

MSCI 641 Assignments: Submission instructions

This document provides detailed submission instructions for the course assignments. Please follow them carefully.

All assignments are due by **11:59 pm EST** on the due day. You must use your computational resources for these assignments.

General instructions:

1. All assignments should be submitted on **mse-msci-641.uwaterloo.ca** server (an account has been created for everyone).
2. SSH to **mse-msci-641.uwaterloo.ca** server using Terminal (Mac, Linux) or SSH Chrome extension (Windows). Use the following credentials:
 - UserID: **<your-UW-username>**
 - Password: **MSCI641@2024** (you'll be prompted to change your password upon first login).
3. Once you log into the server, you can access the template submission folders inside `/DATA1/<your_username>/assignments/`
4. For every assignment, you'll need to update the placeholder content of python files (in the template folder) with your own implementation.
5. The following files have been created inside each template submission folder (there may be more depending on the assignment):
 - a. A `main.py` file that acts as the driver script for that submission.
 - b. A `README.md` file that will serve as your report.
 - c. A `'data'` which contains all the expected files required for that assignment. **DO NOT** make any other subfolders inside the `'data'` folder.
 - d. If you need to use the outputs from your previous assignments, you can easily do so by accessing the earlier submission's data folder. For instance, if you want to use the word2vec model trained in A3 to initialize word embeddings in A4, you can use the `'w2v.model'` (the expected output file for A3) inside `/DATA1/<your_username>/assignments/a3/data/`
5. The command executed by the evaluation system would look something like ``python3 a1/main.py [arg]``. You **must not** change the name of any template file. For instance, do not rename it to `'Main.py'`, `'main.ipynb'`, `'main.txt'`, etc. or the submission won't be evaluated.
6. Make sure that your submission is executable by the test python environment. You can activate the test env by executing ``source /home/gsahu/miniconda3/envs/msci641_env/bin/activate /home/gsahu/miniconda3/envs/msci641_env/`` (**Note:** You may need to run ``bash`` upon login or ``source`` program might be unrecognized)

7. Paths to required data files should **not be** hardcoded. Use a command-line argument, instead. More specific instructions are posted below.
8. All driver scripts should accept **only one** command-line argument. Refer to the respective assignment's section for any exceptions or code requirements.
9. More specific instructions are posted in the respective assignment's guidelines below. **Note** that all the expected output files should be saved to the `data` folder of that assignment. For example, upon running the command `python3 a3/main.py [arg]` the expected output file `w2v.model` should be saved inside the `a3/data/` folder.
10. Finally, stay away from plagiarism.

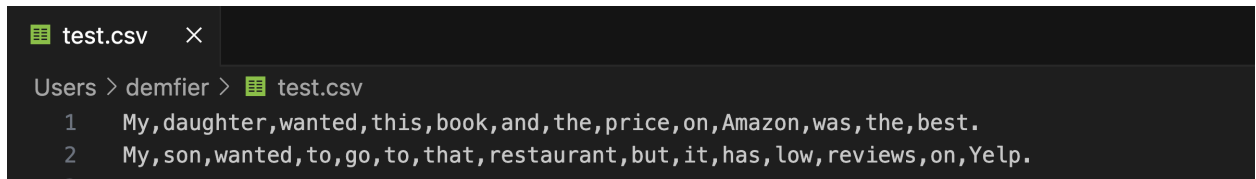
Kindly make sure that you follow the instructions posted in this document when submitting an assignment.

Assignment 0.

1. Install the latest version of all the mentioned Python3 modules. We recommend using Visual Studio Code.

Assignment 1.

1. **Command-line argument:** Path to the folder where `pos.txt` and `neg.txt` reside.
2. You must implement the required techniques in core Python. For instance, you can use Python's `random` module to create the train/val/test split and can read the `.txt` files in Python (using the in-built `open` function) instead of using `pandas`.
3. The tokenized sentences should be stored in a CSV file called `out.csv` where each line contains the tokens for a sentence, separated by a `,`. The contents of your file should look like as follows:



```
test.csv x
Users > demfier > test.csv
1 My,daughter,wanted,this,book,and,the,price,on,Amazon,was,the,best.
2 My,son,wanted,to,go,to,that,restaurant,but,it,has,low,reviews,on,Yelp.
```

