# DSCI 572 Lab 4: RNNs for sentiment analysis

## Assignment Topics

- Train a deep learning system for real-world task
- Recurrent Neural Networks
- RNN, LSTM, GRU

## Software Requirements

- Python (>=3.6)
- PyTorch (>=1.7.1)
- Jupyter (latest)

# Getting Started

In [9]:
```
# all the necessary imports
import torch
from torch.utils.data import Dataset
import torch.nn as nn
from torch import optim
import torch
import pandas as pd
import numpy as np
import spacy
```

# Tidy Submission

rubric={mechanics:4}

To get the marks for tidy submission:

- Submit the assignment by filling in this jupyter notebook with your answers embedded
- Submit a PDF version of your Jupyter notebook.
- Be sure to follow the general lab instructions

# Assignment 1: Kaggle competition on sentiment analysis

(This assignment is based **heavily** on the work of the 2022 DSIC572 teaching team. Thank you Chiyu, Peter and Muhammad!)

In this exercise, you'll be compeating for the best performance in sentiment analysis of Yelp reviews. Remember, this is just a game. Don't put too much pressure on yourself, but do try to come up with a fully functional system for sentiment analysis using the approaches which we've explored in the course. The top-2 submissions will be awarded extra points.

## Yelp sentiment analysis

Millions of people share a great number of reviews about businesses on Yelp.com and Yelp mobile app everyday. We used Yelp APIs (application programming interface) to collect over 35,000 reviews of 1,000 restaurants in New York City. We split this dataset into 90% TRAIN set (28,000 reviews), 10% DEV set (3,500 reviews), and 10% TEST set (3,500 reviews).

The data-points in each dataset contain both an input review and a corresponding label ranging from 1 star (the worst reviews) to 5 stars (the best reviews). This table shows the class distribution of TRAIN and DEV sets.

| Rating | # of Train reviews | # of Dev reviews |
|---|---|---|
| 1star | 5,619 | 683 |
| 2star | 5,616 | 677 |
| 3star | 5,583 | 713 |
| 4star | 5,532 | 733 |
| 5star | 5,650 | 694 |

In directory `data` (which you can find in your DSCI572 student repo `labs/Lab4/data`), we provide the `TRAIN` (`train.tsv`) and `DEV` (`val.tsv`) sets with the corresponding labels for your system development. Please use the TRAIN and DEV sets to develop a classification system for this task. You can use any model we've covered in the course (e.g., linear regression, feed-forward neural network, RNN, GRUs, LSTM) or some machine learning model which we didn't cover. However, your solution has to contain a non-trivial machine learning component.

We also provide the inputs for the `TEST` split. You will use your best trained model to predict the labels of **`TEST` reviews** and submit your predictions to a Kaggle competition.

**The performance of your submitted systems will be evaluated on predictions of rating labels for reviews in TEST set. Macro Averaged F-score will be the evaluation metric.**

Your mark is based on the quality of your work:

- Does your code run?

- Is the code neat, commented and easy to follow?
- Is the machine learning strategy appropriate for this task?
- Are the hyper-parameters tuned? Are you exploring the space of hyper-parameters adequately?

Moreover, particularly interesting and thorough work including thorough analysis of the results will be rewarded. As mentioned above, we will also award a few extra points to the top-2 systems submitted to the Kaggle competition.

Gentle reminder of the rubrics that will be relevant:

Accuracy
Quality
Reasoning
Mechanics

## 1.1 Yelp review rating prediction

rubric={accuracy:10}

### Development Instructions

**1. Data and preprocessing**

The Yelp reviews are untokenized. You should first select a tokenizer for your system (e.g., SpaCy English model, whitespace tokenizer, NLTK word tokenizer, or other). The choice of tokenizer can influence your final performance, so it can be a good idea to try a few different ones and select the best system based on the macro F1 score on the development set.

You should use pandas to read the train, development and test data.

If you use pytorch for this assignment, you should also numericalize your data and organize it into mini-batches.

