Advanced Artificial Intelligence

Part II. Statistical NLP

Markov Models and N-gramms

Wolfram Burgard, Luc De Raedt, Bernhard Nebel, Lars Schmidt-Thieme

Some slides taken from Helmut Schmid, Rada Mihalcea, Bonnie Dorr, Leila Kosseim, Peter Flach and others

Contents

- Probabilistic Finite State Automata
  - Markov Models and N-gramms
  - Based on
    - Jurafsky and Martin, Speech and Language Processing, Ch. 6.
- Variants with Hidden States
  - Hidden Markov Models
  - Based on
    - Manning & Schuetze, Statistical NLP, Ch.9
    - Rabiner, A tutorial on HMMs.

Shannon game
Word Prediction

- Predicting the next word in the sequence
  - Statistical natural language ….
  - The cat is thrown out of the …
  - The large green …
  - Sue swallowed the large green …
  - …

Claim

- A useful part of the knowledge needed to allow Word Prediction can be captured using simple statistical techniques.

- Compute:
  - probability of a sequence
  - likelihood of words co-occurring

- Why would we want to do this?
  - Rank the likelihood of sequences containing various alternative alternative hypotheses
  - Assess the likelihood of a hypothesis

Probabilistic Language Model

- Definition:
  - **Language model** is a model that enables one to compute the probability, or likelihood, of a sentence s, P(s).

- Let’s look at different ways of computing P(s) in the context of Word Prediction

Language Models

How to assign probabilities to word sequences?

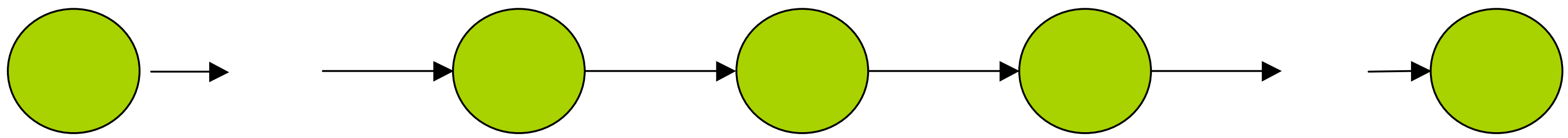
The probability of a word sequence $w_{1,n}$ is decomposed into a product of conditional probabilities.

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

$$= \prod_{i=1..n} P(w_i \mid w_{1,i-1})$$

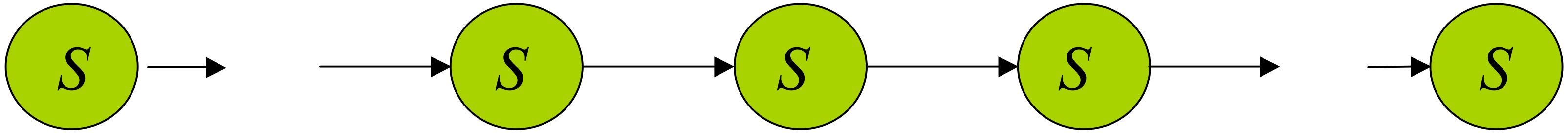
Problems ?

What is a (Visible) Markov Model ?



- Graphical Model (Can be interpreted as Bayesian Net)
- Circles indicate states
- Arrows indicate probabilistic dependencies between states
- State depends only on the previous state
- “The past is independent of the future given the present.” (d-separation)

Markov Model Formalization



- $\{S, \Pi, A\}$
- S : $\{w_1 \ldots w_N\}$ are the values for the states
    - Here : *the words*

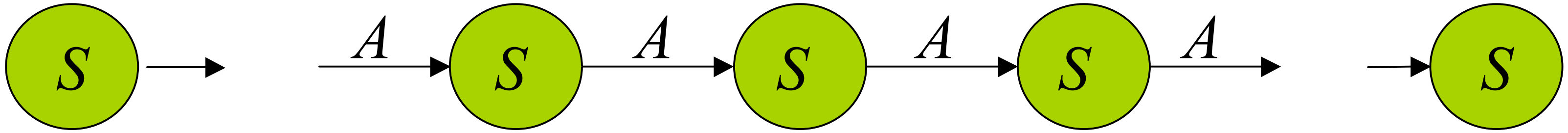
Limited Horizon (Markov Assumption)

