COMP90042 Project 2020:
Climate Change Misinformation Detection

Project type: Individual (non-group)
Due date: 9pm Wed, 13th May 2020
Codalab due date: 1pm Wed, 13th May 2020 *(no extensions possible for competition component)*

Although there is ample scientific evidence on climate change, there is still a lingering debate over fundamental questions whether global warming is human-induced. One factor that contributes to this continuing debate is that ideologically-driven misinformation is regularly being distributed on mainstream and social media.

The challenge of the project is to build a system that detects whether a document contains climate change misinformation. The task is framed as a binary classification problem: a document either contains climate change misinformation or it doesn't.

The caveat of the task is that we only have training documents with positive labels. That is, the training documents provided are articles with climate change misinformation. An example can be seen below:

> New Zealand schools to terrify children about the "climate crisis" Who cares about education if you believe the world is ending?What will it take for sanity to return? Global cooling? Another Ice Age even? The climate lunatics would probably still blame human CO2 no matter what happens.But before that, New Zealand seems happy to indoctrinate an entire generation with alarmist nonsense, and encourage them to wag school to protest for more "action" (reported gleefully by the Guardian, natch):Totally disgraceful, and the fact that the Guardian pushes this nonsensical propaganda as "news" is appalling.You should read the whole thing just to understand how bonkers this is.

We encourage you to have a skim of the documents to get a sense how they read; there are some common patterns that these documents share in challenging climate change, e.g. by fear mongering (like the example article), or by discrediting the science behind climate change.

You're free to explore any learning methods for the task, e.g. by building a supervised binary classifier or developing methods that leverage only the positive label data (one-class classification). You should not, however, attempt to reverse engineer the sources of the documents, and classify using the document source as a feature. Any sign that you have done so will result in significant penalty, as it trivialises the problem. If you want to expand your training data (e.g. to get negative label data), you can crawl the web for additional resources. Alternatively, you can also use publicly released NLP datasets. A list of popular NLP datasets can be seen here. Pre-trained models and word embeddings are also allowed.

You will need to write a report outlining your system, the reason behind the choices you have made, and the performance results for your techniques.

We hope that you will enjoy the project. To make it more engaging we will run this 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 use of codalab, and the marking scheme. Your assessment will be based on your report, your performance in the competition, and your code.

Submission materials: Please submit the following:

- Report (.pdf)
- Python code (.py or .ipynb)
- Scripting code (.sh or similar), if using Unix command line tools for preprocessing
- External training data(.json, <10MB), if used

You should submit these files via Canvas, as a zip archive. Submissions using other formats to those listed, e.g., docx, 7z, rar, etc will not be marked, and given a score of 0. Your external training data json file (if you have one) should not exceed 10MB.

If multiple code files are included, please make clear in the header of each file what it does. If pre-trained models 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: -20% per day

Marks: 40% 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 are free to use public NLP datasets or crawl additional resources from the web. 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).

Evaluation: You will be evaluated based on several criteria: the correctness of your approach, the originality and appropriateness of your method, the performance of your system, and the clarity and comprehensiveness of your final report.

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.json` a set of training documents (all with positive labels)
`dev.json` a set of development documents (mixture of positive and negative labels)
`test-unlabelled.json` a set of test documents (without labels)

All data files are json files, formulated as a dictionary of key-value pairs. E.g.,

```
"train-0": {
  "text": "why houston flooding isn't a sign of climate change...",
  "label": 1
}
```

is the first entry in `train.json`. The dev and test files follow the same format, with the exception that test excludes the label fields. You should use the Python `json` library to load these files.

For the test data, the label fields are missing, but the "id" number (e.g. `test-0`) is included which should be used when creating your codalab submissions.

Each of this datasets has a different purpose. The training data 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 and some common patterns shared by documents with climate change misinformation. As mentioned earlier, you are free to expand the training data by crawling the web or using public NLP datasets.

The development set is formatted like the training set, but this partition contains documents with negative and positive labels. 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, as discussed above, to participate in the codalab competition. You should not *at any time* manually inspect the test dataset; any sign that you have done so will result in loss of marks.

