COMP90042 Project 2018: Question Answering

May 3, 2018

1 Project Outline

Due date: 11pm, Sunday 27th May, 2018

Your challenge is to build a Question Answering (QA) system, based on working as an individual or in a small team of up to three members. More specifically, given a question and a document, the goal is to find the answer to that question in the corresponding document. Your task is to develop an automatic method, or several methods, for solving this problem.

You will need to write a report outlining your system/s 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 kaggle in-class competition. Kaggle is one of the most popular online platforms for running data science competitions. You will be competing with other teams in the class. The following sections give more details on data format, the use of kaggle, and marking scheme. Your assessment will be based on your team's performance in the competition, your code, and your report.

Submission materials: Please submit the following:

- Report (.pdf)
- Python code (.py or .ipynb)

Each group should choose one member to submit these files via the LMS, as a zip archive (submissions using obscure archive format such as 7z, rar, etc will not be marked, and given a score of 0).

If multiple code files are included, please make clear what each of the files do. 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 very much secondary – the primary focus of marking will be your report, and your system performance on Kaggle.

Every member of the group will also be required to perform peer assessment of your colleagues in your team (individual teams are exempted from this), which will be factored into the marking to adjust for individual effort. The peer assessment process will be done via a LMS quiz, which will be announced near to the project deadline.

You must also submit at least one entry to the Kaggle In Class competition.

Late submissions: -20% per day (note that late Kaggle submissions cannot be accepted, i.e., the competition component needs to be completed strictly by the deadline)

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 toolkit. You are free to use the corpora in NLTK. You may also use Python based deep learning libraries: Theano, TensorFlow, Keras or PyTorch.

You are being provided with 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, e-mail us.

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 best system, and the clarity and comprehensiveness of your final report.

Group assessment: In the majority of cases, everyone in the same group will receive the same mark, however, we may make adjustments based on the peer assessment.

To keep things fair, the assessment will be tailored to the size of the group. For teams of two, we will expect a longer report. For teams of three, besides a longer report, teams should also submit two QA systems instead of one. These systems need to be sufficiently different in method, as outlined below. As well as this, team submissions on Kaggle will be evaluated against all other teams of smaller or equal size.

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.

2 Use of Kaggle in Class

Please do the following within the first week after receiving this assignment:

- Setup an account on Kaggle using your unimelb student email, and setting your Kaggle **username** to your unimelb login,[1] and register for the competition here https://www.kaggle.com/t/959a494a05a84083b4eab38ce126b65d. Silly login names might be great fun, but please resist the temptation, as this makes marking difficult. Submissions from non-unimelb accounts will not be marked.
- Form your team of student peers; if you do not form one by Wednesday 9^{th} May, you will be treated a single person team.
- For teams:
  - Connect with your team mates on Kaggle and form a Kaggle team, which you can name according to your whim.
  - Submit your Kaggle team name and list of member names and emails to https://goo.gl/forms/Ge6ChcORyvbbeEls2 Only fill in the form once per team.
- For individuals: Submit your Kaggle name and member name to https://goo.gl/forms/Ge6ChcORyvbbeEls2

Those in teams should only make submissions using the team name, and individual submissions will attract penalties. Those working as an individual, should make their submissions as an individual. Note that you are limited to 5 submissions per day.

You should submit your Kaggle results as a comma separated variable (CSV) file. Your outputs should be output, one entry per line, in a CSV file where each line contains two fields: the question id and the answer. The ids are increasing integers and denote the question in the test dataset, while the answer strings can be words or sequences of words. Here is an example:

```
id,answer
0,the queen of spain
1,eating fish and chips on the pier
2,bananarama
...
```

[1] If your login is already taken, please choose a string that makes it trivial for us to work out who you are, e.g., by adding a suffix to your unimelb login name. E.g., bdylan → bdylan843

See also the sample submission on Kaggle (with garbage answers). Take care with double quotes, commas and other special characters, which will need to be escaped (using the python `csv` library makes this easy). Be sure to include the header line as given above.

The real answers for the test data are hidden from you, but were made available to Kaggle. Each time a submission is made, half of the predictions will be used to compute your public F-score and determine your rank in public leaderboard. This information will become available from the competition page almost immediately. At the same time, the other half of predictions is used to compute a private F-score and rank in private leaderboard, and this information will be hidden from you. At the end of the competition, only private scores and private ranks will be used for assessment. This type of scoring is a common practice and was introduced to discourage overfitting to public leaderboard. A good model should generalize and work well on new data, which in this case is represented by the portion of data with the hidden F-score.

The evaluation score used in this competition is the average F-score over the token strings, which is based on the precision and recall between the bags of predicted tokens versus the text span in the gold standard. For a prediction to be judged completely correct, it must be string identical to the gold answer, meaning that correct tokenisation, including punctuation, determiners, etc are highly important. Before the end of the competition each team will need to choose their best submissions for scoring (this could be your basic QA system or, more likely, one of your extensions.) These do not have to be the latest submission. Kaggle will compute a private F-score for the chosen submissions only, and the best of these used to compute the private leaderboard ranking, which will be used for marking.