In [8]:
```
# your data code here
train = pd.read_csv('data/yelp_review/train.tsv', sep = '\t')
dev = pd.read_csv('data/yelp_review/val.tsv', sep = '\t')
test = pd.read_csv('data/yelp_review/test.tsv', sep = '\t')
```

**2. Model selection and hyper-parameter tuning. You need to select the architecture you want to use. You may need to search the optimal hyper-parameter set to improve your model performance.**

You might want to explore a few different architectures for the sentiment analysis task. One suggestion is to use some sort of RNN sequence classifier, but many other architectures will also be possible and can give you nice performance. If you decide to

implement a neural model, Miikka recommends that you establish a baseline first using a traditional feature-based model like logistic regression.

There are many ways to improve model performance:

- You might want to use pretrained embeddings for this task (e.g., google news word2vec, GloVe, ELMo) instead of randomly initialized embeddings
- You might explore various feature extraction approaches like CountVectorizer and TfidfVectorizer
- You can adjust your vocabulary size (by dropping very frequent or infrequent items)
- If you're using a neural model, you can adjust the number of layers and the size of hidden layers
- You can try out different flavors of neural models (RNN, LSTM, GRU)
- You can apply regularization, e.g. dropout
- Changing training procedure, such as number of epochs, learning rate (with or without scheduling), adding regularization and momentum (or Adam).
- You may find some novel ideas in the state-of-the-art NLP systems here.

You can evaluate the macro averaged F1 score (as compute by sklearn functions) to monitor the impact of various design choices and hyperparameters.

Hint: Due to the potentially intensive computational resource requirements, we suggest that you run your experiments on Google Colab. However, this is not required.

Please read `Colab instructions` for more information on Colab

In [ ]:
```
# your model code here
```

**3. When you get your best model on DEV set, you will use this model to predict the labels of `TEST set` and submit your prediction.**

In [ ]:
```
# your prediction code here
```

**4. For submission of predictions.**

## Offline validation

1. In directory `./data/yelp_review/` , `EXAMPLE_GOLD.txt` and `EXAMPLE_PRED.txt` are examples of gold and prediction files which can be used with the `Scorer.py` provided (description below). Your submission should have excactly the same structure as **`EXAMPLE_PRED.txt`** (i.e., each line contains one predication label without header of the column.) This is important.

2. `./data/yelp_review/Scorer.py`

The scoring script ( `Scorer.py` ) is provided at the root directory of the released data. `Scorer.py` is a python script that will take in two text files containing true labels and predicted labels and will output accuracy, F1 score, precision and recall. (Note that the evaluation metric is F1 score). The scoring script is used for evaluating your TEST prediction. This is an offline scorer to see how you do on the Dev set. Once you are satisfied with your performance you can upload your submission to the Kaggle.

Please make sure to have `sklearn library` installed before running `Scorer.py`.

```
Usage of the scorer:

    python3 Scorer.py  <gold-file> <pred-file>

In the dataset directory, there are example
gold and prediction files. If they are used with the scorer,
they should produce the following results:

python3 Scorer.py EXAMPLE_GOLD.txt EXAMPLE_PRED.txt

OVERALL SCORES:
MACRO AVERAGE PRECISION SCORE: 20.97 %
MACRO AVERAGE RECALL SCORE: 20.97 %
MACRO AVERAGE F1 SCORE: 20.97 %
OVERALL ACCURACY: 20.97 %
```

**Requirements:**

1. Your submission must have the **same** structure as `Tom_Smith_PRED.txt` (the example prediction file in `blank_labs/Lab4/` in the student repo).
2. The predication label must be in the **original label format** ( `i.e., '1star', '2star', '3star', '4star', or '5star'` ).

We provide a funtion `out_prediction()` to generate the predictions in the correct format:

In [2]:
```python
def out_prediction(first_name, last_name, prediction_list):
    """
    out_prediction takes three input varibles: first_name, last_name, predic
    <first_name>, string, your first name, e.g., Tom
    <last_name>, string, your last name, e.g., Smith
    <prediction_list>, list of string which includes all your predications c
                       e.g., ['1star','5star','3star']

    Generate a file is named with <yourfirstname>_<yourlastname>_PRED.txt ir
    """
    output_file = open("{}_{}_PRED.txt".format(first_name,last_name),'w')
    for item in prediction_list:
        output_file.write(item+"\n")
    output_file.close()
```

An example of using `out_prediction` funtion. You can find a file `Tom_Smith_PRED.txt` in your diretory.

