**
  - Estimate the probabilities for just like any other word in the vocabulary.

Open vocabulary

step 1) Choose the vocabulary

Vocabulary = { Escort, escort, claims, Berlusconi's parties, full, of, young, girls, an, who, she, was, paid, to, spend, two, nights, with, the, Italian, Prime, Minister, Berlusconi, has, revealed, how, his, were }

Open vocabulary

step 2) Convert OOV words in the training set

Escort claims Berlusconi's **bunga bunga** parties full of young girls. An escort who claims she was paid **€10,000 (£8,500)** to spend two nights with the Italian Prime Minister **Silvio** Berlusconi has revealed how his parties were full of young girls.

Open vocabulary

step 2) Convert OOV words in the training set

Escort claims Berlusconi's **bunga bunga** parties full of young girls. An escort who claims she was paid **€10,000 (£8,500)** to spend two nights with the Italian Prime Minister **Silvio** Berlusconi has revealed how his parties were full of young girls.

Escort claims Berlusconi's **<UNK> <UNK>** parties full of young girls. An escort who claims she was paid **<UNK> <UNK>** to spend two nights with the Italian Prime Minister **<UNK>** Berlusconi has revealed how his parties were full of young girls.

Open vocabulary

step 3) Estimate the probabilities

Escort claims Berlusconi's **<UNK> <UNK>** parties full of young girls. An escort who claims she was paid **<UNK> <UNK>** to spend two nights with the Italian Prime Minister **<UNK>** Berlusconi has revealed how his parties were full of young girls.

- P(**<UNK>**) = **…**
- P(<UNK>|Berlusconi) = **….**
- P(parties|<UNK>) = **..**

…

Intrinsic vs. Extrinsic evaluation of language models

Extrinsic/in vivo/end-to-end evaluation
- Embed the model in an application
- E.g. in a speech recognition system and measure the quality of the model through the quality of speech recognition
- Not always possible (…costly)

Intrinsic/in vitro evaluation
- We can use an intrinsic measure: **perplexity**
- Not a guarantee of extrinsic improvement
- But often correlated!

Perplexity

definition:

Perplexity (PP) for a sequence of N words:

Normalized by the number of words N

$$PP(w_1 w_2 \ldots w_N) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i \mid w_1 \ldots w_{i-1})}}$$

- In a bigram model:

$$PP(w_1 w_2 \ldots w_N) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i \mid w_{i-1})}}$$

reference to the original paper:
https://pubs.aip.org/asa/jasa/article/62/S1/S63/642598/Perplexity-a-measure-of-the-difficulty-of-speech

Perplexity

definition:

Perplexity for a corpus C, made of **m** sentences and **N** words;

$$\text{Perplexity}(C) = \sqrt[N]{\frac{1}{P(s_1, s_2, \ldots, s_m)}}$$

where, if we consider the sequence of m sentences independent:

$$P(s_1, \ldots, s_m) = \prod_{i=1}^{m} p(s_i)$$

Lower perplexity means a better model

- Example: language models trained on 38 million words from the Wall Street Journal (WSJ) using a 19,979 word vocabulary
  - Evaluated on a disjoint set of 1.5 million WSJ words

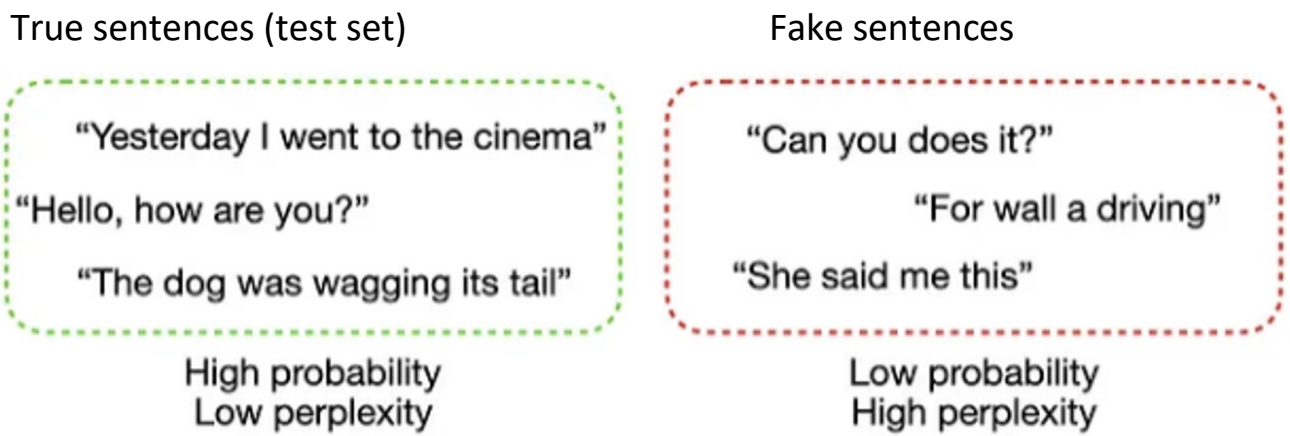
N-gram order	Unigram	Bigram	Trigram
Perplexity	962	170	109

  - The more information the model gives us about the word sequence, the lower the perplexity.

