Information Retrieval SS 2023

Exercise 2: The Vector Space Model, Probabilistic Retrieval[1]

Prof. Goran Glavaš, Benedikt Ebing, Fabian David Schmidt
Chair For Natural Language Processing

Vector space model - TF-IDF

1. Consider the following tables of term frequencies and document frequencies (df), and document collection sizes (N). Calculate for each term-document-pair the tf-idf weight

term	Doc1	Doc2	Doc3	df	N
car	27	4	24	18,165	94,584
auto	3	33	0	6,723	53,814
insurance	0	33	29	19,241	97,226
best	14	0	17	25,235	113,095

Table 1: Corpus statistics

with $idf = ln(N/df)$, use raw term frequencies.

2. In the previous subtask we used the natural logarithm. How does the base of the logarithm affect the tf-idf scores? How does the base of the logarithm affect the relative scores of two documents on a given query?

3. Recall the tf-idf weights computed in the first task. Compute the Euclidean normalized document vectors for each of the documents, where each vector has four components, one for each of the four terms.

4. Using the document vectors you just constructed, rank the documents for the query *car insurance* according to their cosine distances to the query. Represent the query as a binary vector.

5. Compute the vector space similarity between the query "digital cameras" and the document "digital cameras and video cameras" by filling out the empty columns in the table below. Assume N=10,000,000, apply the term frequency scaling, as shown in the lecture, for query and document (wf columns). Do not account for query- or document-length. Apply idf weighting for the query and cosine normalization for the document. Treat *and* as a stop word. Enter term counts in the tf columns. What is the final similarity score?

	query					document			
word	tf	wf	df	idf	q_i=wf-idf	tf	wf	d_i=normalized wf	$q_i \cdot d_i$
digital			10,000						
video			100,000						
cameras			50,000						

6. What is the idf of a term that occurs in every document? Compare this with the use of stop word lists.

[1]Exercise tasks partially based on "An Introduction to Information Retrieval" by Manning, Raghavan and Schütze

7. Why is the idf of a term always finite?

8. What is the minimum and maximum value of the idf that a term can have (assuming a document frequency larger than zero)? In which cases does this happen?

Optimizing Vector Space Retrieval

1. Imagine your are running a rudimentary retrieval engine on a small embedded chip which can barely perform addition and multiplication. To that end, you want to minimize the number of cosine similarity computations but also the number of multiplications and additions in each individual cosine computation. You are given the following toy collection consisting of N=9 documents, with the following TF-IDF vectors (of length k=10):

   d_1 = [0.17, 0.21, 0.35, 0.44, 0.49, 0.39, 0.09, 0.07, 0.37, 0.24]
   d_2 = [0.49, 0.48, 0.44, 0.09, 0.24, 0.2, 0.41, 0.16, 0.1, 0.15]
   d_3 = [0.41, 0.36, 0.27, 0.19, 0.15, 0.42, 0.23, 0.42, 0.02, 0.42]
   d_4 = [0.31, 0.41, 0.21, 0.19, 0.47, 0.28, 0.21, 0.39, 0.16, 0.38]
   d_5 = [0.46, 0.12, 0.21, 0.25, 0.38, 0.38, 0.46, 0.23, 0.31, 0.14]
   d_6 = [0.13, 0.33, 0.28, 0.42, 0.07, 0.13, 0.58, 0.15, 0.0, 0.49]
   d_7 = [0.21, 0.09, 0.07, 0.09, 0.3, 0.54, 0.24, 0.43, 0.51, 0.21]
   d_8 = [0.18, 0.39, 0.42, 0.05, 0.41, 0.1, 0.52, 0.12, 0.14, 0.38]
   d_9 = [0.4, 0.51, 0.01, 0.1, 0.12, 0.22, 0.26, 0.34, 0.42, 0.38]