Scoring Predictions

We provide a scoring script `scoring.py` for evaluation of your outputs. This takes as input two files: the ground truth, and your predictions, and outputs precision, recall and F1 score on the positive class. Shown below is the output from running against random predictions on the development set:

```
$ python3 scoring.py --groundtruth dev.json --predictions dev-baseline-r.json
Performance on the positive class (documents with misinformation):
Precision = 0.5185185185185185
Recall    = 0.56
F1        = 0.5384615384615384
```

Your scores 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 and evaluating your system.

Also provided is an example prediction file, `dev-baseline-r.json`, to help you understand the required file format for your outputs. The label fields are randomly populated for this baseline system.

Misinformation Detection System

You are asked to develop a climate change misinformation detection method, or several such methods. How you do this is up to you. The following are some possible approaches:

- Expand the training data to include documents with negative labels and build a binary supervised classifier;
- Formulate the problem as a one-class classification or outlier detection task, and use the provided positive label data as supervision;
- Build a language/topic model using the positive label data, and classify a document based on how well its language/topic conforms to patterns predicted by the language/topic model.
- A combination of the approaches above.

Note that these are only suggestions to help you start the project. You are free to use your own ideas for solving the problem. Regardless of your approach, the extent of success of your detection system will likely depend on informativeness of the extracted features and novelty of the model design.

If you are at all uncertain about what design choices to make, you should evaluate your methods using the development data, and use the results to justify your choice in your report. You will need to perform error analysis on the development data, where you attempt to understand where your approach(es) work well, and where they fail.

Your approaches should run in a modest time frame, with the end to end process of training and evaluation not taking more than 24 hours of wall clock time on a commodity desktop machine (which may have a single GPU card). You are welcome to use cloud computing for running your code, however techniques with excessive computational demands will be penalised. This time limit includes all stages of processing: preprocessing, training and prediction.

Evaluation

Your submissions will be evaluated on the following grounds:

Component	Marks	Criteria
Report writing	10	clarity of writing; coherence of document structure; use of illustrations; use of tables & figures for experimental results
Report content	20	exposition of technique; motivation for method(s) and justification of design decisions; correctness of technique; ambition of technique; quality of error analysis; interpretation of results and experimental conclusions
Performance	10	positive class F1 score of system

The performance levels will be set such that 0=random; 1-3=simple baseline performance; 4-7=small improvements beyond baseline; 8-10=substantial improvements, where the extent of improvement is based on your relative ranking on the leaderboard. The top end scores will be reserved for top systems on the leaderboard. You must submit at least one competition entry to codalab, and your best result in the leaderboard will be used in marking.

Once you are satisfied that your system is working as intended, you should use the training and development data to do a thorough error analysis, looking for patterns in the kinds of errors your basic system is making. You should consider the various steps in processing, and identify where the most serious problems are occurring. If there are any relatively simple fixes that might have a sizeable impact on performance, you should feel free to note and apply them, but your main goal is to identify opportunities for major enhancements. You should include a summary of this error analysis in your report.

A report should be submitted with the description, analysis, and comparative assessment (where applicable) of methods used. There is no fixed template for the report, but we would recommend following the structure of a short system description paper, e.g., https://www.aclweb.org/anthology/W18-5516. You should mention any choices you made in implementing your system along with empirical justification for those choices. Use your error analysis of the basic system to motivate your enhancements, and describe them in enough detail that we could replicate them without looking at your code. Using the development dataset, you should evaluate whether your enhancements increased performance as compared to the basic system, and also report your relative performance on the codalab leaderboard. Finally, discuss what steps you might take next if you were to continue development of your system (since you don't actually have to do it, feel free to be ambitious!).

For the evaluation, you should generally avoid reporting numbers in the text: include at least one table, and at least one chart. Using the development set, you should report your results, showing precision, recall and F1-score, as appropriate. In addition, you are encouraged to report results with other metrics, where needed to best support your error analysis.

Your 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.

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 follow the ACL style files based on a "short paper" which are available from https://acl2020.org/calls/papers/. We prefer you to use LaTeX, however you also permitted to use Word. You should include your full name and student id under the title (using the `\author` field in LaTeX, and the `\aclfinalcopy` option). We will not accept reports that are longer than the stated limits above, or otherwise violate the style requirements.

Codalab

You will need to join the competition on codalab. To do so, visit

https://competitions.codalab.org/competitions/24205

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 use 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 outputs, by selecting the "Participate" tab 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 the keys in *test-unlabelled.json*. **The system will provide 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" values will be available. 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 used in the report and discussed in your analysis.

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.