Lower perplexity means a better model

"Intuitively, if a model assigns a high probability to the test set, it means that it is **not surprised** to see it (it's not *perplexed* by it), which means that it has a good understanding of how the language works."

https://towardsdatascience.com/perplexity-in-language-models-87a196019a94



Smoothing

- Smoothing is a way to deal with unobserved n-grams

- Since there are a combinatorial number of possible word sequences, many rare (but not impossible) combinations never occur in training, so **MLEs** incorrectly assigns zero to many parameters (a.k.a. ***sparse data problem***)

- If a new combination occurs during testing, it and the entire sequence gets a probability of zero (i.e. *infinite perplexity*)

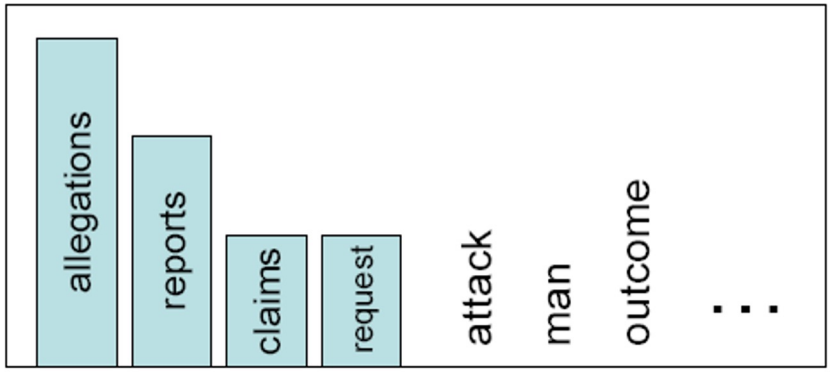
- In practice, parameters are ***smoothed*** (a.k.a. ***regularized***) to reassign some probability mass to unseen events

Smoothing: is like Robin Hood

Steal from the rich and give to the poor (in probability mass)

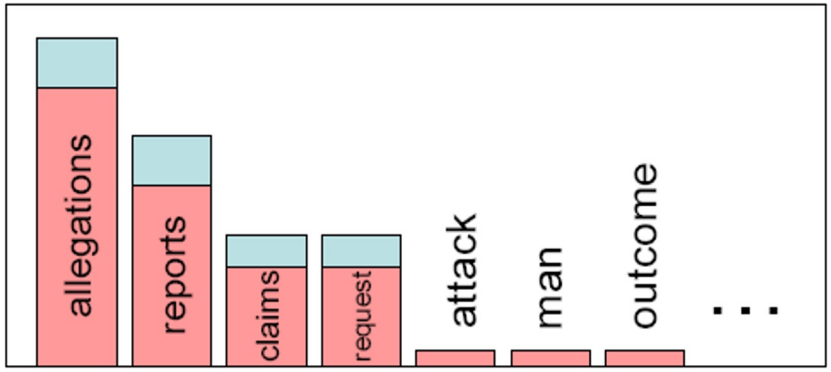
- We often want to make predictions from sparse statistics:

P(w | denied the)
3 allegations
2 reports
1 claims
1 request
7 total



- Smoothing flattens spiky distributions so they generalize better

P(w | denied the)
2.5 allegations
1.5 reports
0.5 claims
0.5 request
2 other
7 total



- Very important all over NLP, but easy to do badly!

Laplace smoothing (unigrams)

- Also called add-one smoothing
- Just add one to all the counts

- MLE estimate: $$P(w_i) = \frac{c_i}{N}$$

N is the number of running words (tokens). V is the number of unique words in the Vocabulary (word types)

- Laplace estimate: $$P_{\text{Laplace}}(w_i) = \frac{c_i + 1}{N + V}$$

- Reconstructed counts $$c_i^* = (c_i + 1)\frac{N}{N + V}$$

Laplace smoothing (bigrams)

The Berkeley Restaurant Project (BERP) corpus

	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

Laplace smoothing (bigrams)

The Berkeley Restaurant Project (BERP) co

	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

we add 1 to the bigram counts

	i	want	to	eat	chinese	food	lunch	spend
i	6	828	1	10	1	1	1	3
want	3	1	609	2	7	7	6	2
to	3	1	5	687	3	1	7	212
eat	1	1	3	1	17	3	43	1
chinese	2	1	1	1	1	83	2	1
food	16	1	16	1	2	5	1	1
lunch	3	1	1	1	1	2	1	1
spend	2	1	2	1	1	1	1	1

Laplace smoothing (bigrams)

The Berkeley Restaurant Project (BERP)

	i	want	to	eat	chinese	food	lunch	spend
i	6	828	1	10	1	1	1	3
want	3	1	609	2	7	7	6	2
to	3	1	5	687	3	1	7	212
eat	1	1	3	1	17	3	43	1
chinese	2	1	1	1	1	83	2	1
food	16	1	16	1	2	5	1	1
lunch	3	1	1	1	1	2	1	1
spend	2	1	2	1	1	1	1	1

V = 1446

$$P^*(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)+1}{C(w_{n-1})+V}$$

	i	want	to	eat	chinese	food	lunch	spend
i	0.0015	0.21	0.00025	0.0025	0.00025	0.00025	0.00025	0.00075
want	0.0013	0.00042	0.26	0.00084	0.0029	0.0029	0.0025	0.00084
to	0.00078	0.00026	0.0013	0.18	0.00078	0.00026	0.0018	0.055
eat	0.00046	0.00046	0.0014	0.00046	0.0078	0.0014	0.02	0.00046
chinese	0.0012	0.00062	0.00062	0.00062	0.00062	0.052	0.0012	0.00062
food	0.0063	0.00039	0.0063	0.00039	0.00079	0.002	0.00039	0.00039
lunch	0.0017	0.00056	0.00056	0.00056	0.00056	0.0011	0.00056	0.00056
spend	0.0012	0.00058	0.0012	0.00058	0.00058	0.00058	0.00058	0.00058

Laplace smoothing (bigrams)

The Berkeley Restaurant Project (BERP) corpus

$$c^*(w_{n-1}w_n) = \frac{[C(w_{n-1}w_n)+1] \times C(w_{n-1})}{C(w_{n-1})+V}$$

	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

The problem with Laplace smoothing

- C(want to) went from 608 to 238!
- P(to|want) from .66 to .26!
- Discount d = c*/c
  - d for "chinese food" =.10!!! A 10x reduction
  - So in general, Laplace is a blunt instrument
  - The big change in counts and probabilities occurs since *too much probability mass is moved to all the zeros*!

