Department of Computing and Information Systems
The University of Melbourne
COMP30018/90049 Knowledge Technologies, Semester 1 2016

Project 1: Which films are good?

Due:	Part A: 5pm, Wed 20 Apr 2016 Part B: 5pm, Wed 27 Apr 2016
Submission materials:	Part A: Source code, `README` to the CIS servers (see below); Part B: PDF Report to Turnitin (also below)
Assessment criteria:	Part A: Software Part B: Analysis, Creativity, Soundness, Report Quality
Marks:	COMP30018: The project will be marked out of 15, and will contribute 15% of your total mark. COMP90049: The project will be marked out of 20, and will contribute 20% of your total mark.

Overview

For this project, we will be working with (textual) reviews from IMDb[1]. Your objective is to consider the application of approximate matching methods for identifying film titles in these reviews, and then assess the "good"ness of some film titles.

You will be required to develop some program(s) which automate various parts of this process, however, the technical merits of your software will not form the bulk of the evaluation: the focus of this project is a critical analysis of methodologies — which is to say, what you have learned about the problem and can pass along to another person attempting it. Impeccably-coded software with little insight will not receive a strong mark.

Resources

You will be given a list of titles of films[2] (`film_titles.txt`, and a set of reviews (`revs/`*num*`.txt`) in which to search for them.

Each line of `film_titles.txt` will correspond to the IMDb-identified name of some film. Note that this only represents a fraction[3] of the entire set of films in the IMDb database.

[1] `http://imdb.com`

[2] Well, there may be some other media, like television shows or video games, but mostly films.

[3] There are roughly 7K film titles given to you out of perhaps 1M titles in total.

For example, the first few lines of the file might look like:

```
For His Son
How It Feels to Be Run Over
Richard III
The House With Closed Shutters
```

Note that film titles often comprise more than one word.

Each review will consist of a plain text file with the content of the review; the review text might contain some HTML mark-up and non-ASCII characters. Here is a sample review:

```
Credited as the earliest complete feature-length American film known
to still exist and restored by the American Film Institute, 'The Life
and Death of King Richard III' is otherwise of little value. Rarely
is Shakespeare nearly as boring. Yes, the film is a symptom of its
time; I also watched 'Queen Elizabeth' (1912) today (it wasn't a very
good day), and both are arid and static adaptations from the stage,
histrionic acting included, but without sound, or any other qualities
of the theatre. Films such as these, however, were coincident with
films by others like D.W. Griffith; one can easily see which was
advancing the medium and which was hampering it.
```

This review appears to be about "`Richard III`" above, and it casts the film in a negative light. (Why?)

The entire collection consists of some 50000 reviews; each student will receive a unique subset of the entire collection. Each review is associated with some title in the list, however, for some reviews, it might be (almost) impossible to determine which title is being reviewed. On the other hand, there is no guarantee that every title in the list has a review in the collection.

Each student's personalised dataset will be stored on the CIS servers (`dimefox` and `nutmeg.eng.unimelb.edu.au`) in the directory: `/home/subjects/comp90049/submission/your-login/` where `your-login` is your central University log-in. There is a thread about connecting to the CIS servers on the LMS Discussion Forum; please get in touch if you are having trouble accessing the data (`nj@unimelb.edu.au`).

Terms of Use

By using this data, you are becoming part of the research community — consequently, as part of your commitment to Academic Honesty, you **must** cite the curators of the dataset in your report:

> Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. (2011). Learning Word Vectors for Sentiment Analysis. *The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011)*.

If you do not cite this work in your report, you will have plagiarised these authors, consequently **your report will be given a mark of 0**.

You have been warned.

The data is freely download-able from the World Wide Web; if you wish to use the entire data set — which we discourage — you may download it from: `http://ai.stanford.edu/%7Eamaas/data/sentiment/`

Note that the review collection is a sub-sample of actual data posted to IMDb, without any filtering whatsoever. As such, the opinions expressed within the documents in no way express the official views of The University of Melbourne or any of its employees, and using them in this teaching capacity does not constitute endorsement of the views expressed within. It is possible that some of the reviews are in poor taste, or that you might find them offensive; please use your discretion when considering the data, and try to look beyond the content to the task at hand. The University of Melbourne accepts no responsibility for offence caused by any content contained within.

If, for some reason, you object to these terms of use, please contact us as soon as possible (`nj@unimelb.edu.au`).

Tasks

There are two sub-tasks in this project: (i) finding film titles, and (ii) deciding whether a film is good.

