AI611μ Word Prediction with
N-Grams Model using Python

Session 2

N-Grams Model Training and Model Evaluation

Sébastien Combéfis

Winter 2020

This work is licensed under a Creative Commons Attribution – NonCommercial – NoDerivatives 4.0 International License.

Objectives

- How to train an N-grams model from and for a given corpus?

  Computing the N-grams probabilities from statistics

- How to evaluate a trained N-grams model?

  Testing the quality of the obtained probabilities for a test set

- How to improve N-grams models by smoothing them?

  Tackling sparse data that can result from bad/small corpus

Model Training

Berkeley Restaurant Project

- Dialogue system that answered questions about restaurants

  From a database of restaurants in Berkeley, California

- A sample of the 9332 user queries in the database
  - can you tell me about any good cantonese restaurants close by
  - mid priced thai food is what i'm looking for
  - tell me about chez panisse
  - can you give me a listing of the kinds of food that are available
  - i'm looking for a good place to eat breakfast
  - when is caffe venezia open during the day

Corpus Statistics

- Bigram counts from a selected set of eight words

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

- Unigram count for the eight selected words from the corpus

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

Bigram Probabilities

- Bigram probabilities obtained after normalisation

  Obtained by dividing the bigram matrix by the unigram vector

- Bigram model captures several linguistic phenomena
  - Strictly syntactic facts such as a verb comes after a pronoun
  - Cultural facts such as low probability of asking English food

	i	**want**	**to**	**eat**	**chinese**	**food**	**lunch**	**spend**
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Sentence Probability

- Two symbols to identify the beginning and end of sentence

  For example, we have $P(\texttt{i}|\texttt{<s>}) = 0.25$ *and* $P(\texttt{</s>}|\texttt{food}) = 0.68$

- Possible to compute the probability of a sentence
  - Assuming $P(\texttt{food}|\texttt{english}) = 0.5$ and $P(\texttt{english}|\texttt{want}) = 0.0011$
  - We have $P(\texttt{<s>i want english food</s>})$
    $= P(\texttt{i}|\texttt{<s>})P(\texttt{want}|\texttt{i})P(\texttt{english}|\texttt{want})P(\texttt{food}|\texttt{english})P(\texttt{</s>}|\texttt{food})$
    $= 0.25 \times 0.33 \times 0.0011 \times 0.5 \times 0.68$
    $= 0.000031$

Unknown Word

- Closed vocabulary when size of vocabulary known in advance

  We know all the words that can occur, in the test set

- Sometimes there are unknown words (OOV out of vocabulary)
  - The OOV rate measure how many such words are present
  - Addition of a special `<UNK>` pseudo-word

- Special training method when in an open vocabulary setting
  - Replace words not in a chosen vocabulary by `<UNK>`
  - Replace the first occurence of every word type by `<UNK>`

Model Evaluation

Training and Test Set

- Probabilities of an N-grams model trained from a corpus

  With sufficient training data, trigram models are better

- Statistical model trained on some data, tested on others
  - Training statistical parameters of the model on training set
  - Computing probabilities on the test set

- Training-and-testing paradigm to compare N-grams models

  An N-grams model can be evaluated with the perplexity

Test Set

- Important to separate the training set from the test set

  Training on the test set can result in biased models

- Possible to have other divisions of data
  - **Held-out** set to compute other parameters of the model
  - Possible to have multiple **test** set to avoid tuning to one
  - **Development test** set compared to a fresh test set

- Keeping a large training set important to train a good model

  80% *for training*, 10% *for development and* 10% *for test*

N-Grams Sensitivity

- N-grams model is very dependent on the training corpus

  Probabilities often encode specific facts about the training corpus

- Better job of modelling the training corpus as N increases

  Can be observed by randomly generating sentences with $N \uparrow$

- Examples with the Wall Street Journal corpus (40e6 words)

  1. Months the my and issue of year foreign new exchange's...
  2. Last December through the way to preserve the Hudson...
  3. They also point to ninety nine point six billion dollars from two...

