



Maschinelle Sprachverarbeitung

Language Models

Ulf Leser

Content of this Lecture

- Language Models
- Markov Models
- Data sparsity
- Language Models for IR

- Most material from [MS99], Chapter 6

Problem

- Given a prefix of a sentence: **Predict the next word**
 - “At 5 o’clock, we usually drink ...”
 - “tea” – quite likely
 - “beer” – quite unlikely
 - “a beer” – slightly more likely, but still
 - “biscuits” – semantically wrong
 - “the windows need cleaning” – syntactically wrong
- Similar to **Shannon’s Game**: Given a series of characters, predict the next one (used in communication theory)
- Abstract formulation: Given a language L and the prefix $S[1..n]$ of a sequence S , $S \in L$: **Predict $S[n+1]$**
- This is a **ranking problem** – no single solution

Applications

- Speech/character recognition
 - Given a **transcribed prefix** of a sentence – which word do we expect next?
- Automatic translation
 - Given a **translated prefix** of a sentence – what do we expect next?
- T9: "... information about common word combinations can also be learned ..."
- General: Use probabilities of next word as a-priori probability for **interpreting the next signal**
 - Helps to **disambiguate** between different options
 - Helps to make useful suggestions
 - Helps to point to likely errors (**observation \neq expectation**)

Language Models

- Usual approach: Learn a **model of the language**
- Classical model: (Deterministic) **Grammars**
 - Regular, context-free, ...
 - Grammars can be learned from examples
 - Not trivial, underdetermined, not covered here
 - Typical language grammars only determine **syntactically correct continuations** but disregard semantics
 - Usually, **many continuations** of a prefix are allowed
 - (Deterministic) Grammars cannot decide upon the **most likely one**

N-Grams over Words

- Popular and simple approach: **N-gram models**
 - “Indeed, it is difficult to beat a trigram model on the purely linear task of predicting the next word” [MS99]
- Definition

*A (word) **n-gram** is a sequence of n words.*
- Usage
 - Given a corpus, perform sentence splitting and tokenization
 - Count frequencies of **all n-grams** in this corpus
 - **Slide window** of size n over sentences and keep counter for each n-gram ever seen
 - Given a sentence prefix, predict **most probable continuation(s)** based on n-gram frequencies – how?

N-Grams for Language Modeling

- Assume a sentence prefix with $n-1$ words $\langle w_1, \dots, w_{n-1} \rangle$
- Look-up counts of all n -grams **starting** with $\langle w_1, \dots, w_{n-1} \rangle$
 - I.e., n -grams $\langle w_1, \dots, w_{n-1}, w_x \rangle$
- Choose w_x whose n -gram is the **most frequent one**
- More formally
 - Compute, for every possibly w_x ,

$$p(w_x) = p(w_x \mid w_1, \dots, w_{n-1}) = \frac{p(w_1, \dots, w_x)}{p(w_1, \dots, w_{n-1})}$$

- Choose w_x which **maximizes** $p(w_x)$

Which n?

- In language modeling, one usually chooses $n=3-4$
- Seems small, but most **language effects are local**
 - But not all:
 - “The day I went to school the first time began with ...”
 - “... I went to school the first time when I was ...”
- Also, we cannot obtain **robust relative counts** for larger n - not enough training data
 - “Data sparsity” problem (see later)
 - In high dimensional problems, training data is always sparse
 - Problem 1: We have no fitting n -grams because we haven’t seen the $n-1$ prefix in our corpus – no prediction possible
 - Problem 2: Almost all n -grams get a count of 1 – we cannot decide which continuation to predict

Content of this Lecture

- Language Models
- **Markov Models**
- Data sparsity
- Language Models for IR

History and Applications

- **Andrej Andrejewitsch Markov** (1856-1922)
 - Russian Mathematician
 - Developed Markov Models (or Markov Chains) as a method for analyzing language
 - Markov, A. A. (1913). "Beispiel statistischer Untersuchungen des Textes ‚Eugen Onegin‘, das den Zusammenhang von Ereignissen in einer Kette veranschaulicht (Original in Russisch)." *Bulletin de l'Academie Imperiale des Sciences de St.-Petersbourg*: 153-162.
- Markov Models and **Hidden Markov Models** are popular
 - Language Modeling, Part-of-speech tagging
 - Speech recognition
 - Named entity recognition / information extraction
 - Biological sequence analysis

