COMP90042 Project 2019: Automatic Fact Verification

Due date: 5pm Friday, 23rd May, 2019

Text Classification

Your challenge is to build a system for fact verification: the task of automatically validating whether a claim is true, false or unverifiable based on the information in a large text corpus. The problem requires finding sentences in the text corpus (here Wikipedia) which relate to the query claim, and then use this evidence to predict the validity of the claim.

For example, consider the claim:

> John Howard acted in the ABC series, Sea Change.

To evaluate the claim,[1] we could first identify that John Howard is the name of an actor (among others sharing the name), he has a Wikipedia page, and that the page references the show "SeaChange":

> John Howard (born 22 October 1952) is an Australian stage and screen actor. Howard is best known for his appearances in the television series SeaChange, Always Greener, All Saints and Packed to the Rafters.
> `https://en.wikipedia.org/wiki/John_Howard_(Australian_actor)`

Together these two sentences SUPPORT the earlier claim, the first verifying that he is an actor and the latter linking him to the show SeaChange. An alternative jusitification would be via the wikipedia page for SeaChange, which begins:

> SeaChange is an Australian television program that ran for 39 episodes from 1998 to 2000 on the ABC. It was created by Andrew Knight and Deborah Cox and starred Sigrid Thornton, David Wenham, William McInnes, John Howard, Tom Long, and Kerry Armstrong. Laura Gibson (Sigrid Thornton), a high-flying city lawyer, is prompted to undergo a 'seachange' with her children Rupert and Miranda after her husband is arrested for fraud and is found to have had an affair with her sister. ...
> `https://en.wikipedia.org/wiki/SeaChange`

The first two sentences together also SUPPORT the claim.

Your job is to automate the above process of verification, creating a system that can judge the validity of a statement and justify its decision based on references to sentences in wikipedia. System predictions must include both the validity label and the supporting evidence, and will be jugdged on the correctness of both (see Scoring Predictions, below).

You will need to write a report outlining your system developed, the reasons 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 teams 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 team's 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

[1]Note that the "truth" of the claim is relative to the information contained in Wikipedia, rather than the underlying truth. For instance, there are mistakes and omissions in Wikipedia; the broader problem of judging the truth of the statement is considerably more complex, and beyond the scope of the project.

Each group should choose one member to submit these files via the LMS, 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).

If multiple code files are included, please make clear in the header of each file what it does. Do not submit any datafiles, your code should assume the data is in the same directory. 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: 30% of mark for class

Materials: See the main class LMS page for information on the basic setup required for this class, 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 the corpora in NLTK, as well as Wikipedia pageview statistics and the SNLI natural language inference corpus. You may also use Python based deep learning libraries: TensorFlow/Keras or PyTorch. Information retrieval tools may also be used, such as pylucene.

You are being provided with various files including a training, a development and a test set. See the main instructions for information on their format and usage. Any other package, tool, or corpus is not allowed except by special permission of the instructors: if there's something else you want to use, 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.

Group assessment: You should form a group of two students, and we expect both of you to contribute equally to the project. In most cases, everyone in the same group will receive the same mark. However, we may make adjustments based on peer assessment.

Each team member will be required to perform peer assessment of your team mate, which will be factored into the marking to adjust for individual effort. We hope that team members work well together and put in equal effort, such that the same mark can be given to each member. If you anticipate a problem in your team, please raise your concern with us before the deadline.

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

Academic Misconduct: Though this is a group project and therefore exchange of code within groups is allowed, reuse of code across groups, 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, 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:

`wiki-text.zip` a collection of wikipedia documents
`training.json` a set of training claims and answers
`devset.json` a set of development claims and answers
`test-unlabelled.json` a set of test claims (without answers)

Most of datafiles are json, formulated as a dictionary of key-value pairs linking a claim identified with its details. E.g.,

```
"75397": {
  "claim": "Nikolaj Coster-Waldau worked with the Fox Broadcasting Company.",
  "label": "SUPPORTS",
  "evidence": [
    ["Fox_Broadcasting_Company", 0],
    ["Nikolaj_Coster-Waldau", 7]
  ]
}
```

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

The entries under *evidence* denote the sentences in the collection which support or refute the claim (no evidence is provided for label=NOT ENOUGH INFO.) Sometimes more than one sentence is needed to form a chain of evidence, however for simplicity we aggregate all relevant sentences to form the evidence set. Each entry in the evidence refers to documents and sentences in the `wiki-text.zip` archive. This includes several shards of wikipedia, named `wiki-nnn.txt` for nnn numbering from 1 to 109. The files contain one wikipedia sentence per line, formatted as *Page identifier, sentence number, sentence text*.

