Exam Portfolio - Language Analytics

Luke Ring (202009983)

2023-08-04

Contents

1 Exam Portfolio - Language Analytics

Luke Ring (202009983)

Date: 6.8.2023

1.1 Repositories

Complete portfolio repository: https://github.com/zeyus/cds-language-exam

- Assignment 1: https://github.com/zeyus/cds-language-exam/tree/main/assignment_1
- Assignment 2: https://github.com/zeyus/cds-language-exam/tree/main/assignment_2
- Assignment 3: https://github.com/zeyus/cds-language-exam/tree/main/assignment_3
- Assignment 4: https://github.com/zeyus/cds-language-exam/tree/main/assignment_4
- Assignment 5: https://github.com/zeyus/cds-language-exam/tree/main/assignment_5

This document is available as a PDF in the repository: https://github.com/zeyus/cds-language-exam/blob/main/README.pdf

2 Assignment 1 - Extracting linguistic features using spaCy

2.1 Original Assignment Description

This assignment concerns using `spaCy` to extract linguistic information from a corpus of texts.

The corpus is an interesting one: *The Uppsala Student English Corpus (USE)*. All of the data is included in the folder called `in` but you can access more documentation via this link.

For this exercise, you should write some code which does the following:

- Loop over each text file in the folder called `in`
- Extract the following information:
  - Relative frequency of Nouns, Verbs, Adjective, and Adverbs per 10,000 words
  - Total number of *unique* PER, LOC, ORGS
- For each sub-folder (a1, a2, a3, …) save a table which shows the following information:

Filename	RelFreq NOUN	RelFreq VERB	RelFreq ADJ	RelFreq ADV	Unique PER	Unique LOC	Unique ORG
file1.txt	—	—	—	—	—	—	—
file2.txt	—	—	—	—	—	—	—
etc	—	—	—	—	—	—	—

2.1.1 Objective

This assignment is designed to test that you can:

1. Work with multiple input data arranged hierarchically in folders;
2. Use `spaCy` to extract linguistic information from text data;
3. Save those results in a clear way which can be shared or used for future analysis

2.1.2 Some notes

- The data is arranged in various subfolders related to their content (see the README for more info). You'll need to think a little bit about how to do this. You should be able do it using a combination of things we've already looked at, such as `os.listdir()`, `os.path.join()`, and for loops.
- The text files contain some extra information that such as document ID and other metadata that occurs between pointed brackets `<>`. Make sure to remove these as part of your preprocessing steps!
- There are 14 subfolders (a1, a2, a3, etc), so when completed the folder `out` should have 14 CSV files.

2.1.3 Additional comments

Your code should include functions that you have written wherever possible. Try to break your code down into smaller self-contained parts, rather than having it as one long set of instructions.

For this assignment, you are welcome to submit your code either as a Jupyter Notebook, or as `.py` script. If you do not know how to write `.py` scripts, don't worry - we're working towards that!

Lastly, you are welcome to edit this README file to contain whatever informatio you like. Remember - documentation is important!

2.2 Assignment 1, Luke Ring

Repository: https://github.com/zeyus/cds-language-exam/tree/main/assignment_1

This repository contains a script for the Cultural Data Science: Language Analytics course at Aarhus University. The script recursively extracts linguistic features from text files in an input folder and saves them in CSV files in an output folder.

2.2.1 Contribution

This assignment was completed by me individually and independently, the code contained in this repository is my own work.

2.2.2 Setup

Using anaconda:

```
conda env create -f environment.yml
conda activate cds-lang-1
```

Using pip:

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

2.2.3 Usage

```
python src/extract_linguistic_info.py -i input_folder -o output_folder <-e encoding> <-m
↪  spacy_model>
```

By default, the spacy models `en_core_web_md`and `en_core_web_lg` are included and can be used for the `-m` flag. If you want to use a different model, you need to install it first:

```
python -m spacy download <model_name>
```

You can run the script with the `-h` flag to see the available options:

```
usage: extract_linguistic_info.py [-h] [-i I] [-o O] [-e E] [-m M]

Extracts linguistic information from text files

options:
  -h, --help  show this help message and exit
  -i I        The directory containing the text files to be processed
  -o O        The directory to write the output files to
  -e E        The encoding of the input files
  -m M        The spacy model to use
```

2.2.3.1 Example

```
python src/extract_linguistic_info.py -i in/USEcorpus -o out -e latin-1 -m en_core_web_md
```

This will provide a progress bar and output something like the following to the terminal:

```
2023-02-25 11:11:38 - INFO - Arguments:
2023-02-25 11:11:38 - INFO - Input directory: in\USEcorpus
2023-02-25 11:11:38 - INFO - Output directory: out
2023-02-25 11:11:38 - INFO - Starting linguistic information extraction
2023-02-25 11:11:38 - INFO - Loading spacy model en_core_web_lg
2023-02-25 11:11:38 - INFO - Using GPU
2023-02-25 11:11:41 - INFO - Getting list of files
2023-02-25 11:11:41 - INFO - Found 1497 files
2023-02-25 11:11:41 - INFO - Extracting linguistic information
100%|-------------------------------------| 1497/1497 [02:28<00:00, 10.06it/s]
2023-02-25 11:14:10 - INFO - Writing summary tables
```

2.2.4 Output

The script creates a CSV file for each text subfolder in the input folder. The CSV files are named after the subfolder.

Each CSV file contains the following columns:

Column	Description
Filename	The name of the text file
RelFreq NOUN	The relative frequency of nouns in the text
RelFreq VERB	The relative frequency of verbs in the text
RelFreq ADJ	The relative frequency of adjectives in the text
RelFreq ADV	The relative frequency of adverbs in the text
Unique PER	The number of unique named entities of type PERSON
Unique LOC	The number of unique named entities of type LOCATION
Unique ORG	The number of unique named entities of type ORGANIZATION

The followg output CSV files are available in the `out` folder:

- a1.csv
- a2.csv
- a3.csv
- a4.csv
- a5.csv
- b1.csv
- b2.csv
- b3.csv
- b4.csv
- b5.csv
- b6.csv
- b7.csv
- b8.csv
- c1.csv

2.2.4.1 Example Output

The following is the contents of b6.csv:

Filename	RelFreq NOUN	RelFreq VERB	RelFreq ADJ	RelFreq ADV	Unique PER	Unique LOC	Unique ORG
0107.b6.txt	1724.14	1238.83	855.68	421.46	1	0	1
0137.b6.txt	1735.65	1241.66	934.58	534.05	1	0	0
0151.b6.txt	1491.23	1353.38	651.63	538.85	3	0	0
0157.b6.txt	1215.47	1381.22	718.23	607.73	2	0	0
0158.b6.txt	1522.49	1257.21	761.25	657.44	2	0	0
0178.b6.txt	1742.34	1140.44	876.45	549.10	2	0	1
0185.b6.txt	1609.20	1379.31	675.29	416.67	2	0	0
0198.b6.txt	1542.94	1222.71	669.58	465.79	2	0	0
0219.b6.txt	1701.53	1311.02	543.93	362.62	2	0	0
0223.b6.txt	1731.01	1232.88	660.02	622.67	3	0	0
0238.b6.txt	1400.97	1417.07	772.95	402.58	2	0	0
0318.b6.txt	1764.71	980.39	813.73	460.78	3	0	0

3 Assignment 2 - Text classification benchmarks

3.1 Original Assignment Description

This assignment is about using `scikit-learn` to train simple (binary) classification models on text data. For this assignment, we'll continue to use the Fake News Dataset that we've been working on in class.

For this exercise, you should write *two different scripts*. One script should train a logistic regression classifier on the data; the second script should train a neural network on the same dataset. Both scripts should do the following:

- Be executed from the command line
- Save the classification report to the folder called `out`
- Save the trained models and vectorizers to the folder called `models`

3.1.1 Objective

This assignment is designed to test that you can:

1. Train simple benchmark machine learning classifiers on structured text data;
2. Produce understandable outputs and trained models which can be reused;
3. Save those results in a clear way which can be shared or used for future analysis

3.1.2 Some notes

- Saving the classification report to a text file can be a little tricky. You will need to Google this part!
- You might want to challenge yourself to create a third script which vectorizes the data separately, and saves the new feature extracted dataset. That way, you only have to vectorize the data once in total, instead of once per script. Performance boost!

3.1.3 Additional comments

Your code should include functions that you have written wherever possible. Try to break your code down into smaller self-contained parts, rather than having it as one long set of instructions.

For this assignment, you are welcome to submit your code either as a Jupyter Notebook, or as `.py` script. If you do not know how to write `.py` scripts, don't worry - we're working towards that!

Lastly, you are welcome to edit this README file to contain whatever informatio you like. Remember - documentation is important!

3.2 Assignment 2, Luke Ring

Repository: https://github.com/zeyus/cds-language-exam/tree/main/assignment_2

3.2.1 Contribution

This assignment was completed by me individually and independently, the code contained in this repository is my own work.

