COMP90042 Project 2023:
Automated Fact Checking For Climate Science Claims

Project type: Individual
Report and code submission due date: 9pm Mon, 15th May 2023
Codalab submission due date: 1pm Mon, 15th May 2023 **(no extensions possible for this component)**

The impact of climate change on humanity is a significant concern. However, the increase in unverified statements regarding climate science has led to a distortion of public opinion, underscoring the importance of conducting fact-checks on claims related to climate science. Consider the following claim and related evidence:

Claim: The Earth's climate sensitivity is so low that a doubling of atmospheric CO2 will result in a surface temperature change on the order of 1°C or less.

Evidence:

1. In his first paper on the matter, he estimated that global temperature would rise by around 5 to 6 °C (9.0 to 10.8 °F) if the quantity of CO 2 was doubled.
2. The 1990 IPCC First Assessment Report estimated that equilibrium climate sensitivity to a doubling of CO 2 lay between 1.5 and 4.5 °C (2.7 and 8.1 °F), with a "best guess in the light of current knowledge" of 2.5 °C (4.5 °F).

It should not be difficult to see that the claim is not supported by the evidence passages, and assuming the source of the evidence is reliable, such a claim is misleading. The challenge of the project is to develop an automated fact-checking system, where given a claim, the goal is to find related evidence passages from a knowledge source, and classify whether the claim is supported by the evidence.

More concretely, you will be provided a list of claims and a corpus containing a large number evidence passages (the "knowledge source"), and your system must: (1) search for the most related evidence passages from the knowledge source given the claim; and (2) classify the status of the claim given the evidence in the following 4 classes: {`SUPPORTS`, `REFUTES`, `NOT_ENOUGH_INFO`, `DISPUTED`}. To build a successful system, it must be able to retrieve the correct set of evidence passages *and* classify the claim correctly.

Besides system implementation, you must also write a report that describes your fact-checking system, e.g. how the retrieval and classification components work, the reason behind the choices you made and the system's performance. We hope that you will enjoy the project. To make it more engaging **we will run the task as a Codalab competition**. You will be competing with other students in the class. The following sections give more details on the data format, system evaluation, grading scheme and use of Codalab. Your assessment will be graded 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/151109/assignments/387847
- 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/151109/assignments/387859

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 shortly 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 do not need to upload your data files in the zip (e.g. the evidence passages). You'll find more information in the submission shells on how to submit the report and code.

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. Note that we may review 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 subject

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 3.8.**

For this project, you're allowed to use pretrained language models or word embeddings. Note, however, that you are **not allowed to**: (1) use closed-source models, e.g. OpenAI GPT-3; (2) models that can't be run on Colab (e.g. very large models that don't fit on the GPU on Colab); or (3) look for more training data in any form (whether they are labelled or unlabelled for a related or unrelated task) to train your system. In other words, you should only train your system using the provided data, which includes a training, a development and a test set; see the instructions in "Datasets" below 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: While you're free to discuss the project with other students, 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. In regards to the use of artificial intelligence tools in the context of academic integrity, please see the university's statement on this: https://academicintegrity.unimelb.edu.au/plagiarism-and-collusion/artificial-intelligence-tools-and-technologies.

Datasets

You are provided with several files for the project:

`[train-claims,dev-claims].json`**:** JSON files for the labelled training and development set;
`[test-claims-unlabelled].json`**:** JSON file for the unlabelled test set;
`evidence.json`**:** JSON file containing a large number of evidence passages (i.e. the "knowledge source");
`dev-claims-baseline.json`**:** JSON file containing predictions of a baseline system on the development set;
`eval.py`**:** Python script to evaluate system performance (see "Evaluation" below for more details).

For the labelled claim files (`train-claims.json`, `dev-claims.json`), each instance contains the claim ID, claim text, claim label (one of the four classes: {`SUPPORTS`, `REFUTES`, `NOT_ENOUGH_INFO`, `DISPUTED`}), and a list of evidence IDs. The unlabelled claim file (`test-claims-unlabelled.json`) has a similar structure, except that it only contains the claim ID and claim text. More concretely, the labelled claim files has the following format:

```
{
  "claim-2967":
  {
    claim_text: "[South Australia] has the most expensive electricity in the world."
    claim_label: "SUPPORTS"
    evidences: ["evidence-67732", "evidence-572512"]
  },
  "claim-375":
  ...
}
```

The list of evidence IDs (e.g. `evidence-67732`, `evidence-572512`) are drawn from the evidence passages in `evidence.json`:

```
{
  "evidence-0": "John Bennet Lawes, English entrepreneur and agricultural scientist",
  "evidence-1": "Lindberg began his professional career at the age of 16, eventually ...",
  ...
}
```

Given a claim (e.g. `claim-2967`), your system needs to search and retrieve a list of the most relevant evidence passages from `evidence.json`, and classify the claim (1 out of the 4 classes mentioned above). You should retrieve at least one evidence passage.

