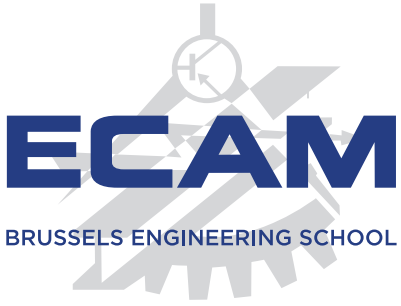

AI611μ Word Prediction with
N-Grams Model using Python

Session 1

Word Prediction Problem and *N*-Grams Model

Sébastien Combéfis

Winter 2020

Objectives

- Understand the automatic word prediction problem

  How to predict the next word of a given sequence of words

- Computing statistics based on a set of corpus

  Collection of data used to train prediction models

- Using *N*-grams models to solve word prediction problem

  Simple N-gram and bigram and N-grams models

Word Prediction

Word Prediction

- How to predict the next word based on $N - 1$ words

  Using probabilistic models to find a very likely word

- What about the sentence *"Please turn your homework..."*?
  - The words *"in"* or *"over"* are very likely possible next words
  - The word *"the"* is probably not a good next word

- Main technique based on probabilistic model

  Based on a statistical analysis of sequences of words

Text Generation

- Possible to generate a text following a trained model

  Generated text will followed the same distribution

- Generating an Eminem style song

"and i should not be a king when you feel em
while he play piano
you better lose yourself and make her fall on her
and its absurd how people hang on every word
off a plank and
look i was but im
teeter totter caught up between being a father and a primadonna
at least once in a while
now who thinks their arms are long enough to one day grow up
but for me to try to get em motivated"

Predictive Keyboard

- Suggesting corrections and possible next words to the user

  Simple embedded dictionary or more sophisticated NLP solutions

- An NLP model should be trained with datasets

  To better understand the user and how he/she writes

- Different possible helps/hints can be provided to the user

  Spelling and auto-correction, prediction and auto-completion

Google Smart Compose

- Google AI-based tool to help drafting emails faster

  Generated text will followed the same distribution

- Operating in background to propose suggestions as you type
  - Predicting next word with *N*-grams, BoW, RNN-LM models
  - Taking into account subject and previous email bodies


Corpus

Corpora (1)

- Corpus is a computer-readable collection of text or speech

  Such as the Brown (text) and the Switchboard (voice) corpora

- Million-word collection of samples from 500 written texts
  - Punctuation is critical for finding boundaries of things (. , ; :)
  - and to identify some aspects of meaning (? ! ")
- 2430 telephone conversations averaging 6 minutes each
  - Two kinds of disfluencies : fragment and fillers/filled pauses

    "I do uh main- mainly business data processing"
  - to remove for automatic dictation or keep to analyse

Corpora (2)

- Word counting can be done in several ways
  - Take into account punctuation, disfluencies or not?
  - Capitalised words, are “they” and “They” different?
  - What about inflected forms such as “cats” vs “cat”?
- For English, ignore case and use wordform (full inflected)

  Possibly: tokenisation, text normalisation, lemmatisation...

Free and Open Source Corpora

- Project Gutenberg provides large collection of books

  60000 free eBooks with several languages

- More formal Google 1 billion word corpus

  Standard corpus to train and evaluate language models

- Brown Univ. Std Corpus of Present-Day American English

  500 samples of English-language text (roughly one million words)

N-Grams Model

N-Grams Model

- An N-token sequence of words is an N-gram

  "please turn" is a bigram, "please turn your" is a trigram...

