

Configuration Manual

MSc Research Project
Data Analytics

Jason Payne
Student ID: 2018543

School of Computing
National College of Ireland

Supervisor: Muhammad Zahid Iqbal


National College of Ireland
Project Submission Sheet
School of Computing



Student Name:	Jason Payne
Student ID:	2018543
Programme:	Data Analytics
Year:	2022
Module:	MSc Research Project
Supervisor:	Muhammad Zahid Iqbal
Submission Due Date:	15/12/2022
Project Title:	Configuration Manual
Word Count:	2,062
Page Count:	6

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:	
Date:	13th December 2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	<input type="checkbox"/>
Attach a Moodle submission receipt of the online project submission , to each project (including multiple copies).	<input type="checkbox"/>
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.	<input type="checkbox"/>

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

Jason Payne
2018543

1 Introduction

This configuration manual describes the required software tools and settings to successfully replicate the experimental setup of this project. High-level information is also provided for the codebase structure to assist with reproducing results.

2 Abbreviations

Abbreviation used in this configuration manual are as follows:

BDA	Baseline Data Augmentation
GloVe	Global Vectors for Word Representation
GPT	Generative Pre-trained Transformer
KWIC	Keyword in Context
LM	Language Model
LSTM	Long Short Term Memory
MRB	Manual Rule Book
OSHA	Occupational Safety and Health Administration
SME	Subject Matter Expert
TC	Topic Classification
TrDA	Transformer-based Data Augmentation

3 Environment Configuration

The research was implemented using Python 3.8. To reproduce the research experiments, the development environments must be recreated. For script-based code, the 'src' environment used for the experiments can be recreated using Conda¹ (for Python version install) and pip (for package dependencies) as follows:

Create conda environment from yml file:

```
conda env create -n projectcodename -f environment.yml
```

Install dependencies from requirements.txt (on windows):

¹<https://docs.conda.io>

```
pip install -r requirements.txt
```

Install dependencies from requirements.txt (on MacOS or Linux)

```
cat requirements.txt | xargs -n 1 pip install
```

The `environment.yml` and `requirements.txt` are located in the code repository.

For notebook code, Google Colab² was used for all experiments. No specific setup steps are required for Google Colab other than some package/resource installs as defined in the applicable notebooks. For completeness, and to assist with reproduction of results at a future date, a separate requirements file (`requirements_colab.txt`) for the Google Colab environment is provided in the project repository.

4 Reproduction Steps

The main steps of this project are summarised as follows:

1. Combine source data free-text description of incidents (remove other columns).
 - Dataset is three columns (row ID, free-text description, and dataset ID).
2. Iterate by randomly selected samples of 100 incidents (rows) and build rules.
 - Rules are added and adjusted during each iteration.
 - In parallel, build domain-specific synonym dictionary.
3. Stop iterating when coverage is above target (e.g., 70%).
 - Coverage is % of sample classified by the rule-book method.
4. Evaluate quality of rule book classifications (i.e., create scored samples).
 - Give classified narratives a score of 1, 2 or 3 by human/SME review.
 - 1: Good, 2: Fair and 3: Bad.
 - Scoring is performed using a scoring script.
 - A detailed output file is created for traceability.
5. Confirm quality is acceptable (target greater than 70% good and less than 5% bad).
6. Loop through rule-by-rule and create a binary dataset for each group/category.
7. Decide on focus (safety leading indicator) groups.
 - Selected groups for the research were minority groups (relatively scarce).
8. For each focus group, fine-tune a GPT-2 (Radford et al., 2019) decoder model.
9. For each focus group, create a set of tailored prompt rules.
10. Use prompt rules and fine-tuned LM to create fake data (for augmentation).
 - Each prompt rule is used to create n fake narratives.
11. Use baseline text augmentation techniques to create modified data.
12. Normalise and denoise data in preparation for TC model development.
13. Train Bi-directional LSTM (Bi-LSTM) model as binary classifier for each group.

²<https://colab.research.google.com>

- Begin with no treatment for class imbalance.
 - Then train base + BDA, base + TrDA and finally base + BDA + TrDA.
14. Evaluate and document performance on hold-out test data.
 - Calculate precision, recall and F1-score using rule-book hits as ground truth.
 - Adjust precision and F1-score by manual correction of false positives.
 15. Compare performance of Bi-LSTM and MRB methods on new unseen dataset.
 - New ‘unseen’ dataset was obtained at the end of the project.
 - The new dataset was unseen by both MRB and Bi-LSTM methods.