$$P(X_{t+1} = w_k \mid X_1, \ldots, X_t) = P(X_{t+1} = w_k \mid X_t)$$

Time Invariant (Stationary) $\qquad = P(X_2 = w_k \mid X_1)$

Transition Matrix *A* $\qquad a_{ij} = P(X_{t+1} = w_j \mid X_t = w_i)$

Markov Model Formalization



- $\{S, \Pi, A\}$
- S : $\{s_1 \ldots s_N\}$ are the values for the states
- $\Pi = \{\pi_i\}$ are the initial state probabilities

$$\pi_i = P(X_1 = w_i)$$

- $A = \{a_{ij}\}$ are the state transition probabilities

Language Model

Each word only depends on the preceeding word

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

- 1st order Markov model, bigram

Final formula: $P(w_{1,n}) = \prod_{i=1..n} P(w_i \mid w_{i-1})$

Markov Models

- Probabilistic Finite State Automaton

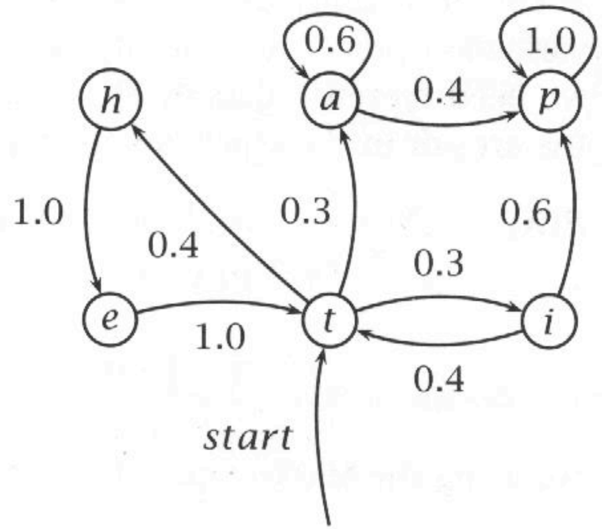


Figure 9.1 A Markov model.

What is the probability of a sequence of states ?

$$
\begin{aligned}
& P(X_1, \ldots, X_T) \\
&= P(X_1) P(X_2 \mid X_1) P(X_3 \mid X_1, X_2) \ldots P(X_T \mid X_1 \ldots, X_{T-1}) \\
&= P(X_1) P(X_2 \mid X_1) P(X_3 \mid X_2) \ldots P(X_T \mid X_{T-1}) \\
&= \pi_{X_1} \prod_{t=1}^{T-1} a_{X_t X_{t+1}}
\end{aligned}
$$

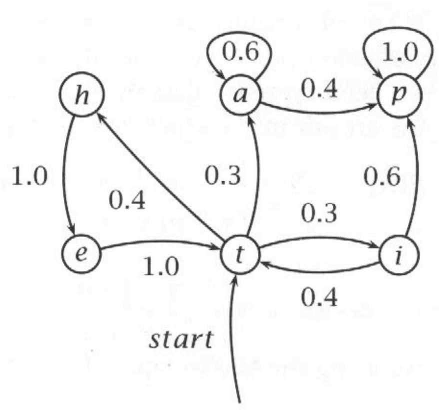


Figure 9.1 A Markov model.

Example

$$
\begin{aligned}
& P(t, i, p) \\
&= P(X_1 = t)P(X_2 = i \mid X_1 = t)P(X_3 = p \mid X_2 = i) \\
&= 1.0 \times 0.3 \times 0.6 \\
&= 0.18
\end{aligned}
$$