Markov Models

- Definition

*Assume an alphabet Σ . A **Markov Model** of order 1 is a sequential stochastic process with $|\Sigma|$ states s_1, \dots, s_n with*

- *Every state emits exactly one symbol w_i from Σ*
- *No two states **emit the same symbol***
- *For a sequence $\langle s_1, s_2, \dots \rangle$ of states, the following holds*

$$p(w_n=s_n/w_{n-1}=s_{n-1}, w_{n-2}=s_{n-2}, \dots, w_1=s_1) = p(w_n=s_n/w_{n-1}=s_{n-1})$$

- For all s_i , it holds that

$$\sum_{s_{i+1}} p(w_{i+1} = s_{i+1} \mid w_i = s_i) = 1$$

- Remarks

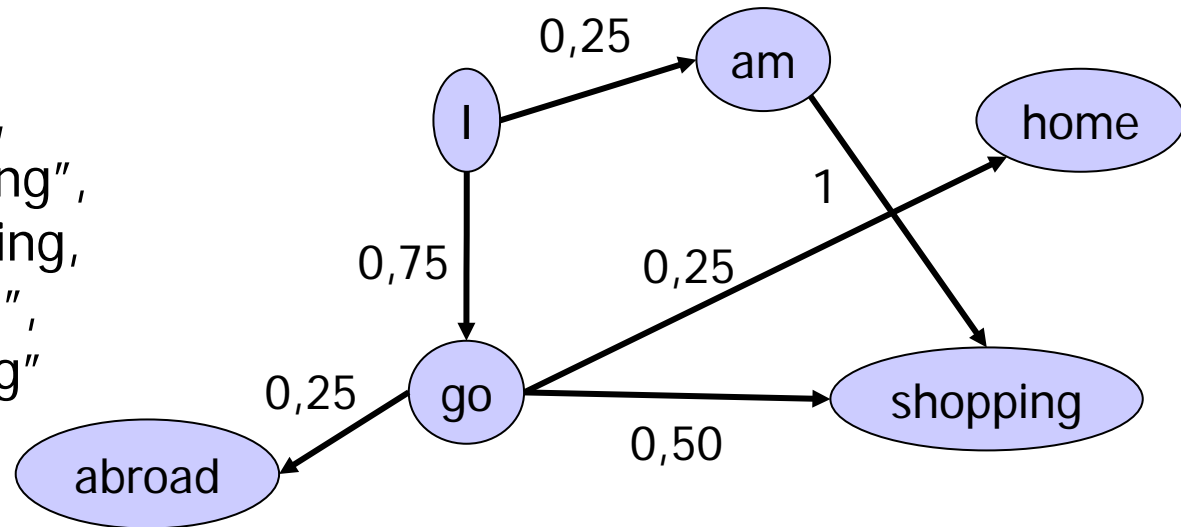
- $a_{i,j} = p(w_n=s_j | w_{n-1}=s_i)$ are called **transition probabilities**
- In language modeling, $\Sigma = \{\text{set of all words of a language}\}$
- Computing good **start probabilities** is an issue we essentially ignore

Visualization

- Since every state emits exactly one word, we can merge states and words
- State transition graph
 - Nodes are states (labeled with their emission)
 - Arcs are transitions labeled with a non-zero probability
 - Probabilities of all outgoing edges of a state sum up to 1

- Example

- “I go home”,
“I go shopping”,
“I am shopping”,
“I go abroad”,
“Go shopping”