Finding titles

Each review in the collection is associated with a single title in the list. Your goal in this task will be to determine — to the best of your ability, or, more accurately, to the best of the computer's ability — which reviews are associated with which films. There is no single "correct" strategy for solving this problem.

As a first attempt, you might try using an **exact string search** method, for example, using regular expressions to search for each title within each review. However, this will **not be adequate** for a complete submission.

Instead, you should attempt to perform some **approximate string search** method(s): students enrolled in COMP30018 should attempt (at least) one approximate matching method; students enrolled in COMP90049 should attempt (at least) two approximate matching methods.

Approximate Matching

As we have discussed in this subject, approximate matching methods can be used to compare some unknown string with known strings from a dictionary — which can allow us to match strings with spelling mistakes, orthographic variants, phonological variants, and so on. In this task, the "dictionary" is the list of titles, and the "unknown" strings are the reviews.

We have seen a selection of methods for approximate matching:

- Neighbourhood search (e.g. `agrep` or regular expressions)
- N-gram distance
- Edit distance
- Phonetic matching (e.g. Soundex or `carney`)

Other approximate matching strategies have been discussed in the literature, you might wish to try them out instead.

One immediate problem that will surface will be that many of the titles comprise multiple words (for example, *Richard III*). You will need to find a way of dealing with this; there are two main options:

1. You may process the reviews (and possibly the titles) into separate tokens ("words") based on whitespace or punctuation. This approach dovetails best with a N-gram distance or Global Edit Distance matching strategy. You will then need some strategy for aggregating matches in sequence.

2. You may treat the review as a string (possibly with spaces), and compare it against each review, suitably interpreted as a string with spaces. This approach dovetails best with a Neighbourhood search or Local Edit Distance matching strategy.

The output of such a system should indicate the "best" match (film title) for each review. You may wish to further incorporate some kind of threshold, where if the top match is not better than some value (for edit distance or n-gram distance), or within some number of edits (for neighbourhood search), then the review cannot be associated with a title.

Since implementation is not our main concern in this project, you may use external packages, standalone programs (like `grep`), or supplied code if you wish; if you do so, the source of the package should be clearly indicated in your `README`.

Evaluating Effectiveness

Once you have one or more system(s) that can associate a review with a film title, you will be required to assess the **effectiveness** of the system.

The recommended method for this will be to actually look at the review text, and to consider whether is it actually reviewing the indicated title. (Note that some film title, or substring thereof, might appear in the text of a review, but the review is actually about some other film.) Obviously, it will not be practical to read many thousands of reviews — you should (randomly) select some modest number of reviews for hand-checking.

You should attempt to assess the performance of each of your systems formally, using one or more of:

- Accuracy: how many of the hand-checked reviews were assigned a correct title by the system?

- Precision: if you're using a threshold, what proportion of above-threshold assignments were correct? — you might consider different thresholds here.

- perhaps "Coverage[4]": what proportion of reviews could be assigned to a title at all?

Note that many "best" matches will be trivial exact matches — for your report, you should choose some examples which demonstrate effective and ineffective approximate matching from the data. (Although some methods might not produce any correct results!)

Your calculated evaluation(s) should be detailed in your report; if you attempt more than one approximate matching method, you should contrast their performance. You should further discuss the examples to contextualise the behaviour of your system(s) — to demonstrate **why** the system performs as it does.

[4]This is a simplification of Recall, which will be impractical to assess in this context.

[Optional] Using external film resources

If you wish, you may use external resources to **supplement** (not replace) your approximate matching strategies.

For example, you might automatically collect lists of actors from the given film from Wikipedia or IMDb. If you attempt to collect any information from the Web automatically, **be very careful**: this can be construed as being against the Terms of Service of these sites, and, in the worst case, a denial-of-service attack, and could result in your or the University's IP address being blocked by the service's host. We **don't recommend** this unless you know what you're doing!

If you download the full dataset from Andrew Maas' website, it is possible to uniquely identify each review with the corresponding film title (via the review's URL). You may do this if you wish — again with the proviso of being careful about how you automatically access the data — if you wish to evaluate your system more thoroughly. **This does not replace the requirement to develop an approximate matching solution.**

Deciding whether a film is good

For the film titles to which the system can assign (correctly or not) one or more reviews, you will be required to determine automatically whether the film is "good" — has positive reviews associated with it or not.