   In order to reduce the cost of computation of individual documents, we need to cut the length of document vectors in half. To achieve this, we perform random projections using the following five random vectors (from the same vector space as the original documents):

   r_1 = [0.33, 0.33, 0.42, 0.12, 0.2, 0.34, 0.58, 0.19, 0.07, 0.24]
   r_2 = [0.29, 0.16, 0.38, 0.48, 0.43, 0.11, 0.12, 0.33, 0.03, 0.44]
   r_3 = [0.01, 0.17, 0.11, 0.27, 0.23, 0.37, 0.35, 0.48, 0.54, 0.24]
   r_4 = [0.09, 0.05, 0.39, 0.25, 0.45, 0.48, 0.04, 0.45, 0.35, 0.12]
   r_5 = [0.13, 0.17, 0.4, 0.4, 0.07, 0.4, 0.35, 0.39, 0.44, 0.06]

   For each initial document, we compute the following hashing functions (based on the dot product with random vectors):

$$h(d_i, r_j) = \begin{cases} 1, & \text{if } d_i \cdot r_j > 0.75 \\ 0, & \text{otherwise} \end{cases}$$

   (a) Compute the hashed (i.e., projected, shortened) document vectors.

   (b) We also need to reduce the total number of cosine comparisons, so we will perform pre-clustering of documents. In the reduced projected vector space, obtained using the random projections from the previous task, you are given the following $\sqrt{N} = 3$ leader vectors that will determine the document clusters:

   l_1 = [0, 0, 1, 1, 0]
   l_2 = [0, 1, 1, 1, 0]

l_3 = [0, 0, 0, 0, 0]

Compute the clusters by assigning each document vector (i.e., it's random projection) to the closest (i.e. most similar) leader vector.

2. You are given the following query vector (in the original vector space):

$$\text{q} = [0.15, 0.39, 0.36, 0.25, 0.36, 0.15, 0.52, 0.37, 0.08, 0.27].$$

Retrieve the top $M = 5$ documents using the random projection vectors of documents from 1a and clusters obtained in 1b. What is the total number of cosine similarities you compute and the total number of element-wise multiplication operations in all dot-products? Compare that with the numbers you would get if we didn't perform any pre-clustering nor random projections.

3. In the previous two tasks we used cluster pruning to avoid computing the distance from the query vector to every document vector. Sketch down an example so that with two leaders, the answer returned by cluster pruning is incorrect (it is not the data point closest to the query vector).

Probabilistic Information Retrieval

1. Consider deriving a language model from the following piece of text:

```
the martian has landed on the latin pop sensation ricky martin
```

   (a) Compute the probabilities $P(the)$ and $P(matrian)$ in the an MLE-eistmated unigram probability model.

   (b) Compute the probabilities $P(pop|the)$ and $P(sensation|pop)$ under an MLE-estimated bigram probability model.

2. Suppose we have a collection D that consists of the four documents given in the following table. Build a query likelihood language model for this document collection. Use

docID	text
1	click go the shears boys click click click
2	click click
3	metal here
4	metal shears click here

Table 2: Document colletion (left) and results table (right).

   the Jelinek-Mercer smoothing model with the interpolation of $\lambda = 0.5$. Use maximum likelihood estimation (MLE) to estimate unigram probabilities. Work out the model probabilities of the queries `click`, `shears`, and `click shears` for each document, and use those probabilities to rank the documents returned by each of the queries.

3. Re-compute the rankings while using the (i) Binary Independence Model, (ii) the Two-Poisson model with $k = 1$ and (iii) the BM25 model with $b = 0.75$.

4. Does the Binary Independence Model make the *document independence assumption*? Does it make the *term independence assumption*? Explain your answer.

5. Why is the (conditional) independence assumption in language models an incorrect assumption that doesn't hold when dealing with natural language? Give an example.

6. What are the differences between standard vector space tf-idf weighting and the BIM probabilistic retrieval model (in the case where no document relevance information is available)?