Add-k smoothing

- An alternative is to use a more fine-grained method (add-k)
  - Adjust Laplace by adding $0 < k < 1$ and normalizing by kV instead of V

$$P^*_{\text{Add-k}}(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n) + k}{C(w_{n-1}) + kV}$$

k is now a parameter of your model one can tune on the development portion of the dataset

- But this also results in an inappropriate discount for many counts…

Model combination

- As N increases, the power (expressiveness) of an N-gram model increases, ***but*** the ability to estimate accurate parameters from sparse data decreases (i.e., the smoothing problem gets worse)
- A general approach is to combine the results of multiple N-gram models of increasing complexity (i.e. increasing N)

- Backoff [Katz, 1987]: **use trigram if you have it, otherwise bigram, otherwise unigram**

- Linear interpolation [Jelinek and Mercer, 1980]: **mix all three**

Linear Interpolation

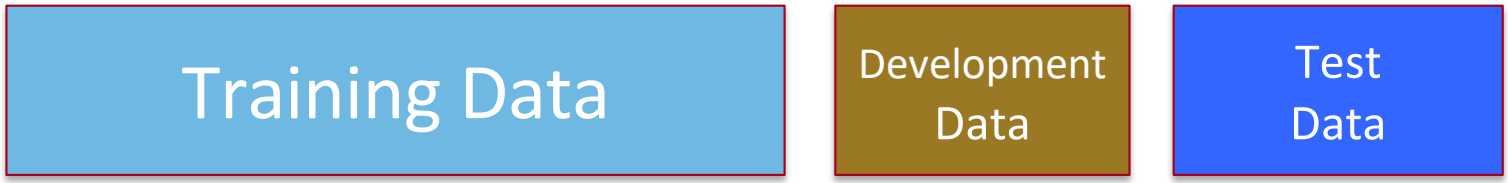
- Linearly combine estimates of N-gram models of increasing order
- Ex: for trigrams:

$$\begin{aligned}\hat{P}(w_n|w_{n-2}w_{n-1}) &= \lambda_1 P(w_n|w_{n-2}w_{n-1}) \\ &+\lambda_2 P(w_n|w_{n-1}) \\ &+\lambda_3 P(w_n)\end{aligned}$$

where

$$\sum_i \lambda_i = 1$$

Linear Interpolation: how to set the lambdas



- Learn the best values for λ_i by maximizing the likelihood of an independent **held-out** – **or development** (a.k.a. ***tuning***) – corpus
  - Fix the N-gram probabilities (on the training data)
  - Then search for λs that give largest probability to held-out set:

$$\log P(w_1 \ldots w_n \mid M(\lambda_1 \ldots \lambda_k)) = \sum_i \log P_{M(\lambda_1 \ldots \lambda_k)}(w_i \mid w_{i-1})$$

Katz back-off

- Only use lower-order model when data for higher-order model is unavailable (i.e. count is zero)
- Recursively back-off to weaker models until data is available

$$P_{\mathrm{BO}}(w_n|w_{n-N+1}^{n-1}) = \begin{cases} P^*(w_n|w_{n-N+1}^{n-1}), & \text{if } C(w_{n-N+1}^{n}) > 0 \\ \alpha(w_{n-N+1}^{n-1}) P_{\mathrm{BO}}(w_n|w_{n-N+2}^{n-1}), & \text{otherwise.} \end{cases}$$

  - Where P* is a discounted probability estimate to reserve mass for unseen events and α_s are back-off weights

Katz back-off

- MLE probabilities sum to 1: $\sum_i P(w_i|w_j w_k) = 1$

- So if we used MLE probabilities but backed off to lower order model when MLE probability is zero, we would be adding extra probability mass, i.e., the total probability of a word would be greater than 1!

- The *P** is used to discount the MLE probabilities and save probability mass for the lower order N-grams.

- The α ensures that the probability mass from all the lower order N-grams sums up to exactly the amount that we saved with the higher-order N-grams.

Stupid back-off [Brants et. al., 2007]

- Works well for web-scale LMs
- No discounting of the higher-order probabilities, just use relative frequencies
- If a higher-order n-gram has a zero count, simply backoff to a lower order n-gram, weighed by a fixed (context-independent) weight

not a true probability distribution

$$S(w_i|w_{i-k+1}^{i-1}) = \begin{cases} \frac{\text{count}(w_{i-k+1}^{i})}{\text{count}(w_{i-k+1}^{i-1})} & \text{if count}(w_{i-k+1}^{i}) > 0 \\ \lambda S(w_i|w_{i-k+2}^{i-1}) & \text{otherwise} \end{cases}$$

Absolute discount interpolation

- Just subtract a fixed (absolute) discount *d* from each count (e.g., 0.75 or some other value for *d*)
- A small discount *d* won't affect estimates for the (very) high counts – which we trust and would like to keep as much as we can
- The discount will mainly modify the smaller counts, for which we don't necessarily trust the estimate anyway

discounted bigram

Interpolation weight

$$P_{\text{AbsoluteDiscounting}}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i) - d}{c(w_{i-1})} + \lambda(w_{i-1})P(w)$$

unigram

Summary

- N-Gram language models: assign probabilities to sequences of words on the basis of n-grams (uni-, bi-, trigrams and so on…)
- Add-1 (or *k*) smoothing
  - a simple yet blunt instrument
- Model combination
  - interpolation
  - backoff
- Stupid backoff
  - Ok for large N-grams models trained on web-scale datasets
- Absolute discounting
  - Simple, yet provides the basis for the most commonly used method, i.e., Extended Interpolated Kneser-Ney [Chen and Goodman,1998]

Exercise:

Given the following corpus C, with the following 5 sentences:

1. I am Sam
2. Sam I am
3. Sam I like
4. Sam I do like
5. do I like Sam

1) **Define the 2-gram (bigram) model on the corpus C.**
2) **What are the most probable words to follow the sequences:**
   a) **<s> Sam . . .**
   b) **<s> Sam I do . . .**
   c) **<s> Sam I am Sam . . .**
   d) **<s> do I like . . .**
3) **Compute the probability of the following sequences:**
   <s> Sam do like </s>
   <s> Sam I am </s>
   <s> I am Sam </s>

Exercise:

1) Define the 2-gram (bigram) model on the corpus C.

