
Advanced Natural Language Processing

Lecture 4

Language Modeling (I)

Philipp Koehn

28 September 2011



Language models

- **Language models** answer the question:

How likely is a string of English words good English?

- Help with reordering

$$p_{\text{LM}}(\text{the house is small}) > p_{\text{LM}}(\text{small the is house})$$

- Help with word choice

$$p_{\text{LM}}(\text{I am going home}) > p_{\text{LM}}(\text{I am going house})$$

N-Gram Language Models

- Given: a string of English words $W = w_1, w_2, w_3, \dots, w_n$
- Question: what is $p(W)$?
- Sparse data: Many good English sentences will not have been seen before

→ Decomposing $p(W)$ using the chain rule:

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

(not much gained yet, $p(w_n|w_1, w_2, \dots, w_{n-1})$ is equally sparse)

Markov Chain

- **Markov assumption:**
 - only previous history matters
 - limited memory: only last k words are included in history (older words less relevant)
- **k th order Markov model**
- For instance 2-gram language model:

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

- What is conditioned on, here w_{i-1} is called the **history**

Estimating N-Gram Probabilities

- Maximum likelihood estimation

$$p(w_2|w_1) = \frac{\text{count}(w_1, w_2)}{\text{count}(w_1)}$$

- Collect counts over a large text corpus
- Millions to billions of words are easy to get
(trillions of English words available on the web)

Example: 3-Gram

- Counts for trigrams and estimated word probabilities

the green (total: 1748)			the red (total: 225)			the blue (total: 54)		
word	c.	prob.	word	c.	prob.	word	c.	prob.
paper	801	0.458	cross	123	0.547	box	16	0.296
group	640	0.367	tape	31	0.138	.	6	0.111
light	110	0.063	army	9	0.040	flag	6	0.111
party	27	0.015	card	7	0.031	,	3	0.056
ecu	21	0.012	,	5	0.022	angel	3	0.056

- 225 trigrams in the Europarl corpus start with *the red*
 - 123 of them end with *cross*
- maximum likelihood probability is $\frac{123}{225} = 0.547$.

How good is the LM?

- A good model assigns a text of real English W a high probability
- This can be also measured with cross entropy:

$$H(W) = \frac{1}{n} \log p(W_1^n)$$

- Or, **perplexity**

$$\text{perplexity}(W) = 2^{H(W)}$$

Example: 3-Gram

prediction	p_{LM}	$-\log_2 p_{\text{LM}}$
$p_{\text{LM}}(\text{i} \langle /s \rangle \langle s \rangle)$	0.109	3.197
$p_{\text{LM}}(\text{would} \langle s \rangle \text{i})$	0.144	2.791
$p_{\text{LM}}(\text{like} \text{i would})$	0.489	1.031
$p_{\text{LM}}(\text{to} \text{would like})$	0.905	0.144
$p_{\text{LM}}(\text{commend} \text{like to})$	0.002	8.794
$p_{\text{LM}}(\text{the} \text{to commend})$	0.472	1.084
$p_{\text{LM}}(\text{rapporteur} \text{commend the})$	0.147	2.763
$p_{\text{LM}}(\text{on} \text{the rapporteur})$	0.056	4.150
$p_{\text{LM}}(\text{his} \text{rapporteur on})$	0.194	2.367
$p_{\text{LM}}(\text{work} \text{on his})$	0.089	3.498
$p_{\text{LM}}(. \text{his work})$	0.290	1.785
$p_{\text{LM}}(\langle /s \rangle \text{work .})$	0.99999	0.000014
average		2.634

Comparison 1–4-Gram

word	unigram	bigram	trigram	4-gram
i	6.684	3.197	3.197	3.197
would	8.342	2.884	2.791	2.791
like	9.129	2.026	1.031	1.290
to	5.081	0.402	0.144	0.113
commend	15.487	12.335	8.794	8.633
the	3.885	1.402	1.084	0.880
rapporteur	10.840	7.319	2.763	2.350
on	6.765	4.140	4.150	1.862
his	10.678	7.316	2.367	1.978
work	9.993	4.816	3.498	2.394
.	4.896	3.020	1.785	1.510
</s>	4.828	0.005	0.000	0.000
average	8.051	4.072	2.634	2.251
perplexity	265.136	16.817	6.206	4.758

