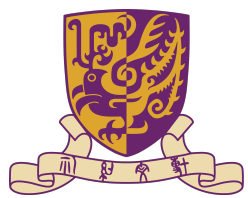
LING3401 Linguistics and Information Technology

Tutorial: Probability, n-gram, and perplexity

Yige Chen

The Chinese University of Hong Kong

January 22, 2025



Acknowledgment

- Some of the tutorial materials are based on: Dan Jurafsky and James H. Martin. *Speech and Language Processing* (3rd ed. draft). 2024.
- I owe my gratitude to my former advisor at the University of Washington. The way I deliver materials is heavily influenced by her approach.
- GPT-4o helped me write more than half of today's codes. Thanks GPT!

About Google Colaboratory (Colab)

- Google Colaboratory is a research tool for machine learning education and research. It's a Jupyter notebook environment that requires no setup to use.
- ...which means that you don't have to worry about setting up the Python environment on your device (at least for now)
- Today's Colab notebook has been posted

Probability: an introduction

- We are not introducing probability theory in a way like those in a math department
- If you are interested in knowing more, here's a very good textbook written by my undergraduate math advisor and his colleagues
  - David F. Anderson, Timo Seppäläinen, Benedek Valkó. *Introduction to Probability*. Cambridge University Press, 2017.
- Or, you can always ask me or GPT!

Basic concepts of probability

- The sample space Ω is the set of all possible outcomes.
  - For instance, $\Omega = \{\text{open}, \text{waitlist}, \text{closed}\}$ is a sample space consisting of all the possible statuses of a course when you register.
- A subset of the sample space is an event.
  - For instance, $A = \{\text{the course is not open}\} = \{\text{waitlist}, \text{closed}\}$ is an event which means that the course is not open (i.e., either closed or waitlist).

Basic concepts of probability

- $P(A)$ is the probability that event A occurs.
  - For instance, $P(A) = 0.2$ means the probability that the course you are looking at is no longer open is 0.2, i.e. about 2 out of 10 times on average you will see the course being closed or waitlist.
    - Note that it doesn't mean that if you randomly search for 10 courses, you'll for sure see 2 courses being closed or waitlist
    - Instead, if you randomly search for 1 course, you will expect a 20% chance that the course is not open
  - For each event A, we have $0 \leq P(A) \leq 1$. $P(A) = 0$ means A is never going to happen, and $P(A) = 1$ means A is always going to happen.
    - So in our case, $P(A) = 0$ means that all the courses always have available seats. What a great university!
    - $P(A) = 1$ means that all the courses are either closed or waitlisted.

Basic concepts of probability

- Two more concepts before we embark!
- $P(AB)$ is the joint probability for events A and B, i.e. the probability that A and B happen at the same time.
  - For instance, A is the event that one got good grades, B is the event that one studied hard, then $P(AB)$ is the probability that you randomly ask a student and found that he or she studied hard and got good grades.
- $P(A \mid B)$ is the conditional probability of A given B, i.e. the probability that A happens given that B happens.
  - For instance, A is the event that one got good grades, B is the event that one studied hard, then $P(A \mid B)$ is the probability that you randomly ask a student who studied hard and found that he or she got good grades.

Calculating probabilities: a demo

- Take a look at Part 1 of today's Colab notebook which includes codes that calculate the probabilities
  - You don't have to understand the mathematics behind the codes if you are not interested

About n-gram LM

- Then we can talk about n-gram and statistical modeling!
- An n-gram is a sequence of n adjacent symbols in particular order
  - i.e. a sequence of n words (or tokens) that appear next to each other in a text
- For example:
  - A 1-gram (unigram) is a single word: "the", "cat", "sat"
  - A 2-gram (bigram) is a pair of consecutive words: "The cat", "cat sat"
  - A 3-gram (trigram) is a sequence of three consecutive words: "The cat sat"

Why do we need n-gram?

- Given previous tokens/words in a sequence/sentence, we want to predict the next word
- For instance, what is the next word if we have the sequence: *Natural language processing is a subfield of computer science and artificial*
  - Suppose we have a large English corpus a statistical LM can learn from
  - If we want to determine the next word of this sequence, we must search for the occurrence of this sequence + a possible next token, namely "*Natural language processing is a subfield of computer science and artificial* [`token`]"
  - That's a mouthful, isn't it? I really doubt an English corpus has exactly the same string that we look for
  - Then the model doesn't know what's the next token since this sequence is not in the corpus! What can we do?