1. I am Sam
2. Sam I am
3. Sam I like
4. Sam I do like
5. do I like Sam

We add the <s> and </s> tags

1. <s> I am Sam </s>
2. <s> Sam I am </s>
3. <s> Sam I like </s>
4. <s> Sam I do like </s>
5. <s> do I like Sam </s>

Exercise:

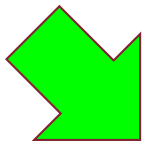
1. <s> I am Sam </s>
2. <s> Sam I am </s>
3. <s> Sam I like </s>
4. <s> Sam I do like </s>
5. <s> do I like Sam </s>

if we adopt the closed vocabulary assumption, the vocabulary is:

<s>
I
am
Sam
</s>
like
do

Exercise:

1. <s> I am Sam </s>
2. <s> Sam I am </s>
3. <s> Sam I like </s>
4. <s> Sam I do like </s>
5. <s> do I like Sam </s>



w	C(w)
<s>	5
I	5
am	2
Sam	5
</s>	5
like	3
do	2

We count $\mathbf{C(w_i)}$

Exercise:

1 <s> I am Sam </s>
2 <s> Sam I am </s>
3 <s> Sam I like </s>
4 <s> Sam I do like </s>
5 <s> do I like Sam </s>

we generate and count all the bigrams

1 (<s>,I)

	<s>	I	am	Sam	like	do	</s>
<s>	0	**1**	0	0	0	0	0
I	0	0	0	0	0	0	0
am	0	0	0	0	0	0	0
Sam	0	0	0	0	0	0	0
like	0	0	0	0	0	0	0
do	0	0	0	0	0	0	0
</s>	0	0	0	0	0	0	0

Exercise:

	<s>	I	**am**	Sam	like	do	</s>
<s>	0	1	0	0	0	0	0
I	0	0	**1**	0	0	0	0
am	0	0	0	0	0	0	0
Sam	0	0	0	0	0	0	0
like	0	0	0	0	0	0	0
do	0	0	0	0	0	0	0
</s>	0	0	0	0	0	0	0

1 <s> I am Sam </s>
2 <s> Sam I am </s>
3 <s> Sam I like </s>
4 <s> Sam I do like </s>
5 <s> do I like Sam </s>

we generate and count all the bigrams

1 (<s>,I) (I, am)

Exercise:

	<s>	I	am	**Sam**	like	do	</s>
<s>	0	1	0	0	0	0	0
I	0	0	1	0	0	0	0
am	0	0	0	**1**	0	0	0
Sam	0	0	0	0	0	0	0
like	0	0	0	0	0	0	0
do	0	0	0	0	0	0	0
</s>	0	0	0	0	0	0	0

1 <s> I am Sam </s>
2 <s> Sam I am </s>
3 <s> Sam I like </s>
4 <s> Sam I do like </s>
5 <s> do I like Sam </s>

we generate and count all the bigrams

1 (<s>,I) (I, am) (am,Sam)

Exercise:

	<s>	I	am	Sam	like	do	</s>
<s>	0	1	0	0	0	0	0
I	0	0	1	0	0	0	0
am	0	0	0	1	0	0	0
Sam	0	0	0	0	0	0	**1**
like	0	0	0	0	0	0	0
do	0	0	0	0	0	0	0
</s>	0	0	0	0	0	0	0

1 <s> I am Sam </s>
2 <s> Sam I am </s>
3 <s> Sam I like </s>
4 <s> Sam I do like </s>
5 <s> do I like Sam </s>

we generate and count all the bigrams

1 (<s>,I) (I, am) (am,Sam) (Sam </s>)

Exercise:

	<s>	I	am	**Sam**	like	do	</s>
<s>	0	1	0	**1**	0	0	0
I	0	0	1	0	0	0	0
am	0	0	0	1	0	0	0
Sam	0	0	0	0	0	0	1
like	0	0	0	0	0	0	0
do	0	0	0	0	0	0	0
</s>	0	0	0	0	0	0	0

1. <s> I am Sam </s>
2. <s> Sam I am </s>
3. <s> Sam I like </s>
4. <s> Sam I do like </s>
5. <s> do I like Sam </s>

we generate and count all the bigrams

1. (<s>,I) (I, am) (am,Sam) (Sam,</s>)
2. (<s>,Sam)

...

Exercise:

	<s>	I	am	Sam	like	do	</s>
<s>	0	1	0	1	0	0	0
I	0	0	1	0	0	0	0
am	0	0	0	1	0	0	0
Sam	0	**1**	0	0	0	0	1
like	0	0	0	0	0	0	0
do	0	0	0	0	0	0	0
</s>	0	0	0	0	0	0	0

1. <s> I am Sam </s>
2. <s> Sam I am </s>
3. <s> Sam I like </s>
4. <s> Sam I do like </s>
5. <s> do I like Sam </s>

we generate and count all the bigrams

1. (<s>,I) (I, am) (am,Sam) (Sam,</s>)
2. (<s>,Sam)(Sam,I)

...

