COMP30018/COMP90049
Knowledge Technologies, Semester 2 2014
Project 2: Geolocation of Twitter Users with Machine Learning

Due:	Stage I: 4pm, Fri 17 October 2014 (Melbourne time) Stage II: 11pm, Thurs 23 October, 2013 (Melbourne time)
Submission Mechanism:	Online submission of project directory package, PDF to Turnitin
Submission Materials:	Source, outputs, `README`; anonymised written report in PDF for which we will supply templates
Assessment Criteria:	Creativity, Critical Analysis, Report Quality

Introduction

In project two, you will build on your processing and observations from project one. In particular, you will be using (almost) the same dataset, to solve what is likely to be a more difficult problem.

This project will give you the opportunity to "road–test" various machine learning classification algorithms on a problem that you are now familiar with. You will also be able to leverage the knowledge that you have gained from processing the dataset to extend a sophisticated representation of the data, hopefully making your choice of classifier(s) more effective.

Overview

This document describes Stages I and II of Project 2. Stage I will ask you to use some machine learning strategies on the data that you were processing in Project 1, and to write a report on your observations. In Stage II, you will read some work by your peers and write a short review of their report.

The objective of Stage 1 is to build a **geolocation classifier for Twitter users**, based on their Tweets (and possibly other meta-data you might have about the user). That is, given a set of tweets associated to a specific user, your system will produce a prediction of where that user is located. For the purposes of this project, we will limit the set of relevant locations (target classes) to the five United States cities that are most heavily represented in the Twitter dataset: Los Angeles (`LA`), New York City (`NY`), Chicago (`C`), Atlanta (`At`), and San Francisco (`SF`).

The technical side of this project will involve constructing features for the data, and using appropriate machine learning packages to apply machine learning algorithms to the data to solve the task. You should explore the impact of different features on the performance of the task. This means that direct evaluation of the technical aspects of the project are fairly limited: the focus of the project will be the report, where you will demonstrate the knowledge that you have gained, in a manner that is accessible to a reasonably informed reader. A strong implementation with a poor report will not receive a good mark.

Tasks

Stage I will comprise two main tasks:

- Feature Engineering, which will be optional for students taking this subject at an undergraduate level, and
- Machine Learning

Stage II will be a reviewing process.

Stage I

Feature Engineering

As discussed in the lectures, the process of engineering features that are useful for discriminating amongst your target class set is inherently poorly-defined. Most machine learning assumes that the attributes are simply given, with no indication from where they came. The question as to which features are the best ones to use is ultimately an empirical one: just use the set that allows you to correctly classify the data.

In practice, the researcher uses their knowledge about the problem to select and construct "good" features. Your experience with Project 1, as well as hearing Tim Baldwin's guest lecture on tweet processing, should have given you some ideas about relevant criteria: what aspects of a tweet itself might indicate a user's location? what aspects of a user's meta-data might be useful to determine that user's location? You can also find ideas in published papers, e.g., [1].

Attributes typically fall into one of three categories: binary (T/F), categorical (a or b or c etc.; these are often special cases of binary attributes), and numerical (both discrete and continuous). All three types can be constructed for the given data. Some machine learning architectures prefer numerical attributes (e.g. k-NN); some work better with categorical attributes (e.g. multivariate Naive Bayes) — you will probably observe this through your experiments.

One final question is **feature selection**: given a (large) set of features, which subset is most appropriate to use for classifying the data? There is a rather large body of literature on this topic: you might like to familiarise yourself with some of the concepts if this is a question you would like to answer. Weka, and many other machine learning packages, have some feature selection mechanisms built-in (more on this below). Again, feature selection is more important for certain machine learning frameworks (e.g. Naive Bayes) than others (e.g. SVMs).

To make this easier for you, we have produced a representation of the data in which we have already performed feature selection. In particular, we have selected 100 words from the dataset having the greatest **mutual information** for each of the five classes — giving 386 distinct features — and recorded the corresponding frequency of each of these words for each tweet instance in the data set.

Each line of the data set will look like the vector space model that we are already familiar with. Obviously, each instance is too long to be sensibly displayed here, but as an example, the first five features might be `id`, `chicago`, `atlanta`, `sf`, `httpdealnaycom`, followed by its `class` (from the five locations above) and a typical tweet would look like:

`4997229212,1,0,0,0,SF` typical feature vector

You are free to use this representation for the purposes of building the classifier, or to produce your own through your own experimentation.

For students taking this subject at a Master's level, you will be required to engineer some new features from the dataset (and possibly use them along with the given features). For students taking this subject at an undergraduate level, this step is optional (although you may find it interesting or instructive to try; in which case, it will count toward your Creativity component).

Machine Learning

Systems

Various machine learning techniques have been discussed in this subject (Naive Bayes, Decision Trees, Support Vector Machines, 0-R, etc.); many more exist. Developing a machine learner (i.e., implementing a machine learning algorithm from scratch) 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.

Some people may not like Weka. Other good packages are available (for example, Orange (`http://orange.biolab.si/`), PyML (`http://pyml.sourceforge.net/`), or `skikit-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 provide a transformation of the data suitable for use in Weka (an arff file) and Orange, and scripts to process the output as necessary.

