National College of Ireland

Configuration Manual

MSc Research Project
MSc in Artificial Intelligence

Manupavan Mulkuri
Student ID: 23301112

School of Computing
National College of Ireland

Supervisor: Abdul Shahid

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Manupavan Mulkuri
Student ID:	23301112
Programme:	MSc in Artificial Intelligence
Year:	2025
Module:	MSc Research Project
Supervisor:	Abdul Shahid
Submission Due Date:	11/08/2025
Project Title:	Configuration Manual
Word Count:	978
Page Count:	4

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Mulkuri Manupavan
Date:	14th September 2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Manupavan Mulkuri
23301112

1 System Requirements

We have run the experiments on the Google colab notebook with run-time type as GPU (T4). You can repeat the experiments in Google Colab or any other environment with the minimum specifications provided to run the code. We recommend a GPU configuration for fine-tuning of the Transformer and quick training cycles. To perform efficient calculations, at least eight-cores CPU and NVIDIA CUDA-compatible processor (e.g., Tesla T4) with a minimum of 16 GB VRAM are, recommended. The amount of RAM necessary to maintain tokenization pipeline and dataloader pipeline should be not less than 12 GB. One should have at least 20 - 50 Gb available to store Hugging Face datasets, the cached RoBERTa model and saved checkpoints.

The experiments are optimized on Ubuntu 18.04/20.04 but work on Windows 10+ and other Linux distributions as well (Google Colab is supported to the fullest extent possible). It requires Python 3.8+ and CUDA toolkit and cuDNN version which is compatible with your PyTorch construct (Colab has all of that ready). Also part of our core frameworks are PyTorch for model training and Hugging Face Transformers Wolf et al. (2020)/Datasets for model and data access.

2 Installation of Dependencies

Before running the code, you have to ensure that all the dependent libraries are installed and ready for use in your python environment. If you are using google colab most of the libraries are readily available in colab runtime environment. if you are not using colab, please ensure pip and python are available.

Try to import all the libraries which we are using in our code and if you encounter any errors while importing then install specific library which is not available using pip to avoid further errors. In some cases, the libraries installed may not be up-to date and may trigger warnings or errors. So, you can update the libraries installed by using the similar command given below.

eg. !pip install –upgrade fsspec datasets for updating existing libraries

!pip install datasets for installing missing libraries.

3 Dataset Configuration

We have used IMDB Maas et al. (2011) and SST2 Socher et al. (2013)datasets for our model training and evaluation. These datasets are loaded directly from the huggingface datasets Lhoest et al. (2021) library using python.

IMDB dataset has only train and test split but no validation set. Glue SST2 Wang et al. (2018) dataset has only train and validation split with labels and test split do not have lables. So, we have used the train split in both datasets as it is and created a seperate validation and test splits using stratified split strategy.

For IMDB: We have used test split and created both test and validation split using stratified 50-50 split.

For SST2: We have used validation split and created both test and validation split using stratified 50-50 split.

The created validation and test splits for both IMDB and SST2 datasets are stored in a folder and then reused wherever required. The training splits are loaded dorectly from huggingface hub.

4 Tokenizer Configuration

Tokenizer in the training and test scripts are configured with the following specifications.

Tokenizer: RobertaTokenizer.from_pretrained('roberta-base', add_prefix_space=True) Liu et al. (2019)

Max sequence length: 256

Truncation: on

Padding: max_length

This ensures uniform input sizes and stable batching.

5 Model Configuration

Firstly, we have to define the ten pre-defined aspects. Aspect-conditioned context vectors are scored and aggregated, and a final linear head produces the binary overall sentiment (negative/positive).

Training configuration :

Optimizer: AdamW

Learning rate: 2e-5

Batch size: 16

Epochs: 3

Loss: Cross-entropy

Device: CUDA if available, otherwise CPU

Checkpoint output: ./absa_roberta_bilstm_imdb_latest.pt

Variations in RoBERTa:

1. Fine - Tuned: All RoBERTa parameters are trainable.

2. Frozen: To freeze only RoBERTa's embedding layer parameters before training, set their requires_grad=False. This reduces trainable parameters and isolates the effect of the BiLSTM Schuster and Paliwal (1997) + attention layers while keeping encoder embeddings fixed.

Both variants use identical data, tokenization, and training schedules to enable fair comparison.

6 TroubleShooting

During the code execution the following errors may be encountered. You can do the following fixes for the relevant errors if encountered.

1. PyTorch Cannot find GPU:

1. check exclusively with the below code if the CUDA is available.

import torch print("CUDA available:", torch.cuda.is_available()) if torch.cuda.is_available(): print("GPU:", torch.cuda.get_device_name(0))

2. When using local machines be sure to check versions of NVIDIA drivers, CUDA toolkit and cuDNN compatibility with your PyTorch wheel.

3. In Colab go to Runtime →Change runtime type → GPU, and re-run the environment cells.

2. Out-of-Memory Error during Training:

1. Shrink batch_size (e.g., 16 -8 -4).

2. Keep max_length=256, but in extreme cases it can go down to 192 with warning (this is a setting change and it should be reported).

3. Make sure to close other processes of GPU when local usage is in progress.

3. Validation and Test split not found Error:

Check you have the validation and test split folder readily available and the path is defined correctly in the code. if you do not have the data split ready, you can use the code commented in the train script or model training notebook for creating a stratified split. verify the path and provide the path correctly to fix the issue.

4. Checkpoint not saved:

1. Verify if the path of the folder defined to save the file has write permissions and also the disk space is sufficient for the storage of the model.

2. Verify if the training cell is executed without errors and the save command works properly by printing the path after saving successfully.

5. Version Mismatch errors:

Some commands may not be compatible with the current versions of some libraries and may produce errors or warnings. So try to update the libraries and restart the environment to apply the changes before running the code again.

References

Lhoest, Q., Del Moral, A. V., Jernite, Y., Thakur, A., Von Platen, P., Patil, S., Chaumond, J., Drame, M., Plu, J., Tunstall, L. et al. (2021). Datasets: A community library for natural language processing, *arXiv preprint arXiv:2109.02846* .

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L. and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach, *arXiv preprint arXiv:1907.11692* .

Maas, A., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y. and Potts, C. (2011). Large movie review dataset, *Sentiment Analysis. Available online: https://ai. stanford. edu/˜ amaas/data/sentiment (accessed on 6 March 2023)* .

Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks, *IEEE transactions on Signal Processing* **45**(11): 2673–2681.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y. and Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank, *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O. and Bowman, S. R. (2018). Glue: A multi-task benchmark and analysis platform for natural language understanding, *arXiv preprint arXiv:1804.07461* .

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M. et al. (2020). Transformers: State-of-the-art natural language processing, *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pp. 38–45.