Exercise:

	<s>	I	**am**	Sam	like	do	</s>
<s>	0	1	0	1	0	0	0
I	0	0	**2**	0	0	0	0
am	0	0	0	1	0	0	0
Sam	0	1	0	0	0	0	1
like	0	0	0	0	0	0	0
do	0	0	0	0	0	0	0
</s>	0	0	0	0	0	0	0

1. <s> I am Sam </s>
2. <s> Sam I am </s>
3. <s> Sam I like </s>
4. <s> Sam I do like </s>
5. <s> do I like Sam </s>

we generate and count all the bigrams

1. (<s>,I) (I, am) (am,Sam) (Sam,</s>)
2. (<s>,Sam)(Sam,I) (I,am)

...

Exercise:

	<s>	I	am	Sam	like	do	</s>
<s>	0	1	0	1	0	0	0
I	0	0	2	0	0	0	0
am	0	0	0	1	0	0	1
Sam	0	1	0	0	0	0	1
like	0	0	0	0	0	0	0
do	0	0	0	0	0	0	0
</s>	0	0	0	0	0	0	0

1. <s> I am Sam </s>
2. <s> Sam I am </s>
3. <s> Sam I like </s>
4. <s> Sam I do like </s>
5. <s> do I like Sam </s>

we generate and count all the bigrams

1. (<s>,I) (I, am) (am,Sam) (Sam,</s>)
2. (<s>,Sam)(Sam,I) (I,am) (am,</s>)

...

Exercise:

	<s>	I	am	**Sam**	like	do	</s>
<s>	0	1	0	2	0	0	0
I	0	0	2	0	0	0	0
am	0	0	0	1	0	0	1
Sam	0	1	0	0	0	0	1
like	0	0	0	0	0	0	0
do	0	0	0	0	0	0	0
</s>	0	0	0	0	0	0	0

1. <s> I am Sam </s>
2. <s> Sam I am </s>
3. <s> Sam I like </s>
4. <s> Sam I do like </s>
5. <s> do I like Sam </s>

we generate and count all the bigrams

1. (<s>,I) (I, am) (am,Sam) (Sam,</s>)
2. (<s>,Sam)(Sam,I) (I,am) (am,</s>)
3. (<s>,Sam)

...

Exercise:

	<s>	I	am	Sam	like	do	</s>
<s>	0	1	0	2	0	0	0
I	0	0	2	0	0	0	0
am	0	0	0	1	0	0	1
Sam	0	**2**	0	0	0	0	1
like	0	0	0	0	0	0	0
do	0	0	0	0	0	0	0
</s>	0	0	0	0	0	0	0

1 <s> I am Sam </s>
2 <s> Sam I am </s>
3 <s> Sam I like </s>
4 <s> Sam I do like </s>
5 <s> do I like Sam </s>

we generate and count all the bigrams

1 (<s>,I) (I, am) (am,Sam) (Sam,</s>) 2 (<s>,Sam)(Sam,I) (I,am) (am,</s>)

3 (<s>,Sam) (Sam,I)

...

Exercise:

	<s>	I	am	Sam	**like**	do	</s>
<s>	0	1	0	2	0	0	0
I	0	0	2	0	1	0	0
am	0	0	0	1	0	0	1
Sam	0	2	0	0	0	0	1
like	0	0	0	0	0	0	0
do	0	0	0	0	0	0	0
</s>	0	0	0	0	0	0	0

1 <s> I am Sam </s>
2 <s> Sam I am </s>
3 <s> Sam I like </s>
4 <s> Sam I do like </s>
5 <s> do I like Sam </s>

we generate and count all the bigrams

1 (<s>,I) (I, am) (am,Sam) (Sam,</s>) 2 (<s>,Sam)(Sam,I) (I,am) (am,</s>)

3 (<s>,Sam) (Sam,I) (I,like)

...

Exercise:

	<s>	I	am	Sam	like	do	</s>
<s>	0	1	0	2	0	0	0
I	0	0	2	0	1	0	0
am	0	0	0	1	0	0	1
Sam	0	2	0	0	0	0	1
like	0	0	0	0	0	0	**1**
do	0	0	0	0	0	0	0
</s>	0	0	0	0	0	0	0

1. <s> I am Sam </s>
2. <s> Sam I am </s>
3. <s> Sam I like </s>
4. <s> Sam I do like </s>
5. <s> do I like Sam </s>

we generate and count all the bigrams

1. (<s>,I) (I, am) (am,Sam) (Sam,</s>)
2. (<s>,Sam)(Sam,I) (I,am) (am,</s>)
3. (<s>,Sam) (Sam,I) (I,like) (like, </s>)

...

Exercise:

	<s>	I	am	**Sam**	like	do	</s>
<s>	0	1	0	**3**	0	0	0
I	0	0	2	0	1	0	0
am	0	0	0	1	0	0	1
Sam	0	2	0	0	0	0	1
like	0	0	0	0	0	0	**1**
do	0	0	0	0	0	0	0
</s>	0	0	0	0	0	0	0

1 <s> I am Sam </s>
2 <s> Sam I am </s>
3 <s> Sam I like </s>
4 <s> Sam I do like </s>
5 <s> do I like Sam </s>

we generate and count all the bigrams

1 (<s>,I) (I, am) (am,Sam) (Sam,</s>) 2 (<s>,Sam)(Sam,I) (I,am) (am,</s>)

3 (<s>,Sam) (Sam,I) (I,like) (like, </s>)

4 (<s>,Sam)

...

