Middle East Technical University **Department of Computer Engineering**

CENG 463

Introduction to Natural Language Processing

Fall 2025-2026

Programming Assignment 1

Due date: 8 November 2025, Saturday, 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.

1 Problem Definition

In this programming assignment, we want to walk you through the building blocks of natural language processing before the success of neural models. You will be working on text preprocessing, textual data representation methods such as bag-of-words, and traditional ML training pipelines.

You will use Python for this task. You can use libraries such as `pandas`, `nltk`, `scikit-learn` etc. for your implementations, or implement your own functions. However, you are expected to analyse and reason about your implementation and results. The assignment consists of 3 questions.

1.1 Q1 - Special token extraction with regular expressions (10 points)

From the `old_tweets.csv` file shared with you, use regular expressions to capture the following:

- usernames (starting with '@', followed by at least 4, at most 15 alphanumeric characters and '_')
- hashtags (starting with '#' followed by any number of alphanumeric characters)
- numbers (different number notations may be captured, such as "1", ".004", "3:00" and "100,323")
- Emojis

Notes and tips

- First tokenizing each tweet by space may be a better approach than using a library tokenizer. You can use the re module to implement your solutions.
- Do not forget to add comments to your code.

1.2 Q2 - Ngrams and perplexity (40 points)

For this task, you will use the `user_reviews_train.csv` file as a corpus to train N-gram language models for N=1 to N=20. Then you will calculate and plot the following:

- Perplexity scores of the trained N-gram LMs for the first review in `user_reviews_test.csv`
- Perplexity scores of the trained N-gram LMs for `trip_review.txt`, which is a "domain-shift" for our LM.

How does our small model perform? Explain the plots and discuss your findings.

Notes and tips

- One approach may be using the `nltk` LM interface to implement the language model and calculate perplexity scores.
- You may follow other approaches, but make sure to explain your work in the discussion and by adding comments in your code.

1.3 Q3 - Classifying user reviews using Bag-of-Words (50 points)

For this final task, you will use the `user_reviews_train.csv` and `user_reviews_test.csv` files to implement a binary classifier with Bag-of-Words representation.

The given dataset includes user reviews and corresponding `sentiment` value for each review, either `1` meaning positive or `0` meaning negative. With this dataset, you are expected to implement a classifier that decides whether a given user review is positive or negative.

1.3.1 Part A - Analysis

In this part, you will analyse the training set given in the `user_reviews_train.csv` file to answer these questions:

- How many instances are there for each class?
- What is the minimum, maximum, and average character length of the reviews?
- Does the average length of a review depend heavily on the class label?
- Any other statistics you can report with minimal processing?

You can use tables or draw plots to show your analyses.

1.3.2 Part B - Implementation

The goal of this part is not to get the highest possible performance, but to see the effect of changing the text representation. Hence, you will train six models:

- `lighlty_preprocessed_most_15`
- `lighlty_preprocessed_most_100`
- `fully_preprocessed_most_15`
- `fully_preprocessed_most_100`
- `fully_preprocessed_most_500`
- `fully_preprocessed_least_15`

For the `lighlty_preprocessed_most_N` models, the preprocessing steps are as follows:

- Tokenize the reviews by separating from whitespaces.
- Lowercase all tokens for normalization.
- Order all unique tokens by frequency, take the most frequently used `N`.
- Use these `N` words as the corpus for Bag-of-Words representation.

For the `fully_preprocessed_most_N` models, the preprocessing steps are as follows:

- Tokenize the reviews by using `word_tokenize()` from `nltk.tokenize`.
- Lowercase all tokens for normalization.
- Remove stopwords by using `stopwords` from `nltk.corpus`.
- Apply lematization to tokens by using `WordNetLemmatizer` from `nltk.stem`.
- Order all unique tokens by frequency, take the most frequently used `N`.
- Use these `N` words as the corpus for Bag-of-Words representation.

For the `fully_preprocessed_least_15` model, the preprocessing steps are as follows:

- Tokenize the reviews by using `word_tokenize()` from `nltk.tokenize`.
- Lowercase all tokens for normalization.
- Remove stopwords by using `stopwords` from `nltk.corpus`.
- Apply lematization to tokens by using `WordNetLemmatizer` from `nltk.stem`.
- Order all unique tokens by frequency, take the **least** frequently used 15.
- Use these 15 words as the corpus for Bag-of-Words representation.

For all six models, after preprocessing, you should implement the model obeying the following:

- You must to use Bag-of-Words representation to represent each data instance. You can use `CountVectorizer` from `scikit-learn` or any other library available to implement this.
- 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_reviews_test.csv` during your training process. You should use `user_reviews_train.csv` only.
- You may add a validation step in your training process. To do this, you can further split the `user_reviews_train.csv` data and apply k-fold cross validation.
- You should analyse the performance of your models with four metrics on the `user_reviews_test.csv` test set: accuracy, precision, recall and F1-score.

1.3.3 Part C - Performance discussion

In this part, you will report the performance of your models with the four metrics mentioned in Part A. You should discuss your findings. Which model performed better, and why is that the case? Is it the expected result, or a surprising one? Discuss the results in detail. For the models with a corpus of length 15, make sure to add the corpus to your discussion.

2 Specifications and Regulations

- You will implement your solutions and write your discussion in the shared `e1234567_ceng463_pa1.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 a 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 PA1 Quiz.