Natural Language Processing

Lecture V. N-gram Language Models and TF-IDF

Forrest Sheng Bao, Ph.D.

September 28, 2022

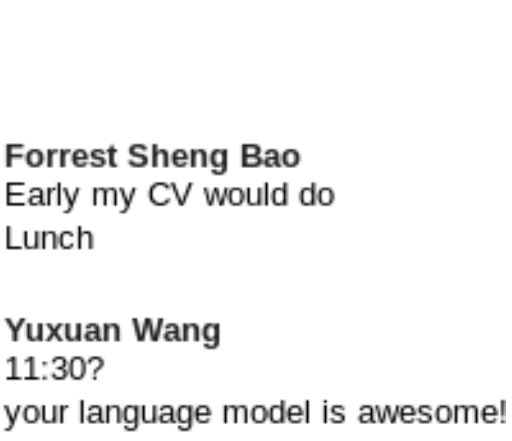
Outline

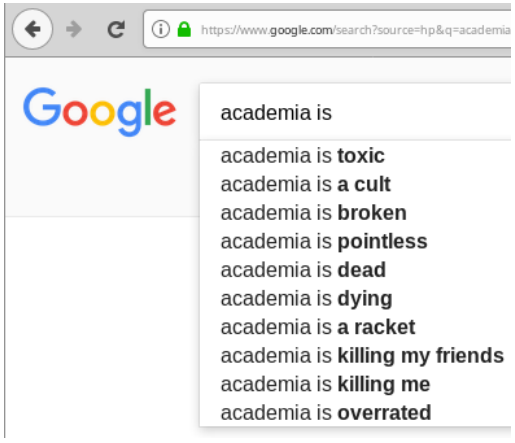
- Why language models
- Language models
- n-gram models
- Smoothing and Discounting
- Naïve Bayes spam filter
- TF-IDF

Why language models

For many applications, the output is a sequence of words.

- Example 1: Machine translation (MT): *Sprechen Sie MATLAB?*
  $\Rightarrow$ Do you speak MATLAB?
- Example 2: Automatic Speech Recognition (ASR):
  
  $\Rightarrow$ "$2M seed from Sequoia?"
- Example 3: Spell checking or spell suggestion





- The system needs to evaluate/rank a set of candidates, e.g., "$2M seed from Sequoia?" over "$2M seed from DFJ!"
- An easy way is to compute the likelihood that a candidate is a "making-sense" sentence.

Ranking candidate text strings

- For example, we can compute a probability for each string below:
- S1: “Please CALL me in 10 minutes.”
- S2: “Please ALL me in 10 minutes.”
- Which one is bigger? $P(S_1)$ or $P(S_2)$?

Language models

- A statistical language model is a probability distribution over sequences of words.
- Given a length l, a language model assigns a probability $P(w_1, \ldots, w_l)$ to the sequence.
- When $l = 1$, we have the probability for each individual word.
- When $l = 2$, we have the probability for each two-word pair.
- ...

Language models (cont.)

- In the example above, the system will compare the probabilities $P(w_1 = \text{“\$2M”}, w_2 = \text{“seed”}, w_3 = \text{“from”}, w_3 = \text{“Sequoia”})$ and $P(w_1 = \text{“\$2M”}, w_2 = \text{“seed”}, w_3 = \text{“from”}, w_4 = \text{“DFJ”})$.
- By chain rule, we have

$$
\begin{aligned}
& P(w_0 = \triangleright, w_1 = \text{“\$2M”}, w_2 = \text{“seed”}, w_3 = \text{“from”}, w_4 = \text{“Sequoia”}, \\
& w_5 = \triangleleft) \\
=& P(w_1 = \text{“\$2M”} | w_0 = \triangleright) \\
\times& P(w_2 = \text{“seed”} | w_1 = \text{“\$2M”}, w_0 = \triangleright) \\
\times& P(w_3 = \text{“from”} | w_2 = \text{“seed”}, w_1 = \text{“\$2M”}, w_0 = \triangleright) \\
\times& P(w_4 = \text{“Sequoia”} | w_3 = \text{“from”}, w_2 = \text{“seed”}, w_1 = \text{“\$2M”}, w_0 = \triangleright) \\
\times& P(w_5 = \triangleleft | w_4 = \text{“Sequoia”}, w_3 = \text{“from”}, w_2 = \text{“seed”}, w_1 = \text{“\$2M”}, \\
& w_0 = \triangleright)
\end{aligned}
$$

