COMP 5790/ 6790/ 6796

Special Topics: Information Retrieval

Instructor: Shubhra ("Santu") Karmaker

Assignment #2: Text Representation + Vector Space Retrieval Model. [100 points]

Notice: This assignment is due **Friday, February 7, 2020 at 11:59pm**.

Please submit your solutions via Canvas (https://auburn.instructure.com/). You should submit your assignment as a **typeset PDF**. Please do not include scanned or photographed equations as they are difficult for us to grade.

This assignment tests your understanding about N-Gram Language Models. In particular, you will work on specific problems related to the estimation of N-Gram Language Model parameters, the issues involved in the estimation process and ways to overcome those issues. You will also work on problems related to Vector Space Model and implement TF-IDF heuristics from scratch.

1. N-Gram Language Model [15 pts]

a. **[5 pts]** What is the central assumption regarding word dependencies that is made in N-Gram Language Models?
b. **[5 pts]** Do you think the assumption made in N-Gram Language Models is reasonable? Why?
c. **[5 pts]** What is the primary benefit of applying the assumption made in N-Gram Language Models?

2. Unigram Language Model [15 pts]

Unigram Language Model is a special class of N-Gram Language Model where the next word in the document is assumed to be independent of the previous words generated by the model. Mathematically, this is written as, $P(w_m|w_{m-1}, ..., w_1) = P(w_m)$.

a. **[10 pts]** We can estimate the parameters of a Unigram Language Model through Maximum Likelihood Estimation. Given a particular document d and the vocabulary set V, the maximum likelihood estimator for a Unigram Language Model is given by the following formula:

$$P_{ML}(w|d) = \frac{c(w,d)}{\sum_{w' \in V} c(w',d)}$$

Now, let us assume that, we have seen the following document d: **"the sun rises in the east and sets in the west"**. Assuming that this document was generated by a Unigram Language Model and words in the document d constitute the entire vocabulary, how many parameters are necessary to specify the Unigram Language Model? Estimate the values of all these parameters using the maximum likelihood estimator.

b. **[5 pts]** Now assume that, the entire vocabulary V consists of the set:

$$\{a, the, from, retrieval, sun, rises, in, BM25, east, sets, and, west\}$$

If we consider the same document dd: **"the sun rises in the east and sets in the west"** and assume again that this document was generated by a Unigram Language Model, how many parameters are necessary to specify the Unigram Language Model in this case? Estimate the values of all these parameters using the maximum likelihood estimator.

3. Bigram Language Model [15 pts]

Bigram Language Model is another special class of N-Gram Language Model where the next word in the document depends only on the immediate preceding word. Mathematically, this is written as the conditional probability, $P(w_m|w_{m-1},...,w_1)=P(w_m|w_{m-1})$.

a. **[8 pts]** Given the same document d from question 2(a) and same vocabulary set V from question 2(b) and assuming the document d is now generated by a Bigram Language Model, how many parameters are necessary to specify the Model? Using the maximum likelihood estimator, estimate the values of the following parameters (assume # to be the start of the sentence marker):

i. $P(sun|the)$
ii. $P(the|in)$
iii. $P(from|the)$
iv. $P(BM25|retrieval)$
v. $P(east|west)$
vi. $P(east|rises)$
vii. $P(sets|\#)$
viii. $P(the|\#)$

b. **[7 pts]** Please provide answers to the following questions:

i. Do you see any general problem associated with the estimation of the parameters of the Bigram Language Model from problem 3(a)?
ii. Do you see the same problem in the estimation process for question 2(b)?
iii. For which model, the problem is more severe? Can you derive some general conclusion based on this comparison?
iv. What can we do to solve this general problem?

3. Vector Space Model [25 pts]

a. **[5 pts]** Here's a query and document vector. What is the score for the given document using dot product similarity?

$$d = \{1, 0, 0, 0, 1, 4\} \qquad q = \{2, 1, 0, 1, 1, 1\}$$

b. **[10 pts]** Let d be a document in a corpus. Suppose we add another copy of d to collection. How does this affect the IDF of all words in the corpus?

c. **[5 pts]** Consider Euclidean distance as our similarity measure for text documents:

$$d(q,d) = \sqrt{\sum_{i=1}^{|V|}(q_i - d_i)^2}.$$

What does this measure capture compared to the cosine measure discussed in this chapter? Would you prefer one over the other? Why?

d. **[5 pts]** In Okapi BM25, how can we remove document length normalization by setting a parameter? What value should it have?

4. TF-IDF Heuristic [30 pts]

In this assignment you will implement the TF-IDF formula and use it to study the topics in State of the Union speeches given every year by the U.S. president.

a. Download the source data file state-of-the-union.csv. This is a standard CSV file with one speech per row. There are two columns: the year of the speech, and the text of the speech. You will write a Python program that reads this file and turns it into TF-IDF document vectors, then prints out some information. Here is how to read a CSV in Python.
b. Tokenize the text corresponding to each speech, to turn it into a list of words. Using the following simple tokenization scheme:
   - convert all characters to lowercase
   - remove all punctuation characters
   - split the string on spaces
c. Compute a TF (term frequency) vector for each document. This is simply how many times each word appears in that document. You should end up with a Python dictionary from terms (strings) to term counts (numbers) for each document.
d. Count how many documents each word appears in. This can be done after computing the TF vector for each document, by incrementing the document count of each word that appears in the TF vector. After reading all documents you should now have a dictionary from each term to the number of documents that term appears in.
e. Turn the final document counts into IDF (inverse document frequency) weights by applying the formula IDF(term) = log(total number of documents / number of documents that term appears in.)
f. Now multiply the TF vectors for each document by the IDF weights for each term, to produce TF-IDF vectors for each document.
g. Then normalize each vector, so the sum of squared weights is 1. You can do this by dividing each component of the document vector by the length of the vector.
h. Congratulations! You have a set of TF-IDF vectors for this corpus. Now it's time to see what they say. Randomly select a speech from the corpus, and print out the highest weighted 20 terms, along with their weights. What do you think this particular speech is about? Write your answer in at most 200 words.
i. Your task now is to see if you can understand how the topics changed since 1900. For each decade since 1900, do the following:
   - sum all of the TF-IDF vectors for all speeches in that decade
   - print out the top 20 terms in the summed vector, and their weights

Now take a look at the terms for each decade. What patterns do you see? Can you connect the terms to major historical events? (wars, the great depression, assassinations, the civil rights movement, Watergate…) Write up what you see in narrative form, no more than 500 words, referring to the terms for each decade.

10. To finish the assignment, upload your code to canvas.