Exercise:

	<s>	I	am	Sam	like	do	</s>
<s>	0	1	0	3	0	0	0
I	0	0	2	0	1	0	0
am	0	0	0	1	0	0	1
Sam	0	**3**	0	0	0	0	1
like	0	0	0	0	0	0	**1**
do	0	0	0	0	0	0	0
</s>	0	0	0	0	0	0	0

1. <s> I am Sam </s>
2. <s> Sam I am </s>
3. <s> Sam I like </s>
4. <s> Sam I do like </s>
5. <s> do I like Sam </s>

we generate and count all the bigrams

1. (<s>,I) (I, am) (am,Sam) (Sam,</s>)
2. (<s>,Sam)(Sam,I) (I,am) (am,</s>)
3. (<s>,Sam) (Sam,I) (I,like) (like, </s>)
4. (<s>,Sam) (Sam,I)

...

Exercise:

	<s>	I	am	Sam	like	**do**	</s>
<s>	0	1	0	3	0	0	0
I	0	0	2	0	1	**1**	0
am	0	0	0	1	0	0	1
Sam	0	3	0	0	0	0	1
like	0	0	0	0	0	0	1
do	0	0	0	0	0	0	0
</s>	0	0	0	0	0	0	0

1 <s> I am Sam </s>
2 <s> Sam I am </s>
3 <s> Sam I like </s>
4 <s> Sam I do like </s>
5 <s> do I like Sam </s>

we generate and count all the bigrams

1 (<s>,I) (I, am) (am,Sam) (Sam,</s>) 2 (<s>,Sam)(Sam,I) (I,am) (am,</s>)

3 (<s>,Sam) (Sam,I) (I,like) (like, </s>)

4 (<s>,Sam) (Sam,I) (I,do)

…

Exercise:

	<s>	I	am	Sam	**like**	do	</s>
<s>	0	1	0	3	0	0	0
I	0	0	2	0	1	1	0
am	0	0	0	1	0	0	1
Sam	0	3	0	0	0	0	1
like	0	0	0	0	0	0	1
do	0	0	0	0	**1**	0	0
</s>	0	0	0	0	0	0	0

1 <s> I am Sam </s>
2 <s> Sam I am </s>
3 <s> Sam I like </s>
4 <s> Sam I do like </s>
5 <s> do I like Sam </s>

we generate and count all the bigrams

1 (<s>,I) (I, am) (am,Sam) (Sam,</s>) 2 (<s>,Sam)(Sam,I) (I,am) (am,</s>)

3 (<s>,Sam) (Sam,I) (I,like) (like, </s>)

4 (<s>,Sam) (Sam,I) (I,do) (do, like)

...

Exercise:

	<s>	I	am	Sam	like	do	</s>
<s>	0	1	0	3	0	0	0
I	0	0	2	0	1	1	0
am	0	0	0	1	0	0	1
Sam	0	3	0	0	0	0	1
like	0	0	0	0	0	0	2
do	0	0	0	0	1	0	0
</s>	0	0	0	0	0	0	0

1. <s> I am Sam </s>
2. <s> Sam I am </s>
3. <s> Sam I like </s>
4. <s> Sam I do like </s>
5. <s> do I like Sam </s>

we generate and count all the bigrams

1. (<s>,I) (I, am) (am,Sam) (Sam,</s>)
2. (<s>,Sam)(Sam,I) (I,am) (am,</s>)
3. (<s>,Sam) (Sam,I) (I,like) (like, </s>)
4. (<s>,Sam) (Sam,I) (I,do) (do, like) (like, </s>)

...

Exercise:

	<s>	I	am	Sam	like	**do**	</s>
<s>	0	1	0	3	0	1	0
I	0	0	2	0	1	1	0
am	0	0	0	1	0	0	1
Sam	0	3	0	0	0	0	1
like	0	0	0	0	0	0	2
do	0	0	0	0	1	0	0
</s>	0	0	0	0	0	0	0

1 <s> I am Sam </s>

2 <s> Sam I am </s>

3 <s> Sam I like </s>

4 <s> Sam I do like </s>

5 <s> do I like Sam </s>

we generate and count all the bigrams

1 (<s>,I) (I, am) (am,Sam) (Sam,</s>)

2 (<s>,Sam)(Sam,I) (I,am) (am,</s>)

3 (<s>,Sam) (Sam,I) (I,like) (like, </s>)

4 (<s>,Sam) (Sam,I) (I,do) (do, like) (like, </s>)

5 (<s>,do)

…

Exercise:

	<s>	I	am	Sam	like	do	</s>
<s>	0	1	0	3	0	1	0
I	0	0	2	0	1	1	0
am	0	0	0	1	0	0	1
Sam	0	3	0	0	0	0	1
like	0	0	0	0	0	0	2
do	0	1	0	0	1	0	0
</s>	0	0	0	0	0	0	0

1. <s> I am Sam </s>
2. <s> Sam I am </s>
3. <s> Sam I like </s>
4. <s> Sam I do like </s>
5. <s> do I like Sam </s>

we generate and count all the bigrams

1. (<s>,I) (I, am) (am,Sam) (Sam,</s>)
2. (<s>,Sam)(Sam,I) (I,am) (am,</s>)
3. (<s>,Sam) (Sam,I) (I,like) (like, </s>)
4. (<s>,Sam) (Sam,I) (I,do) (do, like) (like, </s>)
5. (<s>,do) (do, I)

...

Exercise:

	<s>	I	am	Sam	**like**	do	</s>
<s>	0	1	0	3	0	1	0
I	0	0	2	0	**2**	1	0
am	0	0	0	1	0	0	1
Sam	0	3	0	0	0	0	1
like	0	0	0	0	0	0	2
do	0	1	0	0	1	0	0
</s>	0	0	0	0	0	0	0

1 <s> I am Sam </s>
2 <s> Sam I am </s>
3 <s> Sam I like </s>
4 <s> Sam I do like </s>
5 <s> do I like Sam </s>

we generate and count all the bigrams

1 (<s>,I) (I, am) (am,Sam) (Sam,</s>)

2 (<s>,Sam)(Sam,I) (I,am) (am,</s>)

3 (<s>,Sam) (Sam,I) (I,like) (like, </s>)

4 (<s>,Sam) (Sam,I) (I,do) (do, like) (like, </s>)

5 (<s>,do) (do, I) (I, like)

...

