N-gram Language models and Smoothing

Mittul Singh

16.11.2015

Slides inspired by Philipp Koehn's slides available at:
http://www.statmt.org/book/slides/07-language-models.pdf

Recap

- Language modelling applications
- Perplexity (PPL) : 2^H, where H is cross entropy
- MLE: $P(w_2|w_1) = count(w_2,w_1)/count(w_1)$

Language Models

- Language models answer the question:

  How likely is it that a string of English words is good English?

- Help with reordering

  p_{LM}(the house is small) > p_{LM}(small is the house)
- Help the with word choice

  p_{LM}(easy to recognise speech) > p_{LM}(easy to wreck a nice beach)

- They define the probability of a string of words

N-Gram Language Models

- Given: a string of English Words $W = w_1, w_2, w_3, \ldots, w_n$
- Question: what is $p(W)$?
- Sparse data: Many perfectly good English sentences might not have been recorded (or written)

—> Decomposing $p(W)$ using the chain rule:

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

(not much gained yet, $p(w_n| w_1, w_2, w_3, \ldots, w_{n-1})$ is equally sparse)

Markov Chain

- Markov Assumption:
  - ✳ only recent 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:

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

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

Estimating N-Gram Probabilities

- Maximum likelihood estimation

$$p(w_2|w_1) = count\,(w_1, w_2)/count(w_1)$$

- 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)

word	c.	prob.
paper	801	0,458
group	640	0,367
light	110	0,063
party	27	0,015

the red (total: 225)

word	c.	prob.
cross	123	0,547
tape	31	0,138
army	9	0,040
card	7	0,031

the blue (total: 54)

word	c.	prob.
box	16	0,296
.	6	0,111
army	6	0,111
card	3	0,056

— 225 trigrams in the Europarl corpus start with the red

— 123 of them end with cross

—> maximum likelihood probability is 123/225 = 0,547

Example: 3-Gram

prediction	p_{LM}	$-\log_2 p_{LM}$
p_{LM}(i | </s><s>)	0,109	3,197
p_{LM}(would | <s> i)	0,144	2,791
p_{LM}(like | i would)	0,489	1,031
p_{LM}(to | would like)	0,905	0,144
p_{LM}(commend | like to)	0,002	8,794
p_{LM}(him | to commend)	0,472	2,367
p_{LM}(. | commend him)	0,290	1,785
p_{LM}(</s> | him .)	0,999	0,000
	average	2,513

Perplexity

$$\text{PPL} = 2\verb|^|\left(-\frac{1}{N}\Sigma_i \log_2 p(w_i \mid w_{i-1}\ldots)\right)$$

- where N is the number of tokens in data D, $w_i \in D$
- Shannon-McMillan-Breiman theorem

Comparison 1 to 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
him	10,678	7,316	2,367	0,880
.	4,896	3,020	1,785	1,510
</s>	4,828	0,005	0,000	0,000
average	7,613	3,898	2,513	2,302
perplexity	195,768	14,907	5,708	4,913

Unseen N-Grams

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

—> *p(smooth | i like to) = 0*

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

Seen N-Grams

- *p(I like to commend)* computed on training set
- Is it representative on test set?
    - Does it overfit ?

Add-One Smoothing

- For all possible n-grams, add the count of one

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

  - 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 bigrams:
  - 86700 distinct words
  - 86700^2 = 7516890000 possible bigrams (~ 7,517 billion)
  - but only about 30000000 bigrams (~30 million) in corpus

Add-α Smoothing

- Add $\alpha < 1$ to each count

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

- What is a good value of α ?

- Could be optimised on held-out set

Example: 2-Grams in Europarl

c	$(c+1)\frac{N}{N+v^2}$	$(c+\alpha)\frac{N}{N+\alpha v^2}$	t_c
0	0,00378	0,00016	0,00016
1	0,00755	0,95725	0,46235
5	0,02266	4,78558	4,35234
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 α=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 $C_t(w_1,\ldots, w_n)$
  - number of n-grams with training count r: N_r
  - total times n-grams of training count r seen in held-out data: T_r
- Held-out estimator:

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

- Both halves can be switched and results combined

$$p_h(w_1,\ldots, w_n) = \frac{T^1_r + T^2_r}{N(N^1_r + N^2_r)} \text{ where } count(w_1,\ldots, w_n) = r$$

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
- This smoothing works well for low r
  - It fails for high r, as $N_r = 0$