Unseen N-Grams

- We have seen *i like to* in our corpus
- We have never seen *i like to smooth* in our corpus

→ $p(\text{smooth} | \text{i like to}) = 0$

- Any sentence that includes *i like to smooth* will be assigned probability 0

Add-One Smoothing

- For all possible bigrams, add the count of one.

$$p = \frac{c + 1}{n + v^2}$$

- c = count of n-gram in corpus
- n = count of history
- v = vocabulary size
- But there are many more unseen n-grams than seen n-grams
- Example: Europarl 2-bigrams:
 - 86,700 distinct words
 - $86,700^2 = 7,516,890,000$ possible bigrams
 - but only about 30,000,000 words (and bigrams) in corpus

Adjusted Counts

- Previously, we estimated probabilities based on actual counts

$$p = \frac{c}{n}$$

- Now, we change the formula to estimate smoothed probabilities

$$p_{\text{smoothed}} = \frac{c + 1}{n + v}$$

- Another view: we adjusted the counts c

$$p_{\text{smoothed}} = \frac{c^*}{n} \quad \Rightarrow \quad c^* = n p_{\text{smoothed}} = n \frac{c + 1}{n + v}$$

Add- α Smoothing

- Add $\alpha < 1$ to each count

$$p = \frac{c + \alpha}{n + \alpha v}$$

- What is a good value for α ?
- Could be optimized on held-out set

Example: 2-Grams in Europarl

Count	Adjusted count	
c	$n \frac{c+1}{n+v^2}$	$n \frac{c+\alpha}{n+\alpha v^2}$
0	0.00378	0.00016
1	0.00755	0.95725
2	0.01133	1.91433
3	0.01511	2.87141
4	0.01888	3.82850
5	0.02266	4.78558
6	0.02644	5.74266
8	0.03399	7.65683
10	0.04155	9.57100
20	0.07931	19.14183

- Add- α smoothing with $\alpha = 0.00017$
- t_c are average counts of n-grams in test set that occurred c times in corpus

Deleted Estimation

- Estimate true counts in held-out data
 - split corpus in two halves: training and held-out
 - counts in training $C_t(w_1, \dots, w_n)$
 - number of ngrams with training count r : N_r
 - total times ngrams of training count r seen in held-out data: T_r

- Held-out estimator:

$$p_h(w_1, \dots, w_n) = \frac{T_r}{N_r N} \quad \text{where } \text{count}(w_1, \dots, w_n) = r$$

- Both halves can be switched and results combined

$$p_h(w_1, \dots, w_n) = \frac{T_r^1 + T_r^2}{N(N_r^1 + N_r^2)} \quad \text{where } \text{count}(w_1, \dots, w_n) = r$$

Example: 2-Grams in Europarl

Count	Adjusted count		Test count
c	$n \frac{c+1}{n+v^2}$	$n \frac{c+\alpha}{n+\alpha v^2}$	t_c
0	0.00378	0.00016	0.00016
1	0.00755	0.95725	0.46235
2	0.01133	1.91433	1.39946
3	0.01511	2.87141	2.34307
4	0.01888	3.82850	3.35202
5	0.02266	4.78558	4.35234
6	0.02644	5.74266	5.33762
8	0.03399	7.65683	7.15074
10	0.04155	9.57100	9.11927
20	0.07931	19.14183	18.95948

- Add- α smoothing with $\alpha = 0.00017$
- t_c are average counts of n-grams in test set that occurred c times in corpus

Good-Turing Smoothing

- Adjust actual counts r to expected counts r^* with formula

$$r^* = (r + 1) \frac{N_{r+1}}{N_r}$$

- N_r number of n-grams that occur exactly r times in corpus
- N_0 total number of n-grams

Good-Turing for 2-Grams in Europarl

Count	Count of counts	Adjusted count	Test count
r	N_r	r^*	t_r
0	7,514,941,065	0.00015	0.00016
1	1,132,844	0.46539	0.46235
2	263,611	1.40679	1.39946
3	123,615	2.38767	2.34307
4	73,788	3.33753	3.35202
5	49,254	4.36967	4.35234
6	35,869	5.32928	5.33762
8	21,693	7.43798	7.15074
10	14,880	9.31304	9.11927
20	4,546	19.54487	18.95948

adjusted count fairly accurate when compared against the test count

Derivation of Good-Turing

- A specific n-gram α occurs with (unknown) probability p in the corpus
- Assumption: all occurrences of an n-gram α are independent of each other
- Number of times α occurs in corpus follows binomial distribution

$$p(c(\alpha) = r) = b(r; N, p_i) = \binom{N}{r} p^r (1 - p)^{N-r}$$

Derivation of Good-Turing (2)