Why do we need n-gram?

- For instance, what is the next word if we have the sequence: *Natural language processing is a subfield of computer science and artificial*
  - But if we don't have this super long string in the corpus, can we at least look for part of it?
  - Like, n tokens including the previous $n-1$ tokens from the sequence and a new [`token`]?
  - So let's use 4 previous tokens from the sequence!
    - i.e., *computer science and artificial* + [`token`]
    - This means $n-1=4$ and $n=5$. And it's a 5-gram model!
  - There should be a lot of mentions of "computer science and artificial intelligence" in our corpus! Then why don't take the token intelligence to end the sentence?
    - i.e. $P(\text{intelligence} \mid \text{computer science and artificial})$ is large

Why do we need n-gram?

- For instance, what is the next word if we have the sequence:
  Natural language processing is a subfield of computer science and artificial
  - Or, if we set $n = 2$ (bigram) and only rely on the previous single token, there should be a lot of mentions of "artificial intelligence" in our corpus as well!
    - i.e., The probability of having "intelligence" after "artificial" $P(\text{intelligence} \mid \text{artificial})$ is large
- Therefore, the next word/token predicted by the bigram LM (or the 5-gram LM) is "intelligence"
- And the sentence continues in this way:
  Natural language processing is a subfield of computer science and artificial intelligence

Why n-gram works?

- Why are we confident that considering only the previous $n-1$ tokens is safe?
- The Markov assumption states that the probability of a particular event depends only on a fixed, limited history of prior events.
  - I will skip the math. Shoot me an email if you want to know more about it
- The future state (e.g., next word/token) depends only on a limited number of recent states (e.g., previous word(s)/token(s))
- e.g. "...and the bus came"
  - Bigram: predicting "came" using the previous 1 word ("bus") only, i.e., $P(\text{came} \mid \text{bus})$
  - Trigram: predicting "came" using the previous 2 words ("the bus") only, i.e., $P(\text{came} \mid \text{the}, \text{bus})$

Why n-gram works?

- However, one limitation of using a small n value is that the model struggles to capture long-range context or dependencies.
  - Think about this: which token is a bigram model more likely to produce after "Taking two classes", "is" or "are"?
  - For a more extreme case:
    The book that the professor who won the prestigious award last year wrote during her sabbatical is on the table. What is on the table?

Have a try!

- Take a look at Part 2 of today's Colab notebook, and see how to train/learn an n-gram language model from a text!
  - The sample text we are using is the book *Alice in Wonderland* by Lewis Carroll
    - ...which means the language model we trained will closely reflect the style and characteristics of the book!
  - You can manipulate the size of n to adjust the context window your model takes into account!
  - Then, you can try inputting a string yourself and see what's the most possible next word as predicted by your model

Have a try!

- Congratulations! You have trained your first language model!
  - ...but this LM is quite small and incapable of handling complicated cases

Have a try!

- Then, take a look at Part 3 of today's Colab notebook, where we will ask GPT-2 to predict the next tokens!
  - Try to see how different the predicted next tokens are given the same text!

About perplexity

- Non-NLP sense of perplexity: the state of feeling confused and anxious because you do not understand something
- For NLP: a measure of uncertainty or "confusion" in predictions
  - A "perplexed" person is unsure what to say next
  - A language model with high perplexity struggles to predict the next word accurately
    - i.e., the higher a LM's perplexity is, the worse it performs
- Therefore, perplexity is a commonly used metric to evaluate the performance of a generative language model
- We will skip the math. Just remember that for perplexity, the lower, the better

Have a try!

- Take a look at Part 4 of today's Colab notebook, and see how one can calculate perplexity for a given model and text!
- We are using the n-gram model you have just trained and GPT-2 from Hugging Face
- Importantly, perplexity scores are model-dependent and should be interpreted within the context of each model

Miscellaneous

- **No class next week (Lunar New Year)**
- Please do not hesitate to ask questions
- We enjoy feedback from you, so please let us know if you feel there's anything we could have done better
- It would be great if you'd bring your laptop to the class every week