Trigrams

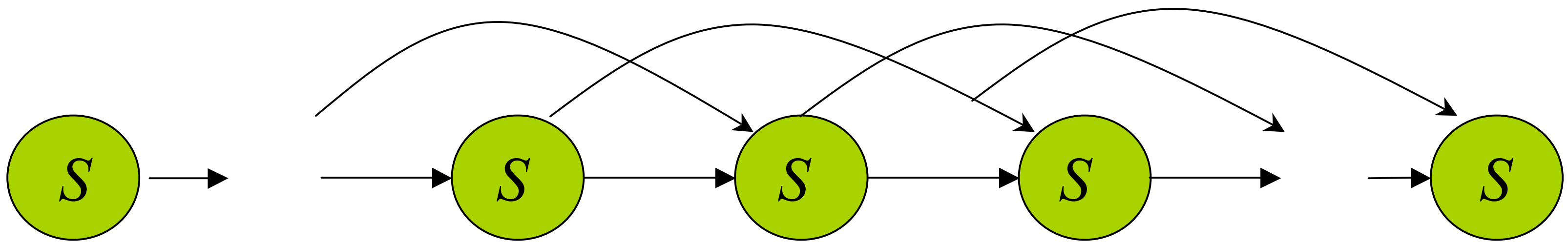
Now assume that

- each word only depends on the 2 preceeding words

  $P(w_i \mid w_{1,i-1}) = P(w_i \mid w_{i-2}, w_{i-1})$

  - 2nd order Markov model, trigram

Final formula: $P(w_{1,n}) = \prod_{i=1..n} P(w_i \mid w_{i-2}, w_{i-1})$



Simple N-Grams

- An N-gram model uses the previous N-1 words to predict the next one:
  - $P(w_n \mid w_{n-N+1}\ w_{n-N+2\ldots}\ w_{n-1})$
- unigrams: P(dog)
- bigrams: P(dog | big)
- trigrams: P(dog | the big)
- quadrigrams: P(dog | chasing the big)

A Bigram Grammar Fragment

Eat on	.16	Eat Thai	.03
Eat some	.06	Eat breakfast	.03
Eat lunch	.06	Eat in	.02
Eat dinner	.05	Eat Chinese	.02
Eat at	.04	Eat Mexican	.02
Eat a	.04	Eat tomorrow	.01
Eat Indian	.04	Eat dessert	.007
Eat today	.03	Eat British	.001

Additional Grammar

<start> I	.25	Want some	.04
<start> I'd	.06	Want Thai	.01
<start> Tell	.04	To eat	.26
<start> I'm	.02	To have	.14
I want	.32	To spend	.09
I would	.29	To be	.02
I don't	.08	British food	.60
I have	.04	British restaurant	.15
Want to	.65	British cuisine	.01
Want a	.05	British lunch	.01

Computing Sentence Probability

- P(I want to eat British food) = P(I|<start>) P(want|I) P(to|want) P(eat|to) P(British|eat) P(food|British) = .25x.32x.65x.26x.001x.60 = .000080
- vs.
- P(I want to eat Chinese food) = .00015

- Probabilities seem to capture “syntactic" facts, “world knowledge"
  - eat is often followed by a NP
  - British food is not too popular

- N-gram models can be trained by counting and normalization

Some adjustments

- product of probabilities… numerical underflow for long sentences
- so instead of multiplying the probs, we add the log of the probs

P(*I want to eat British food)*

Computed using

log(P(I|<s>)) + log(P(want|I)) + log(P(to|want)) + log(P(eat|to)) + log(P(British|eat)) + log(P(food|British))

= log(.25) + log(.32) + log(.65) + log (.26) + log(.001) + log(.6)

= -11.722

Why use only bi- or tri-grams?

- Markov approximation is still costly

  with a 20 000 word vocabulary:
  - bigram needs to store 400 million parameters
  - trigram needs to store 8 trillion parameters
  - using a language model > trigram is impractical

- to reduce the number of parameters, we can:
  - do stemming (use stems instead of word types)
  - group words into semantic classes
  - seen once --> same as unseen
  - ...
- Shakespeare
  - 884647 tokens (words)
    29066 types (wordforms)

unigram

(a) To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have
(b) Every enter now severally so, let
(c) Hill he late speaks; or! a more to leg less first you enter
(d) Will rash been and by I the me loves gentle me not slavish page, the and hour; ill let
(e) Are where exeunt and sighs have rise excellency took of.. Sleep knave we. near; vile like

2. Bigram approximation to Shakespeare

(a) What means, sir. I confess she? then all sorts, he is trim, captain.
(b) Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow.
(c) What we, hath got so she that I rest and sent to scold and nature bankrupt, nor the first gentleman?
(d) Enter Menenius, if it so many good direction found'st thou art a strong upon command of fear not a liberal largess given away, Falstaff! Exeunt
(e) Thou whoreson chops. Consumption catch your dearest friend, well, and I know where many mouths upon my undoing all but be, how soon, then; we'll execute upon my love's bonds and we do you will?
(f) The world shall- my lord!