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

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. This data will be available on CIS servers in the directory `/home/subjects/comp90049/2014-sm2/project2`.

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[1]. If you are using an **unsupervised** classifier, you might skip this step altogether.

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 this, and submit it along with your report; we will use this output to confirm the observations of your approach.

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.

To add a bit of fun to the project, and to give you the possibility of evaluating your models on the test set, we will be setting up this project on Kaggle in Class (`https://inclass.kaggle.com`). You will be able to submit results on the test set there, and get immediate feedback on how well your system is doing on the test data. There is a Leaderboard on the site, that will also allow you to see how well you are doing as compared to other classmates participating on-line. This will be optional, but we encourage you to participate. More details on this will follow next week via the LMS.

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

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

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, 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.

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.

For students taking this subject at a Master's level, the report should be 1000-1500 words; for students taking this subject at an undergraduate level, it should be 700-1000 words. Your report should aim to provide a basic description of the task, your approach to generating features, machine learning approaches, your observations, and your critical analysis. The critical analysis is key; please think carefully about the following questions:

- Does your classifier do a good job at addressing the task? Why or why not?
- Why is the method(s) you explored a reasonable strategy for approaching the task? What advantages does it have over other possible methods?
- If you engineered new features, why did you use them? What aspect of the data set are they attempting to model?
- What evaluation strategy did you use? Based on this evaluation, does your model seem to be a good one?
- Be sure to support your statements and analysis with examples.

Your report should include a bibliography of relevant or important piece of (**peer-reviewed**) academic literature: research is not done in a vacuum, and it is inappropriate, inadequate, or intellectually dishonest not to cite (or to mis-cite) relevant work. Note that Wikipedia is **not** appropriate as a primary reference (for an overview, see `http://en.wikipedia.org/wiki/Citing_Wikipedia`), although it can occasionally be a good place to start. If you directly use information from a website, e.g., a blog or technical forum, please also be sure to cite those resources.

There are numerous texts in the Readings sections of the LMS, from which you can get a sense of good structure and style — the content will probably not be relevant to your project, however. The templates for the Project 1 report should be used for this report as well. **Your report must be submitted in PDF format**: we reserve the right to return reports submitted in any other format than PDF with a mark of 0.

Note: **the report should be anonymous**, i.e. it should have no mention of your name or student number. This is for the reviewing process below.

(continued . . .)

Stage II

After the reports have been submitted, there will be a five day period where you will review 2 papers written by your peers in Stage I.

The 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 few 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 few 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.**

Assessment

For students taking this subject at a Master's level, the project will be marked out of 20, and will be worth 20% of your overall mark for the subject. For students taking this subject at an undergraduate level, the project will be marked out of 15, and will be worth 15% of your mark. Note that there is a hurdle requirement on your combined project mark (20/40 for Master's students; 15/30 for undergraduate students).

The mark breakdown will be as follows:

Category	COMP90049	COMP30018
Critical Analysis	8	7
Feature Engineering	2	0
Report Quality	5	4
Creativity	2	1
Reviews	3	3
Total	20	15

Submission

Submission will take place in two locations:

- You will need to upload your code, README and test outputs to a directory on the CIS servers, in the same manner as for Project 1.
- The PDF of your report will be submitted to the appropriate project within Turnitin. Note that Turnitin will accept other report formats (`doc`, etc.) — we will **not** accept them.
- Your review will also be done via Turnitin, through the corresponding project (Project 2 Peer Review)

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

While it is acceptable to discuss the project with other students in general terms (including the Discussion Forum), excessive collaboration will be consider cheating (in particular, sharing of methodological details or code). We will be carefully examining the output files, README and report for evidence of collusion, and will invoke the University's Academic Misconduct policy (`http://academichonesty.unimelb.edu.au/policy.html`) where inappropriate levels of collaboration or plagiarism are deemed to have taken place.

Late submissions

We will not accept late submissions under **any** circumstances. The reason for this is that submission is late in semester — we would not want to disrupt your exam preparation (and to miss out on getting peer reviews for your paper!). Furthermore, the submission deadline **will not be extended**, because of the timing of the reviewing process.

If there are documentable medical or personal circumstances which have taken time away from your project work, you should contact Jeremy via email (`nj@unimelb.edu.au`) at the earliest possible opportunity (this generally means well before the deadline). We will assess whether special consideration is warranted, and if granted, will scale back the expectations proportionately (e.g. if you have been unable to work on the project for a total of 1 out of 2 weeks, the expectations will be reduced by around 50%). No requests for special consideration will be accepted after the submission deadline.

Changes/Updates to the Project Specifications

We will use the LMS to advertise any (hopefully small-scale) changes or clarifications to the project specifications. Any addendums made to the project specifications via the LMS will supersede information contained within this document.

References

[1] Cheng Z., Caverlee J., and Lee K. You are where you tweet: A content-based approach to geo-locating twitter users. In *CIKM'10*, Toronto, Ontario, Canada, 2010. ACM.