4. Although not required, you can remove any additional punctuation marks such as `'`. Additionally, it doesn't matter if you store those punctuations as a separate token or concatenate it with the previous word.
5. **Expected output files:**
 1. `out.csv`: tokenized sentences w/ stopwords
 2. `train.csv`: training set w/ stopwords
 3. `val.csv`: validation set w/ stopwords
 4. `test.csv`: test set w/ stopwords
 5. `out_ns.csv`: tokenized sentences w/o stopwords
 6. `train_ns.csv`: training set w/o stopwords
 7. `val_ns.csv`: validation set w/o stopwords

8. test_ns.csv: test set w/o stopwords
6. It is recommended that you generate the labels as well in this assignment even though it is not mandatory.

Assignment 2.

1. **Command-line argument:** Path to the folder containing the split from A1.
2. DO NOT tune your models on the **test** set and store the trained models as a pickle (.pkl) file.
3. Include the table with the results and answers to questions in README.md. Do not exceed the word limit (5-6 sentences) per answer.
4. **Output files (must be included in your submission):**
 - a. mnb_uni.pkl: Classifier for unigrams w/ stopwords
 - b. mnb_bi.pkl: Classifier for bigrams w/stopwords
 - c. mnb_uni_bi.pkl: Classifier for unigrams+bigrams w/ stopwords
 - d. mnb_uni_ns.pkl: Classifier for unigrams w/o stopwords
 - e. mnb_bi_ns.pkl: Classifier for bigrams w/o stopwords
 - f. mnb_uni_bi_ns.pkl: Classifier for unigrams+bigrams w/o stopwords
 - g. Store any other files you may need inside the `data` folder
5. **Code requirement:**
 - a. In addition to the general instructions posted at the beginning of the document, you should also submit an `inference.py` script, which classifies a given sentence into a positive/negative class. It should accept the **two** command-line arguments described below:
 - i. arg1: Path to a .txt file, which contains some sentences compiled for evaluation. There will be one sentence per line.
 - ii. arg2: Type of classifier to use. Its value will be one of the following – mnb_uni, mnb_bi, mnb_uni_bi, mnb_uni_ns, mnb_bi_ns, mnb_uni_bi_ns. **Example:** mnb_uni indicates that the MultinomialNB classifier trained on unigram features with stopwords should be selected for classifying sentences in the aforementioned .txt file.

Assignment 3.

1. **Command-line argument:** Path to the folder where pos.txt and neg.txt reside
2. Write the report (README.md) as succinctly as possible and **DO NOT** overshoot the word limit (5-6 sentences).
3. **Output files (must be included in your submission):**
 - a. w2v.model: Word2vec model trained on the entire Amazon corpus (pos.txt + neg.txt)
4. **Code requirement:**

- a. In addition to the general instructions posted at the beginning of the document, you should also submit an ``inference.py`` script, which generates the top-20 most similar words for a given word. It should accept **one** command-line argument described below.
 - i. `arg1`: Path to a .txt file, which contains some words compiled for evaluation. There will be one word per line.

Assignment 4.

1. **Command-line argument:** Path to the folder containing the split from A1
2. DO NOT tune your models on the **test** set.
3. You need to submit only **one** model that is trained on either with or without stopwords version (whichever gives you better performance).
4. Write the report (README.md) as succinctly as possible and **DO NOT** overshoot the word limit.
5. **Output files (must be included in your submission):**
 - a. `nn_relu.model`: Classifier w/ ReLU activation
 - b. `nn_sigmoid.model`: Classifier w/ Sigmoid activation
 - c. `nn_tanh.model`: Classifier w/ Tanh activation
6. **Code requirement:**
 - a. In addition to the general instructions posted at the beginning of the document, you should also submit an ``inference.py`` script, which classifies a given sentence into a positive/negative class. It should accept the **two** command-line arguments described below.
 - i. `arg1`: Path to a .txt file, which contains some sentences compiled for evaluation. There will be one sentence per line.
 - ii. `arg2`: Type of classifier to use. Its value will be one of the following – relu, sigmoid, tanh. **Example:** relu indicates that the neural network with ReLU activation should be selected for classifying sentences in the aforementioned .txt file.