Information Retrieval SS 2023

Exercise 3: Revelance Feedback, Semantic Retrieval, Machine Learning and IR, Evaluation[1]

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

Relevance Feedback

1. Suppose that a user's initial query is **cheap CDs cheap DVDs extremely cheap CDs**. The user examines two documents, d_1 and d_2. She judges d_1, with the content **CDs cheap software cheap CDs** relevant and d_2 with the content **cheap thrills DVDs** nonrelevant. Assume that we are using direct term frequency (with no scaling and no document frequency). There is no need to length-normalize vectors. Using Rocchio relevance feedback what would the revised query vector be after relevance feedback? Assume $\alpha = 1, \beta = 0.75, \gamma = 0.25$.

2. In Rocchio's algorithm, what weight setting for $\alpha/\beta/\gamma$ does a "Find pages like this one" search correspond to?

3. Omar has implemented a relevance feedback web search system, where he is going to do relevance feedback based only on words in the title text returned for a page (for efficiency). The user is going to rank 3 results. The first user, Jinxing, queries for:

   banana slug

   and the top three titles returned are:

   banana slug Ariolimax columbianus
   Santa Cruz mountains banana slug
   Santa Cruz Campus Mascot

   Jinxing judes the first two documents Relevant, and the third Not Relevant. Assume that Omar's search engine uses term frequency but no length normalization nor IDF. Assume that he is using the Rocchio relevance feedback mechanism, with $\alpha = \beta = \gamma = 1$. Show the final revised query that would be run. (Please list the vector elements in alphabetical order.)

Semantic Retrieval - Latent Semantic Analysis

Consider the following collection of documents:

- **Document 1:** *Frodo and Sam were trembling in the darkness, surrounded in darkness by hundreds of blood-thirsty orcs. Sam was certain these beasts were about to taste the scent of their flesh.*

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

- **Document 2:** *The faceless black beast then stabbed Frodo. He felt like every nerve in his body was hurting. Suddenly, he thought of Sam and his calming smile. Frodo had betrayed him.*
- **Document 3:** *Frodo's sword was radiating blue, stronger and stronger every second. Orcs were getting closer. And these weren't just regular orcs either, Uruk-Hai were among them. Frodo had killed regular orcs before, but he had never stabbed an Uruk-Hai, not with the blue stick.*
- **Document 4:** *Sam was carrying a small lamp, shedding some blue light. He was afraid that orcs might spot him, but it was the only way to avoid deadly pitfalls of Mordor.*

1. Your vocabulary consists of the following terms: Frodo, Sam, beast, orc, and blue. Compute the TF-IDF document-term occurrence matrix for given document collection and vocabulary terms.
2. Perform the singular value decomposition of the above matrix and write down the obtained factor matrices U, Σ, and V. You can use some existing programming library to perform the SVD (e.g., `numpy.linalg.svd` in Python).
3. Reduce the rank of the factor matrices to K = 2, i.e., compute the 2-dimensional vectors for vocabulary terms and documents. Show terms and documents as points in a 2-dimensional graph.
4. You are given the query "Sam blue orc". Compute the latent vector for the query and rank the documents according to similarity of their latent vectors with the obtained latent vector of the query.

Semantic Retrieval - Representation Learning

For your semantic retrieval system you are training a CBOW model (windows size=2). Your vocabulary consists of the following terms:

[*"Frodo", "followed", "Sam", "into", "the", "dark", "Mordor", "Ring"*]

You are currently processing the sentence "*Frodo followed Sam into the dark*" and your word vectors are as follows:

$$W = \begin{bmatrix} -0.01 & -0.88 & -1.74 & -0.4 \\ 0.69 & -0.46 & 0.23 & -1.34 \\ -1.34 & -0.54 & 0.29 & 0.01 \\ 0.17 & -0.36 & -0.06 & 0.93 \\ -0.6 & -0.65 & -0.52 & 1.7 \\ 0.17 & -0.61 & -0.54 & -1.35 \\ -1.32 & -0.89 & -2.4 & 0.09 \\ -2.19 & 1.1 & 0.58 & 0.63 \end{bmatrix}, \quad W' = \begin{bmatrix} 0.64 & 1.4 & -0.83 & -0.23 \\ 0.01 & -1.75 & -0.6 & 1.19 \\ -0.83 & -1.58 & 0.67 & 0.37 \\ 0.11 & -0.39 & -0.71 & -0.49 \\ -0.04 & 0.11 & 0.22 & 0.02 \\ -0.64 & -1.47 & 0.45 & -0.96 \\ -1.49 & -1.78 & -0.52 & -2.7 \\ -0.68 & -0.55 & -0.91 & 0.59 \end{bmatrix}$$

1. Which (positive) training examples are derived from the sentence?
2. Which (positive) training examples are derived if we would consider the Skip-gram model?

3. Calculate the output of the last layer (softmax layer) for the current sentence in which *Sam* is the center word.

4. What is the final document embedding if we represent it as the average of its constituent word embeddings?

5. Name one shortcoming of representing documents and queries as average word embeddings and how to overcome it?

6. Use your computed document embedding as a query vector and rank the four document documents from the previous task by their cosine similarities, use the following document embeddings:

   - **Document 1**: $\begin{bmatrix}1.17 & 0.05 & -1.69 & 0.15 & 1.87 & -0.25 & -0.92 & 0.84\end{bmatrix}$
   - **Document 2**: $\begin{bmatrix}-0.88 & -0.65 & -0.51 & -1.08 & -0.25 & 1.01 & 0.54 & -0.7\end{bmatrix}$
   - **Document 3**: $\begin{bmatrix}2.93 & -2.28 & 0.01 & 1.65 & 1.15 & 1.24 & 0.26 & 0.52\end{bmatrix}$
   - **Document 4**: $\begin{bmatrix}1.22 & -1.04 & 0.11 & 0.97 & 0.74 & 0.08 & -1.18 & -0.11\end{bmatrix}$

7. After training your embedding model you obtain word representations for every word you observed (assuming you derived your vocabulary form the corpus). Why shouldn't we use every available word embedding after training?