RPTU
Department of Computer Science
Neural Networks for NLP
Prof. Dr. Sophie Fellenz
Vaishnavi Shirbhate
Mayank C. Ahuja

Neural Networks for NLP - Sheet 2

24.11.2025

Deadline: 8.12.2025 - 13:15, on OLAT as **pdf/.py/.ipynb** and/or after the lecture

Task 1: Analysing Pre-trained Word Embeddings

We will use pre-trained GloVe embeddings for analyzing contextualization of words. The pre-trained GloVe 6B embeddings can be downloaded here.

(a) Create two functions to compute the cosine and Euclidean distance between two vectors respectively. Use only Python's built-in functions and the math module.

(b) Create a function that computes the most similar K words for a given input word. What are the most similar words to "awesome"?

(c) Obtain the GloVe embeddings for the following words: red, orange, yellow, green, blue, purple, pink, brown, black, grey, white, violet, january, february, march, april, may, june, july, august, september, october, november, december. Calculate the cosine and euclidean distance between each pair of word embeddings and state your results. Do the word embeddings agree with your assumption of relatedness between words?

(d) Create a function that computes analogies of type *man is to king like woman is to what?* Create ten other examples and check if the model supports your assumptions.

Task 2: Computing Word Embeddings – Word2Vec

(a) Create a function that pre-processes and tokenizes the contents of the file *tutorial2.txt*. Then, produce a training set that consists of a list of center words and its surrounding context words.

(b) Train a Neural Network to predict the center word given some context words (CBOW). Use a context window of size 1.

(c) Train a Neural Network to predict the context words given some center word (Skip-gram). Use a context window of size 1.

(d) Choose the best Word2Vec model from the previous questions (Skip-gram or CBOW) and justify your choice. Then, train the exact same model for context windows of 2 and 3. Does the context window affect the performance of the model? Explain in not more than 50 words.

Task 3: Implementing RNN-LSTM Classifier

If you read the file *tutorial2.txt* carefully, you might observe that each line represents a movie review, where every odd line is a positive review and the even ones are negative reviews.

(a) Create a movie review dataset having all the positive and negative reviews from *tutorial2.txt*. Then, split the dataset into train (80%) and test (20%) sets. To obtain word embeddings of train and test splits, use the GloVe embeddings (used in Task 1). **Hint:** Don't forget to apply padding to your input sequences so that all the input sequences are of the same size.

(b) Train a simple RNN-LSTM classifier using the layers defined in the *tutorial2_task3.py*. Use the trained model to obtain the classification accuracy on the test set and present the results.

In the Materials folder, you will find a file called *tutorial2_task3.py*. Use it as your base code.

Task 4: Resume Matching using Sentence Embeddings and PCA

In this task, you will work on matching resumes with a job query by using sentence embeddings and Principal Component Analysis (PCA) for visualization.

(a) Load the dataset from the file *resumes_train.csv*. The dataset contains columns `resume` (the resume text) and `role` (the associated role). Use `pandas` to load the data.

(b) Use a pre-trained model *SentenceTransformer* (`all-MiniLM-L6-v2`) to encode the `resume` text into numerical embeddings. Print the shape of the resulting embeddings.

(c) Apply Principal Component Analysis (PCA) to reduce the embedding dimensions to 2. Plot the resumes in a 2D scatter plot where each resume is represented as a point colored by its `role`.

(d) Define a job query of your choice (e.g., *"Data Engineer with Apache Airflow experience"*). Encode the job query into a numerical embedding and compute the Euclidean distance between the job query and each resume embedding.

(e) Sort the resumes based on the computed distances and display the top 10 most relevant roles. Print the resume text for the closest match.