Lecture 10: Precision + Recall / kNN

Created	@July 29, 2024 5:47 PM

[paper notes]

In KNN, we return the k nearest documents in the corpus that have the smallest distance (closest) to the document x_i

We need to figure out how to find the distance between 2 documents as well as how to represent the documents

Document representation

Represent the document as a vector

- Bag-of-words representation
    - you have a dictionary, where the keys are the words, and the values are the counts of the occurrences of that word
    - pros: easy to explain and easy to computer
    - cons: it can count unnecessary/meaningless words such as the, and, and, etc
    - this means that looking at the uncommon words is more useful/interesting since they are usually what uniquely define a text
- TF-IDF
    - our goal is to emphasize the important words

  - the important words are those that are more unique to one document compared to another
  - Term Frequency (TF)
    - word counts dictionary
  - Inverse Document Frequency
    - log(#docs/(1+#docs that have that word))
    - this is the inverse document frequency
    - For every word, do TF * IDF
    - more unique words will have a larger TF-IDF value

Ways of measuring distance

- Euclidean distance (L2 Norm): distance formula (sum of the squares of the distances)
  - smooth voronoi diagram boundaries
- Manhattan distance (L1 norm): sum of the absolute values of the distances
  - jagged voronoi diagram boundaries
- Weighted distances
  - put more weight on the dimension that doesn't change as much as the other ones9for example weight kilometers more over seconds)
  - less weight to the data that is more spread out
  - weighted euclidean distance formula
  -

$$distance(x_i, x_q) = \sqrt{\sum_{j=1}^{D} a_j^2 (x_i[j] - x_q[j])^2}$$

ou can just do F of G because, uh, you are you are just taking the

For feature j: $\frac{1}{\max_i(x_i[j]) - \min_i(x_i[j])} = a_j$

- Similarity
  - NOT a measure of distance
  - higher similarity number is better
  - Similarity metrics:
    - Natural similarity
      - dot product of 2 vectors (multiplying each side by side and then summing it up)
      - helps us see how similar the magnitudes are
    - Cosine similarity
      - take the cosine of the angle between the 2 vectors
      - dot product of the 2 vectors / euclidean distance magnitude
      - this helps us see how similar the directions of the 2 vectors are
      - very efficient for sparse vectors since the dot product is more efficient (mostly zeros)
      - distance = 1 - similarity
    - Jaccard similarity
      - compares the overlap of words between the 2 texts

-

$$J(Doc_i, Doc_j) = \frac{|Doc_i \cap Doc_j|}{|Doc_i \cup Doc_j|}$$

- Normalization of embeddings
  - normalization isn't good for when you have 2 documents of differing sizes
  - maximum cap normalization: cap the maximum word count
- Weighted KNN
  - put more emphasis on closer points than farther points
- Kernel regression
  - use a kernel to weight all training points
- Efficient Nearest Neighbors algorithm
  - no training required
  - however it has an O(n) runtime
  - you can approximate the nearest neighbor instead of actually finding the real one which will improve efficiency

Locality Sensitive Hashing (LSH)