3.2.2 Setup

Clone the repository and install the requirements:

```
git clone https://github.com/zeyus/cds-language-exam
cd cds-language-exam/assignment_2
pip install -r requirements.txt
```

3.2.3 Running text classification benchmarks

There are two main scripts for running the text classification benchmarks:

- `src/txt-benchmark-lr.py` runs a Linear Regression model on the text classification task
- `src/txt-benchmark-nn.py` runs a Neural Network model on the text classification task

By default, the script uses the paths required for the assignment, but can be customized.

Both scripts support the following arguments:

```
usage:  [-h] [--version] [-f FILE] [-m MODEL_SAVE_PATH] [-r REPORT_PATH] [-v {tfidf,count}]

Text classification CLI

options:
  -h, --help            show this help message and exit
  --version             show program's version number and exit
  -f FILE, --file FILE  Path to the CSV file containing the data (default: in\fake_or_real_news.csv)
  -m MODEL_SAVE_PATH, --model-save-path MODEL_SAVE_PATH
                        Path to save the trained model(s) (default: models)
  -r REPORT_PATH, --report-path REPORT_PATH
                        Path to save the classification report(s) (default: out)
  -v {tfidf,count}, --vectorizer {tfidf,count}
                        Vectorizer to use (default: tfidf)
```

The reports contain the following information/columns:

- model: the name of the model
- timestamp: the timestamp of the run
- vectorizer: the name of the vectorizer
- train_accuracy: the accuracy of the model on the training set
- train_precision: the precision of the model on the training set
- train_recall: the recall of the model on the training set
- train_f1: the F1 score of the model on the training set
- test_accuracy: the accuracy of the model on the test set
- test_precision: the precision of the model on the test set
- test_recall: the recall of the model on the test set
- test_f1: the F1 score of the model on the test set
- model_params: the parameters of the model
- vectorizer_params: the parameters of the vectorizer

- train_metrics_report: the classification report of the model on the training set
- test_metrics_report: the classification report of the model on the test set

It's not as pretty as I'd have liked but it can be read into a pandas dataframe for further analysis/summary.

3.2.4 Results

The following results are from classifiers run on the `test` data set.

3.2.4.1 Logistic Regression With Count Vectorizer

Max iterations: 100

```
              precision    recall  f1-score   support

        FAKE       0.89      0.90      0.90       619
        REAL       0.90      0.90      0.90       648

    accuracy                           0.90      1267
   macro avg       0.90      0.90      0.90      1267
weighted avg       0.90      0.90      0.90      1267
```

3.2.4.2 Logistic Regression With TF-IDF Vectorizer

Max iterations: 100

```
              precision    recall  f1-score   support

        FAKE       0.89      0.90      0.89       629
        REAL       0.90      0.89      0.90       638

    accuracy                           0.90      1267
   macro avg       0.90      0.90      0.90      1267
weighted avg       0.90      0.90      0.90      1267
```

3.2.4.3 Neural Network With Count Vectorizer

Max iterations: 1000

```
              precision    recall  f1-score   support

        FAKE       0.89      0.94      0.92       618
        REAL       0.94      0.89      0.92       649

    accuracy                           0.92      1267
   macro avg       0.92      0.92      0.92      1267
weighted avg       0.92      0.92      0.92      1267
```

3.2.4.4 Neural Network With TF-IDF Vectorizer

Max iterations: 1000

```
              precision    recall  f1-score   support

        FAKE       0.91      0.91      0.91       635
        REAL       0.91      0.91      0.91       632

    accuracy                           0.91      1267
   macro avg       0.91      0.91      0.91      1267
weighted avg       0.91      0.91      0.91      1267
```

4 Assignment 3 - Language modelling and text generation using RNNs

4.1 Original Assignment Description

Text generation is hot news right now!

For this assignemnt, you're going to create some scripts which will allow you to train a text generation model on some culturally significant data - comments on articles for *The New York Times*. You can find a link to the data here.

You should create a collection of scripts which do the following:

- Train a model on the Comments section of the data
  - Save the trained model
- Load a saved model
  - Generate text from a user-suggested prompt

4.1.1 Objectives

Language modelling is hard and training text generation models is doubly hard. For this course, we lack somewhat the computationl resources, time, and data to train top-quality models for this task. So, if your RNNs don't perform overwhelmingly, that's fine (and expected). Think of it more as a proof of concept.

- Using TensorFlow to build complex deep learning models for NLP
- Illustrating that you can structure repositories appropriately
- Providing clear, easy-to-use documentation for your work.

4.1.2 Some tips

One big thing to be aware of - unlike the classroom notebook, this assignment is working on the *Comments*, not the articles. So two things to consider:

1) The Comments data might be structured differently to the Articles data. You'll need to investigate that;
2) There are considerably more Comments than articles - plan ahead for model training!

4.1.3 Additional pointers

- Make sure not to try to push the data to Github!
- *Do* include the saved models that you output
- Make sure to structure your repository appropriately
  - Include a readme explaining relevant info
    - E.g where does the data come from?
    - How do I run the code?
- Make sure to include a requirements file, etc… ## Assignment 3, Luke Ring

Repository: https://github.com/zeyus/cds-language-exam/tree/main/assignment_3

4.1.4 Contribution

This assignment was completed by me individually and independently, the code contained in this repository is my own work.

4.1.5 Setup

4.1.5.1 Prerequisites

- Python 3.9
- Optional CUDA compatible GPU for training

4.1.5.2 Clone repository

```
git clone https://github.com/zeyus/cds-language-exam
cd cds-language-exam/assignment_3
```

4.1.5.3 Install dependencies

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

4.1.5.4 Data The dataset should be the NYT comments dataset from Kaggle, which can be found here. By default the `src/text-gen-rnn.py` script expects the dataset to be located in `data/nyt_comments`.

4.1.6 Usage

4.1.6.1 General For help you can run the main script `src/text-gen-rnn.py` with the `--help` flag.

```
python src/text-gen-rnn.py --help
```

```
usage: text-gen-rnn.py [-h] [--version] [-s MODEL_SAVE_PATH] [-d DATASET_PATH] [-b
↪  BATCH_SIZE] [-e EPOCHS] [-o OUT] [-c FROM_CHECKPOINT] [-p PARALLEL] [-t TEMPERATURE] [-n
↪  TOP_N]
                       [-m MIN_LENGTH]
                       {train,predict} [prediction_string]

Text classification CLI

positional arguments:
  {train,predict}       The task to perform
  prediction_string

optional arguments:
  -h, --help            show this help message and exit
  --version             show program's version number and exit
  -s MODEL_SAVE_PATH, --model-save-path MODEL_SAVE_PATH
                        Path to save the trained model(s) (default: models)
  -d DATASET_PATH, --dataset-path DATASET_PATH
                        Path to the dataset (default: data/nyt_comments)
  -b BATCH_SIZE, --batch-size BATCH_SIZE
                        The batch size (default: 64)
  -e EPOCHS, --epochs EPOCHS
                        The number of epochs (default: 10)
  -o OUT, --out OUT     The output path for the plots and stats (default: out)
  -c FROM_CHECKPOINT, --from-checkpoint FROM_CHECKPOINT
                        Use the checkpoint at the given path (default: None)
  -p PARALLEL, --parallel PARALLEL
                        Number of workers/threads for processing. (default: 4)
  -t TEMPERATURE, --temperature TEMPERATURE
                        Temperature for sampling during prediction. (1.0 is deterministic)
↪  (default: 0.8)
  -n TOP_N, --top-n TOP_N
```

```
                        Top N for sampling during sequence-to-sequence prediction. (1 is
↪  equivalent to argmax) (default: 1)
  -m MIN_LENGTH, --min-length MIN_LENGTH
                        Minimum length of generated text (in tokens, not characters).
↪  (default: 0)
```

4.1.6.2 Training

To train a model you can run the main script `src/text-gen-rnn.py` with the `train` argument. Additionally, you can specify `-e EPOCHS` and `-b BATCH_SIZE` to change the number of epochs and batch size respectively.

```
python src/text-gen-rnn.py train -e 10 -b 64
```

This will train the model, save it to `models/` and plot the training history to `out/`. Also the prepared datasets and encoder are saved in `data/`. Therefore, if you change the vocabulary size or sequence length, you should delete the `data/encoder` directory to recompute the encoder (this will also regenerate the datasets, as the datasets are capped to the sequence lenght).

4.1.6.3 Prediction

To predict text you can run the main script `src/text-gen-rnn.py` with the `predict` argument. Additionally, you can specify `-c FROM_CHECKPOINT` to load a model from a checkpoint, `-t TEMPERATURE` to change the temperature for sampling, `-n TOP_N` to change the top N for sampling and `-m MIN_LENGTH` to change the minimum length of the generated text.

Note: if you do not specify a checkpoint, the checkpoint created from the training run included in this repository will be used.