Good-Turing for 2-Grams in Europarl

Count	Count of counts	Adjusted count	Test count
r	***N_r***	***r^****	***t***
0	7514941065	0,00015	0,00016
1	1132844	0,46539	0,46235
5	49254	4,36967	4,35234
8	21693	7,43798	7,15074
10	14880	9,31304	9,11927
20	4546	19,54487	18,95948

adjusted count fairly accurate when compared against the test count

Back-Off

- In given corpus, we may never observe
  - Scottish beer drinkers
  - Scottish beer eaters
- Both have count 0
  - our smoothing methods will assign then same probability
- Better: back-off to bigrams:
  - beer drinkers
  - beer eaters

Interpolation

- Higher and lower order n-gram models have different strengths and weaknesses
  - high-order n-grams are sensitive to more context, but have sparse counts
  - low-order n-grams consider only very limited context, but have robust counts
- Combine them

$$p_I(w_3|w_1,w_2) = \lambda_1 p_1(w_3) + \lambda_2 p_2(w_3|w_2) + \lambda_3 p_3(w_3|w_1,w_2)$$

$$\lambda_1 + \lambda_2 + \lambda_3 = 1$$

Recursive Interpolation

- We can trust some histories $w_{i-n+1},\ldots,w_{i-1}$ more than others
- Condition interpolation weights on history: $\lambda(w_{i-n+1},\ldots,w_{i-1})$
- Recursive definition of interpolation

$$p^I_n(w_i|w_{i-n+1},\ldots,w_{i-1}) = \lambda(w_{i-n+1},\ldots,w_{i-1})\, p(w_i|w_{i-n+1},\ldots,w_{i-1}) + (1-\lambda(w_{i-n+1},\ldots,w_{i-1}))p^I_n(w_i|w_{i-n+2},\ldots,w_{i-1})$$

Example: Recursive Interpolation

Consider a trigram: *in spite of* (BTW: GT = Good Turing)

$$p_I(of \mid in\ spite) = \lambda_{in\ spite}\, p_{GT}(of \mid in\ spite) + (1- \lambda_{in\ spite})p_I(of \mid spite)$$

$$= \lambda_{in\ spite}\, p_{GT}(of \mid in\ spite) \qquad (\because \text{ expanding } p_I(of \mid spite))$$

$$+ (1- \lambda_{in\ spite}) \{ \lambda_{spite}\, p_{GT}(of \mid spite) + (1-\lambda_{spite})p_I(of)\}$$

$$= \lambda_{in\ spite}\, p_{GT}(of \mid in\ spite) \qquad (\because\ p_I(of) = p_{GT}(of))$$

$$+ (1- \lambda_{in\ spite}) \{ \lambda_{spite}\, p_{GT}(of \mid spite) + (1-\lambda_{spite})p_{GT}(of)\}$$

Back-Off

- Trust the highest order language model that contains the n-gram

$$p^{BO}_n(w_i|w_{i-n+1},\ldots,w_{i-1}) =$$

$$\begin{cases} \alpha_n(w_i|w_{i-n+1},\ldots,w_{i-1}) & \text{if } count_n(w_{i-n+1},\ldots,w_i) > 0 \\ d_n(w_{i-n+1},\ldots,w_{i-1})\, p^{BO}_n(w_i|w_{i-n+2},\ldots,w_{i-1}) & \text{else} \end{cases}$$

- Requires
  - adjusted prediction model $\alpha_n(w_i|w_{i-n+1},\ldots,w_{i-1})$
  - discounting function $d_n(w_{i-n+1},\ldots,w_{i-1})$:left over mass from the adjusted predicted model

Back-Off with Good-Turing Smoothing

- Previously, we computed n-gram probabilities based on relative frequency

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

- Good Turing smoothing adjusts counts c to expected counts c^*

$$count^*(w_1,w_2) \leq count(w_1,w_2)$$

- We use the expected counts for the prediction model (but 0^* remains 0)

$$\alpha(w_2|w_1) = \frac{count^*(w_1,w_2)}{count(w_1)}$$

- This leaves probability mass for the discounting function

$$d_2(w_1) = 1- \Sigma_{w2}\, \alpha(w_2|w_1)$$

Example: Back-Off with GT Smoothing

- p_{BO}(of | spite) = α_{GT}(of | spite) [$\because$ c(spite of) > 0]
- p_{BO}(**.** | spite) = α_{GT}(**.** | spite) [$\because$ c(spite **.**) > 0]
- $\alpha_{GT} < p_{MLE}$, to allow for unseen words
- d(spite) = 1 - α_{GT}(of | spite) + α_{GT}(**.** | spite) [a piece of the pie left for unseen words]
- Test set: Cut your nose to *spite your* face
- p_{BO}(your | spite) = d(spite) x p_{GT}(your)

