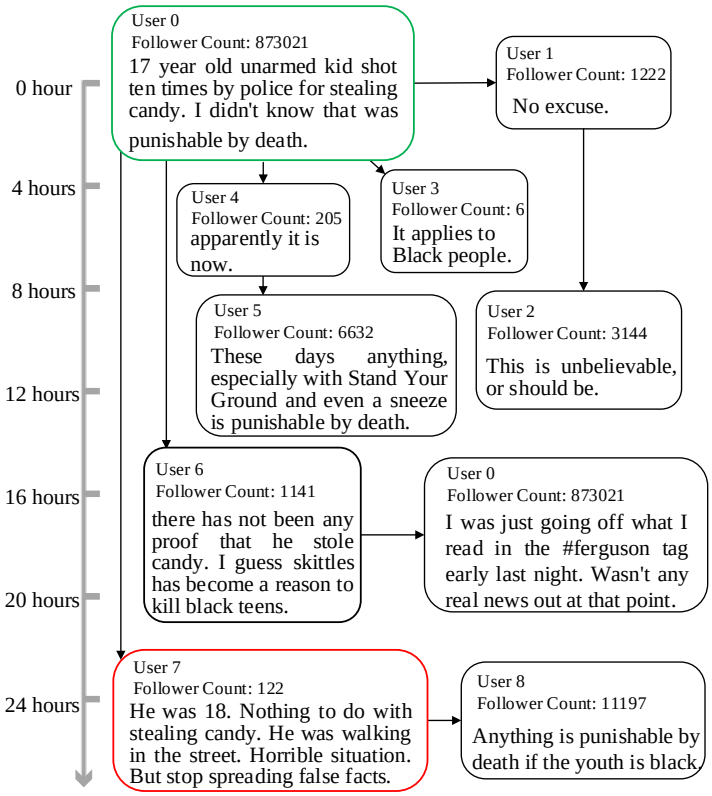
COMP90042 Project 2021:
Rumour Detection and Analysis on Twitter

Project type: Individual (non-group)
Report and code submission due date: 9pm Thu, 13th May 2021
Codalab submission due date: 1pm Thu, 13th May 2021 *(no extensions possible for this component)*

The concept of rumour has a long history, and it is typically defined as an unverified statement or news circulating from person to person. Rumours have the potential to spread quickly through social media, and bring about significant economical and social impact. The figure below illustrates an example of a rumour propagating on Twitter. The source message (green box) started a claim about the cause of Michael Brown's shooting, and it was published shortly after the shooting happened. It claimed that he was shot ten times by the police for stealing candy. The message was retweeted by multiple users on Twitter, and within 24 hours there were about 900K users involved, either by reposting, commenting, or questioning the original source message. From the replies we can see some users (e.g. User 7; red box) questioned the truthfulness of the original message.



The challenge of the project is to develop a rumour detection system and analyse the nature of rumours that are being propagated on Twitter. We will frame this using two tasks: rumour detection and rumour analysis.

Task 1: Rumour Detection

In this task, you will be provided with a set of source tweets and their replies (i.e. the comments we saw in the figure above), and each source tweet is labelled as either a rumour or non-rumour. The task here is to build a binary classifier using this dataset. For each tweet (source tweet or reply tweet), the dataset provides a range of information, including the text of the tweet, information of the user who made the tweet, unique ID of the tweet, etc (more information will be provided in the "Datasets" section below).

You're free to explore any methods or machine learning models for building the binary classifier. To give some ideas, we could model the source tweet and replies as a sequence of tweets using recurrent networks. Alternatively we could also model them based on their propagation structure (like the tree structure of

comments we saw earlier) using recursive networks or graph networks. We might want to consider incorporating some user information, as it could provide hints to the trustworthiness of a user. While you are permitted to use pretrained models or embeddings, you should only use the provided dataset for training the model, i.e. **you should not crawl or search for more training data**. Whatever methods or features you use, **you must at least incorporate the tweet text in your model** (we are doing an NLP project, after all).

Task 2: Rumour Analysis

In this task, you will use your trained rumour classifier from the first task and apply it a set of provided COVID-19 tweets to detect rumours. Given the predicted rumours and non-rumours, the aim here is to perform some analyses to understand the nature COVID-19 rumours and how they differ to their non-rumour counterparts. Below are some questions to get you started:

- What are the topics of COVID-19 rumours? How do they differ from the non-rumours? How do these topics evolve over time?
- What are the popular hashtags of COVID-19 rumours and non-rumours? How much overlap or difference do they share?
- Do rumour source tweets convey a different sentiment/emotion to the non-rumour source tweets? What about their replies?
- What are the characteristics of rumour-creating users, and how do they differ compared to normal users?

