

[Dashboard](#) / [My courses](#) / [ZZEN9444-6275_00043](#) / [Assessment 3: Writing Neural Networks - Language processing \(H522\)](#)


Assessment 3: Writing Neural Networks - Language processing (H522)



Upload your third assessment here.

[My Submissions](#)

[Submit Part 1 - written report PDF](#)[Submit Part 2 - savedModel.pth](#)[Submit Part 3 - Python code student.py](#)

Title	Start Date	Due Date	Post Date	Marks Available
 Assessment 3: Writing Neural Networks - Language processing (H522) - Submit Part 1 - written report PDF	22 Aug 2022 - 00:00	11 Oct 2022 - 16:00	18 Oct 2022 - 16:00	30

TYPE	WEIGHT	DUE	EXPECTED TIME
------	--------	-----	---------------

Writing Neural Networks - Language processing		Tuesday Week 7, 5pm (11 Oct 2022)	4-10 hours
--	--	--------------------------------------	------------

Assessment instructions

What you need

You will need [PyTorch](#) installed on your computer.

Download the archive a3.zip into your own file space and unzip it:

← Assessment 3 files

This should create an a3 directory containing the files:

- main file a3main.py
- configuration file config.py
- skeleton file student.py
- data file train.json

Instructions

For this assignment, you will be writing a Pytorch program that learns to read business reviews in text format and predict a rating (positive or negative) associated with each review, as well as a business category (0=Restaurants, 1=Shopping, 2=Home Services, 3=Health & Medical, 4=Automotive). You will also write a brief report in which you outline how your program works and explain your design decisions. This project covers skills and topics from Weeks 1-4.

In your written report, briefly discuss:

- How you selected your architecture, algorithms, and enhancements.
- How you chose your cost function and optimiser.
- How you determined the dimensionality of the word vectors, metaparameter values, and any data preprocessing that you did or did not use.
- How you employed the validation set, and any additional steps to avoid overfitting not otherwise discussed.

Your task is to complete the file student.py in such a way that it can be run in conjunction with a3main.py by typing:

```
python3 a3main.py
```

You must NOT modify a3main.py in any way. You should ONLY modify student.py (If you wish, you can modify config.py in order to switch between CPU and GPU usage). The provided file a3main.py handles the following:

- Loading the data from train.json
- Splitting the data into training and validation sets (in the ratio specified by trainValSplit)
- Data Processing: strings are converted to lower case, and lengths of the reviews are calculated and added to the dataset (this allows for dynamic padding). You can optionally add your own preprocessing, postprocessing and stop_words (Note that none of this is necessarily required, but it is possible).
- Vectorization, using torchtext GloVe vectors 6B.
- Batching, using the BucketIterator() provided by torchtext so as to batch together reviews of similar length. This is not necessary for accuracy but will speed up training since the total sequence length can be reduced for some batches.

You should aim to keep your code backend-agnostic in the sense that it can run on either a CPU or GPU. This is generally achieved using the .to(device) function. If you do not have access to a GPU, you should at least ensure that your code runs correctly on a CPU.

Please take some time to read through a3main.py and understand what it does.

You will be penalised 5% per day of the marks available for this assessment task if you submit it after the due date, unless you have an approved extension through Special Consideration.

create additional variables, functions, classes, etc., so long as your code runs correctly with `a3main.py` unmodified, and you are only using the approved packages. You must adhere to these constraints:

- Your model must be defined in a class named `network`.
- The `savedModel.pth` file you submit must be generated by the `student.py` file you submit.
- Your submission (including `savedModel.pth`) must be under 50MB and you cannot load any external assets in the `network` class.
- While you may train on a GPU, you must ensure your model is able to be evaluated on a CPU.
- You are restricted to using GloVe vectors 6B, but you are free to specify the value of `dim` (50, 100, 200 or 300). The GloVe vectors are stored (automatically) in a subdirectory called `.vector_cache`

You must ensure that we can load your code and test it. This will involve importing your `student.py` file, creating an instance of your `network` class, restoring the parameters from your `savedModel.pth`, loading our own test dataset, processing according to what you specified in your `student.py` file, and calculating accuracy and score.

You may NOT download or load data other than what we have provided. If we find your submitted model has been trained on external data you will receive zero marks for the assignment.

Approved Packages:

Here is a list of approved packages. If there is some other package you would like to use, let us know and we will consider adding it to the list.

