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

Project 2: Which films are good, revisited?

Due:	5pm, Fri 27 May 2016
Submission materials:	Source code, `README` to the CIS servers (where appropriate); PDF Report to Turnitin
Assessment criteria:	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], similarly to Project 1. This time, we will be attempting to build a more formal "good"ness system, with a clearer definition of "good". Your objectives are: to consider the application of machine learning methods for assessing the "good"ness of some review texts; potentially identifying some interesting patterns within the data; leveraging your knowledge of certain machine learning criteria; and writing a report based on your observations. This aims to reinforce several concepts in data mining and machine learning, including the use and critique of different models, manipulating different data sources, and making conclusions based on observed results.

The technical side of this project is limited: you are highly encouraged to use appropriate machine learning packages in your exploration, which means that you can (almost) "solve" the problem by plugging the relevant data files into an application, and reading off a set of numbers.

However, acquiring knowledge will potentially be quite difficult. Once again, the main focus of our evaluation will be the report detailing the knowledge that you have taken away from your attempts. A strong implementation with a poor report will not receive a good mark.

Resources

The main data set will be a collection of 50000 reviews, which has been split into two halves: a training set and a development set. Each review has been classified as either "positive" (`P`, at least 7 stars) or "negative" (`N`, at most 4 stars)[2]

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

[2] "Neutral" reviews, of 5 or 6 stars, were excluded from the data set by the curators.

The text of the reviews has been made available on the LMS, along with the following derived data sets:

- a "bag–of–words" representation of the training and development data sets, where each file has been tokenised, and the frequency of each token has been recorded. These files (`trainBoW.txt` and `devBoW.txt`) are in LibSVM format, where: each line corresponds to an instance (review); the class (given here as the number of stars[3]) is the first entry in the line; there is a space–delimited list of attribute–value pairs (separated by a colon). For example, `1112:3` means that attribute #1112 occurred 3 times in the text of this review. The attribute numbers refer to line numbers from the (textual) vocabulary file `imdb.vocab`. These values are 0-indexed, so that the first line in the vocabulary file correspond to attribute #0, and so on. The example above would occur on the 1113^{th} line of the file: `books`.

- a number of files in ARFF format (mostly CSV, where the class is the final element; plus a header which names/describes the format (numerical) of each attribute). These files can be loaded directly in Weka (`http://www.cs.waikato.ac.nz/ml/weka/`) so that you can begin experimenting straight away; more on this below. There are a few possibilities:

  - {`train,dev`}`-first10.arff`: For each instance, only the first ten attributes, according to the "bag–of–words" files above, are included. These correspond to the following tokens: `the`, `and`, `a`, `of`, `to`, `is`, `in`, `it`, `i`, `this`[4]. For example, the following line:

    `9,1,4,4,6,4,2,2,4,0,P`

    would indicate that the token `the` was present 9 times, `and` was present once, `this` wasn't present, and so on. This is a positive (`P`) review.
  - {`train,dev`}`-first50.arff`: For each instance, only the first 50 attributes are included.
  - {`train,dev`}`-best.arff`: Each word in the collection has been associated with a score[5] indicating whether it is associated with positive reviews, according to the method described in:

    > Potts, Christopher. 2011. On the negativity of negation. In Nan Li and David Lutz, eds., *Proceedings of Semantics and Linguistic Theory* 20, 636-659.

    For each instance, the 270 attributes which were observed to take the maximum value (calculated as being most well associated with positive reviews) are included.
  - {`train,dev`}`-worst.arff`: For each instance, the 69 attributes which were observed to take the minimum value (most well associated with negative reviews, according to the method above) are included.
  - {`train,dev`}`-best_worst.arff`: For each instance, the 339 attributes which were observed to take either the maximum or the minimum value (most well associated with positive/negative reviews, according to the method above) are included.

  Note that, for all of these files (and the "bag–of–words" above), an instance can be connected to the raw text of its review through the instance number as follows:

[3] Note that this isn't in a suitable format for the Machine Learning task described here, but it is relatively trivial to replace the star values, for example: `sed "s/^[7-9]|^(10)/P/" trainBoW.txt` and similarly with `N`.

[4] You might infer that these are the most frequent tokens in the collection!

[5] The list of values can be obtained by downloading the original data set, if you wish to use it.

- The first 12500 instances are all positive (`P`) reviews, from the `pos` directory, the last 12500 instances are all negative (`N`) reviews, from the `neg` directory.
- Each instance corresponds to the corresponding 0-indexed filename, so that the first positive training instance refers to `pos/0_9.txt`[6]; the second instance refers to `pos/1_7.txt`, and so on. The 12501^{st} training instance, which is the first negative training instance, refers to `neg/0_3.txt`; the final training instance refers to `neg/12499_2.txt`.

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.

Furthermore, if you make use of the `best`, `worst`, or `best_worst` files, these are developed using the method of:

> Potts, Christopher. 2011. On the negativity of negation. In Nan Li and David Lutz, eds., *Proceedings of Semantics and Linguistic Theory* 20, 636-659.

In this case, you should also cite this paper.

The data is freely download-able from the World Wide Web; if you wish to use the entire data set, 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`).

Machine Learning

Systems

Various machine learning techniques are discussed in this subject (Naive Bayes, Decision Trees, Support Vector Machines, Association Rules, etc.); many more exist. Developing a machine learner is

[6] Annoyingly, the name of the file also indicates the number of stars of the review, so that it cannot be automatically determined just from the line number.

likely not to be a good use of your time: instead, you are strongly encouraged to make use of machine learning software in your attempts at this project.

One convenient framework for this is Weka: `http://www.cs.waikato.ac.nz/ml/weka/`. Weka is a machine learning package with many classifiers, feature selection methods, evaluation metrics, and other machine learning concepts readily implemented and reasonably accessible. After downloading and unarchiving the packages (and compiling, if necessary), the Graphical User Interface will let you start experimenting immediately.

