Google Colab notebook link:
https://colab.research.google.com/drive/1mlBDQA9HAPBfHEENDpMo_myWEC48uIYB?usp=sharing

Introduction

158736 - Advanced Machine Learning Assignment 2

In this assignment, you will develop a simple text classification system. You will be using the dataset given in this github repository. The corresponding research paper can be found here (note that you need to access the paper through the Massey network. The paper has been added to Stream for your convenience).

The dataset contains 5000 questions that have been asked by (potential) travellers. Each question has been classified into a coarse-grain class and a fine-grain class. For this assignment, we will only consider the coarse-grain classes as follows:

TTD things to do

TGU travel guide

ACM accommodation

TRS transport

WTH weather

FOD food

ENT entertainment

Preparation

Download the data csv file and remove the fine-grain column.

Read the paper to get an understanding of the dataset. You do not have to understand the full context of the paper, as we are not going to replicate their method. However, you should notice that this is a fine example of a text classification system based on traditional Machine Learning techniques, where a greater emphasis has been placed on input feature engineering and producing a pipe-lined system.

Model Implementation

Similar to assignment 1, your code should run in a colab notebook. Load the dataset into the root directory of your Google drive. Do NOT change the name of the dataset file. Make sure you give me access rights to your notebook. This will allow me to run your code without any problem.

Use Pytorch for implementation. **Add clear instructions/comments to the code.**
Follow the following steps for implementation.
1. Write the logic to prepare data to load into the models. Use 4000 samples for training, 700 for testing and 300 for validation.
2. As a baseline, implement a logistic regression (LR) classifier. You should try the following forms as the input
a. Raw data, which is sent through an embedding layer (as discussed in lectures)
b. Use Word2Vec or FastText representation of the data. You can easily find trained Word2Vec and FastText models for English

3. **Implement an LLM-based classifier. As shown in the lecture 7, you can directly load them from HuggingFace**

a. **Identify five different encoder-based LLMs to begin with. Provide a comparative discussion on how the selected models differ.**

The five different encoder-based LLM I chose are RoBERTa, DistillBERT, XLM, XLM-R, and Glot500.

1. RoBERTa, which stands for a Robustly Optimized BERT Pretraining approach, is designed by Facebook by adjusting various hyperparameters and training datasets. As for BERT (Bidirectional Encoder Representations from Transformers), it is a pre-trained model released by Google. It is designed to overcome the limitations of unidirectionality in standard language models and to improve the fine-tuning-based approaches (Devlin et al., 2019). RoBERTa was proven to outperform BERT. There are four main modifications, namely bigger batches, removing next sentence prediction, longer sequences, and dynamically changing the masking pattern (Liu et al., 2019).
2. DistillBERT, a distilled version of BERT that is 60% faster, 40% lighter in memory, and still retains 97% of BERT's performance (Sanh et al., 2020).
3. XLM refers to cross-lingual language models, which are released by Facebook. Two methods are released to learn XLM namely unsupervised one relying on monolingual data and supervised one with a cross-lingual language labels (Lample & Conneau, 2019).
4. XLM-R is short for XLM-RoBERTa, is also proposed by Facebook. It is a transformer-based multilingual masked language model that has been pre-trained in 100 languages (Conneau et al., 2020). It significantly outperforms multilingual BERT (mBERT) in applications such as cross-lingual classification and question answering.

5. Glot500 is an extended version of XLM-R. Instead of improving the performance in XLM-R's 104 languages, it focuses on pretraining 511 low-resource languages (Imani et al., 2023). It outperforms XLM-R due to the combination of factors such as corpus size, script, related languages assist, and model capacity.

b. Try your text classification system with two of the five LLMs you identified in (a) Provide your justification for model selection. You need to refer to research papers that compare the performance of different models, and cite them in your justification appropriately. (b) do not forget to consider the model size (you may use HF memory calculator). You cannot run very large models in a colab environment. Ideally, you should run each model-specific experiment three times with different seeds. However, you do not have to do that for this assignment, as it is time-consuming.

