Middle East Technical University | Department of Computer Engineering

CENG 463

Introduction to Natural Language Processing

Fall 2025-2026

Programming Assignment 2

Due date: 1 December 2025, Monday, 23.59

Important note: You are expected to implement your solutions and write your discussions in the shared iPython Notebook. Any other form of submission except the shared notebook will receive a zero grade. Questions written in this document are also explained in the shared notebook for your convenience.

Problem Definition

In this programming assignment, you will be dealing with word embeddings and neural networks. You will use Python for this task. You can use libraries such as `pandas`, `nltk`, `numpy` etc. for your implementations, or implement your own functions.

Q1 - Word Embeddings (50 points)

In this question, you will first train a Word2Vec model, then use it to represent and analyse user reviews.

Q1.A - training (10 points)

Load the `user_review_train.csv` file shared with you. Using `Word2Vec` module of `gensim.models`, train a **skip-gram** `Word2Vec` model on the train data.

- Use the given preprocessing function `preprocess_review`.

Q1.B - word similarity (10 points)

Using the trained model, report the following:

- Similarity between "good" and "bad"
- Similar words to "good"
- Similar words to "bad"
- Similar words to "good" but not similar to "bad"
- Similar words to "good" but not similar to "bad"

and discuss the reported words and scores. Is it possible to identify specific good/bad features of the product that is being reviewed? What other words can be looked up to get more insight?

Notes and tips

- Check the documentation of `gensim.models.Word2Vec` to find relevant methods.

Q1.C - representation (15 points)

An important use of word embeddings is representing "documents" (reviews in our case). For this question, before creating the representations, do the following:

- Randomly sample 2 reviews from sentiment label 0, refer to them as `sent0_a` and `sent0_b`.
- Randomly sample 2 reviews from sentiment label 1, refer to them as `sent1_a` and `sent1_b`.

After the sampling, follow these steps to represent each review:

- Preprocess the review with the given `preprocess_review` function.
- For each token in the review, fetch the vector of that token.
- Take the average of the token vectors in the review to represent that review.

Then, calculate and report the cosine similarity of the two vectors representing:

- `sent0_a` and `sent0_b`
- `sent0_a` and `sent1_a`
- `sent1_a` and `sent1_b`

Does this representation work to capture the labels of the reviews? Do you think there is a better way to represent each review instead of taking the average of the word vectors? Discuss your findings with respect to these questions. Repeating the sampling process several times might give you a better insight.

Notes and tips

- You can use `numpy` for your calculations.

Q1.D - training and comparing classifiers (15 points)

For this task, you will use the `user_review_train.csv` and `user_review_test.csv` files to train a binary classification model with Word2Vec representations, and compare its performance with a binary classifier using Bag-of-Words representation. As the Bag-of-Words classifier, you can either choose the best performing classifier you have implemented in Question 3 of Programming Assignment 1, or you can follow these steps:

- Preprocess the review with the given `preprocess_review` function.
- Order all unique tokens by frequency, take the most frequent 100.
- Use these 100 words as the corpus for Bag-of-Words representation.

For the Word2Vec model, represent the reviews by following these steps:

- Preprocess the review with the given `preprocess_review` function.
- For each token in the review that is also in the most frequent 100 tokens, fetch the vector of that token.
- Take the average of the token vectors selected to represent that review.

After training both classifiers on `user_review_train.csv`, test them with `user_review_test.csv` and report the performance of your models with four metrics: accuracy, precision, recall and F1-score. Compare the performance of both models and discuss in detail.

Notes and tips

- You can use `CountVectorizer` from `scikit-learn` or any other library available for Bag-of-Words representation.
- You should select a classification method from the following set of classifiers: `[Naive Bayes, Support Vector Machine, Logistic Regression, Random Forest]`. You can use `scikit-learn`, `nltk`, or any other library for the classifier implementations.
- You should **not** use the test set `user_review_test.csv` during your training process. You should use `user_review_train.csv` only.
- You may add a validation step in your training process. To do this, you can further split the `user_review_train.csv` data and apply k-fold cross validation.

Q2 - Neural Networks for Binary Classification (50 points)

For this task, you will use the `user_review_train.csv` and `user_review_test.csv` files to train two neural network models for the binary classification of user reviews and compare their performances. You are expected to **Q2.A train RNN (20 points)** and **Q2.B train TextCNN (20 points)** models, and report the following:

- Confusion matrix of both models
- Time it took to train both models
- Accuracy, precision, recall, and F1-score of both models
- Other metrics you think are important

Finally as **Q2.C (10 points)**, you should discuss the performance of the models according to your reported results. Try to analyse the models in terms of pros and cons of using each one.

Notes and tips

- For the embedding layers of the models, you are free to use word embedding methods or leave them randomly initialised. Similarly, you can use word-based or character-based embeddings. However, make sure to explain your decisions.
- You are expected to use `tensorflow` for your implementations, but you can use other libraries if you already have a working setup.

1 Specifications and Regulations

- You will implement your solutions and write your discussion in the shared `e1234567_ceng463_pa2.ipynb` file. You should change the student ID placeholder in the file name to your student ID before submitting your solutions.
- Descriptions of the questions and steps for handling libraries, packages and modules are written in the shared notebook. Please obey the directives in the notebook for modifying the cells.
- Please add comments to your code when necessary.
- Limited discussions and codes with no explanation will result in lower grades.
- You are also given the dataset files to be used for the questions. Do not modify the datasets directly.
- Please obey the folder structure of the downloaded student pack to make sure that your notebook can read the dataset files when grading.
- If you have never worked with an iPython Notebook before, check out Jupyter or Google Colab (might require slight changes in the notebook).
- This is an individual work. We expect you to complete this assignment by yourself, ideally with no generative AI tool assistance. You will be responsible from the concepts covered in this assignment in the upcoming PA2 Quiz.