For example, wiki-035.txt line 209223 is as follows (emphasis added):

> Fox_Broadcasting_Company **0** *The Fox Broadcasting Company -LRB- often shortened to Fox and stylized as FOX -RRB- is an American English language commercial broadcast television network that is owned by the Fox Entertainment Group subsidiary of 21st Century Fox .*

The page identifier is Fox_Broadcasting_Company, the sentence index in that page is **0** (the first), and the sentence is shown in *italics*. Note that sentences have already been split, tokenised and preprocessed for your convenience. (The -LRB- and -RRB- tokens denote left and right round brackets, respectively.) The sentence numbers are not consecutive, and you should just treat the sentence number as an identifier.

For the test data, the answer fields are missing, but the "id" number 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 learning. You are encouraged to inspect this data closely to fully understand the task, the types of claims, and the kinds of answers you are expected to predict. The training set is large, and you should not feel you have to use all of the data if it is not feasible to do so. The development set is formatted like the training set, where we have reserved some statements for validation. This will help you make major implementation decisions, 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 `score.py` for evaluation of your outputs. This takes as input two files: the ground truth, and your predictions, and will output a series of evaluation outputs. Shown below is the output from runing with against some nonsense predictions:

```
$ python3 score.py devset.json random-devset.json
Label Accuracy  32.13%
Sentence Precision  0.10%
Sentence Recall  0.07%
Sentence F1  0.08%
Document Precision  0.14%
Document Recall  0.20%
Document F1  0.16%
```

Your scores will hopefully be a good deal higher! We will be focussing on: a) label accuracy, measured over the "label" field, with possible values being SUPPORTED, REFUTED or NOTENOUGHINFO; and b) sentence F1 scores, based on the list of "evidence" predictions. Note that the scoring only uses the first 5 predictions, any remaining values are ignored. The other values are there for your information, and may prove useful in developing and evaluating your system. You should familiarise with further details of the evaluation by looking at the source of `score.py`.

Also provided is an example prediction file, `random-devset.json`, to help you understand the required file format for your outputs. The label and evidence fields are randomly populated.

Fact Verification System

You are asked to develop a fact verification method, or several such methods. How you do this is up to you, and you should start by reviewing the lecture on question answering, a closely related task. You should consider approaches based on

- retrieval techniques for finding the best matching document to the claim;
- retrieval techniques for finding the best matching sentence to the claim, within the candidate documents identified above;
- language processing techniques, such as named entity tagging, based on the idea that the most important parts of the claim are often entities; you might also want to use lexical resources like wordnet, distributional word vectors or similar;
- some form of machine learning method to predict the label based on the above evidence

as well as your own ideas for solving the problem.

This problem and dataset are derived from the FEVER challenge, which is an ongoing research competition. You should draw inspiration from this research, e.g., the details of the dataset https://arxiv.org/abs/1803.05355, and the system description papers for competition entrants. We ask though that you do **NOT** use the official Fever dataset, nor should you use code from other competition entrants as part of your system. You are welcome to run other code as part of benchmarking the quality of your outputs, but you should ensure it is clearly flagged in the report, that is you should not pass this off as your own work. FEVER this is an ongoing competition, and those with particularly innovative and high performing systems should consider entering the competition to compete against the current state-of-the-art.

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. As part of this, you may want to develop method-specific evaluation.

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, indexing, training and prediction.

Evaluation

Your submissions will be evaluated on the following grounds:

Component	Marks	Criteria
Report writing	8	clarity of writing; coherence of document structure; use of illustrations; use of tables & figures for experimental results
Report content	14	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	8	label accuracy and sentence F1 score of system

The performance levels will be set such that 0=random; 1-2=simple baseline performance; 3-5=small improvements beyond baseline; 6-8=substantial improvements, with the top end of the range reserved for systems competitive with the state-of-the-art, under both metrics. 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.

Each team will submit a report 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 label accuracy 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. You should follow the ACL style files based on a "short paper" which are available from `http://www.acl2019.org/EN/call-for-papers.xhtml`. We prefer you to use LaTeX, however you also permitted to use Word. You should include the full names and student ids for both team members 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

Details will be added shortly the `codalab` competition and how to go about submitting your results to the leaderboard. We will make an announcement on the LMS once the competition site is running.