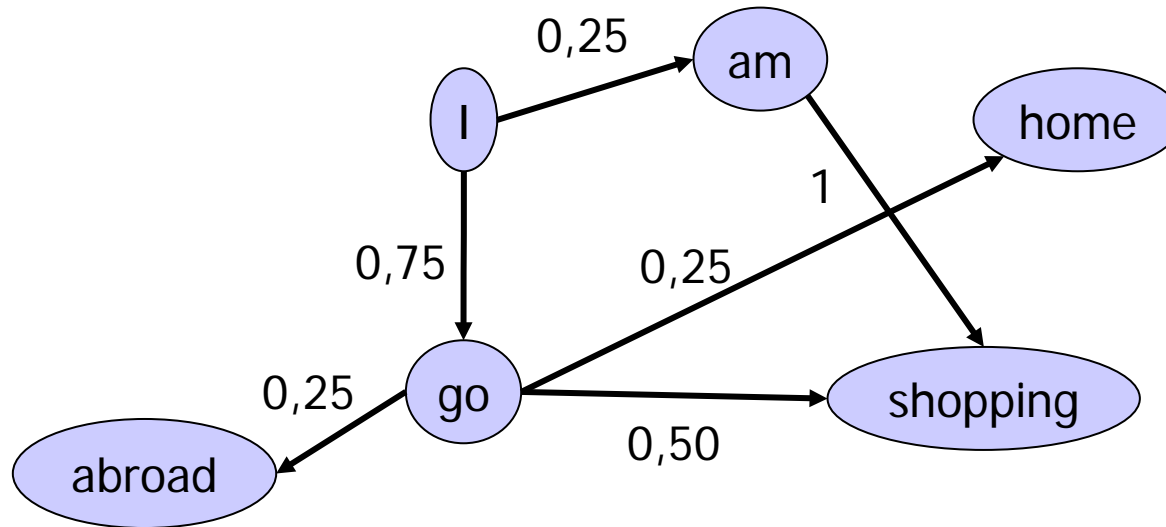


Probability of a Sequence of States (=a Sentence)

- Assume a Markov Model M and a sequence S of states with $|S|=n$
- With which **probability was S generated by M** , i.e., what is the value of $p(S|M)$?

$$\begin{aligned} p(S | M) &= p(w_1 = S[1]) * \prod_{i=2..n} p(w_i = S[i] | w_{i-1} = S[i-1]) \\ &= a_{0,S[1]} * \prod_{i=2..n} a_{S[i-1],S[i]} = a_{0,1} * \prod_{i=2..n} a_{i-1,i} \end{aligned}$$

Example



- $p(\text{"I go home"}) = p(w_1 = \text{"I"} | w_0) * p(w_2 = \text{"go"} | w_1 = \text{"I"}) * p(w_3 = \text{"home"} | w_2 = \text{"go"})$
 $= 1 * 0.75 * 0.25 = 0.1875$
- Problem: **Pairs not** in the training data get probability 0
 - Example: "I am abroad"
 - With this small "corpus", almost all transitions get $p=0$

Stochastic Processes

- Consider language generation as a **sequential stochastic process**
- At each stage, the process generates a new word
 - Like a DFA, but transitions have probabilities
- Question: How big is the **memory**? How many previous words does the process use to determine the next step?
 - 0: **Markov model** order 0: No memory at all
 - 1: Markov model order 1: Next word only depends on prev. word
 - 2: Markov model order 2: Next word only depends on 2 prev. words
 - ...

Higher Order Markov Models

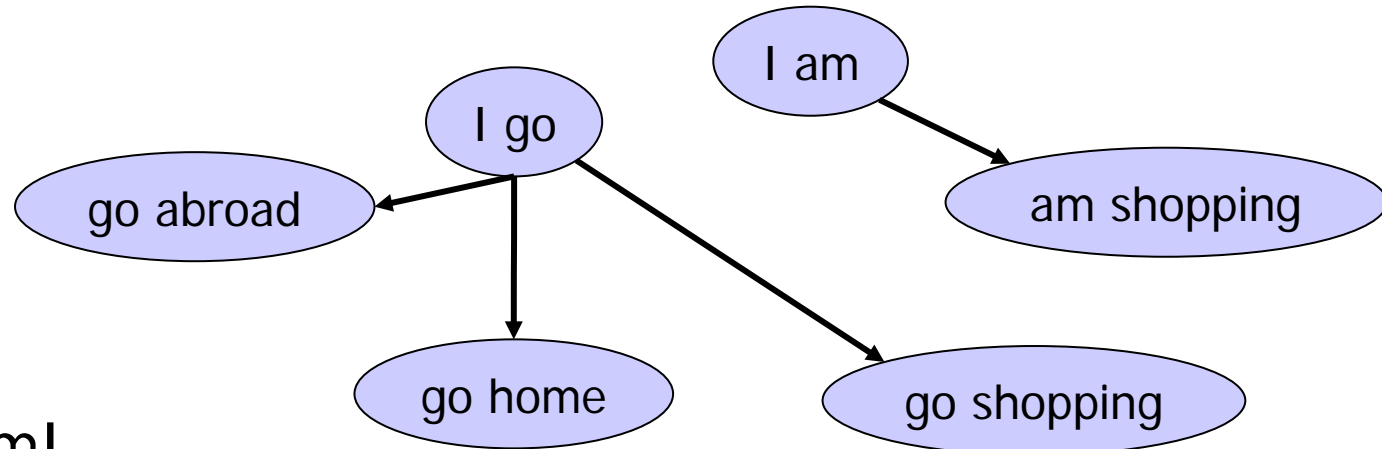
- Markov **Models of order k**: The prob. of state s after n steps depends on the k predecessor states s_{n-1}, \dots, s_{n-k}

