Language Models

Philipp Koehn

5 September 2023

Language models

- **Language models** answer the question:

  How likely is a string of English words good English?

- Help with reordering

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

- Help with word choice

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

N-Gram Language Models

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

$\rightarrow$ Decomposing $p(W)$ using the chain rule:

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

(not much gained yet, $p(w_n|w_1, w_2, ...w_{n-1})$ 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)

  $\rightarrow$ k**th order Markov model**

- For instance 2-gram language model:

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

- 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)

word	c.	prob.
paper	801	0.458
group	640	0.367
light	110	0.063
party	27	0.015
ecu	21	0.012

the red (total: 225)

word	c.	prob.
cross	123	0.547
tape	31	0.138
army	9	0.040
card	7	0.031
,	5	0.022

the blue (total: 54)

word	c.	prob.
box	16	0.296
.	6	0.111
flag	6	0.111
,	3	0.056
angel	3	0.056

  – 225 trigrams in the Europarl corpus start with the red
  – 123 of them end with cross

$\rightarrow$ 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:

$$H(W) = -\frac{1}{n} \log_2 p(W_1^n)$$

- Or, **perplexity**

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

Example: 3-Gram

prediction	p_{LM}	-$\log_2 p_{\text{LM}}$
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}(the|to commend)	0.472	1.084
p_{LM}(rapporteur|commend the)	0.147	2.763
p_{LM}(on|the rapporteur)	0.056	4.150
p_{LM}(his|rapporteur on)	0.194	2.367
p_{LM}(work|on his)	0.089	3.498
p_{LM}(.|his work)	0.290	1.785
p_{LM}(</s>|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

count smoothing

Unseen N-Grams

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

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

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

Add-One Smoothing

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

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

  - 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

efficiency

Managing the Size of the Model

- Millions to billions of words are easy to get
  (trillions of English words available on the web)

- But: huge language models do not fit into RAM

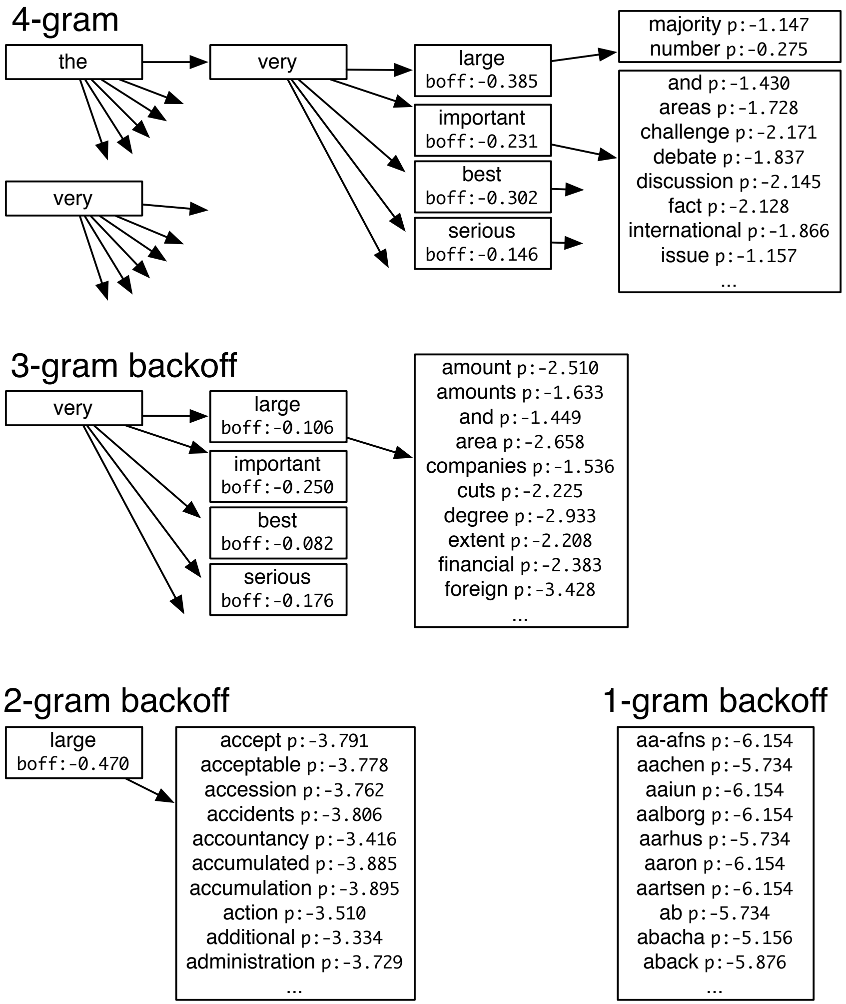
Number of Unique N-Grams

Number of unique n-grams in Europarl corpus
29,501,088 tokens (words and punctuation)

Order	Unique n-grams	Singletons
unigram	86,700	33,447 (38.6%)
bigram	1,948,935	1,132,844 (58.1%)
trigram	8,092,798	6,022,286 (74.4%)
4-gram	15,303,847	13,081,621 (85.5%)
5-gram	19,882,175	18,324,577 (92.2%)

→ remove singletons of higher order n-grams

Efficient Data Structures



- Need to store probabilities for
  - the very large majority
  - the very large number
- Both share history
  the very large

→ no need to store history twice

→ Trie

Reducing Vocabulary Size

- For instance: each number is treated as a separate token
- Replace them with a number token NUM
  - but: we want our language model to prefer

$$p_{\text{LM}}(\text{I pay 950.00 in May 2007}) > p_{\text{LM}}(\text{I pay 2007 in May 950.00})$$

  - not possible with number token

$$p_{\text{LM}}(\text{I pay NUM in May NUM}) = p_{\text{LM}}(\text{I pay NUM in May NUM})$$

- Replace each digit (with unique symbol, e.g., @ or 5), retain some distinctions

$$p_{\text{LM}}(\text{I pay 555.55 in May 5555}) > p_{\text{LM}}(\text{I pay 5555 in May 555.55})$$