COMP10001 Foundations of Computing
Semester 1, 2013
Project 2 Description

— VERSION: 4123, DATE: APRIL 29, 2013 —

Changelog

- updated sample format of qrels CSV file in Q5

1 Task Description

Your task in this project is to build a content-based video search system, that is a system which returns a ranked list of videos for a text query, based on the predicted relevance of each video to that query.

You will be provided with a set of descriptions for 2000 YouTube videos, along with details of the original video, in a file called `english.csv`. These descriptions will form the basis of the video results (intuitively, videos which are more reliably described using the terms in the query should rank higher in the results). These descriptions are a subset of a dataset called the Microsoft Research Video Description Corpus,[1] and in using this data, you are bound by the conditions of the data license linked off that page.

Your solution to each component of the project should take the form of a function, with function name and behaviour as defined below. All of your code should be written exclusively in Python and be contained within a single file, in a dedicated directory that you should create for the project. The code should run successfully to completion within IVLE. As with Project 1, the file should be named as follows:

```
username.py
```

where `username` is your University account name (e.g. `tbaldwin.py` in my case).

1. Write a function `make_index` that takes two arguments — (1) `datafile` stipulating the name of a file containing video descriptions in the local directory where your code is run, and (2) `picklefile` stipulating the name of a cPickle file to store the output of the function in — and "pickles" two dictionaries in `picklefile`, in the following serial order: (1) a dictionary where each key is a video URL and each value is a dictionary of words contained in descriptions of that video (as the key) and the integer frequency (as the value); and (2) a dictionary where each key is a word and the value is the total number of occurrences of that word in all descriptions of all videos. Each dictionary should also include a special word `"__TOTAL__"` which contains the total number of words in each document (in the case of the first dictionary), and the total number of words in the document collection (in the case of the second dictionary). The video URL here and for all following questions should be composed from the video ID and start time (e.g. video ID `jbzaMtPYtl8` and start time `48` map onto URL `http://www.youtube.com/watch?v=jbzaMtPYtl8#t=48s`).

   `datafile` is a CSV file, which is structured as follows:

   ```
   ZbzDGXEwtGc,6,15,155632,clean,51,English,A jet is flying.
   ```

   where the fields are as follows (in order):

   (1) a YouTube video ID (accessible via `http://www.youtube.com/watch?v=VIDEOID`)

   (2) the start time (in seconds) of the video as provided to the annotators

   (3) the end time (in seconds) in the video as provided to the annotators

   (4) the ID of the annotator who provided the description

[1] `http://research.microsoft.com/en-us/downloads/38cf15fd-b8df-477e-a4e4-a4680caa75af/`

(5) whether the description has been checked or not (all descriptions in `english.csv` are labelled as `clean`)

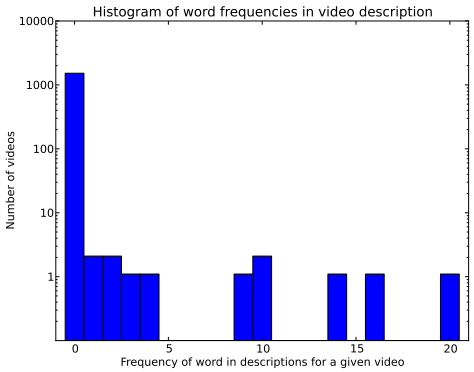
(6) the amount of time taken to write the description (in seconds)

(7) the language the description is written in (all descriptions in `english.csv` are in `English`)

(8) the description of the video segment

[1 mark]

2. Write a function `word_freq_graph` that takes three arguments — (1) `index_fname` stipulating the name of a pickle file containing an index of descriptions, (2) `graph_fname` stipulating a file name in which to store the output graph, and (3) `word` stipulating a word to generate the graph for — and generates a histogram of word frequencies for `word` in `index_fname`. The following is a sample word frequency graph, although you may deviate from this sample if you feel you have come up with a better design:



[1 mark]

3. Write a function `single_word_search` which takes two arguments — (1) `index_fname` stipulating the name of a pickle file containing an index of descriptions, and (2) a single word `word` (in the form of a string) — and returns a list of video URLs ranked in decreasing order of "relevance score" (and sub-ranked in increasing order of video URL in the case of ties in relevance score). Note that only videos with a non-zero relevance score should be included in the list. The relevance score should be calculated based on the following formula:

$$\text{score}(d, t) = \frac{f_{d,t}}{f_d}$$

where d is a document (= video), t is a term, $f_{d,t}$ is the frequency of term t in document d, and f_d is the total frequency of words in document d. Note that the combination of descriptions for a given video should be considered to be a single "document".

[1 mark]

4. Write a function `search` which takes two arguments — (1) `index_fname` stipulating the name of a pickle file containing an index of descriptions, and (2) `query` containing the search query (in the form of a string, with whitespace separating the individual search terms) — and returns a list of video URLs ranked in decreasing order of "relevance score" (and sub-ranked in increasing order of video URL in the case of ties in relevance score). Note that only videos with a non-zero relevance score should be included in the list. The relevance score for each term in `query` should be calculated based on the following formula:

$$w_{d,t} = \frac{f_{d,t}}{f_d} \times \log_e \frac{N}{f_t + 1}$$

where d is a document (= video), t is a term, $f_{d,t}$ is the frequency of term t in document d, f_d is the total frequency of words in document d, N is the total number of documents in the collection and f_t is

the number of documents containing term t. The combined score for the query should be calculated as follows:

$$\text{score}(d, q) = \frac{\sum_{t \in q} w_{d,t}}{\sqrt{\sum_{t \in q} (w_{d,t})^2}}$$

If all these equations look scary, don't panic, as we will go over the details in a forthcoming lecture.