5 Codebase Structure

A folder structure is used in the code repository to separate the main sections/steps of the research and to make reproduction easier. The sub-folders used along with brief descriptions of their purpose and content is as follows:

01_data

The ‘source’ subfolder contains the raw input datasets (OSHA, ORGP and fabricated). All data is in CSV format. ‘ORGP’ is the reference used for data provided by the private company (a multi-national energy services company). There are two ORGP datasets, both (filenames) are tagged with ‘D1’ and date stamped (YYYYMMDD). The oldest and largest dataset is the dataset used for training. The more recent file is the dataset used for final model performance comparison. ‘Fabricated’ data refers to a small dataset manually fabricated narratives to assist classification of the ‘competency’ category.

02_data_preparation

This folder contains scripts and notebooks used to prepare the source data for:

- Topic classification (denoise, normalise, split).
- Augmentation (implementation of random selection (BDA) methods).
- Data exploration/visualisation.

Note that data preparation (creation) using the language model was done in the language model sub-folder (07_language_models).

03_embeddings

Storage of the GloVe pre-trained embedding model (e.g., glove.6B.50d.txt). Due to its relatively large file size (c. 170MB), this file is not stored in the final repo but can be downloaded from [here](https://nlp.stanford.edu/projects/glove/) (hosted by Stanford³).

04_rule_book

Rule book rules (compiled directly in CSV files by human/SME). The ‘_’ character is used to separate rules and the ‘{ }’ characters are used to denote ‘synonym’ or phrase permutation. Separate ‘rule’ files are used for rule book classification and language model

³<https://nlp.stanford.edu/projects/glove/>

prompt generation. A shared `synonym.csv` file (04_rule_book) was used. This file was created by human/SME review of domain data and incident reports.

The folder (04_rule_book) also contains the following scripts:

- `rule_book_funcs.py` (helper functions for rule book implementation).
- `rule_book_kwic.py` (script for running rule book on source data).
- `rule_book_kwic_coverage_chk.py` (calculates coverage using random sampling).
- `rule_book_kwic_create_test_sample.py` (creates sample of n classified reports)
- `rule_book_kwic_score.py` (manually scores the quality of MRB classifications)

05_coverage_checks

Coverage refers to the percentage of narratives assigned at least one label by the MRB method. Coverage checks are performed by running 30 randomly selected samples (each of size 100). Running all rules on all data (over 93k narratives) would take several days of steady connection. This folder contains coverage results for the 30 samples.

06_scored_samples

This folders contains the scored files for both MRB labelled samples and Bi-LSTM false positives. MRB labelled samples contain 100 incident narratives, the assigned MRB labels and the human/SME assigned score (1, 2 or 3). Running `rule_book_kwic_score.py` from the '04_rule_book' folder creates the scored MRB files that are saved to this folder. Running `tc_model_score.py` (09_topic_classification) creates the scored false positive files that are written to this folder. All files are datetime stamped according to run time.

07_language_models

The `GPT_fine_tuning_and_fake_generation.ipynb` notebook is used in Google Colab to fine-tune the GPT-2 decoder model on category-focused datasets created by MRB classification. This creates an LM that creates fake incident narratives in the context of a selected focus group. This notebook also contains the block of code that creates the fake narratives from input prompts. The `language_model_prompt_generation.py` script takes a 'rule book' style CSV input (`language_prompts.csv` from '04_rule_book') and mutates them to create an output list of prompts (`prompts.csv` in '04_rule_book').

08_output

This folder is used to store output from analysis, post-processing and modelling steps. Tokenisers and trained models are saved to this location as datetime stamped pickle format files. The binary labelled focus datasets created from running the rule book classification script (`rule_book_kwic.py`) are also stored here.

09_topic_classification

This folder contains notebooks for training topic classification models. The main model selected is a bi-directional LSTM model using GloVe embeddings. Various iterations of the model are presented:

- Base model with no treatment for class imbalance.

- Model with data augmentation by ‘standard’ (BDA) techniques.
- Model with data augmentation by transformer based augmentation (TrDA).