3 Datasets

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

`documents.json` a collection of wikipedia documents
`training.json` a set of training questions and answers
`devel.json` a set of development questions and answers
`testing.json` a set of test questions, with no answers given

Each of datafiles is a json list, the first comprising a list of documents, where each document has an identifier field, "docid", and "text" which contains a list of paragraph strings comprising the document. The other datafiles contain list of questions, with fields "question", containing a text question, "docid" identifying the relevant document, and in the case of `testing.json`, a question "id" which is used in evaluating your outputs using Kaggle. For the training and test data, each question additionally includes a gold standard answer in the "text" field, as well as the "answer_paragraph", which identifies the paragraph in the document where the answer can be found. There is an exact string match of the answer in the answer sentence, however please be aware that the tokenisation of the "text" answers will often differ from the untokenised text in the document. For the test data, these answer fields are missing, but the "id" number is included which should be used when creating your Kaggle submissions. You should use the Python `json` library to load these files.

Each of this datasets has a different purpose. The training data should be used for building your QA models, e.g., for use in supervised learning. You are encouraged to inspect this data closely to fully understand the task, the types of questions, and the kinds of answers you are expected to predict. The training set is large, and you should not feel you do not 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 documents 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 Kaggle 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.

4 QA System

You are asked to develop one or more QA systems, depending on the size of your group. How you do this is up to you, and you should start by reviewing the lecture on QA and the suggested reading carefully. In particular, you should consider approaches based on

- retrieval techniques for finding the best matching paragraph, sentence or smaller span based on the content terms in the question and the text; this might serve as the first stage in proposing candidate spans, for use in later stages of processing

- language processing techniques, such as parsing, named entity tagging, based on the idea that the answer is often a constituent or an entity; you might also want to use lexical resources like wordnet, distributional word vectors or topic models
- some form of machine learned method, such as learning the expected answer type (e.g., a person entity, a number, a place name) from the way the question is phrased (e.g., what "wh" word is used, if any)

and combinations thereof, as well as your own ideas for solving the problem.

Teams of 3 members will be required to develop diverse approaches, such that the techniques are not just minor riffs on the same idea. Accordingly, such teams should produce at least one predominantly retrieval based method, and one based on richer NLP annotations. Significant parts of your preprocessing and evaluation code will be shared between your methods, however the core of the techniques should differ.

If you are at all uncertain about what design choices to make, you should evaluate your methods using the dev 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, such as evaluating retrieval accuracy.

Your approaches should run in a modest time frame, with the end to end process of training and evaluation not taking more than 8 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.

5 Evaluation

Your submissions will be evaluated on the following grounds:

Component	Marks	Criteria
Report writing	6	clarity of writing and document structure; exposition of methodology; display of experimental results
Report content	12	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*
Competition	12	Kaggle ranking in cohort; absolute score of system

*: For these criteria, the expectations applied to team submissions will be judged more stringently than for individuals. For instance, the greater manpower available to teams, means that we expect more advanced and elaborate techniques, judged under "ambition". This also applies to the level of detail in the error analysis, and degree of comparative analysis in the results and conclusions, which will need to have greater depth to achieve a high grade.

One 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 it should start with a very brief introduction of the problem. You should mention any choices you made in implementing your QA 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 dev dataset, you should evaluate whether your enhancements increased performance as compared to the basic system, and also report your relative performance on the Kaggle 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 dev set, you should report your results, based on the average F-score. 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 references to 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:

Team Size	Content Pages	Reference Pages
1	2	1
2	3	1
3	4	1

where page counts refer to single-sides of A4, using a font size of 11pt and margins of at least 1cm (much like this document). "Content pages" should contain your brief introduction, method description, plots, tables, discussion of results, and conclusions. "Reference Pages" may contain only references. You do not need to include a cover page, but please ensure that all usernames are clearly stated on the first page, as well as your Kaggle team name. If a report is longer than the stated limits above, we will only read and assess the report up to the page limit and ignore further pages.

Competition component

This component of your assessment is based on the results achieved by your method(s) in your final submission to the Kaggle competition. Before the end of the competition you should choose two of your systems to put forward for the final evaluation on the private data.

You will be evaluated in two ways:

by ranking where each competition entry is ranked in terms of the private evaluation score, such that the team with the best score gets a rank of 1 and the worst scoring team a rank of N. Any ties will be assigned the lower (better) rank. These ranks will then be mapped linearly to produce a mark between 0 and 6, i.e., rank 1 gets a mark of 6, and rank N gets a mark of 0. We will round fractional marks to the nearest half mark.

Note that in determining rankings we will compare your approach only against teams of equal or smaller size. This way individuals will not be penalised for underperforming against much larger teams. Note however that larger teams will have a tougher time, as they effectively compete against a larger cohort.

by absolute score where the mark is set based on your F1 score on the private test data, based on the following levels:

private test F1	<1%	1-5%	5-9%	9-12%	12-15%	15-18%	>18%
marks	0	1	2	3	4	5	6

Although you will not see the private test results until the project deadline, your public test results should give you a reasonable idea of your performance. The benchmark system labelled "heuristic" in the leaderboard is a method we implemented, and will falls roughly in the middle of the above range. You should be able to beat with some effort.

Together these two marks will be combined to give an overall evaluation score out of 12.