Spring 2025 | DATA 266 | Homework -1

Deadline – 11.59 PM – 03/01/2025

20 Points

Question 1 (2+ 6 +2 = 10 pts): This assignment examines the impact of fine-tuning on sentiment classification performance using pre-trained transformer models on IMDB dataset. You will compare zero-shot classification (without fine-tuning) against fine-tuned models and analyze the effects of byte pair encoding (BPE) vs. WordPiece tokenization on downstream performance.

From the IMDB dataset, select a training sample of at least 3,000 examples and a test sample of at least 300 examples. The training data will be used for fine-tuning, while the same test set must be used for evaluation in both scenarios (with and without fine-tuning) to maintain consistency. DistilBERT used WordPiece tokenization and DistilRoBERTa used BPE tokenization during training and inference.

A. Use DistilBERT and DistilRoBERTa to classify the test data without fine-tuning and evaluate performance in terms of accuracy, f1-score, and inference time.
B. Fine-tune DistilBERT and DistilRoBERTa separately on the training dataset using both BPE and WordPiece-based tokenization. In total, 4 combinations. Train each model for at least 3 epochs and evaluate performance using the same metrics.
C. Finally, perform a comparative analysis and discussion. Assess the impact of fine-tuning by comparing model performance before and after training. Evaluate the trade-offs between BPE and WordPiece in sentiment classification across the two models.

Question 2 (1 + 3 +1 = 5 pts): Use a multiclass sentiment analysis dataset from Hugging Face and select at least 2000 and 200 samples for training and evaluation, respectivly. Perform sentiment classification using two different approaches and compare their effectiveness.

A. OpenAI API-based Classification: Use any available OpenAI model (e.g., GPT-4o-mini, GPT-4-turbo or GPT-3.5) to classify sentiment for the test data. Evaluate model performance using accuracy and recall.
B. Train a Recurrent Neural Network (RNN)-based model using different pre-trained word embeddings: Word2Vec, GloVe, and FastText. Train the model using training data and evaluate the model on the same test dataset, and report performance in terms of accuracy and recall.
C. Finally, Compare OpenAI API performance vs. RNN models with different embeddings.

Question 3 (1 + 4 = 5 pts): Use at least five chapters from the book MOBY-DICK to develop an n-gram-based language model for text generation.

1. Text Preprocessing (1 point):
    a. Tokenization
    b. Convert to lowercase
    c. Expand contractions
    d. Remove punctuation
    e. Apply lemmatization/stemming

2. Develop unigram, and bigram language model. Generate minimum 50 words (use seed text = "call me") using greedy and beam generation (beam width = 4) approaches. Present perplexity score for all of the models.

Useful link-

Question 1: Helper

```
Dataset loading:
 ! pip install datasets
from datasets import load_dataset
# Load IMDb dataset
dataset = load_dataset("imdb")
```

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:

Dataset: https://huggingface.co/datasets/Sp1786/multiclass-sentiment-analysis-dataset

Question 3 Helper:

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

You are required to follow:

1. Submit **one** MS/PDF/Scanned document:
   - Include all the steps of your calculations.
   - Include the summary of the model.
   - Attach screenshots of your code.
   - Attach screenshots – showing first few epochs of model training.
   - Attach screenshots of the important code outputs such as confusion matrices, learning curves, and classification reports.
2. Source code:
   a. Python (Jupyter Notebook)
   b. Ensure it is well-organized with comments and proper indentation.
- **Failure to submit the source code will result in a deduction of full/partial points.**

- Format your filenames as follows: "your_last_name_HW1.pdf" for the document and "your_last_name_HW1_source_code.ipynb" for the source code.
- 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.