10_performance

This folder contains model output files (label predictions/assignments) used as input to scoring scripts (`rule_book_kwic_score.py`) (04_rule_book) and (`tc_model_score.py`) (09_topic_classification). As performance measures for the classification models are based on precision, recall and F1-score calculated on the basis that MRB classifications are ‘true’, there was a requirement to calculate adjusted measures by manual review of false positives. The spreadsheet `results_collation.xlsx` was used for the calculation of adjusted measures, using the (`rule_book_kwic_score.py`) (04_rule_book) script to expedite the manual review process and generate new counts.

11_plots

Plots generated with required formatting for insertion in the project (LaTeX) report.

12_general_resources

General resources and tools used in certain steps, e.g., GB to US english dictionary.

13_test_samples

Storage area for rule book classified test samples.

6 Manual Rulebook (MRB)

The rule-book method is run from the `rule_book_kwic.py` script (04_rule_book). Three run-mode choices must be selected to initiate the run. The first is data choice (original or latest), the second determines if the run should be executed on all data or a randomly selected sample, and the third defines rule group extent (all rules or just one rule group). Running all rules on all data is not recommended, as it will take several days to complete. Hence, for trial purposes, selecting either the sample or single rule group mode of operation is recommended. When data is run for all data, the labelled output is written to ‘08_output’. The other rule-book scripts (test sample creation, coverage check and scoring) run similarly to the base script, with run options prompted for user input.

7 Language Model Fine-tuning

LM fine-tuning was performed using the `GPT_fine_tuning_and_fake_generation.ipynb` notebook in Google Colab. The main Python packages used are pytorch (deep learning) and transformers (pre-trained models and tokenisers, [Wolf et al. \(2019\)](#)). A focus category is selected by the user at the start of the notebook and this loads the appropriate ‘training trues’ dataset for LM fine-tuning, i.e., the subset of the main dataset that contains narratives labelled with the focus category. After model training, the `prompts.csv` (04_rule_book) is loaded and used to generate approximately 8,000 fake narratives using the fine-tuned LM. The fabricated narratives are automatically written to the ‘01_data\fabricated’ folder location.

8 Baseline Data Augmentation (BDA)

Base line data augmentation is performed using the `baseline_data_augmentation.ipynb` notebook (02_data_preparation). The main Python package used is nltk with the ‘stop-words’, ‘wordnet’, ‘omw-1.4’ and ‘punkt’ resources downloaded to the environment. The default target number of augmentations is 8,000. Once augmentations have been generated, they are written to ‘01_data\fabricated’.

9 Supervised Topic Classification

There are four variations of the `safety_BiLSTM.ipynb` notebook located in the ‘09 topic classification’ folder. These cover ‘no augmentation’ plus three levels of augmentation (BDA, TrDA and BDA + TrDA). Additional identifying text is added to the filename to identify which notebook applies to which augmentation level. The main Python packages used are sklearn (for data pre-processing) and tensorflow (for model training). A focus category is selected at the start of the notebook and this loads the appropriate binary classified prepared dataset. The input dataset is created initially by MRB labelling and then cleaned by normalisation and denoising steps using the `create_data_for_model.ipynb` notebook. Datasets for augmentation are then loaded where applicable. Input datasets for BDA augmentation are created from the `baseline_data_augmentation.ipynb` notebook (02_data_preparation). Input datasets for TrDA augmentation are created from the `GPT_fine_tuning_and_fake_generation.ipynb` notebook (07_language_models). Both augmenting datasets are loaded from ‘01_data\fabricated’. Trained models are written to a pickle file in ‘08.output\models’.

10 Inference for New Data

A comparison was also performed for MRB versus Bi-LSTM performance on a new unseen dataset. The new dataset was firstly labelled using `rule_book_kwic.py` (04_rule_book) running in ‘selected rule’ mode. This limits the rule-book to only checking for the presence of a single defined focus category. This script outputs a binary classified dataset to ‘08.output’. This output dataset is then loaded into `create_data_for_model.ipynb` (02_data_preparation) to normalise and denoise raw data in preparation for inference by the Bi-LSTM model. The `safety_BiLSTM_BDA_trDA.ipynb` is then used in ‘load model’ mode which loads the trained model from ‘08.output\models’.

References

- Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. and Sutskever, I. (2019). Language models are unsupervised multitask learners.
- Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M. et al. (2019). Huggingface’s transformers: State-of-the-art natural language processing, *arXiv preprint arXiv:1910.03771*.