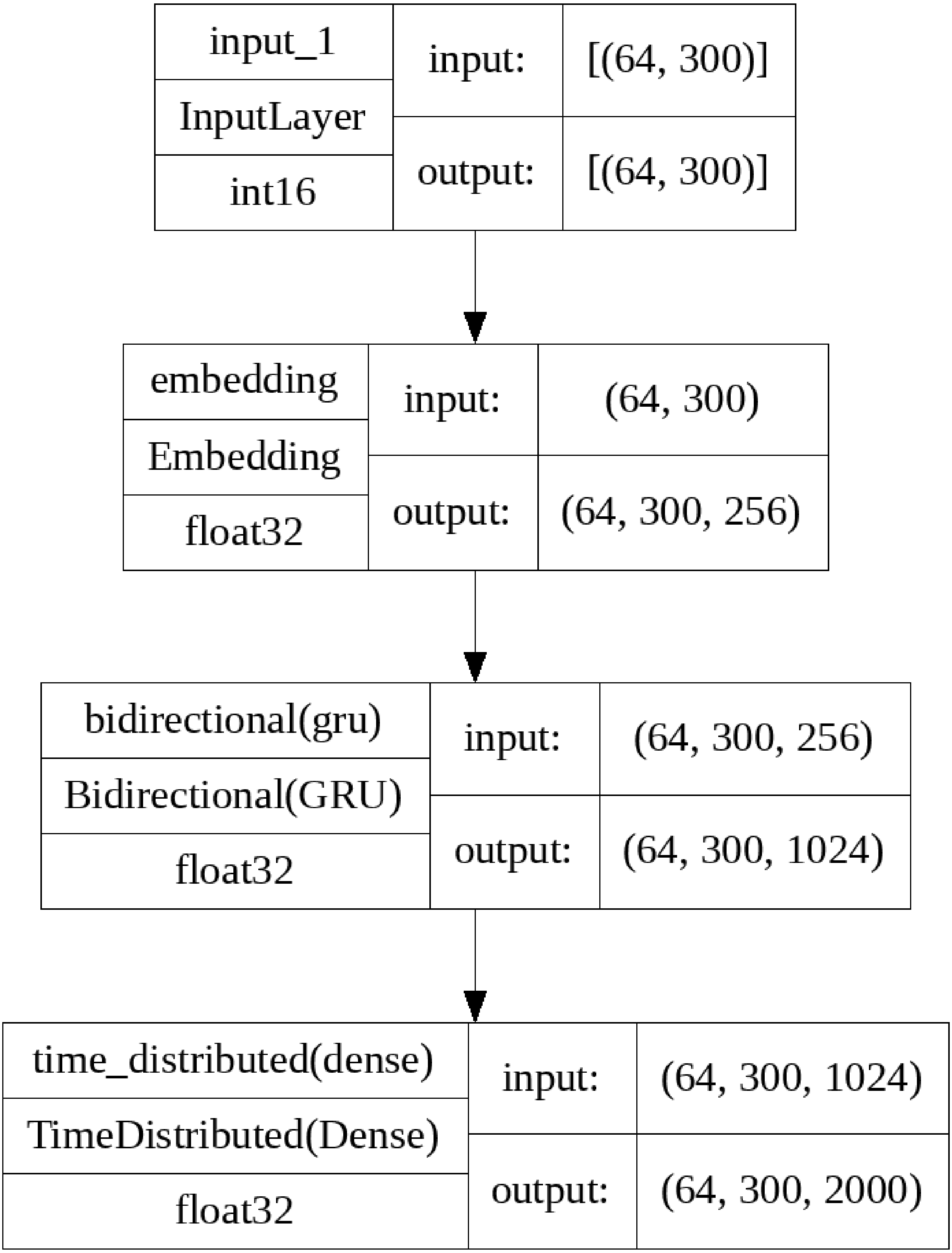
```
python src/text-gen-rnn.py -c models/rnn/20230502_095615_rnn_2000x300_batch64_iter10.h5 -t
↪  0.8 -n 1 -m 100 predict "some text to start the prediction with"
```

4.1.7 Results

4.1.8 Model Architecture

The model was designed to be a simple RNN with a single bi-directional GRU layer. It was trained with the idea of a sequence-to-sequence architecture, thus the final dense layer was wrapped in a `TimeDistributed` layer.

The model architecture can be seen in the following figure:



The model was configured with a maximum vocabulary size of 2000, and a maximum sequence length of 300.

4.1.8.1 Training

4.1.8.1.1 Data preprocessing The data were split into training and validation sets, with a 90/10 split. The dataset `x` consisted of either:

- the article headline, keywords and abstrict concatenated; or
- a top level comment

The dataset `y` consisted of either:

- the top level comments of an article; or
- the replies to a top level comment

Each `x` entry was preceded by a special token `<ITEM>` and each `y` entry was wrapped with the special tokens `<START>` and `<END>`.

Data were tokenized using the keras `TextVectorizer`, and padded to the maximum sequence length.

4.1.8.1.2 Model training The training was done on a GTX 1070 with 8GB of VRAM. The training took about 1 hour per epoch, and the model was trained for 10 epochs with a batch size of 64. Unfortunately the training history was lost due to a bug in the script, but the training loss was around 4.2x after 10 epochs. Next time I'll definitely use the CSVLogger callback to save the training history, that way graphs can be generated later and the history is guaranteed to be saved. :)

4.1.8.2 Model Metrics The model was evaluated using the perplexity metric. The perplexity was calculated on the training set and validation set, with the model optimizing for the training perplexity. The model loss function was the categorical crossentropy loss function, and perplexity was calculated as `exp(mean(categorical_crossentropy))`. At the end of training, the model trainig perplexity was around 750.

4.1.8.3 Prediction The model outputs both a word-by-word prediction, and a sequence-to-sequence prediction. The word-by-word prediction is done by sampling from the output distribution of the model, the sequence-to-sequence prediction samples the most likely word at each timestep, and if the output is shorter than the minimum length, the model is re-run with the previous output as input until the minimum length is reached.

4.1.8.3.1 Examples Unfortunately most of what the model outputs is somewhat context-aware gibberish, but there are some examples where the model accidentally outputs something that makes a bit of sense. Increasing the models complexity (vocabulary size, adding statefulness, etc.) might improve the results.

The following examples are generated using the default model checkpoint with a temperature of 1.0 for deterministic output (at least for the sequence-to-sequence output, as the word-by-word output is randomly sampled from the timesteps).

prompt: “” (empty string, model latent space)

```
INFO:root:Sequence to sequence result:
 the

INFO:root:Word by word result:
```

```
 want bring such possibly rising offers dc driving leaders influence shut israel ! china
↪  because discussion ones cares easier only left poverty fire cut privacy campaign basic
↪  led heart rosenstein truth officials means english exercise main missile himself clear
↪  amendment taking board economics makes expect federal cars nomination according com
↪  votes l month 60 that goes 2010 pruitt god theres james re catholic scott fbi hasnt back
↪  general red back lobbying 1971 housing chaos .gun criticism press dems choose sent
↪  killed ability douglas find members california fraud never target poor lobbying gop aid
↪  event gain progressive womens head simply 30 rest p either crisis abuse nunes steve
↪  conservative wrote value passed practice ex born businesses us common 2018 f wall
↪  comment 15 meet paul childhood basis legislatures statement old secrets act pretty
↪  because legislatures whole yes land consequences associates far rod destroy goes off
↪  krugman movie deal men de name writing hes story roberts worry half education unless
↪  million hours ending trust unemployment city legislation h end challenge canada income
↪  seem power understand speak waiting uses colleges ideas according step term officers
↪  mexican able displaced interest remain called always done patients editorial knew shut
↪  tweets 1977 costs idea walk fraud than airplanes charge obamas refugees according
↪  experts interests situation became school bigger warfare future energy worst few
↪  religious create fair promise market talks marriage rather pence special what officials
↪  added model start total stage teachers judges vladimir thus a parenting profit term rex
↪  lawmakers crazy decision jews rifle lost reports decades do university millions short
↪  knowing complex accused comey 1979 wife life believe killing imagine syria issues white
↪  re share disaster freedom the officials happens job knowing rule write scandal age
↪  expression happen write fires warming meet question find supporting against soon pass at
↪  2 study hear
```

prompt: `the`

```
INFO:root:Sequence to sequence result:
the christopher the

INFO:root:Word by word result:
the .news green intelligence tariffs changes support daily closely first 1950 estate bill
↪  colleges sean guns focus fired listen trends iii profits conference hitler senior
↪  example telling jefferson crimes potential up firing leading becomes intellectual kim
↪  talks treason connections fired for research television whose secret equal everyone
↪  kushner note workers try progress paid france nearly still guilty terrorism be london
↪  check any dead lives democracy top emmanuel krugman tariffs clearly author family
↪  western vice individual shame guess doctor show the created assault writing transgender
↪  existing pen ms league assad excellent scandal remember tough everyone blame student
↪  affairs waiting dying things beginning otherwise cuts obamacare wonder devices liberal
↪  passed airplanes drug here wait wars .crossword despite minutes share four created raise
↪  there until choices cares count groups reporting low impact program isis knowing win
↪  president inside war congressional intellectual changing fund ? interested win
↪  discrimination there values gave supporters lived hes democrat be his united general
↪  strike donald easily space she building path prices administrations forces intelligence
↪  prove misinformation rise immigrants others gain 25 leaving rosenstein shut stock groups
↪  pay anymore book parkland he chemical shows official responsible nyc progressive enter
↪  further result fair son military worst ?united safety student mention cut message after
↪  effort felt tweets fun wake religious conservative abortion myself served vladimir
↪  kislyak putin staff s reach basis parts foreign actually husband border enter high know
↪  caused deals unfortunately quality himself pruitt daughter this russia bring la ground
↪  near in subject mention investigation lie leaving fox industry done otherwise comments
↪  natural eat sad best prime majority talking true remember fully 2018 levels highest
↪  somehow abuse since page agents later question impossible by judiciary attorneys con
↪  residential trip music car rather this kushner security company then see atlantic nobody
↪  infrastructure times situation please syria drugs cold end google
```