- An N-grams model computes the last word of an N-gram

  Predictive model making computation based on the previous ones

- Language models computes probability of a sequence of words

  N-gram models closely related to such statistical models

Simple N-Grams (1)

- Simple (unsmoothed) N-grams model based on a probability

  Highlight some intuitive motivations for N-grams

- Probability of a word w given some history h

  $P_1 = P(\textit{the}|\textit{its water is so transparent that})$

- Simple idea is to estimate P_1 from relative frequency count

  - Need a large enough corpus to compute statistics on
  - Estimate $P_1 = \dfrac{C(\textit{its water is so transparent that the})}{C(\textit{its water is so transparent that})}$

Simple N-Grams (2)

- Statistics on corpus not enough since language is creative

  New sentences are created all the time

- Need for cleverer ways to estimate two probabilities
  - Probability of a word w given a history h
  - Probability of an entire word sequence W

- Sequence of N words represented as $w_1 \ldots w_n$ or w_1^n

$$P(w_1^n) = \prod_{k=1}^{n} P(w_k | w_1^{k-1}) = P(w_1)P(w_2|w_1)P(w_3|w_1^2)\ldots P(w_n|w_1^{n-1})$$

(chain rule of probability)

Bigram Model (1)

- Bigram model approximates history by just the previous word

$$P(w_n \mid w_1^{n-1}) \approx P(w_n|w_{n-1})$$

- This is exactly what is called the Markov assumption
  - We can predict the probability of some future unit...
  - ...without looking too far in the past
- Simplification of the probability of a complete word sequence

$$P(w_1^n) \approx \prod_{k=1}^{n} P(w_k|w_{k-1}) = P(w_1)P(w_2|w_1)P(w_3|w_2)...P(w_n|w_{n-1})$$

Bigram Model (2)

- Bigram probabilities estimated with an MLE estimate

  Normalisation of the counts obtained from a corpus (to $\in [0, 1]$*)*

- Count of a particular bigram followed by a normalisation
  - Sum of all bigrams that share the same first word
  - $P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{\sum_w C(w_{n-1}w)} = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$

Bigram Model Example

- Bigram model example for a corpus with a single sentence

  "I train to drive a train."

	I	train	to	drive	a
I	0	0	0	0	0
train	1	0	0	0	1
to	0	0.5	0	0	0
drive	0	0	1	0	0
a	0	0	0	1	0

$$P(to|train)$$

$$=$$

$$\frac{C(train\ to)}{C(train)}$$

N-Grams Model

- Bigram model can be generalised to a N-grams model

  Looks into $N-1$ *words in the past to estimate* $P(w|h)$

- General equation for the N-grams approximation becomes

  $$P(w_n|w_1^{n-1}) \approx P(w_n|w_{n-N+1}^{n-1})$$

- MLE N-grams parameter estimation becomes

  - $$P(w_n|w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n-1} w_n)}{C(w_{n-N+1}^{n-1})}$$

    - MLE maximises likelihood of the training T given the model M

References

- Daniel Jurafsky, & James H. Martin (2008). *Speech and Language Processing* (Second Edition), Pearson, ISBN: 978-0-135-04196-3.
- Jose Corbacho (2019). *Android Predictive Keyboard*, January 12, 2019. https://proandroiddev.com/android-predictive-keyboard-e6c9df01e527
- Yonghui Wu (2018). *Smart Compose: Using Neural Networks to Help Write Emails*, May 16, 2018. https://ai.googleblog.com/2018/05/smart-compose-using-neural-networks-to.html
- Ashwin M J (2018). *Next Word Prediction using Markov Model*, March 16, 2018. https://medium.com/ymedialabs-innovation/next-word-prediction-using-markov-model-570fc0475f96
- Usman Malik (2019). *Python for NLP: Developing an Automatic Text Filler using N-Grams*, July 16, 2019. https://stackabuse.com/python-for-nlp-developing-an-automatic-text-filler-using-n-grams
- Jason Brownlee (2017). *Making Predictions with Sequences*, September 4, 2017. https://machinelearningmastery.com/sequence-prediction
- Sayantini Deb (2019). *Explore Markov Chains With Examples? Markov Chains With Python*, July 2, 2019. https://medium.com/edureka/introduction-to-markov-chains-c6cb4bcd5723

Credits

- Stefan Lins, February 28, 2007, https://www.flickr.com/photos/mrlins/425103299.
- Burns Library, Boston College, September 10, 2013, https://www.flickr.com/photos/bc-burnslibrary/9727280574.
- Randomthoughtstome, August 17, 2013, https://www.flickr.com/photos/90560049@N03/9532248976.