The training set (`train-claims.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-claims.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-claims-unlabelled.json`) to participate in the Codalab competition. For this reason no labels (i.e. the evidence passages and claim labels) are provided for this partition. You are allowed (and encouraged) to train your final system on both the training and development set so as to maximise performance on the test set, but you should not **at any time** manually inspect the test dataset; any sign that you have done so will result in loss of marks. In terms of the format of the system output, we have provided `dev-claims-predictions.json` for this. Note: you'll notice that it has the same format as the labelled claim files (`train-claims.json` or `dev-claims.json`), although the `claim_text` field is optional (i.e. we do not use this field during evaluation) and you're free to omit it.

Evaluation

We provide a script (`eval.py`) for evaluating your system. This script takes two input files, the ground truth and your predictions, and computes three metrics: (1) F-score for evidence retrieval; (2) accuracy for claim classification; and (3) harmonic mean of the evidence retrieval F-score and claim classification accuracy. Shown below is the output from running predictions of a baseline system on the development set:

```
$ python eval.py --predictions dev-claims-baseline.json --groundtruth dev-claims.json
Evidence Retrieval F-score (F)     = 0.3377705627705628
Claim Classification Accuracy (A) = 0.35064935064935066
Harmonic Mean of F and A           = 0.3440894901357093
```

The three metrics are computed as follows:

1. `Evidence Retrieval F-score (F)`: computes how well the evidence passages retrieved by the system match the ground truth evidence passages. For each claim, our evaluation considers *all* the retrieved evidence passages, computes the precision, recall and F-score by comparing them against the ground truth

passages, and aggregates the F-scores by averaging over all claims. E.g. given a claim if a system retrieves the following set {`evidence-1`, `evidence-2`, `evidence-3`, `evidence-4`, `evidence-5`}, and the ground truth set is {`evidence-1`, `evidence-5`, `evidence-10`}, then precision $= 2/5$, recall $= 2/3$, and F-score $= 1/2$. The aim of this metric is to measure how well the retrieval component of your fact checking system works.

2. `Claim Classification Accuracy (A)`: computes standard classification accuracy for claim label prediction, ignoring the set of evidence passages retrieved by the system. This metric assesses solely how well the system classifies the claim, and is designed to understand how well the classification component of your fact checking system works.

3. `Harmonic Mean of F and A`: computes the harmonic mean of the evidence retrieval F-score and claim classification accuracy. Note that this metric is computed at the end after we have obtained the aggregate (over all claims) F-score and accuracy. This metric is designed to assess both the retrieval and classification components of your system, and as such will be used as **the main metric for ranking systems on Codalab**.

The first two metrics (F-score and accuracy) are provided to help diagnose and develop your system. While they are not used to rank your system on the leaderboard, you should document them in your report and use them to discuss the strengths/weaknesses of your system.

The example prediction file, `dev-claims-baseline.json`, is the output of a baseline system on the development set. 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).

Note that this is not a *realistic* baseline, and you might find that your system performs worse than it. The reason for this is that this baseline constructs its output in the following manner: (1) the claim labels are randomly selected; and (2) the set of evidence passages combines several randomly selected ground truth passages and several randomly selected passages from the knowledge source. We created such a 'baseline' because a *true random baseline* that selects a random set of evidence passages will most probably produce a zero F-score for evidence retrieval (and consequently zero for the harmonic mean of F-score and accuracy), and it won't serve as a good diagnostic example to explain the metrics. To clarify, this baseline will not be used in any way for ranking submitted systems on Codalab, and is provided solely to illustrate the metrics and an example system output.

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	H. Mean of F and A	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. 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). 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**. **Appendix is not allowed** — you should therefore consider carefully the information that you want to include in the report to build a *coherent* and *concise* narrative.

You should use the ACL template when writing your report. 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 on the leaderboard and r is your system rank. E.g. if $N = 100$ and you're the top-ranked system (#1), you will score 5.0; but if you're ranked #50, you will score 2.55.

Codalab

You will need to join the competition on Codalab to submit your fact checking system. The Codalab competition link will be announced on Canvas at a later date.

You must use your **student.unimelb.edu.au** address email to join the competition (via the "Participate" tab), and you are not permitted to join the competition with multiple email accounts. **Any student who is found to have participated with multiple accounts will be automatically suspended from the competition and graded zero for the project**.

Once you have joined the competition, please edit your account details by clicking on your login in the top right corner and selecting "Settings". **Set your team name using your student number. Submissions which have no team name will not be marked.**

To submit your test output, select the "Participate" tab, click the "Ongoing evaluation" button, and then click "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-claims-predictions.json`. The JSON file should produce the claim labels and evidence passages for all the claims in `test-claims-unlabelled.json`. The format of the JSON file should follow the format of the provided baseline system (i.e. `dev-claims-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". The competition ends at 1pm on 15th May, after which submissions will no longer be accepted (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 of your best submission(s) can now be discussed in the report**, which is due at 9pm on the same day (15th 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. **Please do not over-tune your system based on the ongoing test set**, as it is very likely to see a performance drop when it's evaluated on the final test set, since it probably has overfitted on the ongoing test set (we see this every year in COMP90042 projects, where systems that have a large number of submissions during ongoing evaluation see a large drop in ranking once the final evaluation results are released). Note that Codalab 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.