prompt: Why do we spend so much energy on pop-stars when there are bigger issues to worry about?

```
INFO:root:Sequence to sequence result:
why do we spend so much energy on pop - stars when there are bigger issues to worry about ?
↪  neil between 10 season church church season season the
```

```
INFO:root:Word by word result:
why do we spend so much energy on pop - stars when there are bigger issues to worry about ?
↪  wages suggests began 7 jefferson quite spy missile successful speak understand
↪  constitutional millions bit meddling personal responsible finally wish questions being
↪  must promised help targetblankhttpswww months absolutely read medical but deferred thats
↪  losing ignore creating greed man iii today riots seven super garland program faith 1950
↪  justice fired adultery that believe .comey mass hasnt w western proposed world tax
↪  decision perhaps emigration b 1977 cyberwarfare obvious himself c half john .comey
↪  imagine help name stay ill heart moral programs wanted hold russia truly grow sets
↪  anyone due surprised nyt nature people ourselves recently bureau yourself comeys
↪  candidate reduce need worked was voice whole despite although enemy allies provide
↪  period just advisers mistake green 100 re warfare question land george congressional
↪  base status .federal forces harassment working conflicts number sort voter racism
↪  lobbying father correct ties the ryan food study my border elected response roberts
↪  critical salaries course top using barack union 1984 60 awards uber press fail elite
↪  bannon says am millions show subject under hour constitutional office decade divide .gun
↪  loyalty return cars consequences france conditions enter stephen correct product food
↪  sort lack dog yet citizens disorders ignore life member connections riots done total
↪  almost americans battle risk intellectual rid took get ever credit someone speaking
↪  taxation purpose other huge did supreme surprise pro everyone think several finding
↪  choose missing enforcement suffering fire personally teachers far f doesnt reach funny
↪  listen sick got rosenstein old violations benjamin total bush training alternative co
↪  steel al position economy older statements young barriers keeping future easily
↪  misinformation taxes while effect along catholic nations rights considering cnn
↪  parenting constitutional secretary t especially people 1979 killed follow whites final
↪  another shes department nyc
```

The following examples are generated using the model checkpoint with a temperature of 0.8, and a minum length of 50 (the minimum length only applies to the sequence-to-sequence output).

prompt: The biggest issue facing humanity today is

```
INFO:root:Sequence to sequence result:
the biggest issue facing humanity today is editorial author the bureau author the bureau
↪  author the bureau father author the bureau father author the bureau father author author
↪  the bureau father author author author author author the bureau father author author the
↪  bureau father author author author the bureau father author author author author author
↪  author the cities police author author author author the

INFO:root:Word by word result:
```

```
the biggest issue facing humanity today is century lie wife although return details
↪  education world chose replace lives gender doctors - integrity column mother spending
↪  unfortunately days ourselves elections start brain better matters opinion value
↪  judiciary crimes blow 2020 middle why certain v reality donald obamacare piece fuel
↪  football much source russian smart replace disaster mueller mother ny propaganda
↪  something krugman needs supreme asking proposed computers interference creating by
↪  subject area judiciary pruitt violence suffer promised belief save together clintons new
↪  now progress wiretapping forces todays puzzles kislyak integrity biggest sense friends
↪  situation ive air expensive mike amendment wanted said ignore add fury politicians
↪  thomas hillary barriers americas p wouldnt person litigation federal been away loyalty
↪  universities managed fbi vice expect islamic officials .united minority cabinet ensure
↪  approach led senator working sad areas cuomo robert ignorant pro wait cities marine
↪  longer 2010 follow result same relationships expensive near west pelosi missiles
↪  individuals republic saying articles night liberal sanctions going ms fury find de
↪  removed peace deportation pro victory promised republic showed pretty credit asking
↪  propaganda stephanie heres won ivanka police so accused liar meet otherwise taxation
↪  communications 2 highest 3 individuals warren hold hes process planned amount no was
↪  questions americans thomas moral small e choice plans female caption code they
↪  misconduct 1958 they sanctions parenting sides devin shouldnt care disaster required
↪  speaking wiretapping opportunity numbers levels itself ms ready powerful interview which
↪  over jefferson november family safety rising lying meaning amp l my 2017 walk conway
↪  status makes - vote days levels short dangerous tough however fellow pick began simple
↪  changes space serious serious season correct mention doing leadership real leads
↪  residential society bad rifle clear highest party bank some times criminal wasnt outrage
↪  born allow we turned 15 flight replace jr aides created devos isis damage often
```

prompt: While the author describes certain advantages that come from using electric vehicles, I disagree entirely.

```
INFO:root:Sequence to sequence result:
while the author describes certain advantages that come from using electric vehicles , i
↪  disagree entirely . french silence french kushner french french the french the french
↪  the french the french the french the french the french the french macron french the
↪  french macron french macron french the french macron french macron french macron french
↪  the french macron french macron french macron french the french macron french macron
↪  french macron french macron french the

INFO:root:Word by word result:
```

```
while the author describes certain advantages that come from using electric vehicles , i
↪ disagree entirely . muslim education across existing age learn details human residential
↪ crisis avoid protection often politics campaign voter minority housing single words
↪ flight planning once rural mental push works rosenstein airlines network wouldnt
↪ necessary king look back critical speech disease 60 brought led elected instead growing
↪ parents devices islamic wanted found doctor 60 refugees without systems friday 12
↪ warming students id grow added putting challenge had skills tests showed talks articles
↪ integrity become k wanted strong corrupt destroy economic several column supreme lost
↪ program etc waste another required yourself our also paris absolutely markets third vote
↪ thus facing welfare several anymore also personally chinese facing think beginning
↪ tweets votes opposition should even memo biological defense effect have getting king she
↪ respect leave treated de caption apply prison price you knows completely sets abuse
↪ majority opposition year g grow more young purchased changes times changes difficult
↪ expensive nice european possibly account pass seem due thursday source claims rex column
↪ false last speech days cambridge verdicts do rural memo served shutdown programs offered
↪ consider big got ability leadership melania airport morning doctors shouldnt bottom
↪ retirement true board rate search homeland ?the decide do unemployment male hate enough
↪ regardless obama tower property course we friday neil oreilly average caused ! starting
↪ column china short pennsylvania came pass choice final pre marjory t last personal
↪ except ivanka schools approach deserve afford infrastructure already march propaganda
↪ surprise pollution movement sanders played husband 100 watching millions situation
↪ officers lie threat room remember let above work least deferred seeking if 2017 london
↪ complex future third stephanie consequences provide reform long looks city week nuclear
↪ heres until walk 000 happy step say served reporting lawyer con serve criminal purchased
↪ gone sense key
```

4.1.9 Final note

Althoguh the predicted text weren't that great, it seems that the biggest issue with the sequence to sequence model is repetition, especially when you want to generate a specific length of text. I think the best approach to remedy this would be to add statefulness to the model and potentially a beam search. Either way, I have tried a bunch of different model architectures and options and some of them failed to generate anything beyond a bunch of commas, or "the", etc.

5 Assignment 4 - Using finetuned transformers via HuggingFace

5.1 Original Assignment Description

In previous assignments, you've done a lot of model training of various kinds of complexity, such as training document classifiers or RNN language models. This assignment is more like Assignment 1, in that it's about *feature extraction*.

For this assignment, you should use `HuggingFace` to extract information from the *Fake or Real News* dataset that we've worked with previously.

You should write code and documentation which addresses the following tasks:

- Initalize a `HuggingFace` pipeline for emotion classification
- Perform emotion classification for every *headline* in the data
- Assuming the most likely prediction is the correct label, create tables and visualisations which show the following:
  - Distribution of emotions across all of the data
  - Distribution of emotions across *only* the real news
  - Distribution of emotions across *only* the fake news
- Comparing the results, discuss if there are any key differences between the two sets of headlines

5.1.1 Tips

- I recommend using `j-hartmann/emotion-english-distilroberta-base` like we used in class.
- Spend some time thinking about how best to present you results, and how to make your visualisations appealing and readable.
- **MAKE SURE TO UPDATE YOUR README APPROPRIATELY!**

5.2 Assignment 4, Luke Ring

Repository: https://github.com/zeyus/cds-language-exam/tree/main/assignment_4

5.2.1 Contribution

This assignment was completed by me individually and independently, the code contained in this repository is my own work.

