Advanced Methods in Text Mining
Winter Semester 2022 / 2023
Assignment 2 – LSI, GloVe, and RNNs

Instructor: Dr. Rafet Sifa, Tobias Deußer
University of Bonn

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 3.9.

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 (10 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.

3. GloVe

3.1 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?

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

3.2 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. Recurrent Neural Networks

4.1 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?

4.2 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?

5. Revisiting the SMS spam dataset (30 Pts. + 5 bonus Pts.)

Remember the SMS spam detection pipeline we built last assignment? Take your code from that exercise (or write it anew, up to you) and replace the tf-idf embedder with a GloVe embedder. Furthermore, swap the logistic regression with a multilayer perceptron. Therefore, the pipeline should be:

1. Reading in the raw data (SMS spam dataset[2] introduced by Almeida et al. 2011).
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 one of the following methods we discussed during the lecture:
   (a) *Mean-pooling*: Represents each sentence by the mean of the word embeddings.
   (b) *Randomized Elman Networks*: Represents each sentence by the final hidden state evaluated considering the RandomizedSentenceEmbeddings algorithm (note that you have to adjust the spectral radius of the weight matrix corresponding to the hidden state – see the lecture slides for more information). Using this approach correctly earns you **5 bonus points**.
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).

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

[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. Try to beat your baseline (logistic regression from the previous assignment) here.

6. Named Entity Recognition with Recurrent Neural Networks

Consider the task of named entity recognition (NER), where the objective is to extract key information in form of entities, e.g. persons, locations, companies, numeric values, etc., from a document, sentence, or similar. Take this example from Deußer et al. 2022, in which we want to extract key performance indicators (KPIs) and their current year (cy) and prior year (py) values:

> “In 2021 and 2020 the $\underset{\text{kpi}}{\text{total net revenue}}$ was \$$\underset{\text{cy}}{100}$ million and \$$\underset{\text{py}}{80}$ million, respectively.”

6.1 Implementation (27 Pts.)

With this definition of NER in mind, construct a full NER pipeline for the CoNLL-2003 dataset, introduced in Tjong Kim Sang et al. 2003. This pipeline should integrate the following steps:

1. Reading in the raw data[4].
2. Encoding your textual data using a **GloVe** encoder.
3. Tagging the entities in the *IOB*[5] scheme, as introduced in Ramshaw et al. 1999.
4. Training a **Recurrent Neural Network**, use either a *LSTM* or *GRU*, to sequentially predict each token in the sentence. Use a *grid-search* on the validation set to find a decent hyperparameter configuration (use your own judgement when deciding on how many combinations you want to test).
5. Predicting on the test set and reporting your achieved accuracy and F_1 score.

6.2 Intuition (3 Pts.)

What do you think is the advantage of using a recurrent neural network when tagging NER tokens as done in the previous section?

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.

Deußer, Tobias, Syed Musharraf Ali, Lars Hillebrand, Desiana Nurchalifah, Basil Jacob, Christian Bauckhage, and Rafet Sifa (2022). “KPI-EDGAR: A Novel Dataset and Accompanying Metric for Relation Extraction from Financial Documents”. In: *Proc. ICMLA*.

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.

Ramshaw, Lance A and Mitchell P Marcus (1999). “Text chunking using transformation-based learning”. In: *Natural language processing using very large corpora*. Springer, pp. 157–176.

Tjong Kim Sang, Erik F. and Fien De Meulder (2003). “Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition”. In: *Proc. HLT-NAACL*, pp. 142–147.

[4] see `https://www.clips.uantwerpen.be/conll2003/ner/` for the website or `https://huggingface.co/datasets/conll2003` for the Hugging Face site.

[5] For a quick overview, see `https://en.wikipedia.org/wiki/Inside-outside-beginning_(tagging)`