- `json`
- `math`
- `mlflow`
- `nltk*`
- `numpy`
- `os`
- `pandas`
- `random`
- `re`
- `sklearn`
- `spacy*`
- `string`
- `sys`
- `time`
- `torch`
- `torchtext**`
- `unicodedata`

* Note that, for packages like `nltk` and `spacy`, you are only allowed to use methods which do not try to download additional data at runtime from external sources. For example, if you used them to get an initial list of stop words, you should include the list explicitly in your code rather than downloading it again at run-time.

** Note: You will need to install `torchtext` 0.10 (not 0.11 or 0.12) in order to use the required features in `torchtext.legacy`.

General advice:

- Use the variable `trainValSplit` to help you make design decisions aimed to avoid overfitting to the training data. At the very end, you may wish to re-train using the entire training set.
- Try to be methodical in your development. Blindly modifying code, looking at the output, then modifying again can cause you go around in circles. A better approach is to keep a record of what you have tried, and what outcome you observed. Decide on a hypothesis you want to test, run an experiment and record the result. Then move on to the next idea.
- Do Not leave this assignment to the last minute. Get started early.

Common questions:

Can I train on the full dataset if I find it?

No. You should NOT attempt to reconstruct the test set by searching the Internet. We will retrain a random selection of submissions, as well as those achieving high accuracy. If your code attempts to search or load external assets, or we find a mismatch between your submitted code and saved model, you will receive zero marks.

Can we assume you will call `net.eval()` on our model prior to testing?

Yes.

Can we assume a max length on the reviews?

No. But nothing will be significantly longer than what is present in the training set.

Presentation style/format

You must submit three files to Turnitin below:

1. written report saved as a .pdf
2. trained model saved Model.pth
3. Python code student.py

How to submit

Submit all three files via Turnitin on this page. You may only submit a single file to each Turnitin part - use the tabs at the top of this page to navigate to each submission area for each part.

You may re-submit multiple times up to the due date. If you submit for the first time after the due date, you may only submit once (if you have submitted before the due date, you will be unable to re-submit after the due date). Normal late penalties apply for submission after the due date.

Additional information may be found in the [Ed discussion board](#) and will be considered as part of the specification for the project. You should check that page regularly.

Marking and feedback

After submissions have closed, your code will be run on a holdout test set (i.e. a set of reviews and ratings that we do not make available to you, but which we will use to test your model). Marks will be allocated as follows:

- 14 marks for your written report
- 2 marks for coding style and comments
- 14 marks based on performance on the (unseen) test set

The performance mark will be a function of the Weighted score, which is:

$$(1.0 \times \text{Percentage with correct rating and correct category}) \\ + (0.5 \times \text{Percentage with correct category but incorrect rating}) \\ + (0.1 \times \text{Percentage with correct rating but incorrect category})$$

MARKING CRITERIA	UNSATISFACTORY - ZERO MARKS	SATISFACTORY - HALF MARKS	HIGH DISTINCTION - FULL MARKS
Selection of architecture, algorithms and enhancements 4 marks	The overall approach to select architecture, algorithms and enhancements is incorrect.	Choice of architecture, algorithms and enhancements are reasonable, but not clearly explained.	Clear justification for architecture, algorithms, and enhancements selected, including consideration of possible alternatives.
Selection of cost function and optimiser 3 marks	The overall approach to select cost function and optimiser is incorrect.	Choice of cost function and optimiser is reasonable, but not clearly explained.	Clear justification for choice of cost function and optimiser, including the consideration of possible alternatives.

other metaparameters, and data preprocessing

4 marks

vectors and other metaparameters is incorrect. Preprocessing (or lack thereof) is not clearly explained.

metaparameters, and any preprocessing steps used are reasonable, but not clearly explained.

vectors with clear explanation. Choice of metaparameter values is adequately explained. Clear explanation for why data preprocessing steps were used or not used.

Use of validation set and any additional steps taken to avoid overfitting and improve generalisation (apart from the criteria listed above)

3 marks

Incorrect justification of use of validation set. Inadequate steps taken to avoid overfitting and improve generalisation.

Validation set and any additional steps taken to avoid overfitting and improve generalisation were used correctly, but not clearly explained.

Validation set was used correctly with clear explanation of how it was used. Any additional steps taken to avoid overfitting are either explained here, or included in the criteria listed above.

Style and commenting of Python code

2 marks

Code is difficult to read and not adequately commented.

Code is either difficult to read, or not adequately commented.

Code is well-structured, readable, and adequately commented.

Performance on the (unseen) test set

14 marks

Code performs poorly on test set.

Code achieves moderate score on test set.

Code achieves high score on test set.

 Refresh Submissions

Submission Title

Turnitin Paper ID

Submitted

Grade

Overall Grade

--

--

--

--

--

--

Submit Paper 

--

--