5.2.2 Setup

This assignment uses PyTorch and HuggingFace Transformers. Fine tuning was done using CUDA 11.8 on an NVIDIA GeForce GTX 1070 GPU with 8GB VRAM on a system with 25GB RAM.

5.2.2.1 Prerequisites

- Python 3.11

5.2.2.2 Installation Clone the repository:

```
git clone https://github.com/zeyus/cds-language-exam
cd cds-language-exam/assignment_4
```

Install requirements:

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

5.2.3 Usage

The script can be run from the command line as follows:

```
python3 src/ftt.py
```

It is also possible to specify arguments to the script, which can be seen by running:

```
python3 src/ftt.py --help
```

Output:

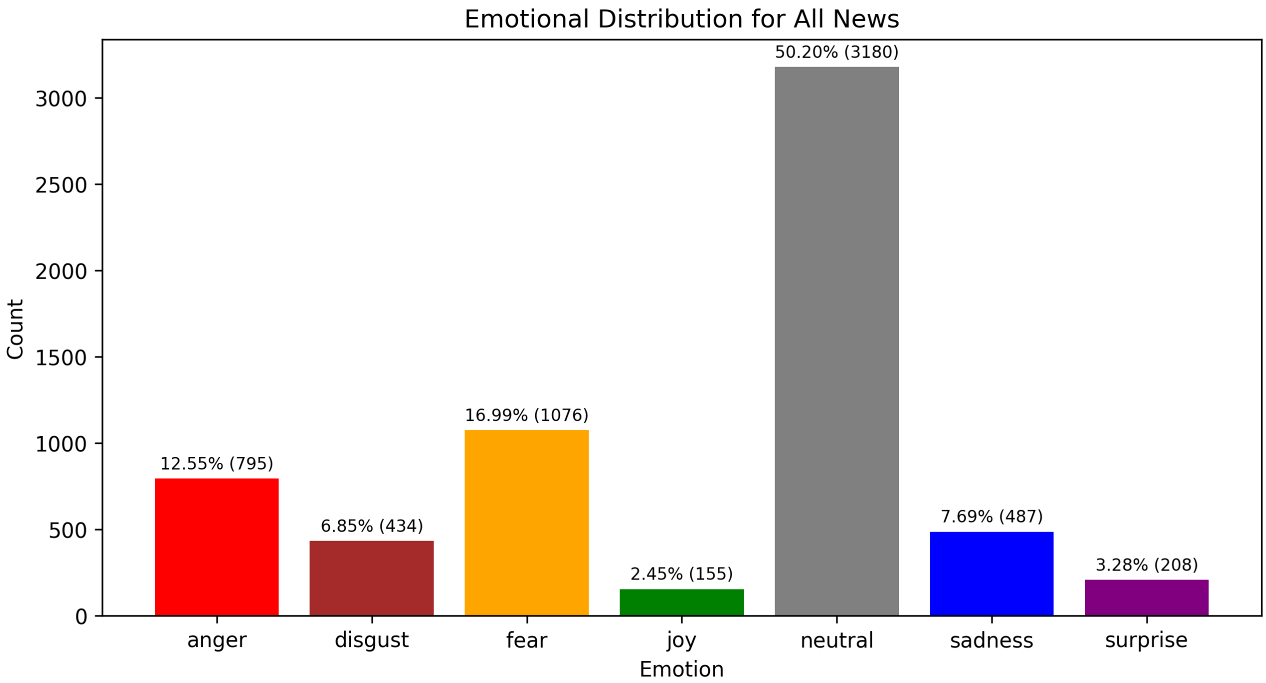
```
usage: ftt.py [-h] [--version] [-o OUTPUT_PATH] [-d DATASET_PATH] [-V]

Text classification CLI

options:
  -h, --help            show this help message and exit
  --version             show program's version number and exit
  -o OUTPUT_PATH, --output-path OUTPUT_PATH
                        Path to save the output, figures, stats, etc. (default: out)
  -d DATASET_PATH, --dataset-path DATASET_PATH
                        Path to the dataset (default: data)
  -V, --visualize-data  Visualize the dataset (default: False)
```

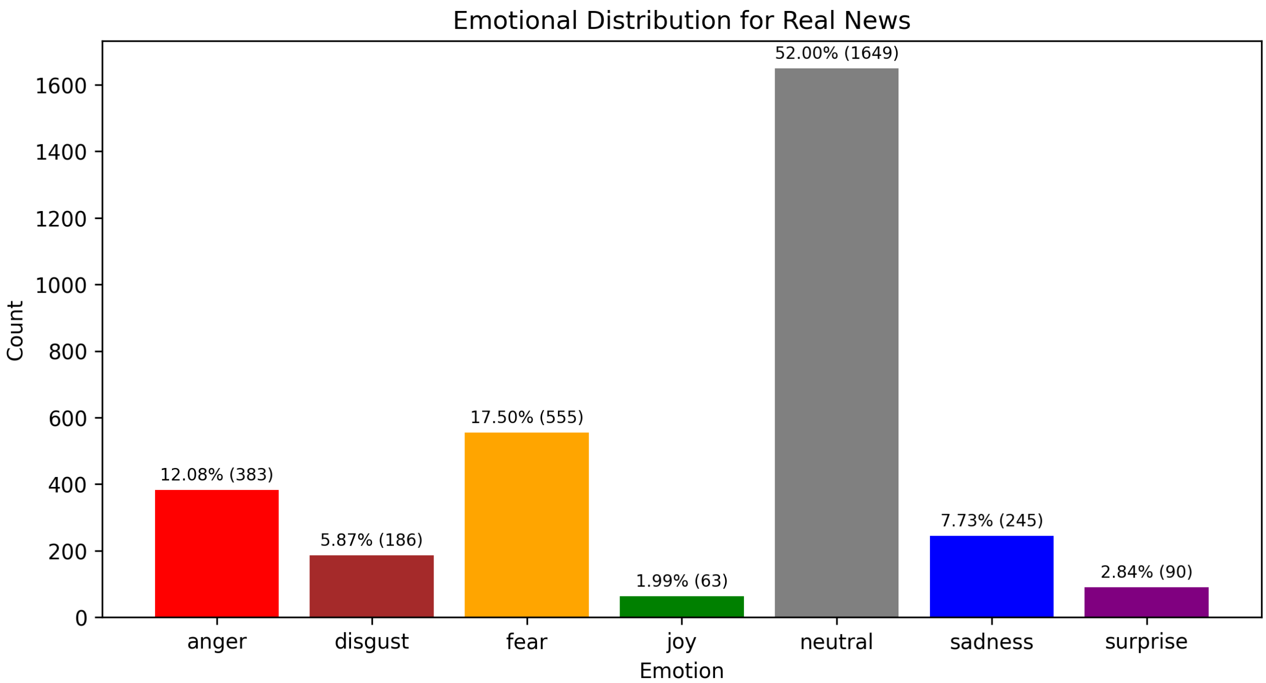
5.2.4 Implementation

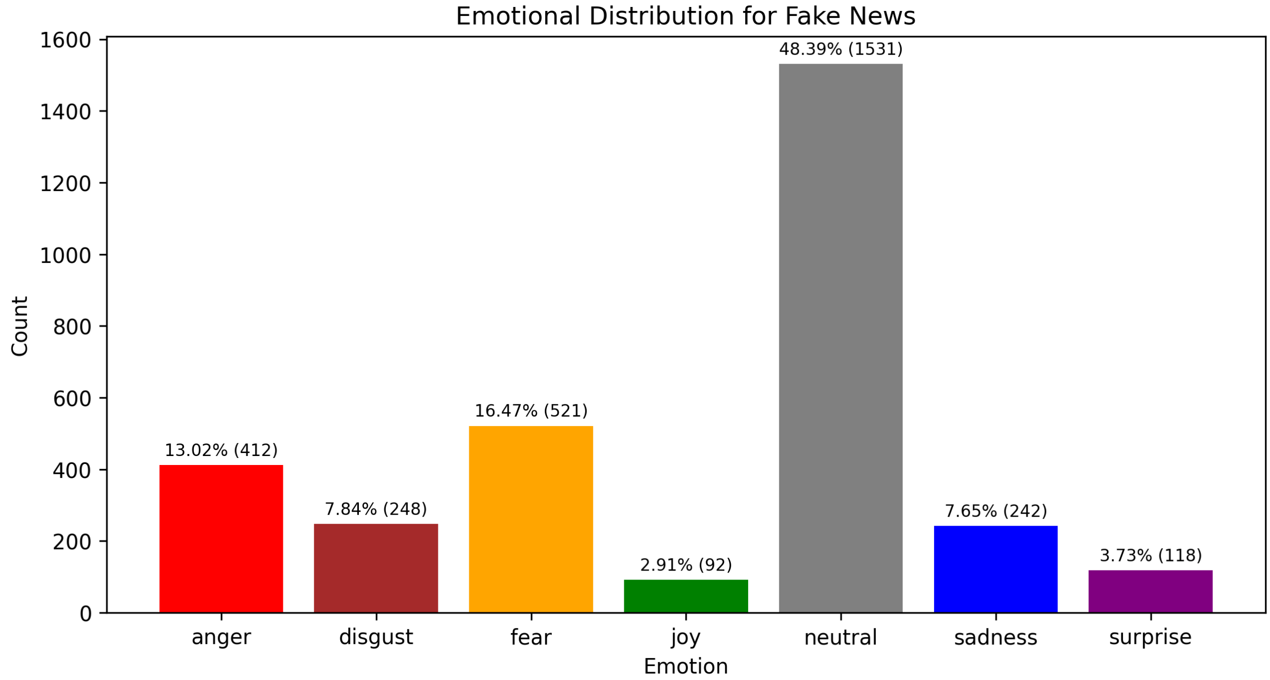
The headlines from the `fake_or_real_news.csv` file are loaded into a `pandas.DataFrame` object. A `Dataset` is created from using the `Dataset.from_pandas` method. The model used is the recommended `j-hartmann/emotion-english-distilroberta-base` model. The model is then used to predict the emotion of each headline in the dataset using an `inference` function that tokenizes the input and returns a `softmax` of the predictions. The predictions are then saved to a `pandas.DataFrame` object and saved to the `out/news_emotions.csv` file.