Exercise:

	<s>	I	am	**Sam**	like	do	</s>
<s>	0	1	0	3	0	1	0
I	0	0	2	0	2	1	0
am	0	0	0	1	0	0	1
Sam	0	3	0	0	0	0	1
like	0	0	0	**1**	0	0	2
do	0	1	0	0	1	0	0
</s>	0	0	0	0	0	0	0

1. <s> I am Sam </s>
2. <s> Sam I am </s>
3. <s> Sam I like </s>
4. <s> Sam I do like </s>
5. <s> do I like Sam </s>

we generate and count all the bigrams

1. (<s>,I) (I, am) (am,Sam) (Sam,</s>)
2. (<s>,Sam)(Sam,I) (I,am) (am,</s>)
3. (<s>,Sam) (Sam,I) (I,like) (like, </s>)
4. (<s>,Sam) (Sam,I) (I,do) (do, like) (like, </s>)
5. (<s>,do) (do, I) (I, like) (like, Sam)

...

Exercise:

	<s>	I	am	Sam	like	do	</s>
<s>	0	1	0	3	0	1	0
I	0	0	2	0	2	1	0
am	0	0	0	1	0	0	1
Sam	0	3	0	0	0	0	**2**
like	0	0	0	1	0	0	2
do	0	1	0	0	1	0	0
</s>	0	0	0	0	0	0	0

1. <s> I am Sam </s>
2. <s> Sam I am </s>
3. <s> Sam I like </s>
4. <s> Sam I do like </s>
5. <s> do I like Sam </s>

we generate and count all the bigrams

1. (<s>,I) (I, am) (am,Sam) (Sam,</s>)
2. (<s>,Sam)(Sam,I) (I,am) (am,</s>)
3. (<s>,Sam) (Sam,I) (I,like) (like, </s>)
4. (<s>,Sam) (Sam,I) (I,do) (do, like) (like, </s>)
5. (<s>,do) (do, I) (I, like) (like, Sam) (Sam, </s>)

...

Exercise:

w	C(w)
<s>	5
I	5
am	2
Sam	5
</s>	5
like	3
do	2

	<s>	I	am	Sam	like	do	</s>
<s>	0/5	1/5	0/5	3/5	0/5	1/5	0/5
I	0/5	0/5	2/5	0/5	2/5	1/5	0/5
am	0/2	0/2	0/2	1/2	0/2	0/2	1/2
Sam	0/5	3/5	0/5	0/5	0/5	0/5	2/5
like	0/3	0/3	0/3	1/3	0/3	0/3	2/3
do	0/2	1/2	0/2	0/2	1/2	0/2	0/2
</s>	0/5	0/5	0/5	0/5	0/5	0/5	0/5

we divide each cell (u,w) by the c(u)

...

Exercise:

w	C(w)
<s>	5
I	5
am	2
Sam	5
</s>	5
like	3
do	2

	<s>	I	am	Sam	like	do	</s>
<s>	**0**	**1/5**	**0**	**3/5**	**0**	**1/5**	**0**
I	**0**	**0**	**2/5**	**0**	**2/5**	**1/5**	**0**
am	**0**	**0**	**0**	**1/2**	**0**	**0**	**1/2**
Sam	**0**	**3/5**	**0**	**0**	**0**	**0**	**2/5**
like	**0**	**0**	**0**	**1/3**	**0**	**0**	**2/3**
do	**0**	**1/2**	**0**	**0**	**1/2**	**0**	**0**
</s>	**0**	**0**	**0**	**0**	**0**	**0**	**0**

now each cell (u,w) = P(w|u)

...

Exercise:

2. What are the most probable words to follow the sequences:

a. **<s> Sam . . .**
b. **<s> Sam I do . . .**
c. **<s> Sam I am Sam . . .**
d. **<s> do I like . . .**

Exercise:

2. What are the most probable words to follow the sequences:

a. **<s> Sam . . .**
b. **<s> Sam I do . . .**
c. **<s> Sam I am Sam . . .**
d. **<s> do I like . . .**

Markov assumption with a bigram model

	<s>	I	am	Sam	like	do	</s>
<s>	0	1/5	0	3/5	0	1/5	0
I	0	0	2/5	0	2/5	1/5	0
am	0	0	0	1/2	0	0	1/2
Sam	0	**3/5**	0	0	0	0	2/5
like	0	0	0	1/3	0	0	2/3
do	0	1/2	0	0	1/2	0	0
</s>	0	0	0	0	0	0	0

a) **<s> Sam …**
we must search **w** with the highest P(**w** |Sam)

we have only one w= I
P(I | Sam) = ⅗

Exercise:

2. What are the most probable words to follow the sequences:

- **a. <s> Sam . . .**
- **b. <s> Sam I do . . .**
- **c. <s> Sam I am Sam . . .**
- **d. <s> do I like . . .**

Markov assumption with a bigram model

	<s>	I	am	Sam	like	do	</s>
<s>	0	1/5	0	3/5	0	1/5	0
I	0	0	2/5	0	2/5	1/5	0
am	0	0	0	1/2	0	0	1/2
Sam	0	3/5	0	0	0	0	2/5
like	0	0	0	1/3	0	0	2/3
do	0	**1/2**	0	0	**1/2**	0	0
</s>	0	0	0	0	0	0	0

b) **<s> Sam I do …**

we must search **w** with the highest P(**w** | do)

we have two **w**

P(I | do) = P(like | do) = ½

Exercise:

2. What are the most probable words to follow the sequences:

a. <s> Sam . . .
b. <s> Sam I do . . .
c. <s> Sam I am Sam . . .
d. <s> do I like . . .