3. Trigram approximation to Shakespeare

    (a) Sweet prince, Falstaff shall die. Harry of Monmouth's grave.
    (b) This shall forbid it should be branded, if renown made it empty.
    (c) What is't that cried?
    (d) Indeed the duke; and had a very good friend.
    (e) Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done.
    (f) The sweet! How many then shall posthumus end his miseries.

4. Quadrigram approximation to Shakespeare

    (a) King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in;
    (b) Will you not tell me who I am?
    (c) It cannot be but so.
    (d) Indeed the short and the long. Marry, 'tis a noble Lepidus.
    (e) They say all lovers swear more performance than they are wont to keep obliged faith unforfeited!
    (f) Enter Leonato's brother Antonio, and the rest, but seek the weary beds of people sick.

Building n-gram Models

- Data preparation:
  - Decide training corpus
  - Clean and tokenize
  - How do we deal with sentence boundaries?
    - I eat. I sleep.
      - (I eat) (eat I) (I sleep)
    - <s>I eat <s> I sleep <s>
      - (<s> I) (I eat) (eat <s>) (<s> I) (I sleep) (sleep <s>)
- Use statistical estimators:
  - to derive a good probability estimates based on training data.

Maximum Likelihood Estimation

- Choose the parameter values which gives the highest probability on the training corpus
- Let $C(w_1,..,w_n)$ be the frequency of n-gram $w_1,..,w_n$

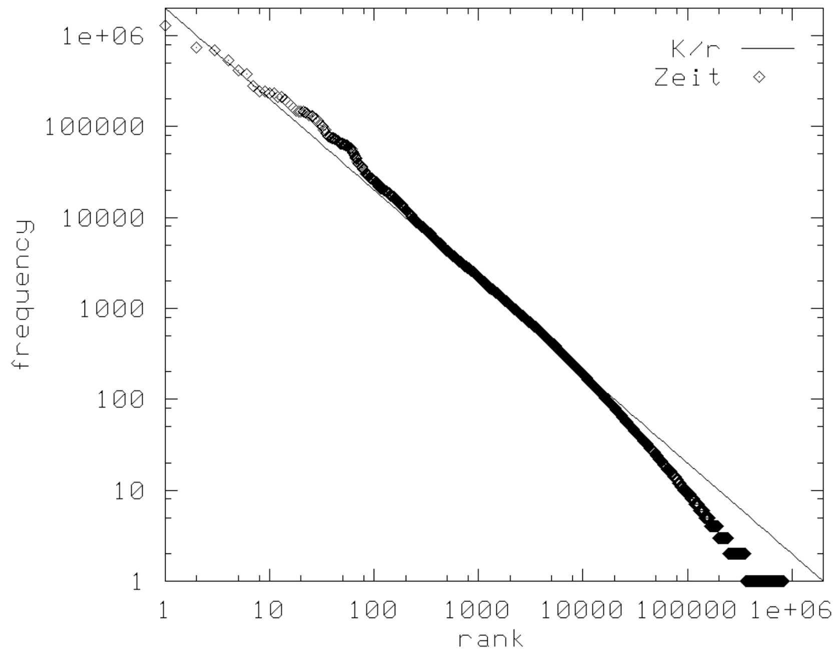
$$P_{MLE}(w_n | w_1,..,w_{n-1}) = \frac{C(w_1,..,w_n)}{C(w_1,..,w_{n-1})}$$

Example 1: P(event)

- in a training corpus, we have 10 instances of "*come across*"
  - 8 times, followed by "*as*"
  - 1 time, followed by "*more*"
  - 1 time, followed by "*a*"
- with MLE, we have:
  - P(as | come across) = 0.8
  - P(more | come across) = 0.1
  - P(a | come across) = 0.1
  - P(X | come across) = 0 where X $\neq$ "*as*", "*more*", "*a*"
- if a sequence never appears in training corpus? P(X)=0
- MLE assigns a probability of zero to unseen events …
- probability of an n-gram involving unseen words will be zero!

Maybe with a larger corpus?