After the csv file has been created, the script can be run with the `-V` argument to create visualizations of the emotional distribution of the headlines. The visualizations are saved to the `out` directory.

5.2.5 Results







Sample of the predictions (complete predictions can be found in the news_emotions.csv file):

text	emotion	label
You Can Smell Hillary's Fear	fear	FAKE
Watch The Exact Moment Paul Ryan Committed Political Suicide At A Trump Rally (VIDEO)	sadness	FAKE
Kerry to go to Paris in gesture of sympathy	joy	REAL
Bernie supporters on Twitter erupt in anger against the DNC: 'We tried to warn you!'	anger	FAKE
The Battle of New York: Why This Primary Matters	neutral	REAL
Tehran USA	neutral	FAKE
Girl Horrified At What She Watches Boyfriend Do After He Left FaceTime On	fear	FAKE
'Britain's Schindler' Dies at 106	sadness	REAL
Fact check: Trump and Clinton at the 'commander-in-chief' forum	neutral	REAL
Iran reportedly makes new push for uranium concessions in nuclear talks	neutral	REAL

The results look quite good, apart from the third sample "Kerry to go to Paris in gesture of sympathy" which has been clasified as "joy", but in my opinion would be better classified as sadness, although it's easy to imagine that "going to Paris" is usually something associated with holidays and pleasant experiences.

6 Assignment 5: Unsupervised Pre-training Using Flan-T5

6.1 Description

This assignment self-assigned, and is the final assignment for Cultural Data Science - Language Analytics.

For this assignment, I wanted to try and create a way to interface with my notes, in particular I use Obsidian which uses Markdown files for notes. As such, there are many approaches to creating a language model that can be used to glean information from these notes, including using the "summary/summarize" method, although this requires a lot of manual work, or using another network to generate summaries of the notes. Instead, I wanted to see if there was a way to add the information in the notes in an unsupervised way, which is possible using different methods. Taking inspiration from (Suichan Li et al., 2021), I wanted to see if I could create a language model that could be used to interact with my notes. This can easily be updated and potentially integrated directly into Obsidian by creating a plugin to interface with the model.

Suichan Li et al., 2021. Unsupervised Finetuning. arXiv:2110.09510 [cs.CV] https://arxiv.org/abs/2110.09510

6.2 Assignment 5, Luke Ring

Repository: https://github.com/zeyus/cds-language-exam/tree/main/assignment_5

6.2.1 Contribution

This assignment was completed by me individually and independently, the code contained in this repository is my own work.

6.2.2 Setup

This assignment uses PyTorch and HuggingFace Transformers. Fine tuning was done using CUDA 11.8 on an NVIDIA GeForce GTX 1070 GPU with 8GB VRAM on a system with 24GB RAM.

6.2.2.1 Prerequisites

- Python 3.11

6.2.2.2 Installation

Clone the repository:

```
git clone https://github.com/zeyus/cds-language-exam
cd cds-language-exam/assignment_5
```

Install requirements:

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

If you wish to use rouge to evaluate the model, you will need to install the nltk punkt package:

```
python -m nltk.downloader punkt
```

Additionally, apex is required for using the Fused AdamW optimization algorithm (and apex provides additional performance enhancements to the base T5 model), this can be compiled in windows by following my instructions or for other operating systems, please use the official documentation available on the NVIDIA/apex github repository.

6.2.3 Usage

The script can be run from the command line as follows:

```
python src/obsidianlm.py
```

The arguments available can be found by running:

```
python src/obsidianlm.py --help
```

Output:

```
usage: obsidianlm.py [-h] [--version] [-o OUTPUT_PATH] [-b BATCH_SIZE] [-V VAULT_PATH] [-r]
↪  [-e] [-i INFERENCE] [-p PROMPT]

ObsidianLM: Create a model of your brain.

options:
  -h, --help            show this help message and exit
  --version             show program's version number and exit
  -o OUTPUT_PATH, --output-path OUTPUT_PATH
                        Path to save the output, figures, stats, etc. (default: out)
  -b BATCH_SIZE, --batch-size BATCH_SIZE
                        Batch size for training. (default: 2)
  -V VAULT_PATH, --vault-path VAULT_PATH
                        Path to your obsidian vault. (default: vault)
  -r, --rouge           Calculate rouge scores. (default: False)
  -e, --do-eval         Run evaluation during training. (default: False)
  -i INFERENCE, --inference INFERENCE
                        Run a test inference on the specified model checkpoint. (default:
↪  None)
  -p PROMPT, --prompt PROMPT
                        Prompt to use for generation. (default: None)
```

6.2.4 Implementation

6.2.4.1 Model The base model used for this project is the Flan-T5-Base model, it is interesting because it can respond to various prompts and perform different language tasks.

This implementation used the T5ForConditionalGeneration class which includes a language modelling head from the HuggingFace Transformers library.

6.2.4.2 Data In order to adapt the model to my notes, the markdown files are loaded from my Obsidian Notes directory, the text is read in and used as the dataset for the model. Using this data for fine tuning should allow the model to give more relevant results if I want to get summaries of my notes or questions about specific sections in the notes.

6.2.4.2.1 Data Preprocessing As part of the preprocessing steps, the special sentinal tokens are inserted into the text and the corresponding values are added to the labels. This masking is done in a probabilistic way (by default with a 0.15 probability of masking) and sequential token masking is prevented. A simple representation of what the masked input and labels would look like is shown below:

```
Raw input: "The cat sat on the mat."
Masked input: "The <extra_id_1> sat on the <extra_id_2>."
Labels: "<extra_id_1> cat <extra_id_2> mat"
```

The `<extra_id_n>` tokens are the sentinel tokens where the model should fill in the blanks. The labels are the tokens that should be predicted by the model. The `T5Tokenizer` class allows up to 100 sentinel tokens, and the preprocessing steps ensured that there were never more than 100 sentinel tokens in the input.

6.2.4.3 Unsupervised Training and Evaluation The models were trained for up to 4 epochs, and batched with either a batch size of 1 or 4 or 8. The batch size could be 8 when no evaluation was done, 4 when the evaluation dataset was extremely small, otherwise a batch size of 1 due to insufficient GPU memory available for the evaluation step.

Training the model with my notes was done in an unsupervised, programmatic way, where the model by default during training will try to minimize the Cross-Entropy Loss, and the authors describe that one of the advantages of the T5 model is that the same loss function can be used across various tasks[1] . While this means that evaluation isn't strictly necessary, I was interested in seeing the results of different evaluation metrics, and how they compared to the model's loss.

6.2.4.3.1 Evaluation metrics Training runs were done with and without evaluation, evaluation runs were performed for both the ROUGE and TER metrics.

Rouge is a metric that measures the overlap between the model's output and the reference. The higher the score, the more overlap there is between the model's output and the reference, indicating a better match. Rouge can provide various scores, including Rouge-1, Rouge-2, and Rouge-L, which are the unigram, bigram, and longest common subsequence scores respectively. The metric calculation used the Porter stemmer so that word suffixes were ignored when comparing the model's output and the reference, this is done by passing the `use_stemmer=True` argument to the `rouge.compute()` function.