n-gram models

- Let's generalize: $P(w_1, \cdots, w_l) = \prod_{i=1}^{l} P(w_i | w_1, \ldots, w_{i-1})$
- Use a shorter history (n-th order Markov property):
  $\prod_{i=1}^{l} P(w_i | w_1, \ldots, w_{i-1}) \approx \prod_{i=1}^{l} P(w_i \mid w_{i-(n-1)}, \ldots, w_{i-1})$
- Conditional probability from counting:
  $P(w_i \mid w_{i-(n-1)}, \ldots, w_{i-1}) = \frac{\text{count}(w_{i-(n-1)}, \ldots, w_{i-1}, w_i)}{\text{count}(w_{i-(n-1)}, \ldots, w_{i-1})}$
- When $n = 1$, 2 or 3, it's unigram, bi-gram or tri-gram respectively.
- Yes, unigram means no history but just the word itself. No order between words is considered.
- N-gram is one (simple but widely used) way to represent language models.

Bag of words

The bag of words and the background words as seen in Gates building, CMU

Bag of words (cont.)

- A bag-of-words (BOW) model is basically a uni-gram model: an orderless representation of a document.
- A common use of a BOW model is create feature vectors using the frequencies of words, i.e., term frequency.
- Background words are those of extreme high frequency. They are earlier called stop words.
- When many researchers use the term “bigram” or “trigram”, they are actually talking about bag-of-double-words or bag-of-triple-words.
- See https://en.wikipedia.org/wiki/Bag-of-words_model.
- Also, how to build a spam filter using BOW: https://en.wikipedia.org/wiki/Naive_Bayes_spam_filtering

skip-grams and CBOW

- We may also add a gap between words.
- 1-degree Skip-grams from the sentence “I am a PhD student”: (“I”, “a”), (“am”, “PhD”), (“a”, “student”).
- A language model using skip-gram:

$$P(w_i|w_{i-k},\ldots,w_{i-1},\quad,w_{i+1},\ldots,w_{i+k})$$

- NOT THE CASE!
- Instead of estimating the probability giving the neighboring words, a skip-gram language model actually estimates the probabilities of neighboring words given a word:
-

$$\sum_{j\in[-k..-1]\cup[1..k]} \log P(w_{i+j}|w_i)$$

- The first equation is actually called Continues BOW (CBOW) model.

Order of n-grams and interpolation

- Larger n: the string is specific but sparse, e.g., “academia is vicious” may not be in training data.
- Smaller n: dense but general, e.g., lots of “academia is” or maybe “is vicious”
- To balance, mix different orders by interpolation:

$$\lambda_3 P(w_i|w_{i-1}, w_{i-2}) + \lambda_2 P(w_i|w_{i-1}) + \lambda_1 P(w_i)$$

- The challenge is how to choose weights. Many people use brutal force.
- Better methods to be presented later.

Smoothing and Discounting

- What about unknown words?
- The problem is known as OOV (out of vocabulary)
- Lazy way: ignore them – closed vocabulary.
- For uni-gram:

$$P_{\text{smoothed}}(w_i) = \lambda P_{\text{model}}(w_i) + (1 - \lambda)\frac{1}{N},$$

  where

  - N is the number of total vocabulary,
  - λ is the probability that a word is known to the model (it can be totally arbitrary, e.g., 0.95),
  - $P_{\text{model}}(w_i)$ is the probability of the word in a unigram model (thus, 0 if the word is not in the model).

- It means that all unknown words have the equal probability $\frac{1-\lambda}{N}$.

Smoothing for bi-grams

- Observation: some words have more words following it while others have fewer.
- Make the smoothing depend on the context, i.e., λ is not the same for all words.
- Also leverages the unigram model, i.e., interpolation.
- $P_{\text{smoothed}}(w_i|w_{i-1}) = \lambda_{w_{i-1}} P_{\text{model}}(w_i|w_{i-1}) + (1 - \lambda_{w_{i-1}}) P_{\text{model}}(w_i)$
- Witten-Bell smoothing:

$$\lambda_{w_{i-1}} = \frac{c(w_{i-1})}{u(w_{i-1}) + c(w_{i-1})}$$

where $u(w_{i-1})$ is the number of unique words after w_{i-1} and $c(w_{i-1})$ the total count of w_{i-1} in the training corpus.

Evaluating a language model

- Likelihood: Given a test set W_{test}, compute the score $\prod_{\mathbf{w} \in W_{test}} P(\mathbf{w}|M)$ where M is the model and $\mathbf{w}$ is a sequence of words.
- Example: Let $W_{test} =$ {"I am a Linuxer", "I am an entreprenuer", "I am a Pythonista"}. The score is P("I am a Linuxer") $\times$ P("I am an entreprenuer") $\times$ P("I am a Pythonista")
- But, what if the model covers some sentences very well but not others?
- Entropy
- And, coverage.

