IR Assignments #3 and #4

17 November 2019 22:45

Assignment #3:
Random Indexing

Objective is to find words with similar semantics. We have to use Random Indexing for this.

1. Take a large set of documents.
2. Scan through the set and pre-process the documents (remove special characters, split properly, bring in standard format etc.)
3. Scan through the document and make an index vector for each term.
   a. Index vector is a vector of a constant size that contains all zeroes and about 5-10% entries are randomly turned to 1 or -1
   b. It is necessary to create a unique index vector for each term. For this we can use seeds of random numbers. If a unique seed is given to the random number generator, then it is guaranteed that the generated sequence of numbers will also be unique.
      In python seed can be given by random.seed(number). So we need to generate a unique number for every term. This can be accomplished by taking ascii values of each character. Example: hello = [1*(72), 2*69, 3*76, 4*76, 5*79] = [0072,0138, 0228, 0304, 0395]. Thus hello can be converted to 00720138022803040395. This will be the seed. (Credits to jitesh for this method)
   c. The size of the index vector needs to be sufficiently large. Sir told to use 1024 in class.
   d. Of course, step 2 and step 3 can be done at the same time.

4. Now we need to choose a window size (K). This will define the number of nearby words that we give weightage. If K is 2, we take 2 nearby words (2 from left, and 2 from right) for every term.
   a. For this assignment, K can be taken between 2 and 4
5. Make Context Vectors for each Term.
   a. Initially take context vectors same as the index vectors
   b. Then for each term, accumulate the IVs of nearby terms into CV of this term.
      i. Example: Sentence is:

         The quick brown fox jumped over the lazy dog.

         $CV_{fox} = CV_{fox} + 0.25*(IV_{quick}) + 0.5*(IV_{brown}) + 0.5*(IV_{jumped}) + 0.25*(IV_{over})$

         The terms that are more near are given more weightage. In this case **K=2**

      This needs to be done for each document in the text. Make sure to slit properly on sentences so that Words of another sentence is not added.

         Ex: *Hello World. Happy to help.*
         CV of "World" should include only "Hello". CV of "happy" should accumulate only "to". World should not include IV of Happy as they are in separate sentences.

6. Now find similarities between each pair of CVs of terms. (Find Cosine)
7. For each term, print top X terms who have CV most similar to that term.

Points to note:

1. This is a highly computationally intensive assignment.

   Especially step 6 is highly intensive as complexity runs into $O(T^2N)$ where T is number of terms and N is size of the CV. N will be 1024 and T can be anything between 5k to 100k depending on dataset.
2. Thus it is better to use GPUs for computation and use libraries like NumPy which are much faster.
3. Taking small dataset may result in weird results.
4. If anyone finds any other useful article/video on Random Indexing, please share it as it will help give better clarity.

Assignment #4:
Near duplicate detection

Objective is to find near duplicate documents from a given set.
Near duplicates are the documents that many not necessarily contain the exact same text but have similar content/ meaning.
Best example is 2 news articles covering the same story.

1. We have to take a large collection of news documents.
    1. Probably sir will give the dataset.
2. Use boilerplate to convert information to pure text files (Optional step. Depends on dataset)
3. Apply Near duplicate detection algorithm to find near duplicates.

Points to note:

1. There are multiple algorithms available for doing this.
2. Finding suitable method is part of the assignment.
3. This could potentially take a lot of time to do.