$$p(w_n=s_n/w_{n-1}=s_{n-1}, w_{n-2}=s_{n-2}, \dots, w_1=s_1) = p(w_n=s_n/w_{n-1}=s_{n-1}, \dots, w_{n-k}=s_{n-k})$$

- We can transform any order k model M ($k > 1$) into a Markov Model of order 1 (M')
 - M' has $|M|^k$ states (all combinations of states of length k)

- **Example**

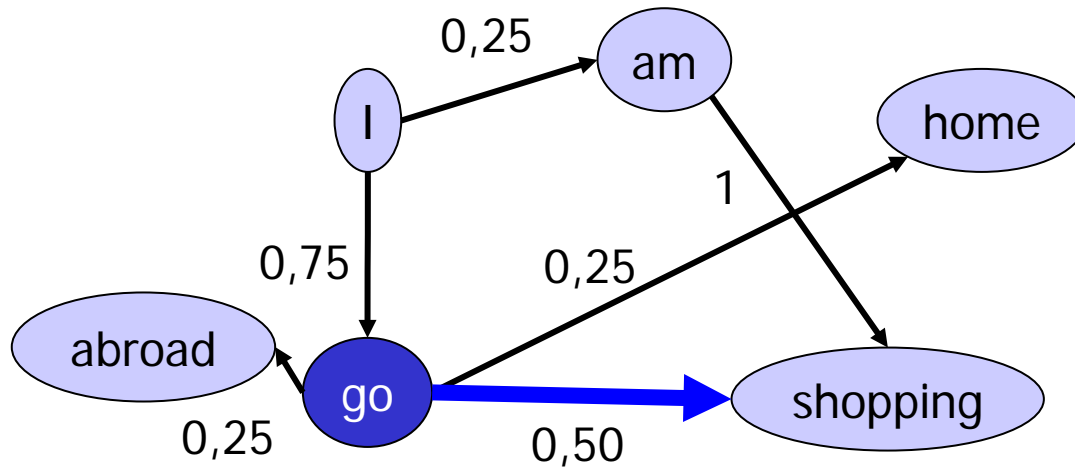
"I go home",
"I go shopping",
"I am shopping",
"I go abroad",
"Go shopping"



Sparsity problem!

Predicting the Next State

- For language modeling, we do not need the probability of an entire sequence, but we only reason about the **next state** given some previous states
- Consider an order-1 Markov Model



$$\begin{aligned} p(w_n) &= p(w_n \mid w_1, \dots, w_{n-1}) \\ &= p(w_n \mid w_{n-1}) \\ &= \frac{p(w_{n-1}, w_n)}{p(w_{n-1})} \\ &\sim p(w_{n-1}, w_n) \end{aligned}$$

This is the **most frequent bi-gram** with prefix w_{n-1}

Content of this Lecture

- Language Models
- Markov Models
- **Data sparsity**
- Language Models for IR

Problem

- We learn our transition probabilities from **a limited sample**
- Thus, we only **estimate the true transition** probabilities
- Introduces a systematic error which we can try to alleviate
 - **Sample selection** is important
 - Problem is researched a lot in statistics
- Extreme cases: Transitions we do not see in the corpus
 - Get a probability of 0
 - Will never be predicted
 - This does not mean that they are non-existing in the language
- Our model (yet) cannot adequately cope with **data sparsity**

