Advanced Methods in Text Mining
Summer Semester 2024
Assignment 2 – LSI, GloVe, and RNNs

Instructor: Prof. Rafet Sifa, Dr. Lorenz Sparrenberg,
Armin Berger, Maren Pielka, Tobias Deußer
University of Bonn

Deadline: 19. 6., 11:59 pm
Send your solution (as .zip) to: amllab@bit.uni-bonn.de
(the title of the e-mail should be "<group_name>: Assignment 2" and all group members should be mentioned in the e-mail)

Introduction

In this assignment, we will explore some basic concepts of Natural Language Processing (NLP). This assignment will have 6 main parts totalling 100 points (pts.). The coding exercises will be implemented using the programming language Python.

1. Orthonormal Vectors (3 Pts.)

Assume you have two orthonormal vectors, $\mathbf{a}$ and $\mathbf{b}$. Calculate c and d, which are defined as:

1. $c = ||\mathbf{a}|| + ||\mathbf{b}||$,
2. $d = \text{cosine similarity}(\mathbf{a}, \mathbf{b})$.

Explain your solution!

2. Latent Semantic Indexing

2.1 Theory (4 Pts.)

Explain how we can use matrix factorization and singular value decomposition to analyze a matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$, holding the *bag-of-words* representation of m documents and n unique words.

2.2 Implementation (5 Pts.)

Download the IMDb Movie Reviews dataset[1], introduced by Maas et al. 2011, create *bag-of-words* representations for the *train* set, and apply truncated singular value decomposition with $k = 2$ to your thus created matrix. Take 10 randomly selected observations from the test set, generate the document embeddings for these and plot them on a graph with colour coding for their respective review. Explain and interpret your results.

[1]official homepage: `https://ai.stanford.edu/~amaas/data/sentiment/`; Hugging Face: `https://huggingface.co/datasets/imdb`

3. Backpropagation (10 Pts.)

Derive the following equations as seen during the Exercise session on Backpropagation:

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l \tag{BP3}$$

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1}\delta_j^l \tag{BP4}$$

4. GloVe

4.1 Basic Idea (2 Pts.)

Explain the basic idea behind GloVe, introduced by Pennington et al. 2014. What problems of Latent Semantic Indexing are addressed by it?

4.2 Theory (10 Pts.)

Derive the function $g(\mathbf{w}_i, \mathbf{w}_j, \mathbf{v}_l)$ that models the relationships between two words i and j against a context word l in the GloVe model.

4.3 Applied Theory (10 Pts.)

GloVe relies on a word co-occurrence matrix, which is built using a context window around each word. Examine the impact of the context window size on the quality of the word embeddings as discussed in Pennington et al. 2014.

1. What is the context window in the GloVe model, and how does its size affect the word embeddings?
2. Compare the advantages and disadvantages of using short (e.g., 2-5 words) versus long (e.g., 10-20 words) context windows in building the co-occurrence matrix.
3. Summarize any empirical findings or theoretical insights that support the choice of context window size in the GloVe model, based on the literature or studies discussed in the lecture.
4. In what types of text (e.g., technical documents, literature, social media) would different context window sizes be more appropriate, and why?

4.4 Application (20 Pts.)

Write a SMS spam classification model with the SMS spam dataset[2] introduced by Almeida et al. 2011. The pipeline should be:

1. Reading in the raw data.
2. Randomly splitting the dataset into training (80% of the data) and test (20% of the data).
3. Applying preprocessing.
4. Encoding your textual data using a **GloVe** encoder. To that end use both of the following methods we discussed during the lecture and compare the results:
   (a) *Mean-pooling*: Represents each sentence by the mean of the word embeddings.
   (b) *Max-pooling*: Represents each sentence by the maximum of the word embeddings.

[2] `https://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection`

5. Training a **multilayer perceptron**[3] model on predicting whether the input is `spam` or `ham` using only your training set.
6. Predicting on your hold-out test set and reporting your achieved accuracy.
7. Compare your results to your results from the previous assignment (a tf-idf encoder with either a logistic regression or a k-nearest neighbour search with at least $k = 3$ neighbours).

5. Recurrent Neural Networks

5.1 Concepts (5 Pts.)

Compare a *vanilla* (a.k.a. simple) recurrent neural network (RNN) to a multilayer perceptron (MLP). What can an RNN model that an MLP is unable to? How does it achieve this feat?

5.2 Extensions (6 Pts.)

Discuss the two famous extensions to the *vanilla* RNNs:

1. Long short-term memory (LSTM), introduced by Hochreiter et al. 1997, and
2. Gated recurrent unit (GRU), introduced by Cho et al. 2014.

What are the main differences between LSTMs and GRUs?
What problem do they attempt to solve and how are they accomplishing this?

6. LLMs + Movie Recommendations

Build a system that recommends movies to a user considering a short input question/prompt about the content of the movie. Exemplary inputs could be:

- What movie is set in ancient Rome (similar to Gladiator)?
- Please recommend me some movies that are both romantic and funny.
- Which movies are about astronauts and starships?

Please provide your code, as well as some exemplary output. For each input statement, return the titles of the 5 most relevant movies, using your approach. You should test the input examples stated above, as well as 5 additional user inputs you come up with yourself, for each sub-task individually.

6.1 Embeddings and Retrieval (12 Pts.)

Based on a data set of movie synopses[4], solve the problem using an embedding model of your choice (e.g. OpenAI Ada embeddings, GloVe, SentenceBERT). The recommendations should be made by calculating the similarities between the embeddings of the user query and the synopsis texts in the data base, and sorting the movies accordingly (recommend the one with highest similarity first, second-highest second, etc.). You are free to use a vector database (e.g. ChromaDB[5]) to optimize the retrieval performance.

[3] You are free to use either PyTorch or Tensorflow, but we recommend using Pytorch (all NN-based examples in the lecture are based on it and the provided solutions to this assignment will use it as well). The hyperparameter configuration is also up to you, but you should have at least one hidden layer.

[4] `https://huggingface.co/datasets/mt0rm0/movie_descriptors_small`

[5] `https://docs.trychroma.com/`

6.2 LLMs as Recommender Systems (12 Pts.)

Solve the problem using only a generative large language model of your choice. You can use any existing open source model or API (e.g. GPT-2, LLama2, Microsoft Phi). Please do not only use the ChatGPT web interface, a valid solution needs to contain code ;) (Hint: This task is only about prompt engineering, you do not need to consider the data set from the other sub-task, and it is also not necessary to fine-tune the model).

References

Almeida, Tiago A, José María G Hidalgo, and Akebo Yamakami (2011). "Contributions to the study of SMS spam filtering: new collection and results". In: *Proc. DocEng*, pp. 259–262.

Cho, Kyunghyun, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio (2014). "On the Properties of Neural Machine Translation: Encoder–Decoder Approaches". In: *Proc. SSST-8*, pp. 103–111.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory". In: *Neural computation* 9.8, pp. 1735–1780.

Maas, Andrew, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts (2011). "Learning word vectors for sentiment analysis". In: *Proc. ACL-HLT*, pp. 142–150.

Pennington, Jeffrey, Richard Socher, and Christopher D Manning (2014). "GloVe: Global vectors for word representation". In: *Proc. EMNLP*, pp. 1532–1543.