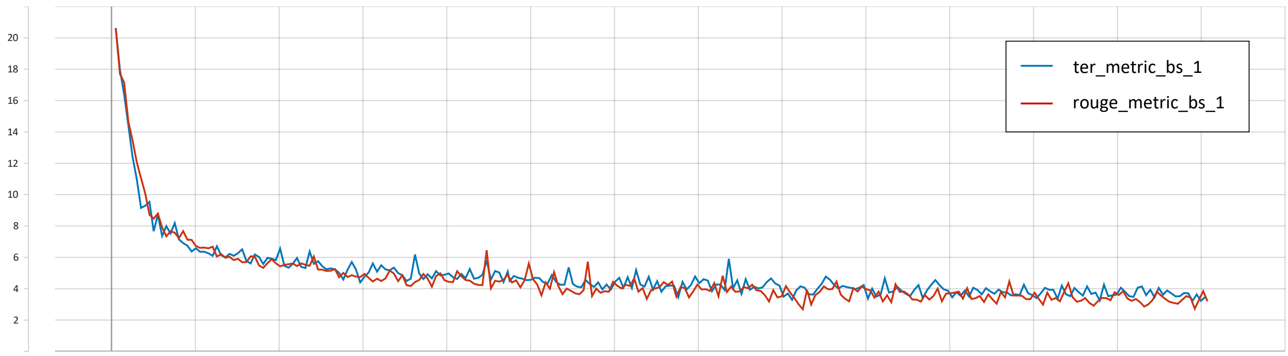
TER measures the number of edits required to transform the model's output to match the reference. The lower the score, the less changes are required to transform the model output to the supplied reference, indicating a better match. The metric calculation was done in a case-insensitive way, and was configured to ignore punctuation and normalize the sentences by passing the `ignore_punctuation=True` and `normalize=True` arguments to the `ter.compute()` function.

Using runs with both of the metrics as well as runs without evaluation, the results can be compared to see if the evaluation metrics are useful for determining the best model. Intuitively it would be easy to assume that the evaluation metrics should help determine the best model, but for tasks like summarization and question answering, there could be many "correct" answers that are not the same as a provided reference which would give the model a lower score despite it answering correctly.

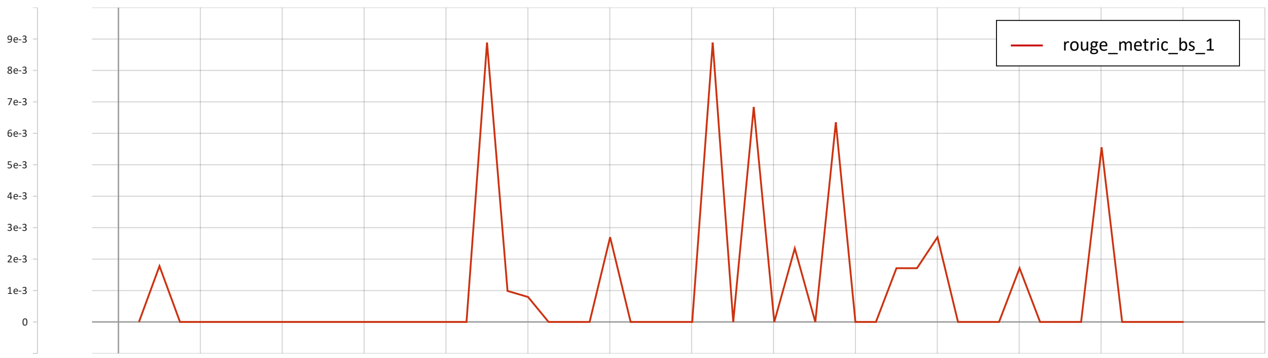
Due to the rouge score calulation taking an extremely long time, rouge was only calculated for one run with a batch size of 1.

6.2.5 Results

6.2.5.1 Batch Size 1 Training loss:



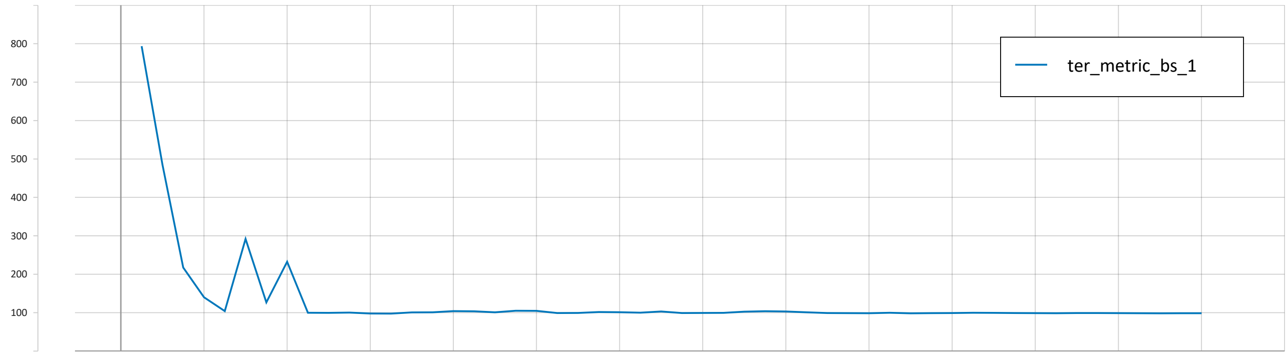
6.2.5.1.1 Rouge Rouge1 Score:



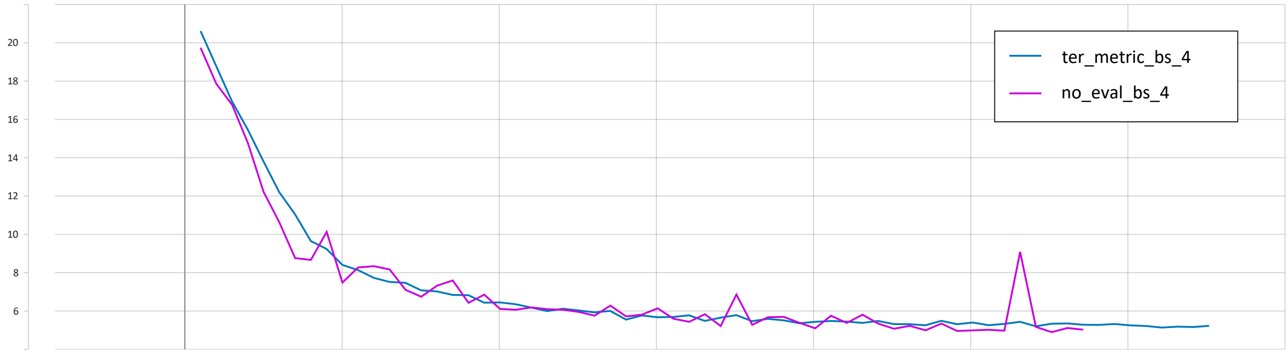
Rouge2 Score:

[1] Raffel et al, 2020: Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. arXiv:1910.10683

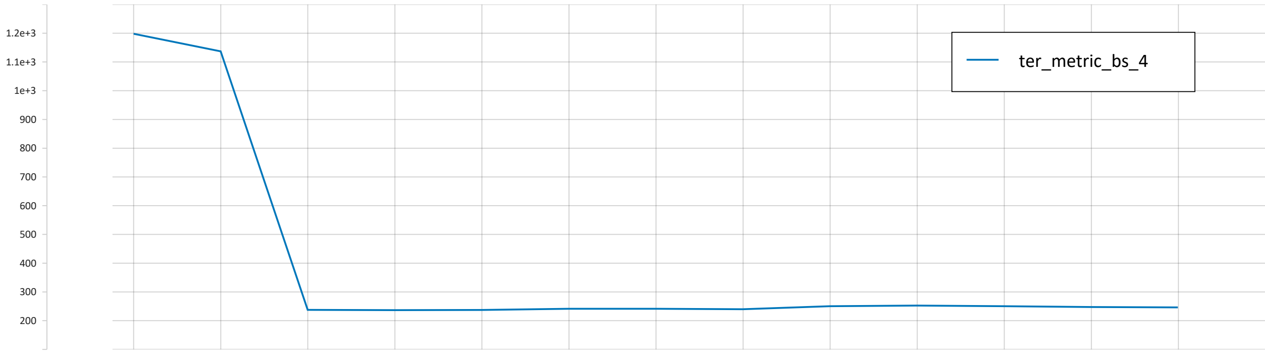
6.2.5.1.2 TER TER Score:



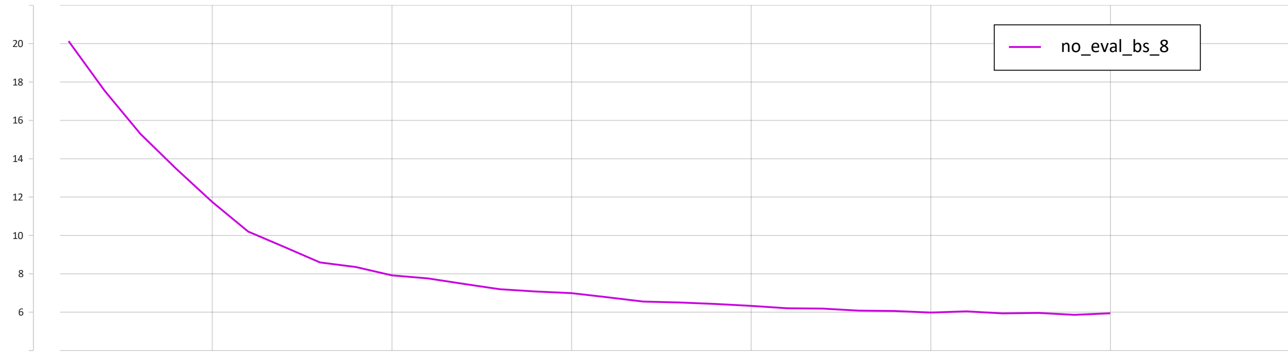
6.2.5.2 Batch Size 4 Training loss:



6.2.5.2.1 TER TER Score:



6.2.5.3 Batch Size 8 Training loss:



6.2.6 Discussion

The graphs above show that the evaluation scores never got to a decent level for either the TER or ROUGE metrics, but in all models, the loss continued to decrease as the training continued. This indicates that the model is learning, but the evaluation metrics are possibly not so useful for determining the best model for this particular case.