You may choose any method for this that you like, which is generally consistent with the topics we discuss in Knowledge Technologies.
This is an opportunity for you to be creative! If you do not wish to be creative at this point, here is one possible method:

- Identify some positive and negative words (e.g. *good*, *bad*)
- Use your approximate matching strategies from above to search for these words in the review
- If the review contains more positive words than negative words, then this review is a positive indicator for the film
- If the review collection contains more positive reviews for a film title than negative reviews, that film is "good"

You will not be required to assess this formally, but you should discuss the output of this system in your report — which films it believes to be good and **why**; and whether you feel this is a useful strategy to solving this knowledge problem (and **why**).

Report

You will write a report detailing your analysis, which should focus mostly on the application of approximate matching methodology(ies) to this problem. This should be a structured technical report, roughly in the style of the sample papers; a sample structure might be as follows:

1. A basic description of the problem and data set;
2. An overview of your approximate matching methods. You can assume that the reader is familiar with the methods discussed in this subject, and instead focus on how they are applied to this task, including some indication of how you dealt with multi-word titles (if necessary);

3. A discussion of the effectiveness of the approximate matching method(s) you chose, including a formal evaluation, and some examples to illustrate where the method(s) was/were effective/ineffective;

4. A short sketch of your mechanism and motivations for deciding which films are "good", as well as a discussion of some "good" films;

5. Some conclusions about the problem of searching for approximate matches to film titles within a corpus of IMDb reviews, and automatically determining which films are "good"

The report should consist of about 750–1500 words. This is quite short: you will need to be concise, and you should use tables or graphs to present the data more compactly where appropriate. You should not discuss the technical details of your implementation, unless they are novel or crucial to your analysis of the methods; you can assume that the reader is familiar with the methods we have discussed in this subject. **Overly long reports will be penalised.**

Note that we will be looking for evidence that you have thought about the task and have determined reasons for the performance of the methods involved. A report that simply records data without corresponding analysis will not receive a strong mark.

You should include a bibliography and citations to relevant research papers. A good place to begin is probably with the paper that describes the data set:

> Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. (2011). Learning Word Vectors for Sentiment Analysis. *The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011)*.

You might also consider the Approximate Matching Readings linked on the LMS, in particular:

> Zobel, Justin and Philip Dart. (1996). Phonetic String Matching: Lessons from Information Retrieval. In *Proceedings of the Eighteenth International ACM SIGIR Conference on Research and Development in Information Retrieval*. Zürich, Switzerland. pp. 166–173.

Note that, in general, **you should not cite Wikipedia**. For more information, see the Caution in `http://en.wikipedia.org/wiki/Wikipedia:Citing_Wikipedia` and more generally `http://en.wikipedia.org/wiki/Wikipedia:Academic_use`.

Both of the above articles are excellent sample reports, you should be able to get a sense of style from them. We will make report templates (in Rich-Text Format (for MS Word or similar) and LaTeX) available; it would be preferred for you to use these templates when writing your report. Please do not include a title page, abstract, or table-of-contents. You report **must be submitted in PDF format**. We reserve the right to return reports submitted in any other format with a mark of 0.

Submission

Submission will entail two parts (note that they are due at different times):

Part A. The code for your approximate matching system[5] (for titles) and a `README` file which briefly explains how to compile and run your submission and the location and format of the output. These should be uploaded to the CIS servers, into the same directory where you obtained the data, namely: `/home/subjects/comp90049/submission/your-login/`

[5] You don't need to submit your "good"ness system; it just needs to be described in your report.

Your software can be implemented in any programming language or combination of programming languages. There is no requirement that it runs on the CIS servers; please be considerate of other users if you run your system there — the task may be very memory–intensive.
This Part of the Project will be due at 5pm on Wed 20 Apr 2016.

Part B. Your written report, as a single file in Portable Document Format (PDF).
This will be submitted via Turnitin, on the LMS — the link will be made available in the "Assessment" area shortly before submission.
This Part of the Project will be due at 5pm on Wed 27 Apr 2016.

The marks available will be as follows:

	COMP30018	COMP90049
A: Software	4	5
B: Analysis	4	6
Creativity	1	2
Soundness	3	4
Report Quality	3	3
Total	15	20

We will post a marking rubric to indicate what we will be looking for in each of these categories when marking.

Changes/Updates to the Project Specifications

If we require any (hopefully small-scale) changes or clarifications to the project specifications, they will be posted on the LMS. Any addendums will supersede information included in this document.

Academic Misconduct

For most people, collaboration will form a natural part of the undertaking of this project. However, it is still an individual task, and so reuse of code or excessive influence in algorithm choice and development will be considered cheating. We will be checking submissions for originality and will invoke the University's Academic Misconduct policy (`http://academichonesty.unimelb.edu.au/policy.html`) where inappropriate levels of collusion or plagiarism are deemed to have taken place.