[2 marks]

5. Write a function `rr` that takes three arguments — (1) `query` containing a search query (in the form of a string, with whitespace separating the individual search terms), (2) `doc_ranking` containing a (ranked) list of documents for `query`, and (3) `qrels` stipulating the name of a file containing queries paired with relevant video IDs (see below for details) — and returns the "reciprocal rank" (RR) for `doc_ranking` based on `qrels`. The reciprocal rank is calculated by determining the rank of the first result in `doc_ranking` which is "relevant", and calculating the inverse of that rank (e.g. if the second document were the highest-ranking relevant document, the RR would be $\frac{1}{2}$).

   The file `qrels` will take the form of a CSV file, with one relevant query and video ID/start time per line (and any documents which aren't listed for a given query being implicitly "irrelevant"), e.g.:

```
"bird tree",-8rBYNP0UKM,194
"bird tree",1dmVuwO1RZk,0
```

[1 mark]

6. Write a function `batch_evaluate` that takes a pre-computed index `index_fname`, a list of queries `queries` (made up of string-based queries), a file containing relevance judgements `qrel`, and the name of a file to save the HTML output to, and generates two graphs and an HTML page with the following contents:

   - a table with each column made up of a single query, the number of results returned for it, and its reciprocal rank (following the same order as in `queries`), as well as a final column containing the mean reciprocal rank (or MRR) averaged across all the queries
   - a bar graph which plots the RR for each query, again following the same ordering of queries as in `queries`; the bar graph should be generated in your local file space and named `USERNAME-rr.svg` (e.g. `tbaldwin-rr.svg` in my case)
   - a bar graph which plots the MRR for queries of different length in `queries` (e.g. the MRR for the subset of queries which have length 1, 2 and 3, respectively); the bar graph should be generated in your local file space and named `USERNAME-mrr.svg` (e.g. `tbaldwin-mrr.svg` in my case)

[2 marks]

For all graphs, make sure to label all axes appropriately, and include a caption describing the content of the graph. Marks for questions involving graphs will be awarded in part based on the appropriateness and aesthetics of the presentation. Similarly, questions involving HTML output will be marked in part based on the correctness of the HTML.

As with Project 1, an additional 1 mark will be awarded for code stylistics, including:

- readability and commenting (incl. appropriate naming of variables/functions, and adherence to the code style guide)
- code structure and design (incl. appropriate use of data structures, and lack of redundancy in your code)

[1 mark]

Optional Bonus Question

Extend the basic document scoring function to make use of descriptions in other languages, and a "translation graph" from English terms to terms in each of the other languages. See the Projects page for details.

[2 marks]

2 Assessment

The project will be marked out of 10 (not including the bonus question), and is worth 10% of your overall mark for the subject. Note that there is a hurdle requirement on your combined worksheet and project mark, of 20/40 (of which this project will contribute 10 marks).

I will not accept late submissions. If there are documentable medical or personal circumstances which have taken time away from your project work, you should contact Tim via email at the earliest possible opportunity (this generally means well before the deadline). I will assess whether special consideration is warranted, and if granted, scale up your marks relative to the amount of "down time" from the project. No requests for special consideration will be accepted after the submission deadline.

Be warned that IVLE is often heavily loaded near project deadlines, and unexpected network or system down-time can occur. You should plan ahead to avoid unexpected problems at the last minute. System downtime or failure will generally not be considered as grounds for special consideration.

This project should be completed on an *individual* basis. While it is acceptable to discuss the project with others in general terms, excessive collaboration and any sharing of code are considered a breach of the University's Academic Misconduct policy (`http://academichonesty.unimelb.edu.au/policy.html`). Submission of your project constitutes a declaration of academic honesty and that your submission is strictly your own work (with the exception of the library made available to you). I will be vetting system submissions for originality and will invoke the policy for all involved students (whether you were the provider or recipient of the code) where inappropriate levels of collaboration or plagiarism are deemed to have taken place.

3 Submission

Submission will be via IVLE, using the same mechanism as for Project 1: you should create a dedicated project directory in "mywork" on IVLE, "Add" and "Commit" the directory (via the "More actions..." menu), create a single file in the directory (named as specified above), and "Add" that file. "Commit" the version of the file you wish to submit (and any versions of the file you want to back-up along the way), and finally "Submit" the file.

Once you have submitted, you should "Verify" your submission using the link on IVLE. Clicking this link should display the version of the file that you have submitted for marking. If you are unable to verify your submission, it means your submission has not been received and will not be marked. It is your responsibility to submit and verify before the project due date.

4 Getting Help

Getting a bit lost in the details of the subject? Having problems with a particular concept or question? Have a burning question you want to ask more generally about Computing? The primary places to get personalised help with the subject are:

- The subject forums on the LMS
- Weekly workshops
- After lectures
- Revision lectures, every second Friday (in odd weeks)

- During Tim's office hours

And as a general piece of advice: if you are feeling lost, get help now, rather than sitting on it and getting irrevocably lost as the subject continues to plough ahead.

5 Changes/Updates to the Project Specifications

I will use the LMS to advertise any (hopefully small-scale) changes or clarifications in the project specifications. Any addendums made to the project specifications via the LMS will supersede information contained in the hard-copy version of the project.

6 Important Dates

Release of Project 2 details	29 April, 2013
Deadline for submission of Project 2	13 May, 2013 (9:00am)