The models I chose are DistillBERT and RoBERTa because the other models (XLM, XLM-R, and Glot500) are all multilingual language models which are unnecessary for this text classification task. The reason to choose DistillBERT is that it maintains 97% of BERT's performance while 60% faster and 40% lighter in the memory (Sanh et al., 2020). As for the comparison between BERT and RoBERTa, RoBERTa outperforms BERT since it is based on BERT and optimizes key hyperparameters. Specifically, in the evaluation based on three benchmarks namely GLUE (the General Language Understanding Evaluation), SQuAD (the Stanford Question Answering Dataset) and RACE (the ReAding Comprehension from Examinations), RoBERTa outperforms BERT on both GLUE and SQuAD when controlling training data, and matches BERT when training larger datasets with 88.5 compared to 88.4 on GLUE (Liu et al., 2019).

c. Try several different values for the hyper-parameters and record the selected values.

d. Try global fine-tuning as well as feature-based fine-tuning (by the time you submit the file, make sure the code is set to feature-based fine-tuning and the smallest model)

4. Report Precision, Recall, F1 and Accuracy (as a percentage) as results, as well as model fine-tuning time.

a. You can report results in a tabular format as follows. Highlight the best result.

b. Provide a discussion of the results. You may consider discussing (but not limited to)

i. Impact of using pre-trained embeddings in the LR model

Pre-trained embeddings like FastText, Word2Vec, and GloVe might help improve the performance since they are trained on large and diverse corpora so that they can better capture context and meaning.

ii. The results differences you see between different LLMs/baselines and your reasoning for the observations.

However, the performance of FastText pre-trained word embeddings is worse than that of raw data embedding layers in the logistic regression classifier. The reason might be domain-specific data and data size. Pre-trained word embeddings like FastText are trained on large corpora of text data, but they might not capture the nuances specific to this tourism dataset. Additionally, this dataset only contains 5000 texts, which is limited to FastText.

As for the performance comparison between DistillBERT and RoBERTa, RoBERTa is better than DistillBERT with 86.7% accuracy compared to 85.4%. This is probably because of RoBERTa's architecture and improved training methods. RoBERTa is built on the BERT architecture and is trained with more complex data. In addition, it is trained with improved training strategies including dynamic masking, large batch training, and longer training times.

Table 1

Results of Different Models

Model	Accuracy	Precision	Recall	F1	Time (s)
LR (embedding layer)	76	74	81	77	13
LR(pre-trained word embeddings)	54	49	49	48	10
DistillBERT (global FT)	85.4	85.5	85.4	85.3	28139
RoBERTa (global FT)	86.7	86.8	86.7	86.7	14294

iii. Your observations against observations reported in research papers

Cortiz(n.d.) compared results among RoBERTa, DistillBERT, XLNet and BERT in emotion recognition and showed that RoBERTa is the best. My experiments of RoBERTa and DistillBERT show consistent results.

iv. Difference between global and feature-based fine-tuning.

In global fine-tuning, all the layers of the model are fine-tuned on the new dataset, while in feature-based fine-tuning, the lower layers are kept frozen to extract general features from the input data and other layers are fine-tuned.

Due to the extremely long fine-tuning time of the entire dataset, I used a small sample data to compare the global fine-tuning and feature-based fine-tuning results of DistillBERT and RoBERTa, as shown in Table 2. It shows that feature-based fine-tuning influences the performance with a lower accuracy but it reduces the processing time compared to the global fine-tuning, with 104s compared to 568s in DistillBERT, and 206s compared to 467s in RoBERTa.

Table 2

Comparison between Global FT and Feature-based FT

Model	Accuracy	Precision	Recall	F1	Time (s)
DistillBERT(global FT)	20	9.8	20	8.0	568
DistillBERT (feature-based FT)	18	11.5	18	9.3	104
RoBERTa(global FT)	24	11.1	24	14.3	467
RoBERTa (feature-based FT)	15	2.3	15	3.9	206

v. Time taken to fine-tune different models and the value-addition brought by larger models at the cost of fine-tuning time.