Note that these are just some suggestions, and you are free to explore any questions to understand COVID-19 rumours on Twitter. As before, while you are free to do any analyses that may not be text-related (e.g. propagation analysis), most of your analyses should involve the text of the tweets. You should do your analyses using only the provided data.

You will complete two tasks, and write a report that details: (1) your detection system (first task), e.g. the reason behind the choices you have made and its performance; and (2) findings from your analyses (second task). Both tasks are equally important, so you should put similar efforts to them when you're working on the project.

We hope that you will enjoy the project. To make it more engaging **we will run the first task as a Codalab competition**. You will be competing with other students in the class. The following sections give more details on data format, the grading scheme and the use of Codalab. Your assessment will be based on your report, your performance in the competition, and your code.

Submission materials: Please submit the following:

- Report (.pdf): https://canvas.lms.unimelb.edu.au/courses/121115/assignments/213183
- A zip file (.zip) containing your python code (.py or .ipynb) and scripting code (.sh or similar) if using Unix tools: https://canvas.lms.unimelb.edu.au/courses/121115/assignments/213184

Note that there are two different submission links/shells for the project report and code. The reason for separating these two submissions is that we will be running peer reviewing for the report one week after the project has completed. Please note that you should be uploading a single pdf document for your report and a single zip file for your code; **all other formats are not allowed**, e.g. docx, 7z, rar, etc. Your submission will not be marked and will be given a score of 0 if you use these other formats. You should not have any external data files, as you should use only the provided data for model training and rumour analysis.

If multiple code files are included, please make it clear in the header of each file what it does. If pre-trained models or embeddings are used, you do not need to include them as part of the submission, but make sure your code or script downloads them if necessary. We should be able to run your code, if needed, however note that code is secondary — the primary focus of marking will be your report, and your system performance on Codalab.

You must submit at least one entry to the Codalab competition.

Late submissions: -10% per day

Marks: 35% of mark for class

Materials: See Using Jupyter Notebook and Python page on Canvas (under Modules>Resources) for information on the basic setup required for COMP90042, including an iPython notebook viewer and the Python packages NLTK, Numpy, Scipy, Matplotlib, Scikit-Learn, and Gensim. For this project, you are encouraged to use the NLP tools accessible from NLTK, such as the Stanford parser, NER tagger etc, or you may elect to use the Spacy or AllenNLP toolkit, which bundle a lot of excellent NLP tools. You may also use Python based deep learning libraries: TensorFlow/Keras or PyTorch. **You should use Python 3.**

You are being provided with various files including a training, a development and a test set. See the instructions below (section "Datasets") for information on their format and usage. If there's something you want to use and you are not sure if it's allowed, please ask in the discussion forum (without giving away too much about your ideas to the class).

Grading: You will be graded based on several criteria: clarity of expressions of your report, soundness and novelty of your methods, substance of your work, interpretation of your results and performance of your system (section "Grading" below will provide more details).

Updates: Any major changes to the project will be announced via Canvas. Minor changes and clarifications will be announced in the discussion forum on Canvas; we recommend you check it regularly.

Academic Misconduct: This is an individual project, and while you're free to discuss the project with other students, this is ultimately an individual task, and so reuse of code between students, copying large chunks of code from online sources, or other instances of clear influence will be considered cheating. Do remember to cite your sources properly, both for research ideas and algorithmic solutions and code snippets. We will be checking submissions for originality and will invoke the University's Academic Misconduct policy where inappropriate levels of collusion or plagiarism are deemed to have taken place.

Datasets

You are provided with several data files for use in the project. :

`[train,dev,test].data.jsonl`**:** tweet data for the first task (rumour detection);
`[train,dev].label.json`**:** rumour labels for the first task (rumour detection);
`covid.data.jsonl`**:** COVID-19 tweet data for the second task (rumour analysis).

All data files (`[train,dev,test,covid].data.jsonl`) are JSONL files, where each line is a JSON string. For these files, each line is an **event**: a list of tweets where the first tweet is a source tweet and the rest are reply tweets. When classifying whether a source tweet is spreading a rumour, we recommend that you use both source and reply tweets from the event.[1] Note that the reply tweets are sorted arbitrarily — they are not ordered chronologically by their timestamp, so you need to be careful if you are modelling them as a sequence of tweets.