Other language models

Other ways to estimate $P(w_1, \ldots, w_l)$ beyond n-gram:

- Exponential language models: maximum entropy language models, log-bilinear language models.
- neural language models: based on neural networks, embedding

TF-IDF and document retrieval I

- Let's put the BOW model into application.
- An example is document retrival, or more general, **information retrieval (IR)**.
- The goal of document retrival is to return the most relavant, thus expected, documents to users based on their queries.
- Google is an example of IR.
- For example, if a user searches for "matrix" among the user manuals of Pytorch, Numpy, SpaCy, then the manual of Numpy should (probably) be ranked the highest, then PyTorch in the middle, and finally Spacy.
- If the query is "language model", the order might be flipped.
- How do we do it?

TF-IDF and document retrieval II

- Intuition 1: “matrix” occurs the most frequent in the manual of Numpy, and then PyTorch, finally Spacy.
- This is the idea of **term frequency** or TF:

$$\frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

  where d is a document, t is a specific term, t' is any term in the document d, and f is the frequency of a term in a document.
- However, Intuition I does not consider the occurence of the query term is other documents.

TF-IDF and document retrieval III

- Intuition 2: If a term occurs in many documents, then this term is not specific to a particular document. Hence, its importance should be less.
- This is idea of **inverse document frequency**:

$$\mathrm{idf}(t, D) = \frac{|D|}{|d \in D : t \in d|}$$

  where D is the set of all documents under consideration, e.g., the user manuals of many packages.

- Put them together: TF-IDF

$$\mathrm{tfidf}(t, d, D) = \mathrm{tf}(t, d) \cdot \mathrm{idf}(t, D)$$

- In other words, IDF is a weighing factor for TF.

Naïve Bayes spam filter

- Another classical application of the BOW model
- Two classes: Spam (S) vs ham (H, non-spam)
- Based on Bayes theorem, given one word w, the chances that a message is spam: $P(S|w) = \frac{P(S,w)}{P(w)} = \frac{P(w|S)P(S)}{P(w|H)P(H)+P(w|S)P(S)}$
- Usually we further assume that $P(H) = P(S)$. Thus $P(S|w) = \frac{P(w|S)}{P(w|H)+P(w|S)}$. Takeaway: all you need is $P(w|H)$ and $P(w|S)$ which can be obtained via counting.
- Naïve Bayes spam filters assume that words appear in an email indepdently. So for two independent words w_1 and w_2 (not necessarily sequential), we have: $P(S|w_1, w_2) = \frac{P(w_1|S)P(w_2|S)}{P(w_1|S)P(w_2|S)+(1-P(w_1|S))(1-P(w_2|S))}$
- Detailed derivation at http://www.paulgraham.com/naivebayes.html and https://www.mathpages.com/home/kmath267/kmath267.htm

How to evaluate a model

- How do you know the performance of a spam filter or a document retriever?
- They are two kinds of tasks: classification and ranking.
- Their judgements are different

Judging a classification model

- Classification: map samples to different classes (discrete).
- The spam filtering example falls under a special, but common classification problem category: **binary classfication**.
- We have two ground truth labels (usually denoted as y): true or false.
- We also have two prediction labels (usually denoted as $\hat{y}$): true or false.
- But some prediction true are not really true while some prediction false are not really false.
- So we have:
  - True positive (indeed a spam)
  - True negative (indeed a ham)
  - False positive (a ham, but predicted as spam)
  - False negative (a spam, but predicted as ham)
- Thus the confusion matrix link

Accuracy is not enough

- If we have 90 spams and 10 hams. If the model predicts all spams correctly but all hams wrongly, the accuracy is 90%.
- If the model truely good?
- Accuracy makes less sense when the dataset is **unbalanced** meaning that samples of different labels have unequal amounts.

Sensitivity, Recall, Specificity, Precision

- A fairer approach is to look at the performance on each **class**.
- Again, this link for their definitions.
- In documental retrieval, precision and recall are more frequently used. See link.

Overfitting and generalization

- The naive Bayes spam filter is trained: it counts words over a corpus
- In the spam filter case, what if the spam sender changes the strategy to draft a spam using different words, words frequencies, or styles?
- It may fail as the frequencies of words in a new spam differ greatly from those in the training corpus.
- Training vs test: getting the parameters of a model vs. checking the model's performance
- If training and test data/set are the same, it makes less sense.
- If a model only works well on the training set, we say it **overfits**.
- If a model works well on the test set as well, we say it **generalizes**.

Juding a ranking model

- Precision@N.