That said, all of the models are able to provide answers to some questions as well as summarize some notes, which is great and shows potential for further improvement in the training and inference steps.

6.2.6.1 Inference Examples Examples are generated from the ROUGE evaluation run with a batch size of 1 as well as the TER evaluation run with a batch size of 4 and the no evluation run with a batch size of 8. I will use the last checkpoint form each run to generate the examples, as the "best score" isn't necessarily the best model, and the last checkpoint will have one of the lower loss scores for each batch size.

The examples are generated by running the following:

```
python src/obsidianlm.py -p "<prompt>" -i "<path/to/checkpoint/for/inference>"
```

The format is based on some of the training examples from the FLAN templates.

Prompt 1

> Question: Which models are used in this course? Context: This course wraps up the series of methods courses. We look at modelling from a birds-eye view, introducing advanced concepts and revisiting methods introduced in earlier courses from a comprehensive Bayesian perspective. We introduce causal reasoning using directed acyclic graphs (DAGs), mixture models, Gaussian processes, learn to deal with measurement error and missing data; and we revisit regression modelling, generalized linear models, multilevel modelling, Markov chain Monte Carlo sampling, learning to implement them using probabilistic programming.

Response 1 (batch size 1)

> Bayesiansssssssssssssss.sssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss.sssssssssssssssssssssssssss

Response 1 (batch size 4)

> directed acyclic graphs (DAGs), mixture models, Markov chain Monte Carlo

Response 1 (batch size 8)

> directed acyclic graphs (DAGs), mixture models, Gaussian processes, learn to deal with measurement error and missing data; and we revisit regression modelling, generalized linear models, multilevel modelling, Markov chain Monte Carlo sampling, learning to implement them using probabilistic programming models models models models models

Prompt 2

Question: Which frequencies are used in this experiment? Context: The experimental design will both replicate and augment a prior study[1] that used 4 tones with fixed frequencies (40 Hz, 500 Hz, 5000 Hz and 15,000 Hz) played back to participants for at least 15 seconds each while their eyes are closed. Our experiment will use those same 4 tones, and will add additional tones with smaller frequency differences to attempt to find a minimum frequency difference that is distinguishable from EEG readings. In order to train a machine learning model using a single participant, the range of tones will be played multiple times (within the time constraints) and models will be trained on labelled subsets of the recorded EEG data.

Response 2 (batch size 1)

40 Hz, 500 Hz, 5000 Hz and 15,000 Hz to. to. to. to. . . . .

Response 2 (batch size 4)

40 Hz, 500 Hz, 5000 Hz and 15,000 Hz

Response 2 (batch size 8)

fixed frequencies (40 Hz, 500 Hz, 5000 Hz and 15,000 Hz) played back to participants for at least 15 seconds each while their eyes are closed. Our experiment will use those same 4 tones, and will add additional tones with smaller frequency differences to attempt to find a minimum frequency difference that is distinguishable from EEG readings. In order to train a machine learning model using a single participant, the range of tones will be played multiple times (within the time constraints) and models will be trained on labelled subsets of the recorded EEG data.„

Prompt 3

Write a question you would see in a school textbook.

Response 3 (batch size 1)

(no output)

Response 3 (batch size 4)

Which of the following statements is true?

Response 3 (batch size 8)

Which of the following is a characteristic characteristic of a living thing?. . . . .

Prompt 4

Generate a context and a hypothesis.

Response 4 (batch size 1)

Context Context Context Context Context Context Context Context Context Context Context Context Context Context Context Context Context Con Context Con

Response 4 (batch size 4)

Context: This is a list of the United States Air Force bases in the United States of America. The United States Air Force base in the United States of America is located in Washington, D.C. The United States Air Force base in Washington, D.C. is located in Washington, D.C. Hypothesis: The United States Air Force base in Washington, D.C. is located in Washington, D.C.: The United States Air Force base in Washington, D.C. is located in Washington, D.C.: The United States Air Force base in Washington, D.C. is located in Washington, D.C.: The United States Air Force base in Washington, D.C. is located in Washington, D.C.: The United States Air Force base in Washington, D.C. is located in Washington, D.C. Hypothesis: The United States Air Force base in Washington, D.C. is located in Washington, D.C.: The United States Air Force base in Washington, D.C. is located in Washington, D.C.: The United States Air Force base in Washington, D.C. is located in Washington, D.C.: The United States Air Force base in Washington, D.C. is located in Washington, D.C.: The United States Air Force base in Washington, D.C. is located in Washington, D.C. Hypothesis: The United States Air Force base in Washington, D.C. is located in Washington, D.C.: The United States Air Force base in Washington, D.C. is located in Washington, D.C.: The United States Air Force base in Washington, D.C. is located in Washington, D.C.: The United States Air Force base in Washington, D.C. is located in Washington, D.C.: The United States Air Force base in Washington, D.C. is located in Washington, D.C.: The United States Air Force base in Washington, D.C. is located in Washington, D.C.: The United States Air Force base in Washington, D.C. is located in Washington, D.C.: The United States Air Force base in Washington, D.C. is located in Washington, D.C.: The United States Air Force base in Washington,

Response 4 (batch size 8)

Context: How to prepare for a weddingbr>Determine what type of wedding you are planning to have. If you are planning on having a wedding in the near future, you may want to consider having a wedding in the near future. Hypothesis: Weddings in the near future are more likely to have a wedding in the near future.t weddings in the near future are more likely to have a wedding in the near future.t weddings in the near future are more likely to have a wedding in the near future.t weddings in the near future are more likely to have a wedding in the near future.t weddings in the near future are more likely to have a wedding in the near future.t weddings in the near future are more likely to have a wedding in the near future.t weddings in the near future are more likely to have a wedding in the near future.t weddings in the near future are more likely to have a wedding in

> the near future.t weddings in the near future are more likely to have a wedding in the near future.t weddings in the near future are more likely to have a wedding in the near future.t weddings in the near future are more likely to have a wedding in the near future.t weddings in the near future are more likely to have a wedding in the near future.t weddings in the near future are more likely to have a wedding in the near future.t weddings in the near future are more likely to have a wedding in the near future.t weddings in the near future are more likely to have a wedding in the near future.t weddings in the near future are more likely to have a wedding in the near future.t weddings in the near future are more likely to have a wedding in the near future.t weddings in the near future are more likely to have a wedding in the near future.t weddings in the near future are more likely to have a wedding in the near future.t weddings in the near future are more likely to have a wedding in the near future.t weddings in the near future are more likely to have a wedding in the near future.t weddings in the near future are more likely to have

Prompt 5

> Complete the following sentence.
>
> Cognitive Science is a

Response 5 (batch size 1)

> :...

Response 5 (batch size 4)

> scientific field...

Response 5 (batch size 8)

> scientific discipline..

6.2.6.2 Discussion of Inference Examples The above examples show some quite good output, but also indicate a few issues, including the model with batch size of 1 seems to be corrupted, and all examples seem to have trailing or repetitive output. Although I am using the model's tokenizer to decode the output, it could be that the end of string token is either not generated correctly, or the decoding also includes tokens after the EOS token. ### Future Improvements

There are many ways this problem could be approached, including trying a different model completely, but assuming that Flan-T5-Base is used, there are a few ways that the model could be improved.

- Data preprocessing: Instead of treating each file individually, the files could be concatenated first into a single long string, and then chunked into the max token length (512 for Flan-T5-Base). This would allow the model to learn from the context of the entire vault, instead of just the individual files.
- Labelled data: Manual (or model-created, curated) summaries could be added to the data as a way to improve the model's ability to summarize the notes, specifically in the context of my notes from university.
- specific tasks: The model could be trained on specific tasks, such as summarization, or question answering, which would allow the model to be more specific in its responses, rather than filling in the blanks.

6.2.7 Conclusion

Using a Flan-T5 based model and fine tuning it on personalized data seems like a promising option to help with reviewing and finding information in notes, emails, etc. With little effort, the context could be provided as part of a plugin, meaning you could use the model to interact directly with the application.

The whole process has given me a much deeper insight into language modelling and what is involved, including performance optimization and memory management, as well as programatically creating the sentinel tokens, and how different parameters such as attention masking are used.