Importance of Data Sparsity

- How many n-grams do exist in principle?
 - Assume a language of 20.000 words
 - $n=1$: 20.000, $n=2$: $4E8$, $n=3$: $8E12$, $n=4$: $1.6E17$, ...
 - Very bad estimates: Natural languages have many more words, but most combinations are not allowed
- In “normal” corpora over natural languages, **almost all n-grams** with $n > 4$ are very sparse
 - Exponential growth cannot be balanced by “use larger corpora”
 - Rare (and therefore very specific) n-grams are missed
- **Trade-off: N-gram models**
 - Large n : More expressive model, but bad estimations
 - Small n : Less expressive model, but better estimates

Example

- Unigrams (order 0): Always the most frequent word in the corpus, does not differentiate

<i>In</i>	<i>person</i>	<i>she</i>	<i>was</i>	<i>inferior</i>	<i>to</i>	<i>both</i>	<i>sisters</i>					
1-gram	$P(\cdot)$	$P(\cdot)$	$P(\cdot)$	$P(\cdot)$	$P(\cdot)$	$P(\cdot)$	$P(\cdot)$					
1	<i>the</i>	0.034	<i>the</i>	0.034	<i>the</i>	0.034	<i>the</i>	0.034				
2	<i>to</i>	0.033	<i>to</i>	0.033	<i>to</i>	0.033	<i>to</i>	0.033				
3	<i>and</i>	0.030	<i>and</i>	0.030	<i>and</i>	0.030	<i>and</i>	0.030				
4	<i>of</i>	0.029	<i>of</i>	0.029	<i>of</i>	0.029	<i>of</i>	0.029				
...												
8	<i>was</i>	0.015	<i>was</i>	0.015	<i>was</i>	0.015	<i>was</i>	0.015				
...												
13	<i>she</i>	0.011		<i>she</i>	0.011	<i>she</i>	0.011	<i>she</i>	0.011			
...												
254				<i>both</i>	0.0005	both	0.0005	<i>both</i>	0.0005			
...												
435				<i>sisters</i>	0.0003		sisters	0.0003	<i>sisters</i>	0.0003		
...												
1701				inferior	0.00005			<i>inferior</i>	0.00005			
2-gram	$P(\cdot person)$	$P(\cdot she)$	$P(\cdot was)$	$P(\cdot inferior)$	$P(\cdot to)$	$P(\cdot both)$	$P(\cdot sisters)$					
1	<i>and</i>	0.099	<i>had</i>	0.141	<i>not</i>	0.065	to	0.212	<i>be</i>	0.111	<i>of</i>	0.066
2	<i>who</i>	0.099	<i>was</i>	0.122	<i>a</i>	0.052			<i>the</i>	0.057	<i>to</i>	0.041
3	<i>to</i>	0.076			<i>the</i>	0.033			<i>her</i>	0.048	<i>in</i>	0.038
4	<i>in</i>	0.045			<i>to</i>	0.031			<i>have</i>	0.027	<i>and</i>	0.025
...												
23	she	0.009							<i>Mrs</i>	0.006	<i>she</i>	0.009
...												
41									<i>what</i>	0.004	sisters	0.006
...												
293									both	0.0004		
...												
∞												
				inferior	0							
3-gram	$P(\cdot In, person)$	$P(\cdot person, she)$	$P(\cdot she, was)$	$P(\cdot was, inf.)$	$P(\cdot inferior, to)$	$P(\cdot to, both)$	$P(\cdot both, sisters)$					
1	UNSEEN	<i>did</i>	0.5	<i>not</i>	0.057	UNSEEN	<i>the</i>	0.286	<i>to</i>	0.222		
2		<i>was</i>	0.5	<i>very</i>	0.038		<i>Maria</i>	0.143	<i>Chapter</i>	0.111		
3				<i>in</i>	0.030		<i>cherries</i>	0.143	<i>Hour</i>	0.111		
4				<i>to</i>	0.026		<i>her</i>	0.143	<i>Twice</i>	0.111		
...												
∞				inferior	0		both	0	sisters	0		
4-gram	$P(\cdot u, l, p)$	$P(\cdot l, p, s)$	$P(\cdot p, s, w)$	$P(\cdot s, w, l)$	$P(\cdot w, l, t)$	$P(\cdot t, l, b)$	$P(\cdot l, b, s)$					
1	UNSEEN	UNSEEN	<i>in</i>	1.0	UNSEEN	UNSEEN	UNSEEN					
...												
∞			inferior	0								

Table 6.3 Probabilities of each successive word for a clause from *Persuasion*. The probability distribution for the following word is calculated by Maximum Likelihood Estimate n -gram models for various values of n . The predicted likelihood rank of different words is shown in the first column. The actual next word is shown at the top of the table in italics, and in the table in bold.