- Goal of Good-Turing smoothing: compute *expected count* c^*
- Expected count can be computed with help from binomial distribution:

$$\begin{aligned} E(c^*(\alpha)) &= \sum_{r=0}^N r p(c(\alpha) = r) \\ &= \sum_{r=0}^N r \binom{N}{r} p^r (1-p)^{N-r} \end{aligned}$$

- Note again: p is unknown, we cannot actually compute this

Derivation of Good-Turing (3)

- Definition: expected number of n-grams that occur r times: $E_N(N_r)$
- We have s different n-grams in corpus
 - let us call them $\alpha_1, \dots, \alpha_s$
 - each occurs with probability p_1, \dots, p_s , respectively
- Given the previous formulae, we can compute

$$\begin{aligned} E_N(N_r) &= \sum_{i=1}^s p(c(\alpha_i) = r) \\ &= \sum_{i=1}^s \binom{N}{r} p_i^r (1 - p_i)^{N-r} \end{aligned}$$

- Note again: p_i is unknown, we cannot actually compute this

Derivation of Good-Turing (4)

- Reflection
 - we derived a formula to compute $E_N(N_r)$
 - we have N_r
 - for small r : $E_N(N_r) \simeq N_r$
- Ultimate goal compute expected counts c^* , given actual counts c

$$E(c^*(\alpha)|c(\alpha) = r)$$

Derivation of Good-Turing (5)

- For a particular n-gram α , we know its actual count r
- Any of the n-grams α_i may occur r times
- Probability that α is one specific α_i

$$p(\alpha = \alpha_i | c(\alpha) = r) = \frac{p(c(\alpha_i) = r)}{\sum_{j=1}^s p(c(\alpha_j) = r)}$$

- Expected count of this n-gram α

$$E(c^*(\alpha) | c(\alpha) = r) = \sum_{i=1}^s N p_i p(\alpha = \alpha_i | c(\alpha) = r)$$

Derivation of Good-Turing (6)

- Combining the last two equations:

$$\begin{aligned} E(c^*(\alpha) | c(\alpha) = r) &= \sum_{i=1}^s N p_i \frac{p(c(\alpha_i) = r)}{\sum_{j=1}^s p(c(\alpha_j) = r)} \\ &= \frac{\sum_{i=1}^s N p_i p(c(\alpha_i) = r)}{\sum_{j=1}^s p(c(\alpha_j) = r)} \end{aligned}$$

- We will now transform this equation to derive Good-Turing smoothing

Derivation of Good-Turing (7)

- Repeat:

$$E(c^*(\alpha) | c(\alpha) = r) = \frac{\sum_{i=1}^s N p_i p(c(\alpha_i) = r)}{\sum_{j=1}^s p(c(\alpha_j) = r)}$$

- Denominator is our definition of expected counts $E_N(N_r)$

Derivation of Good-Turing (8)

- Numerator:

$$\begin{aligned}\sum_{i=1}^s N p_i p(c(\alpha_i) = r) &= \sum_{i=1}^s N p_i \binom{N}{r} p_i^r (1 - p_i)^{N-r} \\ &= N \frac{N!}{N - r! r!} p_i^{r+1} (1 - p_i)^{N-r} \\ &= N \frac{(r + 1)}{N + 1} \frac{N + 1!}{N - r! r + 1!} p_i^{r+1} (1 - p_i)^{N-r} \\ &= (r + 1) \frac{N}{N + 1} E_{N+1}(N_{r+1}) \\ &\simeq (r + 1) E_{N+1}(N_{r+1})\end{aligned}$$

Derivation of Good-Turing (9)

- Using the simplifications of numerator and denominator:

$$\begin{aligned} r^* &= E(c^*(\alpha) | c(\alpha) = r) \\ &= \frac{(r + 1) E_{N+1}(N_{r+1})}{E_N(N_r)} \\ &\simeq (r + 1) \frac{N_{r+1}}{N_r} \end{aligned}$$

- QED