Evaluating *N*-Grams

- Extrinsic evaluation for end2end evaluation of language model
    - Embed it in application and measure total performance of it
    - Also referred to as an *in vivo* evaluation
    - Only way to know if improvement really help the task at hand
- Independent quality measure with intrinsic evaluation
    - Quickly evaluate potential improvements in a language model
    - Often correlates with extrinsic improvements

Perplexity Measure

- Perplexity measure best prediction on test set $W = w_1 w_2 ... w_N$

$$PP(W) = P(w_1 w_2 ... w_N)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1 w_2 ... w_N)}}$$

- Perplexity can be computed thanks to the chain rule

$$PP(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_1 ... w_{i-1})}}$$

- Perplexity is simplified in the case of bigram model

$$PP(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_{i-1})}}$$

Perplexity Properties

- The higher the conditional probability, the lower the perplexity

  Minimising perplexity maximises test set probability

- No guarantee on extrinsic improvement with perplexity

  But perplexity correlates with extrinsic improvements

- For example, perplexities of a 1.5 millions words WSJ test set

	Unigram	**Bigram**	**Trigram**
Perplexity	962	170	109

Language Branching Factor

- Perplexity can be seen as weighted average branching factor

  Number of possible next words that can follow any word

- For example, consider strings of digits of length N

  $$PP(W) = P(w_1 w_2 ... w_N)^{-\frac{1}{N}} = \left(\frac{1}{10}^N\right)^{-\frac{1}{N}} = \frac{1}{10}^{-1} = 10$$

- What if digit zero occurs 10 times more often than others?

  The perplexity is expected to be lower...

Smoothing

Smoothing

- Sparse data problem since MLE based on particular training

  Perfectly acceptable English word sequences may be missing

- Modification on the MLE estimates with smoothing
  - Focus on N-grams events that were incorrectly assumed $P = 0$
  - Shaving a little bit of probability mass, piling it on zero counts

- Laplace smoothing adds 1 to counts before normalisation
  - Does not perform well enough for modern N-grams models
  - Introduces many of the concepts seen in other algorithms

Unigram Laplace Smoothing

- Adding one to count and adding V new observations

$$P(w_i) = \frac{c_i}{N} \qquad \textit{becomes} \qquad P_{Laplace}(w_i) = \frac{c_i + 1}{N + V}$$

- Easier definition if thinking with adjusted count
  - $c_i^* = (c_i + 1)\dfrac{N}{N + V}$
  - Normalisation of c_i^* by N gives probabilities P_i^*

Bigram Laplace Smoothing

- Updating bigrams from Berkeley Restaurant Project with $+1$

	i	want	to	eat	chinese	food	lunch	spend
i	6	828	1	10	1	1	1	3
want	3	1	609	2	7	7	6	2
to	3	1	5	687	3	1	7	212
eat	1	1	3	1	17	3	43	1
chinese	2	1	1	1	1	83	2	1
food	16	1	16	1	2	5	1	1
lunch	3	1	1	1	1	2	1	1
spend	2	1	2	1	1	1	1	1

- Normalisation with updated unigram counts with $+V$

$$P^*_{Laplace}(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n) + 1}{C(w_{n-1}) + V}$$

- Very big changes to the counts !
  - $C(\texttt{want to})$ changed from 608 to 238
  - $P(\texttt{to}|\texttt{want})$ decreases from 0.66 to 0.26

References

- Daniel Jurafsky, & James H. Martin (2008). *Speech and Language Processing* (Second Edition), Pearson, ISBN: 978-0-135-04196-3.
- Shashank Kapadia (2019). *Language Models: N-Gram: A step into statistical language modeling*, March 26, 2019. https://towardsdatascience.com/introduction-to-language-models-n-gram-e323081503d9
- arvindpdmn (2020). *N-Gram Model*, January 25, 2020. https://devopedia.org/n-gram-model
- David Masse (2018). *Using Perplexity to Evaluate a Natural-Language Model*, September 11, 2018. https://medium.com/@davidmasse8/using-perplexity-to-evaluate-a-word-prediction-model-8820cf3fd3aa
- Desiré De Waele (2016). *Building an N-gram Model*, September 2, 2016. http://rstudio-pubs-static.s3.amazonaws.com/211935_7d57493909b7452196cf3e585c32ffa5.html

Credits

- Apionid, May 2, 2015, https://www.flickr.com/photos/apionid/16720897144.
- Kaja Avberšek, May 19, 2011, https://www.flickr.com/photos/kaja_a/5776006254.
- Noj Han, March 7, 2009, https://www.flickr.com/photos/nojhan/3392016808.