Weka is dauntingly large: you will probably not understand all of its functionality, options, and output based on the concepts covered in this subject. The good news is that most of it will not be necessary to be successful in this project. A good place to start is the Weka wiki (`http://weka.wikispaces.com/`), in particular, the primer (`http://weka.wikispaces.com/Primer`) and the Frequently Asked Questions. If you use Weka, please do not bombard the developers or mailing list with questions — the LMS Discussion Forum should be your first port of call. We will post some basic usage notes and links to some introductory materials on the LMS Discussion Forum.

Some people may not like Weka. Other good packages are available (for example, Orange (`http://orange.biolab.si/`) or scikit-learn (`http://scikit-learn.org/`). One caveat is that you will probably need to transform the data into the correct syntax for your given package to process them correctly (usually `csv` or `arff`); we will upload suitable versions of the dataset, if requests are made on the Discussion Forum.

Alternatively, you might try implementing certain components yourself. This will probably be time-consuming, but might give you finer control over certain subtle aspects.

Modes

There are numerous modes by which you might possibly approach this problem: in this Project, we recommend you focus primarily on (supervised) **Classification**.

In classification, you will treat the dataset as a number of (marginally relevant) attributes that you can use to help predict the "class" — "Is this review positive (`P`) or negative (`N`)?" You can use one or more of the (numerous) classification algorithms covered in this subject, or otherwise (for example, Naive Bayes, Decision Trees, k-Nearest Neighbour, 1-R, etc.). It is recommended to use methods that you understand (at least conceptually), otherwise it will be difficult to assess **why** the methods work or not, and your discussion will be limited.

You should employ the `train` data to build a model (according to you chosen classification algorithm(s)), and the `dev` data to evaluate the model, according to one or more evaluation metrics (Accuracy, Precision, Recall, F-Score, Area under the ROC curve, ...). Your evaluation of the system(s) should be included in the report, as support for your discussion of the suitability of the machine learning method(s) you consider.

All of the above can be accomplished more–or–less trivially from the Weka GUI.

Features and Feature Engineering

It is possible that the given files are adequate for you to derive some knowledge about the problem of determining the "good"ness of a review. However, you might find that you wish to include (or remove, which is easy in Weka) some other attributes to help your model classify reviews correctly.

If it simply the case that there is some word in the collection (`imdb.vocab`) that you wish to include in your model, this can be determined by extracting the corresponding entry from the

`{train,dev}BoW.txt` files. We will upload a small Perl utility for automatically scraping the values and modifying an ARFF file accordingly; this will be described on the LMS Discussion Forum.

If, however, you wish to add in some completely different feature (perhaps based upon some important observation you made in Approximate Matching on this dataset!), that is not based around the presence of some individual words, you will need to write a program to modify the given data accordingly. (In Weka, this is the CSV section of the ARFF file; the ARFF file header also would need to be updated to indicate the name of the attribute and the data type — this is described in detail in the Weka documentation.) If you do this, then you should upload your program(s) to the CIS servers, in the same way as the code for Project 1.

Any attempt to expand upon the given feature sets (`first10`, `best`, etc.) will count toward the Creativity component for this project.

Report

The report should be broadly similar to your report for Project 1, including the templates, structure, and sample papers. Carefully chosen tables of results and figures will help you to spend your word count in discussing the relevance of your observations. We expect that the system description and evaluation will be quite terse; consequently, the length requirement for this report will be 750–1350 words [edited 23 May 2016].

In your report, you should describe your approach and observations. Only describe points that are relevant to your paper (and not, say, common to everyone's paper) — in particular, do not discuss the internal structure of your classifiers, unless it is substantially different to the ones we discuss in this subject (and then a primary citation should be adequate), or its internals are important for connecting the theory to your practical observations.

Remember that your goal here is **knowledge** — the (detailed) discussion in the sections above is to allow you to set up the problem in a sensible manner, but actually running the software and getting some results will mostly be a trivial matter. Consequently, reports which only describe results of algorithms without corresponding analysis are trivial, and have missed the entire point of the project!

Don't forget that you need to cite the curators of 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)*.

This paper contains some analysis of various methods for classifying the data (which we do not expect you to reproduce); you might use these authors' observations as a competitive **benchmark** for your system.

Submission

You will submit your report to Turnitin, similarly to Project 1. Your report must be in PDF; we reserve the right to return reports in other formats (most notably DOC/DOCX/RTF) unmarked.

The marks available will be as follows; we will post a marking rubric to indicate what we will be looking for in each of these categories when marking.

	COMP30018	COMP90049
Analysis	6	9
Creativity	3	5
Soundness	3	3
Report Quality	3	3
Total	15	20

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.

Late Submission Policy

You are strongly encouraged to submit by the time and date specified above, however, if circumstances do not permit this, then the marks will be adjusted as follows:
Each business day (or part thereof) that the report is submitted after the due date (and time) specified above, 10% will be deducted from the marks available (COMP30018: 15; COMP90049: 20), up until 5 business days (1 week) has passed, after which regular submissions will no longer be accepted.
Updated 23 May 2016: Due to various issues, we have decided to waive the late penalty for the first business day, upon request. If you wish to take advantage of this, please send a (short) email to `nj@unimelb.edu.au` with the subject line *"Knowledge Technologies Project 2 Late Submission"*; we will acknowledge receipt of the email, at which point you may submit without penalty up until 5pm Monday 30 May 2016.
Submissions beyond this deadline will accrue penalties commensurate to the total number of days late — irrespective of whether the waiver was granted — starting at 2 days (20%) after 5pm, Monday 20 May 2016.