Example

- Bi-grams (order 1): Correct word often ranks high, but not always

<i>In</i>	<i>person</i>	<i>she</i>	<i>was</i>	<i>inferior</i>	<i>to</i>	<i>both</i>	<i>sisters</i>			
1-gram	$P(\cdot)$	$P(\cdot)$	$P(\cdot)$	$P(\cdot)$	$P(\cdot)$	$P(\cdot)$	$P(\cdot)$			
1	the	0.034	the	0.034	the	0.034	the	0.034		
2	to	0.032	to	0.032	to	0.032	to	0.032		
3	and	0.030	and	0.030	and	0.030	and	0.030		
4	of	0.029	of	0.029	of	0.029	of	0.029		
...										
8	was	0.015	was	0.015	was	0.015	was	0.015		
...										
13	she	0.011			she	0.011	she	0.011		
...										
254					both	0.0005	both	0.0005		
...										
435					sisters	0.0003		sisters	0.0003	
...										
1701					inferior	0.00005				
2-gram	$P(\cdot person)$	$P(\cdot she)$	$P(\cdot was)$	$P(\cdot inferior)$	$P(\cdot to)$	$P(\cdot both)$				
1	and	0.09	not	0.065	to	0.212	be	0.111	of	0.066
2	who	0.09	was	0.122	a	0.052	the	0.057	to	0.041
3	to	0.075			the	0.033	her	0.048	in	0.038
...					to	0.031	have	0.027	and	0.025
...										
2	she	0.009					Mrs	0.006	she	0.009
...										
41							what	0.004	sisters	0.006
...										
293							both	0.0004		
...										
∞					inferior	0				
3-gram	$P(\cdot In, person)$	$P(\cdot person, she)$	$P(\cdot she, was)$	$P(\cdot was, inf.)$	$P(\cdot inferior, to)$	$P(\cdot to, both)$				
1	UNSEEN	did	0.5	not	0.057	UNSEEN	the	0.286	to	0.222
2		was	0.5	very	0.038		Maria	0.143	Chapter	0.111
3				in	0.030		cherries	0.143	Hour	0.111
4				to	0.026		her	0.143	Twice	0.111
...										
∞				inferior	0		both	0	sisters	0
4-gram	$P(\cdot u, l, p)$	$P(\cdot l, p, s)$	$P(\cdot p, s, w)$	$P(\cdot s, w, l)$	$P(\cdot w, l, l)$	$P(\cdot l, l, b)$				
1	UNSEEN	UNSEEN	in	1.0	UNSEEN	UNSEEN				
...										
∞			inferior	0						

Table 6.3 Probabilities of each successive word for a clause from *Persuasion*. The probability distribution for the following word is calculated by Maximum Likelihood Estimate n -gram models for various values of n . The predicted likelihood rank of different words is shown in the first column. The actual next word is shown at the top of the table in italics, and in the table in bold.