Diversity of Histories

- Consider the word *York*
  - fairly frequent word in Europarl, occurs 477 times
  - as frequent as *foods*, *indicates* and *provides*
  - in unigram language model: a respectable probability
- However, it almost always directly follows *New* (473 times)
- Recall: unigram model only used, if the bigram model inconclusive
  - *York* unlikely second word in unseen bigram
  - in back-off unigram model, *York* should have low probability

Kneser-Ney Smoothing

- Kneser-Ney smoothing takes diversity of histories into account
- Count of histories for a word

$$N_{1+}(\bullet w) = \left|\{w_i : c(w_i, w) > 0\}\right|$$

- Recall: maximum likelihood estimation of unigram language model

$$p_{ML}(w) = \frac{c(w)}{\Sigma_i \, c(w_i)}$$

- In Kneser-Ney smoothing, replace raw counts with count of histories

$$p_{KN}(w) = \frac{N_{1+}(\bullet w)}{\Sigma_i \, N_{1+}(\bullet w_i)}$$

Example: Kneser-Ney Smoothing

I can't see without my _______

$p(\text{York} \mid \text{my}) = \alpha_{GT}(\text{my York}) + d_{my} \times \alpha_{GT}(\text{York})$

$$= \frac{c^*(\text{my York})}{c(\text{my})} + d_{my} \times \frac{c^*(\text{York})}{N}$$

=> $p(\text{York} \mid \text{my}) > p(\text{glasses} \mid \text{my})$

Applying Kneser-Ney …

$p(\text{York} \mid \text{my}) = \alpha_{KN}(\text{my York}) + d_{my} \times \alpha_{KN}(\text{York})$

$$= \frac{c^*(\text{my York})}{c(\text{my})} + d_{my} \times \frac{N_{1+}(\bullet\,\text{York})}{N_{1+}(\bullet\bullet)}$$

=> $p(\text{York} \mid \text{my}) < p(\text{glasses} \mid \text{my})$

Modified Kneser Ney Smoothing

- Absolute discounting: subtract a fixed *D* from all non-zero counts

$$\alpha(w_n|w_1,\ldots,w_{n-1}) = \frac{c(w_1,\ldots,w_n) - D}{\sum_w c(w_1,\ldots,w_{n-1},w)}$$

- Refinement: three different discount values

$$D(c) \begin{cases} D_1 \text{ if } c=1 \\ D_2 \text{ if } c=2 \\ D_{3+} \text{ if } c>=3 \end{cases}$$

Discount Parameters

- Optimal discounting parameters D_1,D_2,D_{3+}can be computed quite easily

$$Y=\frac{N_1}{N_1+2N_2}$$

$$D_1=1-2Y\frac{N_2}{N_1}$$

$$D_2=2-3Y\frac{N_3}{N_2}$$

$$D_{3+}=3-4Y\frac{N_4}{N_3}$$

- Values N_c are the counts of n-grams with exactly count *c*

Interpolated Back-Off

- Back-off models use only highest order n-gram
  - if sparse, not very reliable
  - two different n-grams with same history occur once —> same probability
  - one may be an outlier, the other under-represented in training
- To remedy this, always consider the lower-order back-off models
- Adapting the α function into interpolated α_I function by adding back-off

$$\alpha_I(w_n|w_1,\ldots,w_{n-1}) = \alpha(w_n|w_1,\ldots,w_{n-1}) + d(w_1,\ldots,w_{n-1})p_I(w_n|w_2,\ldots,w_{n-1})$$

- Note that *d* function needs to adapted as well

Evaluation

Evaluation of smoothing methods:

Perplexity for language models trained on the Europarl corpus

Smoothing method	bigram	trigram	4-gram
Good-Turing	96,2	62,9	59,9
Modified Kneser-Ney	95,4	61,6	58,6
Interpolated Modified Kneser Ney	94,5	59,3	54,0

Summary

- Language models: How likely is a string of English words good English ?
- N-Gram models (Markov Assumption)
- Count smoothing
  - add-one, add-α
  - deleted estimation
  - Good Turing
- Interpolation and back off
  - Good Turing
  - Kneser-Ney