School of Computing and Information Systems
The University of Melbourne
COMP90049 Knowledge Technologies, Semester 1 2017

Project 2: R ur tweeps as mad as u think? #analysis

Due:	Report: 3pm (15h00 UTC+10), Wed 24 May 2017
	Reviews: 8.59am (08h59 UTC+10), Mon 29 May 2017
Submission materials:	System outputs, code & `README` (where necessary) to the MSE servers (see below);
	Anonymous Report (in PDF), and Reviews to Turnitin
Assessment criteria:	Analysis, Engineering, Outputs, Report Quality, Reviews
Marks:	The project will be marked out of 20, and will contribute 20%
	of your overall mark for the subject.

Overview

For this project, we will be working with (actual) short messages ("tweets") from Twitter, under the auspices of a shared task, with the data provided by the 2017 SemEval conference (`http://alt.qcri.org/semeval2017/`)[1]. Your objectives are: to consider the application of machine learning methods for assessing the sentiment of tweets; 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 about 33K tweets, which has been split into three parts: a training set, and a development set, and a test set. Each tweet has been manually assigned a sentiment: either "positive", "negative" or "neutral"[2].

[1]This is a modified form of Task 4, Subtask A; you can read more about it about it at `http://alt.qcri.org/semeval2017/task4/` — note that we have altered the data, so that your results will not be directly comparable with the other submissions. Unfortunately, the accepted articles will not be published until August; at which point, you might wish to read some of them, and consider how serious researchers approached this problem. With some effort, you could probably get the data from SemEval, find the test instances, and "cheat" by submitting the gold–standard test labels to Kaggle — please don't do this; there are no marks for the accuracy of your system on the test data, and it might actually make it *more* difficult to get useful knowledge.

[2]Undoubtedly, there will be some tweets where you might think that the sentiment should be labelled differently, but such is the nature of Knowledge Technologies! `:-)`

The tweets have been made available on the LMS, as an archive of several files, which are explained in `README.txt`; briefly:

- There are `tweets` files, which contain the raw text of the training, development, and test tweets
- There are `labels` files, which contain the manually–assigned sentiments of the training and development tweets (*n.b.* not the test tweets)
- There are `arff` files, which we have generated. These are suitable for use with Weka, and will allow you to get started analysing the data straight–away.
  To generate the ARFF files, we first needed to engineer some features: we considered the frequency of all of the various tokens in the collection, after having folded case and removing non-alphetic characters (there are about 60K).
  Next, we performed feature selection[3]. In particular, we have applied the methods of **mutual information**[4] and **Pearson's χ–squared test**[5] to determine which tokens are best correlated with the given classes. We then selected the best 20 tokens for each class, according to each method — 46 distinct features — and recorded the corresponding frequency of each of these words in each tweet instance.

You are free to use the given ARFF representation for the purposes of building the classifier — you can just consider the features as given according to a statistical measure of "goodness" — but you should examine the data further in your analysis.

Alternatively, you can produce your own representation through experimentation. That is, you are not required to use the given ARFF files if you do not wish to.

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:

> Rosenthal, Sara, Noura Farra, and Preslav Nakov (2017). SemEval-2017 Task 4: Sentiment Analysis in Twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval '17)*. Vancouver, Canada.

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**.

Note that the tweet collection is a sub-sample of actual data posted to Twitter, 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 tweets 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.

Also note that you may not re-distribute the data without prior permission.

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

[3]You might like to peruse `https://en.wikipedia.org/wiki/Feature_selection` for a brief overview.
[4]See, for example: `https://en.wikipedia.org/wiki/Mutual_information`
[5]See, for example: `https://en.wikipedia.org/wiki/Pearson_chi-squared_test`

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 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, most notably, scikit-learn (`http://scikit-learn.org/`) is quite well–regarded, if you are already familiar with Python. Another possibility is Orange (`http://orange.biolab.si/`). 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 LibSVM); we will upload suitable versions of the feature engineered dataset described above, 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 tweet positive, neutral, or negative?" 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 your 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.

Phases

The objective of your learner will be to predict the classes of unseen data (hopefully in an accurate manner!). We will use a **holdout** strategy: the entire collection of data is split into three parts: a **training** set, a **development** set, and a **test** set.

The **training phase** will involve training your classifier: for a Decision-Tree classifier, this means building a tree based on the exemplars in the training data; for a Naive Bayes classifier, this means calculating the probabilities of various events; and so on. Parameter tuning (where required) also happens here[6].

The **testing phase** is where you observe the performance of the classifier. The development data is labelled: you should run the classifier that you built in the training phase on this data to calculate your preferred evaluation metric(s). The test data is unlabelled.