An event (list of tweets) is a JSON string, and looks like the following:


```
[
  {
    "created_at": "Wed Jan 07 12:01:03 +0000 2015",
    "id_str": "552797058990870528",
    "in_reply_to_user_id_str": null,
    "text": "Spread this cover in solidarity with the victims at Charlie Hebdo. Don't let the
      sword conquer the pen. http:\/\/t.co\/XVkPPbkLhn",
    "user": {
      "id": 11345012,
      "id_str": "11345012",
```


[1] While you can technically build a classifier that use only the source tweets to detect rumours, its performance is likely to be poor, as the reply tweets will provide helpful information whether the source tweet is spreading a rumour, as we saw in the example figure earlier.

```
      "name": "Ivo Vegter",
      ...
    },
    ...
  },
  {
    "created_at": "Wed Jan 07 17:39:24 +0000 2015",
    "id_str": "552882207426355201",
    "in_reply_to_status_id_str": "552797058990870528",
    "text": "@IvoVegter @pankajchandak #IMwithCharlieHebdo",
    "user": {
      "id": 1083066139,
      "id_str": "1083066139",
      "name": "Solid Item",
      ...
    },
    ...
  }
]
```


Here the event consists of just 2 tweets, one source tweet and a reply tweet. Notice that the source tweet's `in_reply_to_user_id_str` has a `null` value, while the reply tweet's `in_reply_to_user_id_str` has a value of `552797058990870528`, which is the `id_str` of the source tweet. If you are modelling the propagation structure of the comments (i.e. to recover the tree structure that we saw in the example figure earlier), you will need to use this key. There are many more key-value pairs for each tweet object; you should consult the Twitter API documentation for their definitions. You are free to use any of these key-value pairs in the tweet object as features for your rumour detection system.

The label files (`[train,dev].label.json`), on the other hand, are standard JSON files:


```
{
  "552800070199148544": "non-rumour",
  "544388259359387648": "non-rumour",
  "552805970536333314": "non-rumour",
  "525071376084791297": "rumour",
  "498355319979143168": "non-rumour",
  "553591259672379392": "rumour",
  "580342401392312321": "non-rumour",
  "498609222020780033": "non-rumour",
  ...
  "552797058990870528": "non-rumour",
  ...
}
```


Each entry gives the ground truth label for a source tweet, e.g. here we can see that the source tweet earlier (`552797058990870528`) is a non-rumour. Note that we are using the string version of the ID (`id_str` rather than `id`) for matching source tweets between the data (e.g. `train.data.jsonl`) and label file (e.g. `train.label.json`).

For all files (JSONL data files or JSON label files), you should use the Python `json` library to load them.

Training, Development and Test

For the first task (rumour detection), there are 3 data partitions: training, development and test. Each of this partitions has a different purpose. The training set (`train.data.jsonl` and `train.label.json`) should be used for building your models, e.g., for use in development of features, rules and heuristics, and for supervised/unsupervised learning. You are encouraged to inspect this data closely to fully understand the task.

The development set (`dev.data.jsonl` and `dev.label.json`) is formatted like the training set. This will help you make major implementation decisions (e.g. choosing optimal hyper-parameter configurations), and should also be used for detailed analysis of your system — both for measuring performance and for error analysis — in the report.

You will use the test set (`test.data.jsonl`) to participate in the Codalab competition. For this reason no labels are provided for this partition. You should not *at any time* manually inspect the test dataset; any sign that you have done so will result in loss of marks.

For the second task (rumour analysis), there is only 1 file: `covid.data.jsonl`. You will use your trained rumour detection system from the first task to classify these COVID-19 tweets and do some analyses to understand the nature of COVID-19 rumours.

Evaluation Script

For the first task, we provide a script (`eval.py`) for evaluating your rumour detection results. This script takes two input files: the ground truth and your predictions, and computes precision, recall and F1 score on the rumour class. Shown below is the output from running against random predictions on the development set:

```
$ python eval.py --predictions dev.baseline.json --groundtruth dev.label.json
Performance on the rumour class:
Precision = 0.3120567375886525
Recall    = 0.47058823529411764
F1        = 0.3752665245202559
```

Your rumour detection system will hopefully be a good deal higher! We will be focussing on F1 scores on the positive class, and the precision and recall performance are there for your information, and may prove useful in developing your system.