Markov assumption with a bigram model

	<s>	I	am	Sam	like	do	</s>
<s>	0	1/5	0	3/5	0	1/5	0
I	0	0	2/5	0	2/5	1/5	0
am	0	0	0	1/2	0	0	1/2
Sam	0	3/5	0	0	0	0	2/5
like	0	0	0	1/3	0	0	2/3
do	0	1/2	0	0	1/2	0	0
</s>	0	0	0	0	0	0	0

c. <s> Sam I am Sam . . .
d. <s> do I like . . .

do it yourself

Exercise:

3) Compute the probability of the following sequences:

<s> Sam do like </s>
<s> Sam I am </s>
<s> I am Sam </s>

P(<s> Sam like </s>)=
=P(</s>|like)*P(like|Sam)*P(Sam|<s>) =
= ⅔ * ⅓ * ⅗ = 0,134

P(<s> Sam I am </s>)=

P(<s> I am Sam </s>)=

do it yourself

Exercise:

4) Apply the Laplace smoothing on the language model
5) What are the new probabilities P* for:

<s> Sam do like </s>
<s> Sam I am </s>
<s> I am Sam </s>

do it yourself

Exercise:

6) Define a Markov Chain representing the bigram-model

Exercise:

6) Define a Markov Chain representing the bigram-model

Formally, a Markov chain is specified by the following components:

$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{11} a_{12} \ldots a_{N1} \ldots a_{NN}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$
$\pi = \pi_1, \pi_2, \ldots, \pi_N$	an **initial probability distribution** over states. π_i is the probability that the Markov chain will start in state i. Some states j may have $\pi_j = 0$, meaning that they cannot be initial states. Also, $\sum_{i=1}^{n} \pi_i = 1$

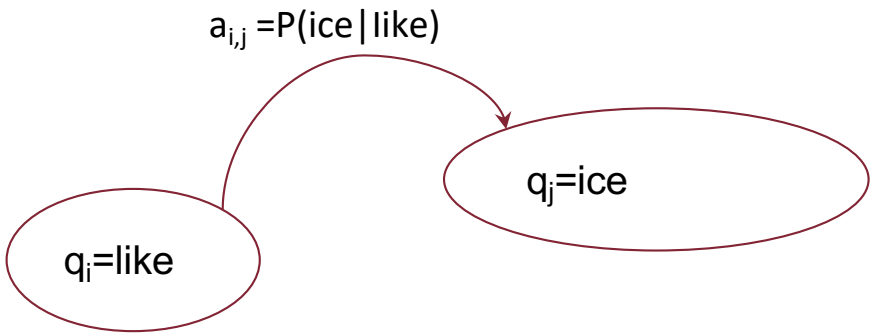
Exercise:

6) Define a Markov Chain representing the bigram-model

Formally, a Markov chain is specified by the following components:

$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{11} a_{12} \ldots a_{N1} \ldots a_{NN}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$
$\pi = \pi_1, \pi_2, \ldots, \pi_N$	an **initial probability distribution** over states. π_i is the probability that the Markov chain will start in state i. Some states j may have $\pi_j = 0$, meaning that they cannot be initial states. Also, $\sum_{i=1}^{n} \pi_i = 1$

e.g.:



do it yourself

Q&A

Resources and References

https://github.com/iacopomasi/NLP/blob/main/notebooks/Part_1_7_Language%20Models.ipynb

Resources and References

[Jurafsky&Martin, 2022] Jurafsky and Martin. Speech and Language Processing, Prentice Hall, third edition https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf

[Kukich, 1992] Kukich, K. (1992) Techniques for Automatically Correcting Words in Text. ACM Computing Surveys, 24, 377-439.
https://doi.org/10.1145/146370.146380

[Damerau, 1964] Fred J. Damerau. 1964. A technique for computer detection and correction of spelling errors. Commun. ACM 7, 3 (March 1964), 171–176. https://doi.org/10.1145/363958.363994

[Katz, 1987] S. Katz, "Estimation of probabilities from sparse data for the language model component of a speech recognizer," in IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 35, no. 3, pp. 400-401, March 1987, doi: https://doi.org/10.1109/TASSP.1987.1165125.

[Jelinek & Mercer, 1980]
F. Jelinek & R. Mercer (1980). Interpolated estimation of Markov source parameters from sparse data. : E. Gelsema & L. Kanal, , *Pattern recognition in practice*, 381-397. North-Holland Publishing Company, Amsterdam.

[Brants et. al.,2007]
Thorsten Brants et a(2007) l **"Large language models in machine translation"**, In Proceedings of EMNLP/CoNLL 2007, http://www.aclweb.org/anthology/D07-1090.pdf

[Chen and Goodman,1998]
Chen, S.F., and J. Goodman. An empirical study of smoothing techniques for language modeling. Technical Report TR–10–98, Center for Research in Computing Technology (Harvard University), August 1998. https://dash.harvard.edu/handle/1/25104739

**Credits

The slides of this part of the course are the result of a personal reworking of the slides and of the course material from different sources:

1. The NLP course of Prof. Roberto Navigli, Sapienza University of Rome
2. The NLP course of Prof. Simone Paolo Ponzetto, University of Mannheim, Germany
3. The NLP course of Prof. Chris Biemann, University of Hamburg, Germany
4. The NLP course of Prof. Dan Jurafsky, Stanford University, USA

Highly readable font Biancoenero© by biancoenero edizioni srl, designed by Umberto Mischi, with the consultancy of Alessandra Finzi, Daniele Zanoni and Luciano Perondi (Patent no. RM2011OOOO128).

Available free of charge for all institutions and individuals who use it for non-commercial purposes.