You should collect the output of your classifier on the test data, and submit it along with your report; we will use this output to confirm the observations of your approach. We will also set up a Kaggle In-class playground, to which you can submit your system outputs if you wish.

The purpose of this partition of the data (into three parts) is to reduce **overfitting**: having a classifier which reproduces the training data, but does not generalise to unseen data. There is still a risk of overfitting, however: by choosing the algorithm or features which perform best on the development set, your system might not predict the test set accurately. You should keep this in mind for development and your report.

Feature Engineering

All of the above can be performed with the ARFF files, as given. However, these 46 attributes are just one possible representation of the data; whereas there are also many others. Consequently, you should try to engineer some feature(s) of your own devising, with the aim of aiding the classifier in predicting the labels of the test instances. You can then evaluate your classifier with just your newly engineered feature(s), or by combining them with the given 46 attributes.

This will contribute to the "Engineering" of your assessment. To get full marks here, the new feature(s) that you attempt should be based on something other than what we have already examined, namely, the frequency of tokens within the tweets.

Technical Notes

You should submit the program(s) which transform the data into a format that you can use in your machine learning system, where necessary. You should also submit a README that briefly describes how you generate your features (the rationale should be explained in your report), and the purposes of important scripts or external resources, if necessary.

You will not be required to submit scripts which generate the output of your system; any manual tuning (cheating!) will be quite obvious. You should discuss the systems that you used (and the relevant parameter settings, where necessary) in your report. You should detail the various files (which are the outputs on the accompanying test data) in your README: which model and parameters were used to generate it. Technical details of the implementation (system settings, resource limits, etc.) should be discussed in the README. Performance over the development data and your observations should be included in the report.

Report

You will submit an **anonymised** report (i.e. don't include your name or student ID), which should describe your approach and observations, both in engineering (and selecting, if necessary) features, and the machine learning algorithms you tried. The technical details of constructing the features and so on should be left out, unless it is particulary interesting or novel. Even then, they probably belong in your README.

[6]Occasionally, there is yet another partition of the data where parameter tuning takes place.

Your aim is to provide the reader with knowledge about the problem, in particular, critical analysis of the techniques you have attempted (or maybe some that you haven't!). You may assume that the reader has a cursory familiarity with the problem — any concepts that are common to most papers (e.g. well-formed CSV) can be assumed or glossed over in a sentence or two. The internal structure of well-known classifiers should only be discussed if it is important for connecting the theory to your practical observations.

Once more, 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. A short background of related research;
3. An overview of your feature engineering and/or machine learning method(s). 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;
4. A discussion of the effectiveness of the machine learning method(s) you chose, ideally, including some example instances to illustrate where the method(s) was/were effective/ineffective;
5. Some conclusions about the problem of using machine learning methods to perform sentiment analysis on tweets.

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 aim to gloss over the technical details, 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; or have discerned inherent properties of the data; or can sensibly critique the problem framework. Namely, that you have acquired some **knowledge** that you can supply to the reader. 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 previous SemEval conferences. It should be much easier to find relevant peer–reviewed articles for this task than Project 1, consequently, we will be expecting some effort into synthesising a **short** background section. (i.e. It should comprise a few sentences. This shouldn't be a **thorough** background, as it would take more than 1500 words to do justice to all of the related research.)

Reviews

After the reports have been submitted, there will be a five day period where you will review 2 papers anonymously written by your peers. (Incidentally, this is the purpose of having anonymous reports.) Each review should be about 200-400 words. In your review, you should aim to have the following structure:

- A couple of sentences describing what the author has done.
- A couple of sentences detailing what you think the author has done well, for example, novel use of features, interesting methodology, or insightful discussion. **You should indicate why you feel this to be the case.**

- A couple of sentences suggesting weak points or further avenues for interesting research. You should focus on the feature or algorithm or discussion side, and less on the quality of the report itself. **You should indicate why or how your suggestions are relevant or interesting.**

Submission

Submission will entail three parts:

- On the MSE servers, in your project submission directory:
  `/home/subjects/comp90049/submission/your-login/`
  You should upload your code used to generate the engineered features, a short README, and the output of your classifier(s) on the test data.
- 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.
- Your reviews will also be submitted to Turnitin: they will be automatically assigned to you once report submissions are closed.

The marks available will be as follows:

Analysis	7
Engineering	2
Outputs	1
Report Quality	6
Reviews	4
Total	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.

Late Submission Policy

You are strongly encouraged to submit by the time and date specified above, however, if circumstances do not permit this, then:
Each business day (or part thereof) that this project is submitted after the due date (and time) specified above, 10% will be deducted from the marks available, up until 5 business days (1 week) has passed, after which regular submissions will no longer be accepted.