The example prediction file, `dev-baseline.json`, is the output of a baseline system where the labels are randomly populated. This file will help you understand the required file format for creating your development output (for tuning your system using `eval.py`) and your test output (for submission to the Codalab competition).

Grading

Your submissions will be graded as follows:

Component	Criteria	Description	Marks
Writing	Clarity	Is the report well-written and well-structured?	5
	Tables/Figures	Are tables and figures interpretable and used effectively?	3
Content	Soundness	Are the experiments sound? Are methods justified and used correctly?	7
	Substance	How much work is done? Is there enough substance?	5
	Novelty	How novel or ambitious are the techniques or methods?	5
	Results	Are the results and findings convincing? Are they well articulated?	5
Performance	Rumour class F1	Graded based on Codalab leaderboard ranking	5

A report should be submitted with the description, analysis, and comparative assessment of methods used. You should describe your methods in enough detail that we could replicate them without looking at your code. For the first task, you should mention any choices you made in implementing your system along with empirical justification for those choices using the development set. You should also detail both your development performance and the "Final Evaluation" performance on the Codalab leaderboard (details in the section below). For the second task, you should articulate the intention of your analyses (i.e. the questions they are addressing), and explain your findings. For both tasks, you should use tables and the appropriate charts to report your results/findings.

The description of your method should be clear and concise. You should write it at a level that a Masters student could read and understand without difficulty. If you use any existing algorithms, you do not have to rewrite the complete description, but must provide a summary that shows your understanding and you should provide a citation to reference(s) in the relevant literature. In the report, we will be very interested in seeing evidence of your thought processes and reasoning for choosing one approach over another (as indicated by the heavier weighting of the "soundness" criteria).

The report should be submitted as a PDF, and be no more than four A4 pages of content (excluding references, for which you can use the 5th page). You should use the ACL template when writing your report, which is available as an Overleaf template or offline Latex/Word files. We prefer you to use LaTeX, but you are permitted to use Word. **You must include your student number under the title** (using the `\author` field in LaTeXand enabling the `\aclfinalcopy` option), **but not your name** so as to facilitate anonymous peer reviewing. We will not accept reports that are longer than the stated limits above, or otherwise violate the style requirements.

For the performance component, you will be graded based on the relative ranking of your system, computed as: $\frac{N-r+1}{N} \times 5$, where N is the total number of systems/students on the leaderboard and r is your system rank. E.g. if $N = 300$ and you're the top-ranked system (#1), you will score 5.0; but if you're ranked #100, you will score 3.35.

Codalab

You will need to join the competition on Codalab to submit your rumour detection system for the first task. To do so, visit:

> https://competitions.codalab.org/competitions/30503?secret_key=83e0d8d6-1ac2-4fb4-ba53-93618034e870

and sign up for a Codalab account **using your @student.unimelb.edu.au email address**, and request join the competition. You can do this using the "Participate" tab. Only students enrolled in the subject will be permitted to join, and this may take a few hours for us to process so please do not leave this to the last minute.

Please edit your account details by clicking on your login in the top right corner and selecting "Settings". Please provide your **student number** as your Team name. Submissions which have no team name will not be marked.

You can use this system to submit your test output, by selecting the "Participate" tab and then clicking the "Ongoing evaluation" button, and then "Submit". This will allow you to select a file, which is uploaded to the Codalab server, which will evaluate your results and add an entry to the leaderboard. Your file should be a **zip archive** containing a single file named *test-output.json*, which has a label prediction for all source tweets in *test.data.json* (with matching `id_str` keys). The format of the output file should follow the format of the provided baseline system in the "Evaluation Script" section above (`dev.baseline.json`). **The system will produce an error message if the filename is different, as it cannot process your file**.

The results are shown on the leaderboard under the "Results" tab, under "Ongoing Evaluation". To start you off, I have submitted a system that gives random predictions, to give you a baseline. The competition ends at 1pm on 13th May, after which submissions will no longer be accepted (individual extensions can not be granted to this deadline). At this point the "Final Evaluation" results will be revealed. These two sets of results reflect evaluation on different subsets of the test data. The best score on the ongoing evaluation may not be the best on the final evaluation, and we will be using the final evaluation scores in assessment. **The final result for your best submission(s) can now be discussed in the report**, which is due at 9pm on the same day (13th May).

Note that Codalab allows only 3 submissions per user per day, so please only upload your results when you have made a meaningful change to your system. Note that the system is a little slow to respond at times, so you will need to give it a minute or so to process your file and update the result table.