In [8]:
```
out_prediction("Tom", "Smith", ['1star','5star','3star'])
```

More detailed instructions on uploading your predictions to Kaggle along with the link to the Kaggle competition will be given in the lab session on Thursday.

### Colab Instructions

Google Colab will allow you to train your model on a GPU.

You can follow the steps to use Colab:

1. We provide a new notebook `(lab4_colab.ipynb)` for your experiments on Colab. You should develop your system on `lab4_colab.ipynb` instead of current jupyter notebook.
2. Go to Google colab.
3. Create an account or login your account.
4. Select "UPLOAD" and upload `lab4_colab.ipynb`, again please don't upload current notebook (Lab4.ipynb).
5. Set the hardware: **Go to the navigation bar, click Runtime --> Change runtime type --> Hardware accelerator --> Select GPU.**
6. You don't need to install any packages. Google prepared everything for you.
7. You can find all your generations in `Files`. You can download your notebook and files.

Suggestion:

1. You can download the notebook from Colab and overwrite your local version of **`lab4_colab.ipynb`**.
2. If you train your model on GPU, please make sure your model, input and loss is sent to GPU using XXX.to(device) where device is `cuda`.
3. If you want to send the GPU varibles to CPU, please use XXX.cpu() to detach from GPU. You can find more related information here.
4. Google colab keeps disconnecting automatically after 30 mins without activity. You can find some solutions here.

`Warning`: Running on Colab CPU will be slow (easily slower than running on your laptop).

## 1.2 Spark Bonus (Optional)

rubric={spark:5}

We'll award 5 extra points to the top-2 systems and/or for particularly good work.

## 1.3 Please clearly describe the system you submitted in 1.1 (i.e., your best model) in a maximum of 400 words.

rubric={reasoning:5}

Hints:

1. Describe your overall approach to sentiment analysis. You should also mention strategies which you attempted but which did not eventually end up being successful.
2. Describe all the hyper-parameters of your submitted system and describe the hyper-parmeter tuning process.
3. Format things to make them easy to understand.
4. Provide some error analysis. Do some of the classes (1-5 stars) attain better performance than others. Why do you think that might be?
5. Mention a few ideas which might help to improve your current system

YOUR ANSWER HERE

## Model Performance

My best model is actually the logistic regression with TfidfVectorizer, the macro f1 score is *59.3%* which is quite close to 60%.

I tried CBOW Neural Network with spacy pretrained weights matrix and BiLSTM with spacy pretrained weights matrix + hyperparameter tuned, and none of the model perform a better macro f1 score on the development set.

For CBOW, I think the f1 score is low makes sense because CBOW did not consider the order of the sentence, hence 'not good, so bad' and 'not bad, so good' are the same to CBOW, hence it does make sense that CBOW has lowest f1 score, *44%*.

For BiLSTM, I believe it has worse performance, macro f1 score 57%, because the word embeddings focus more on the semantic instead of the vocabulary like 'good', 'bad', 'soso' due to smooth.

I also tuned the hyperparameter of BiLSTM, including the hiddenlayer size, num layers, and dropout probability, it does increase the macro f1 score by 2%, but still fall in the overfitting before it exceed the performance of logistic model.

## Prediction analysis

From the predictions, the highest and lowest ratings normally have the highest accuracy, this is because they have strong sentiments. Meanwhile for 2, 3, and 4, they contains

both compliments and criticise so models can not identify them as in highest and lowest ratings.

### Something that can improve my predictions

I'm wondering if we can add attention into BiLSTM so that BiLSTM no longer forget contextual/vocabulary information, meanwhile, I believe adding more data into the training set will also increase the macro f1 score.

## 1.4 Please organize your code in `Lab4.ipynb` only keeping the code that you used to train your submitted system in 1.1. Submit `Lab4.ipynb` with the repo.

rubric={quality:5,mechanics:5}

Make sure that your code is commented and neatly organized. Also, make sure that you use descriptive variable names and split your code into functions where appropriate.

Please submit your code in `Lab4.ipynb` even if you used Google Colab.