Example

<i>In</i>	<i>person</i>	<i>she</i>	<i>was</i>	<i>inferior</i>	<i>to</i>	<i>both</i>	<i>sisters</i>					
1-gram	$P(\cdot)$		$P(\cdot)$	$P(\cdot)$	$P(\cdot)$	$P(\cdot)$	$P(\cdot)$					
1	the	0.034	the	0.034	the	0.034	the	0.034				
2	to	0.032	to	0.032	to	0.032	to	0.032				
3	and	0.030	and	0.030	and	0.030	and	0.030				
4	of	0.029	of	0.029	of	0.029	of	0.029				
...												
8	was	0.015	was	0.015	was	0.015	was	0.015				
...												
13	she	0.011		she	0.011	she	0.011	she	0.011			
...												
254				both	0.0005	both	0.0005	both	0.0005			
...												
435				sisters	0.0003			sisters	0.0003			
...												
1701				inferior	0.00005							
2-gram	$P(\cdot person)$		$P(\cdot she)$		$P(\cdot inferior)$		$P(\cdot both)$		$P(\cdot sisters)$			
1	and	0.099	had	0.141	not	0.065	to	0.212	be	0.111	of	0.066
2	who	0.099	was	0.122	a	0.052	the	0.057	the	0.057	to	0.041
3	to	0.076			the	0.033	her	0.048	her	0.048	in	0.038
4	in	0.045			to	0.031	have	0.027	have	0.027	and	0.025
...												
23	she	0.009					Mrs	0.006	Mrs	0.006	she	0.009
...												
41							what	0.004	what	0.004	sisters	0.006
...												
293							both	0.0004	both	0.0004		
...												
∞					inferior	0						
3-gram	$P(\cdot person)$		$P(\cdot person, she)$		$P(\cdot she, was)$		$P(\cdot was, to)$		$P(\cdot inferior, to)$		$P(\cdot to, both)$	
1	UNSEEN		did	0.5	not	0.057	UNSEEN		the	0.286	to	0.222
2			was	0.5	very	0.038			Maria	0.143	Chapter	0.111
3					in	0.030			cherries	0.143	Hour	0.111
4					to	0.026			her	0.143	Twice	0.111
...												
∞					inferior	0			both	0	sisters	0
4-gram	$P(\cdot s, l, p)$		$P(\cdot l, p, s)$		$P(\cdot p, s, w)$		$P(\cdot s, w, l)$		$P(\cdot w, l, l)$		$P(\cdot l, l, b)$	
1	UNSEEN		UNSEEN		in	1.0	UNSEEN		UNSEEN		UNSEEN	
...												
∞					inferior	0						

- Tri-grams (order 2): Has a 50% hit, but already suffers from sparsity (unseen)
- Four-grams: Unusable
- Corpus: Fraction of Jane Austen's oeuvre, ~600.000 tokens, data from [MS99]

Table 6.3 Probabilities of each successive word for a clause from *Persuasion*. The probability distribution for the following word is calculated by Maximum Likelihood Estimate n -gram models for various values of n . The predicted likelihood rank of different words is shown in the first column. The actual next word is shown at the top of the table in italics, and in the table in bold.

Some Practical Solutions

- Reduce the number of words using **stemming**
 - Might help to go from $n=3..4$ to $n=4..5$
 - Important grammatical clues are lost
- More abstract: Use some form of “binning” of **tokens into classes** and compute n-grams over token classes, not token
 - All numbers -> one class
 - All verbs -> one class (POS tags)
 - All verbs related to “movement” -> one class

Statistical Estimators

- We were a bit sloppy so far
- We want

$$p(w_n) = p(w_n | w_1, \dots, w_{n-1}) = \frac{p(w_1, \dots, w_n)}{p(w_1, \dots, w_{n-1})}$$

- But we only have $count(w_1, \dots, w_n)$
- So far, we always **implicitly assumed**

$$p(w_1, \dots, w_n) = \frac{count(w_1, \dots, w_n)}{N}$$

- N: all observed n-grams

MLE for N-gram Models

- This is called a **Maximum Likelihood Estimator** (MLE)
- MLE gives **maximum likelihood** to the training data
 - Gives zero probability to all events not in the training data
 - The **probability mass** is spent entirely on the training data
 - Gives optimal results when applied to the training data
 - Overfitting
- Need to **smooth** the estimates to account for the limitations of the sample

Smoothing I: Laplace's Law

- Give some **probability mass to unseen events**
- Oldest (and simplest) suggestion: "Adding one"

$$p_{LAP}(w_1, \dots, w_n) = \frac{\text{count}(w_1, \dots, w_n) + 1}{N + B}$$

- Where B is the number of possible n-grams, i.e., K^n
- All n-grams get a probability $\neq 0$
- But – **moves too much mass** to the unknown
 - Estimates for seen n-grams are scaled down dramatically
 - Estimates for unseen n-grams are small, but there are so many
 - And many of them are truly impossible
 - In a corpus of 40 M words with $K \sim 400T$, **99.7% of the total probability mass** is spend in unseen events

Smoothing II: Lidstone's Law

- Laplace not suitable if there are **many events, but few seen**
- Lidstone's law gives less probability mass to unseen events

$$p_{LIP}(w_1, \dots, w_n) = \frac{\text{count}(w_1, \dots, w_n) + \lambda}{N + \lambda * B}$$

- Small λ : More mass is given to seen events
- Typical estimate is $\lambda=0.5$
- Appropriate values can be **learned** (next slide)
- Still: Estimate of seen events is linear in the MLE estimate
 - Not a good approximation of empirical distributions
- Other: Good-Turing Estimator, n-gram interpolations, ...