- Some words or word combinations are unlikely to appear !!!
- Recall:
  - Zipf's law
  - f ~ 1/r



Problem with MLE: data sparseness (con't)

- in (Balh et al 83)
  - training with 1.5 million words
  - 23% of the trigrams from another part of the same corpus were previously unseen.
- So MLE alone is not good enough estimator

Discounting or Smoothing

- MLE is usually unsuitable for NLP because of the sparseness of the data
- We need to allow for possibility of seeing events not seen in training
- Must use a **Discounting** or **Smoothing** technique
- Decrease the probability of previously seen events to leave a little bit of probability for previously unseen events

Statistical Estimators

- Maximum Likelihood Estimation (MLE)
- Smoothing
  - Add one
  - Add delta
  - Witten-Bell smoothing
- Combining Estimators
  - Katz’s Backoff

Add-one Smoothing (Laplace's law)

- Pretend we have seen every n-gram at least once
- Intuitively:
  - new_count(n-gram) = old_count(n-gram) + 1

- The idea is to give a little bit of the probability space to unseen events

Add-one: Example

unsmoothed bigram counts:

2nd word

1st word

	I	want	to	eat	Chinese	food	lunch	…	Total (N)
I	8	1087	0	13	0	0	0		3437
want	3	0	786	0	6	8	6		1215
to	3	0	10	860	3	0	12		3256
eat	0	0	2	0	19	2	52		938
Chinese	2	0	0	0	0	120	1		213
food	19	0	17	0	0	0	0		1506
lunch	4	0	0	0	0	1	0		459
…									

unsmoothed normalized bigram probabilities:

	I	want	to	eat	Chinese	food	lunch	…	Total
I	.0023 (8/3437)	.32	0	.0038 (13/3437)	0	0	0		1
want	.0025	0	.65	0	.0049	.0066	.0049		1
to	.00092	0	.0031	.26	.00092	0	.0037		1
eat	0	0	.0021	0	.020	.0021	.055		1
Chinese	.0094	0	0	0	0	.56	.0047		1
food	.013	0	.011	0	0	0	0		1
lunch	.0087	0	0	0	0	.0022	0		1
…									

Add-one: Example (con't)

add-one smoothed bigram counts:

	I	*want*	*to*	*eat*	*Chinese*	*food*	*lunch*	…	**Total (N+V)**
I	~~8~~ 9	~~1087~~ 1088	1	14	1	1	1		~~3437~~ 5053
want	~~3~~ 4	1	787	1	7	9	7		2831
to	4	1	11	861	4	1	13		4872
eat	1	1	23	1	20	3	53		2554
Chinese	3	1	1	1	1	121	2		1829
food	20	1	18	1	1	1	1		3122
lunch	5	1	1	1	1	2	1		2075

add-one normalized bigram probabilities:

	I	*want*	*to*	*eat*	*Chinese*	*food*	*lunch*	…	**Total**
I	.0018 (9/5053)	.22	.0002	.0028 (14/5053)	.0002	.0002	.0002		1
want	.0014	.00035	.28	.00035	.0025	.0032	.0025		1
to	.00082	.00021	.0023	.18	.00082	.00021	.0027		1
eat	.00039	.00039	.0012	.00039	.0078	.0012	.021		1
Chinese	.0016	.00055	.00055	.00055	.00055	.066	.0011		1
food	.0064	.00032	.0058	.00032	.00032	.00032	.00032		1
lunch	.0024	.00048	.00048	.00048	.00048	.0022	.00048		1

Add-one, more formally

$$P_{Add1}(w_1\ w_2\ \ldots w_n) = \frac{C\ (w_1\ w_2\ \ldots w_n)+1}{N\ +\ B}$$

N: nb of n-grams in training corpus

B: nb of bins (of possible n-grams)

B = V^2 for bigrams

B = V^3 for trigrams etc.

where V is size of vocabulary

Problem with add-one smoothing

- bigrams starting with *Chinese* are boosted by a factor of 8 ! (1829 / 213)

unsmoothed bigram counts:

1st word

	I	*want*	*to*	*eat*	*Chinese*	*food*	*lunch*	…	**Total (N)**
I	8	1087	0	13	0	0	0		**3437**
want	3	0	786	0	6	8	6		**1215**
to	3	0	10	860	3	0	12		**3256**
eat	0	0	2	0	19	2	52		**938**
Chinese	2	0	0	0	0	120	1		**213**
food	19	0	17	0	0	0	0		**1506**
lunch	4	0	0	0	0	1	0		**459**

add-one smoothed bigram counts:

1st word

	I	*want*	*to*	*eat*	*Chinese*	*food*	*lunch*	…	**Total (N+V)**
I	9	1088	1	14	1	1	1		**5053**
want	4	1	787	1	7	9	7		**2831**
to	4	1	11	861	4	1	13		**4872**
eat	1	1	23	1	20	3	53		**2554**
Chinese	3	1	1	1	1	121	2		**1829**
food	20	1	18	1	1	1	1		**3122**
lunch	5	1	1	1	1	2	1		**2075**

Problem with add-one smoothing (con't)

- Data from the AP from (Church and Gale, 1991)
  - Corpus of 22,000,000 word tokens
  - Vocabulary of 273,266 words (i.e. 74,674,306,760 possible bigrams - or bins)
  - 74,671,100,000 bigrams were unseen
  - And each unseen bigram was given a frequency of 0.000295

Freq. from training data

Freq. from held-out data

Add-one smoothed freq.

f_{MLE}	$f_{empirical}$	$f_{add\text{-}one}$
0	0.000027	**0.000295**
1	0.448	0.000589
2	1.25	0.008884
3	2.24	0.00118
4	3.23	0.00147
5	4.21	0.00177

too high

too low

- Total probability mass given to unseen bigrams =
  (74,671,100,000 x 0.000295) / 22,000,000 ~**0.9996** !!!!

Problem with add-one smoothing

- every previously unseen n-gram is given a low probability, but there are so many of them that too much probability mass is given to unseen events
- adding 1 to frequent bigram, does not change much, but adding 1 to low bigrams (including unseen ones) boosts them too much !
- In NLP applications that are very sparse, Laplace's Law actually gives far too much of the probability space to unseen events.

Add-delta smoothing (Lidstone's law)

- instead of adding 1, add some other (smaller) positive value λ

$$P_{AddD}(w_1\ w_2\ \ldots w_n) = \frac{C\,(w_1\ w_2\ \ldots w_n) + \lambda}{N + \lambda B}$$

- Expected Likelihood Estimation (ELE) $\lambda = 0.5$
- Maximum Likelihood Estimation $\lambda = 0$
- Add one (Laplace) $\lambda = 1$

- better than add-one, but still…

Witten-Bell smoothing

- intuition:
  - An unseen n-gram is one that just did not occur yet
  - When it does happen, it will be its first occurrence
  - So give to unseen n-grams the probability of seeing a new n-gram
- Two cases discussed
  - Unigram
  - Bigram (more interesting)

Witten-Bell: unigram case

- N: number of tokens (word occurrences in this case)
- T: number of types (diff. observed words) - can be different than V (number of words in dictionary
- Total probability mass assigned to zero-frequency N-grams:

:
$$\sum_{i:c_i=0} p_i^* = \frac{T}{N+T}$$

- Z: number of unseen N-gramms

$$Z = \sum_{i:c_i=0} 1$$

- Prob. unseen

$$p_i^* = \frac{T}{Z(T+N)}$$

- Prob. seen

$$p_i^* = \frac{c_i}{N+T}$$

Witten-Bell: bigram case condition type counts on word

- N(w): # of bigrams tokens starting with w
- T(w): # of different observed bigrams starting with w
- Total probability mass assigned to zero-frequency N-grams:

$$\sum_{i:c(w_i,w_x)=0} p^*(w_i|w_x) = \frac{T(w_x)}{N(w_x)+T(w_x)}$$

- Z: number of unseen N-gramms

$$Z(w_x) = \sum_{i:c(w_i,w_x)=0} 1$$

Witten-Bell: bigram case condition type counts on word

- Prob. unseen

$$p^*(w_i|w_x) = \frac{T(w_x)}{Z(w_x)(N(w_x)+T(w_x))}$$

- Prob. seen

$$p^*(w_i|w_x) = \frac{c(w_x, w_i)}{N(w_x)+T(w_x)}$$

The restaurant example

- The original counts were:

	I	want	to	eat	Chinese	food	lunch	…	N(w) seen bigram tokens	T(w) seen bigram types	Z(w) unseen bigram types
I	8	1087	0	13	0	0	0		3437	95	1521
want	3	0	786	0	6	8	6		1215	76	1540
to	3	0	10	860	3	0	12		3256	130	1486
eat	0	0	2	0	19	2	52		938	124	1492
Chinese	2	0	0	0	0	120	1		213	20	1592
food	19	0	17	0	0	0	0		1506	82	534
lunch	4	0	0	0	0	1	0		459	45	1571

- T(w)= number of different <u>seen</u> bigrams types starting with w
- we have a vocabulary of 1616 words, so we can compute
- Z(w)= number of <u>unseen</u> bigrams types starting with w

  Z(w) = 1616 - T(w)

- N(w) = number of bigrams <u>tokens</u> starting with w

Witten-Bell smoothed probabilities

Witten-Bell normalized bigram probabilities:

	I	*want*	*to*	*eat*	*Chinese*	*food*	*lunch*	…	**Total**
I	.0022 (7.78/3437)	.3078	.000002	.0037	.000002	.000002	.000002		1
want	.00230	.00004	.6088	.00004	.0047	.0062	.0047		1
to	.00009	.00003	.0030	.2540	.00009	.00003	.0038		1
eat	.00008	.00008	.0021	.00008	.0179	.0019	.0490		1
Chinese	.00812	.00005	.00005	.00005	.00005	.5150	.0042		1
food	.0120	.00004	.0107	.00004	.00004	.00004	.00004		1
lunch	.0079	.00006	.00006	.00006	.00006	.0020	.00006		1

Witten-Bell smoothed count

- the count of the unseen bigram "*I lunch*"

$$\frac{\mathrm{T(I)}}{\mathrm{Z(I)}}\mathrm{x}\frac{\mathrm{N(I)}}{\mathrm{N(I)+T(I)}} = \frac{95}{1521}\mathrm{x}\frac{3437}{3437+95} = 0.06$$

- the count of the seen bigram "*want to*"

$$\mathrm{count(want\ to)x}\frac{\mathrm{N(want)}}{\mathrm{N(want)+T(want)}} = 786\mathrm{x}\frac{1215}{1215+76} = 739.73$$

Witten-Bell smoothed bigram counts:

	I	*want*	*to*	*eat*	*Chinese*	*food*	*lunch*	…	***Total***
I	7.78	1057.76	.061	12.65	.06	.06	.06		**3437**
want	2.82	.05	739.73	.05	5.65	7.53	5.65		**1215**
to	2.88	.08	9.62	826.98	2.88	.08	12.50		**3256**
eat	.07	.07	19.43	.07	16.78	1.77	45.93		**938**
Chinese	1.74	.01	.01	.01	.01	109.70	.91		**213**
food	18.02	.05	16.12	.05	.05	.05	.05		**1506**
lunch	3.64	.03	.03	.03	.03	0.91	.03		**459**

Combining Estimators

- so far, we gave the same probability to all unseen n-grams
  - we have never seen the bigrams
    - *journal of* $P_{unsmoothed}(of \mid journal) = 0$
    - *journal from* $P_{unsmoothed}(from \mid journal) = 0$
    - *journal never* $P_{unsmoothed}(never \mid journal) = 0$
  - all models so far will give the same probability to all 3 bigrams
- but intuitively, "*journal of*" is more probable because...
  - "*of*" is more frequent than "*from*" & "*never*"
  - unigram probability P(*of*) > P(*from*) > P(*never*)

Combining Estimators (con't)

- observation:
  - unigram model suffers less from data sparseness than bigram model
  - bigram model suffers less from data sparseness than trigram model
  - …
- so use a lower model estimate, to estimate probability of unseen n-grams
- if we have several models of how the history predicts what comes next, we can combine them in the hope of producing an even better model

Simple Linear Interpolation

- Solve the sparseness in a trigram model by mixing with bigram and unigram models
- Also called:
  - linear interpolation,
  - finite mixture models
  - deleted interpolation
- Combine linearly

$$P_{li}(w_n|w_{n-2},w_{n-1}) = \lambda_1 P(w_n) + \lambda_2 P(w_n|w_{n-1}) + \lambda_3 P(w_n|w_{n-2},w_{n-1})$$

  - where $0 \leq \lambda_i \leq 1$ and $\Sigma_i \lambda_i = 1$

Backoff Smoothing

Smoothing of Conditional Probabilities

p(Angeles | to, Los)

If *„to Los Angeles“* is not in the training corpus, the smoothed probability p(Angeles | to, Los) is identical to p(York | to, Los).

