Fall 2025 | DATA 266 | Homework -1
Deadline – 11.59 PM – 09/24/2025
20 Points

Question 1 – Transformer and Neural Network Models for Sentiment Classification

In this assignment, you will investigate how different models, tokenization schemes, and embeddings affect text classification performance. You will use the Amazon Polarity dataset (available on Hugging Face). First, randomly select 12,000 samples and then split them into 80% training (9,600 samples) and 20% testing (2,400 samples). For splitting, use a random seed equal to the final two digits of your student ID.

Part A – Zero-Shot Baseline (No Fine-Tuning) 2pts

1. Use DistilBERT (WordPiece) and DistilRoBERTa (BPE) to classify the test data without fine-tuning.
2. Evaluate both models in terms of:
   - o Accuracy
   - o F1-score
   - o Inference time (per 100 samples)

Part B – Fine-Tuning Transformer Models 4pts

1. Fine-tune DistilBERT and DistilRoBERTa on the training set.
2. Training requirements:
   - o Minimum 20 epochs
   - o Batch size ≥ 16
   - o Optimizer: AdamW
3. Report Accuracy, Precision, Recall, and F1-score.
4. Present the loss curve and discuss the loss curve.

Part C – Error Analysis 2pts

1. Identify 5 test samples misclassified by both models.
2. For at least 2 of these, show how each tokenizer splits the input into subwords.
   - o Example: “unbelievable” → WordPiece: ["un", "##believable"], BPE: ["un", "believ", "able"]
3. Discuss how tokenization differences may have contributed to the errors.

Part D – Closed-Source vs Open-Source Zero-Shot LLMs 3pts

1. Use the following models for zero-shot classification:
   - o Closed-source model: OpenAI GPT-4o-mini (via API).
   - o Open-source model: LLaMA (or equivalent, via Hugging Face).
2. Provide the prompt templates you used for classification.
3. Evaluate both models in terms of:
   - o Accuracy
   - o F1-score
   - o Inference time (per 100 samples)

Part E – RNN with Pre-trained Word Embeddings 4pts

1. Build an RNN-based classifier (RNN, GRU, or LSTM — specify your choice).
2. Initialize the embedding layer with Word2Vec, GloVe, and FastText (keep embeddings trainable).
3. Train on the same training set (9,600 samples) and test on the same test set (2,400 samples).
4. Training requirements:
   o Minimum 20 epochs
   o Batch size ≥ 32
   o Optimizer: Adam
5. Evaluate each embedding setup in terms of:
   o Accuracy
   o Recall
6. Present results in a comparison table.
7. Provide analysis:
   o Which embedding performed best overall?
   o Did FastText handle noisy/rare words better?
   o Show at least 2 misclassified examples where embedding choice influenced the result.

Question 2 – N-Gram Language Modeling with Moby-Dick (5pts)

In this task, you will build n-gram language models and evaluate their ability to generate text.

Dataset Preparation

1. Select at least 6 consecutive chapters from Moby-Dick (public domain, e.g., from Project Gutenberg).
2. Preprocess the text:
   o Lowercase all words.
   o Remove punctuation and special symbols.
   o Tokenize text into words.

Model Development

1. Develop a bigram model.

Text Generation

1. Use the seed text: “call me”.
2. Generate at least 50 words using the model.
3. Apply and compare three decoding strategies:
   o Greedy decoding
   o Beam search decoding (beam width = 3)
   o Sampling with top K = 5

Evaluation

1. Compute the perplexity score for the three decoding methods.
2. Present the generated text (first 50 words).

Useful link-

Question 1: Helper

Dataset link: https://huggingface.co/datasets/mteb/amazon_polarity?utm_source=chatgpt.com

You can use PreTrainedTokenizerFast for easy implementation. Set vocabulary size to 30,000 tokens for both tokenizers. Define:

Unk_token = "[UNK]",
special_tokens=["[PAD]", "[UNK]", "[CLS]", "[SEP]", "[MASK]"]
"pad_token": "[PAD]"

Example Code Sequence Classification: https://huggingface.co/blog/sentiment-analysis-python
BPE & WordPiece Tokenizer example: https://huggingface.co/docs/transformers/en/fast_tokenizers

Question 2 Helper:

MOBY-DICK: https://cs.brown.edu/courses/csci0931/2014-spring/2-text_analysis/HW2-2/MobyDick.txt

You are required to follow:

You must submit both source code and a single written report (MS Word or PDF). Submissions that do not follow this format will receive a penalty.

1. Source Code

- Python (Jupyter Notebook): Must be well-organized, properly indented, and include clear comments
- The notebook should run from start to finish without manual intervention

2. Written Report (MS Word or PDF)

Your report must be a single consolidated document and include the following:

1. Summary of each model and Mention hyperparameters used
2. Outputs and results: Include tables summarizing performance metrics (Accuracy, Precision, Recall, F1, Perplexity, etc. as applicable)
3. Screenshots: You must include screenshots to verify that your code actually ran:
    - Training logs showing first few epochs
    - Training logs showing completion of all epochs
    - Screenshots of important outputs such as:
        - Confusion matrices
        - Learning curves (loss/accuracy plots)
        - Classification reports

3. Submission Format

- One Jupyter Notebook (.ipynb) containing all source code
- One report (.docx or .pdf) with results, tables, screenshots, and analysis

- Both files must be clearly named:
    - HW1_LastName_FirstName.ipynb
    - HW1_LastName_FirstName.pdf

- **Failure to submit the source code will result in a deduction of full/partial points.**
- Before submitting the source code, please double-check that it runs without any errors.
- Must submit the files separately.
- Do not compress into a zip file.
- HW submitted more than 24 hours late will not be accepted for credit.