Learning Appropriate Values for λ

- We “simulate” seen and unseen events
- Divide corpus in two disjoint parts C_1 and C_2
- Count frequencies of n-grams in C_1
- Let c be the number of n-grams from C_1 not present in C_2
- Set $\lambda=c/B$
 - The probability of an n-gram (in C_2) to be considered as not existing although in reality it does exist

Option III: Back-Off Models

- If we cannot find a n-gram with count≠0, use a (n-1)-gram
 - Or an n-2 gram, ...
- Thus, in case there is no $p(w_1, \dots, w_n) \neq 0$, we “back off” to a simpler model

$$p(w_n | w_1, \dots, w_{n-1}) = \frac{p(w_1, \dots, w_n)}{p(w_1, \dots, w_{n-1})} \approx \frac{p(w_2, \dots, w_n)}{p(w_2, \dots, w_{n-1})} \approx \frac{p(w_3, \dots, w_n)}{p(w_3, \dots, w_{n-1})} \approx \dots$$

- Stop at the first (n-k)-gram with non-zero count
- Alternative: Always look at different n's
 - With different weights

$$p(w_n) = \lambda_1 \frac{p(w_{n-2}, w_{n-1}, w_n)}{p(w_{n-2}, w_{n-1})} + \lambda_2 \frac{p(w_{n-1}, w_n)}{p(w_{n-1})} + \lambda_3 p(w_n)$$

Content of this Lecture

- Language Models
- Markov Models
- Data sparsity
- Language Models for IR

New IR Model

- Recent trend in IR: Relevance based on language models
- Idea: See a **document as a “language”**
 - Learn a model of this language
 - See with which probability this **model has generated the query**
 - Rank documents based on these probabilities
- Sounds weird, but leads to a simple and well justified approach
- Very successful in recent evaluations
- **Smoothing is crucial** – docs are too small

Approach

- As docs are small, only **unigram models** are sensible
- Model of a doc: Relative frequencies of all its words
- Compute

$$p(d | q) = \frac{p(q | d) * p(d)}{p(q)} \sim p(q | d) * p(d) \sim p(q | d)$$

- $p(q)$ is equal for all d – irrelevant for ranking
- $p(d)$ can be used to incorporate a-prior knowledge (e.g. prestige), but often is set to uniform – irrelevant for ranking
- We **replace d with its model** and obtain

$$p(q | d) = p(q | M_d) = p(k_1, k_2, \dots, k_n | M_d) = \prod_{k \in q} p(k | M_d) = \prod_{k \in q} \frac{tf_{k,d}}{|d|}$$

Discussion

- **Very simple**
- Principled approach to justify usage of tf values
- More powerful for longer queries
- Problems
 - Words in q not in d: **Smoothing**
 - Where is idf gone?

Smoothing a Language Model for IR

- For instance, if $k \notin d$, set $p(k|M_d) = df_k/|D| = p(k|M_D)$
 - Token that are in d are counted with tf values (and not discounted with idf); tokens **not in d are counted with df values**
- More tunable parameters: **Linear interpolation**

$$p'(k | M_d) = \lambda * p(k | M_d) + (1 - \lambda) * p(k | M_D)$$

- Combine **relevance of k in document** and **relevance of k in corpus**
 - Large λ : More weight to the document, less weight to background
 - λ may vary, for instance with query size
- We are back at something similar to TF*IDF, but with a **probabilistic interpretation**, not a geometric one

Self Assessment

- What is language modelling about?
- Define a Markov model
- How can you turn a Markov model of order 4 into one of order 1?
- What is the data sparsity problem (in language modeling)?
- What is the disadvantage of Laplace smoothing?
- Explain how we can use language models for information retrieval