However, the actual probability is probably close to the bigram probability p(Angeles | Los).

Backoff Smoothing

(Wrong) **Back-off Smoothing** of trigram probabilities

if C(w‘, w“, w) > 0

P*(w | w‘, w“) = P(w | w‘, w“)

else if C(w“, w) > 0

P*(w | w‘, w“) = P(w | w“)

else if C(w) > 0

P*(w | w‘, w“) = P(w)

else

P*(w | w‘, w“) = 1 / #words

Backoff Smoothing

Problem: not a probability distribution

Solution:

Combination of Back-off and frequency discounting

$P(w \mid w_1,\ldots,w_k) = C^*(w_1,\ldots,w_k,w) / N$ if $C(w_1,\ldots,w_k,w) > 0$

else

$$P(w \mid w_1,\ldots,w_k) = \alpha(w_1,\ldots,w_k)\, P(w \mid w_2,\ldots,w_k)$$

Backoff Smoothing

The backoff factor is defined s.th. the probability mass assigned to unobserved trigrams

$$\sum_{w:\, C(w_1,\ldots,w_k,w)=0} \alpha(w_1,\ldots,w_k)\, P(w \mid w_2,\ldots,w_k))$$

is identical to the probability mass discounted from the observed trigrams.

$$1 - \sum_{w:\, C(w_1,\ldots,w_k,w)>0} P(w \mid w_1,\ldots,w_k))$$

Therefore, we get:

$$\alpha(w_1,\ldots,w_k) = \left(1 - \sum_{w:\, C(w_1,\ldots,w_k,w)>0} P(w \mid w_1,\ldots,w_k)\right) / \left(1 - \sum_{w:\, C(w_1,\ldots,w_k,w)>0} P(w \mid w_2,\ldots,w_k)\right)$$

Spelling Correction

- They are leaving in about fifteen *minuets* to go to her house.
- The study was conducted mainly *be* John Black.
- Hopefully, all *with* continue smoothly in my absence.
- Can they *lave* him my messages?
- I need to *notified* the bank of….
- He is trying to *fine* out.

Spelling Correction

- One possible method using N-gramms
- Sentence $w_1, \ldots, w_n$
- Alternatives $\{v_1, \ldots v_m\}$ may exist for w_k
  - Words sounding similar
  - Words close (edit-distance)
- For all such alternatives compute
- $P(w_1, \ldots, w_{k-1}, v_i, w_{k+1}, \ldots, w_n)$ and choose best one

Other applications of LM

- Author / Language identification
- hypothesis: texts that resemble each other (same author, same language) share similar characteristics
  - In English character sequence "*ing*" is more probable than in French
- Training phase:
  - construction of the language model
  - with pre-classified documents (known language/author)
- Testing phase:
  - evaluation of unknown text (comparison with language model)

Example: Language identification

- bigram of characters
  - characters = 26 letters (case insensitive)
  - possible variations: case sensitivity, punctuation, beginning/end of sentence marker, …

	A	B	C	D	…	Y	Z
A	0.0014	0.0014	0.0014	0.0014	…	0.0014	0.0014
B	0.0014	0.0014	0.0014	0.0014	…	0.0014	0.0014
C	0.0014	0.0014	0.0014	0.0014	…	0.0014	0.0014
D	0.0042	0.0014	0.0014	0.0014	…	0.0014	0.0014
E	0.0097	0.0014	0.0014	0.0014	…	0.0014	0.0014
…	…	…	…	…	…	…	0.0014
Y	0.0014	0.0014	0.0014	0.0014	…	0.0014	0.0014
Z	0.0014	0.0014	0.0014	0.0014	0.0014	0.0014	0.0014

1. Train a language model for English:

2. Train a language model for French

3. Evaluate probability of a sentence with LM-English & LM-French

4. Highest probability -->language of sentence

Claim

- A useful part of the knowledge needed to allow Word Prediction can be captured using simple statistical techniques.

- Compute:
  - probability of a sequence
  - likelihood of words co-occurring

- It can be useful to do this.