By comparing the global fine-tuning time between DistillBERT and RoBERTa in Table 2, DistillBERT takes a longer fine-tuning time compared to RoBERTa, with 568s and 467s respectively. However, comparing the feature-based fine-tuning time shows the opposite results, with 104s of DistillBERT and 206s of RoBERTa. Theoretically, RoBERTa should take longer due to its deeper architecture and a larger amount of hyperparameters. The value addition brought by RoBERTa at the cost of fine-tuning time is its superior performance, with 24% accuracy and 14.3% F1 score, compared to 20% and 8% of DistillBERT.

vi. Impact of hyper-parameters.

I adjusted three hyperparameters including learning rate, batch size, and epochs, the results are shown in Table 3.

Firstly, the learning rate determines the step size at each iteration during the optimization process. When I change it from “2e-5” to “3e-5”, accuracy remains the same, while F1 score decreases from 14.3% to 9.3%. It is weird that the fine-tuning processing time increases from 467s to 515s since a higher learning rate represents quicker convergence.

Secondly, batch size determines the number of training samples utilized in one iteration. When changing it from 16 to 32, both accuracy and F1 score increases, with 24% to 25% and 14.3% to 14.9% respectively.

Lastly, the number of epochs represents the number of complete passes through the whole training dataset. When adjusting it from 2 to 3, the results illustrate that A higher number of epochs improves the performance by achieving 35% accuracy and 23.9% F1 score, while taking a longer time of 705s compared to 467s.

Table 3

Results of Different Hyperparameters

RoBERTa Model Global Fine-tuning	Accuracy	Precision	Recall	F1	Time (s)
Original parameters (lr:2e-5,batch size:16, epochs:2)	24	11.1	24	14.3	467
Only change lr: 2e-5 to 3e-5	24	5.8	24	9.3	515
Only change batch size: 16 to 32	25	10.8	25	14.9	520
Only change epochs: 2 to 3	35	38.2	35	23.9	705

Lr*= learning rate.

vii. Rough comparison with the results reported in the paper. You cannot do a one-to-one comparison, as you do not know their data splits

Kahaduwa et al. (2017) achieved a maximum accuracy of 87% for the coarse class using the traditional SVM classifier and different combinations of features. In my experiment, the best global fine-tuning RoBERTa achieves 86.7% accuracy, which proves that the transformer-based RoBERTa has enhanced performance in the text classification domain.

A report that carries

1. the link to the colab notebook.
2. Answers to questions 3.a, 3.b, 3.c and 4.a, 4.b

References

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., & Stoyanov, V. (2020). *Unsupervised Cross-lingual Representation Learning at Scale* (arXiv:1911.02116). arXiv. http://arxiv.org/abs/1911.02116

Cortiz, D. (n.d.). *Exploring Transformers in Emotion Recognition: A comparison of BERT, DistillBERT, RoBERTa, XLNet and ELECTRA*.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*.

Imani, A., Lin, P., Kargaran, A. H., Severini, S., Sabet, M. J., Kassner, N., Ma, C., Schmid, H., Martins, A. F. T., Yvon, F., & Schütze, H. (2023). Glot500: Scaling Multilingual Corpora and Language Models to 500 Languages. *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1082–1117. https://doi.org/10.18653/v1/2023.acl-long.61

Kahaduwa, H., Pathirana, D., Arachchi, P. L., Dias, V., Ranathunga, S., & Kohomban, U. (2017). Question Answering system for the travel domain. *2017 Moratuwa Engineering Research Conference (MERCon)*, 449–454. https://doi.org/10.1109/MERCon.2017.7980526

Lample, G., & Conneau, A. (2019). *Cross-lingual Language Model Pretraining* (arXiv:1901.07291). arXiv. http://arxiv.org/abs/1901.07291

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). *RoBERTa: A Robustly Optimized BERT Pretraining Approach* (arXiv:1907.11692). arXiv. http://arxiv.org/abs/1907.11692

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2020). *DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter* (arXiv:1910.01108). arXiv. http://arxiv.org/abs/1910.01108