Department of Computing and Information Systems
The University of Melbourne
COMP30018/90049 Knowledge Technologies, Semester 2 2016

Project 1: Misspelled Location Names

Due: Friday 02 September 12pm

Submission: Source code, README to the MSE servers;
PDF Report to Turnitin

Assessment criteria: Critical Analysis; Soundness; Report Quality; Creativity.

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 tweets from Twitter. Your objective is to 1 develop software that identifies misspelled location names in these tweets, and to 2 analyse the performance of your tool.

The particular 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. Technical merit and good programming practice will be not be directly relevant to the assessment of your submission. Impeccably-coded software with little insight will not receive a strong mark.

Resources

There are two datasets: 1 a dictionary ("gazetteer") of US location names, and 2 a number of documents ("tweets") in which to search for the location names. Note that many tweets will not contain any location names at all.

You may find the datasets on the MSE servers:

- The gazetteer can be found in `/home/subjects/comp90049/2016-sm2/project1/geonames/` — there is a full version called `US.txt`, described in the READMEs in that directory. As well, we have provided a simplified version called `US-loc-names.txt` where we have extracted the `asciiname` field, sorted the location names, and removed duplicates. This latter file, comprising about 1.3M locations, one per line, is the main dictionary for this project; the larger file contains other ancillary information that you may use as you wish.
- The tweets have been personalised, and can be found in the following directory: `/home/subjects/comp90049/submission/your-login/` where `your-login` is your central University log-in (and the one you use to connect to the MSE servers).

There is a set of about 400K tweets called `your-login_tweets.txt` and a small (~4K) random sample called `your-login_tweets_small.txt` in that directory.
In the tweet file, each tweet is separated by a new-line character[1] and has some metadata as follows:
`user_id (tab) tweet_id (tab) tweet_text (tab) time_stamp (newline)`

Terms of Use

As part of the terms of use of Twitter, in using the data you agree to the following:

- You are strictly forbidden from redistributing (sharing) the dataset with others or reusing it for any purpose other than this project.
- You are strictly forbidden from reproducing documents in the document collection in any publication, other than in the form of isolated examples.

Additionally note that the document 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 my using them does not constitute endorsement of the views expressed within. We recognise that some of you may find certain of the documents in bad taste and possibly insulting, but please look beyond this to the task at hand. The University of Melbourne accepts no responsibility for offence caused any content contained in the documents.

Task 1: Software

地名词典

You should aim to write a (simple) software system which automatically attempts to find location names in tweets based on the gazetteer. The location names may be misspelled or otherwise vary from the official name; your system should allow for this. There is no single "best" way of doing this; you might like to take some of the following ideas into account:

- 非字母字符
  Some of the tweets contain non-alphabetic characters: you might wish to remove them
- The location names generally begin with capital letters, but sometimes users on Twitter might write them in lower-case 首字母大写 / 小写
- Some location names comprise more than one word; a large number of location names will trivially not be found anywhere in the collection
  琐细的
- The dictionary is roughly the same size as the document collection; consequently it is probably of similar computational complexity to compare each entry from the dictionary against the entire collection, as it is to compare each entry from the document collection to the entire dictionary
- 原型
  The document collection is somewhat large; you might find it useful to build a prototype on the `small` dataset, and only run it on the larger dataset when it is working to your satisfaction. If working with the entire dataset is too challenging, you may reduce it as necessary — if you do so, you should indicate in your report how much data you analysed, and how it was generated (as necessary)

[1] Beware, Windows users, that there is no carriage-return, so some text editors won't render it properly.

In the lectures, we have discussed a number of methods for attempting to solve the approximate string search problem.

- Students taking **COMP30018** should attempt **one** (or more) of these methods.
- Students taking **COMP90049** should attempt **two** (or more) of the methods, and *suitably contrast them.*

1. You may process the texts (and possibly the queries) into separate tokens ("words"). If you use this approach, you should identify its advantages and disadvantages in your report. You might then:

   (a) Use a Global Edit Distance matching strategy for each token. You will need to choose appropriate values for the operation scores, and threshold sensibly. Looking at all of the tokens may take a long time.

   (b) Use a N-gram Distance matching strategy for each token. Thresholding is the most difficult problem here. This might be the fastest approach, but may give poorer results.

2. You may treat the entire collection as a monolithic entity 整块实体, and apply the approximate matching technique(s) to this directly. If you use this approach, you should identify its advantages and disadvantages in your report.

   (a) Use the Neighbourhood Search. `agrep` is a good tool for this, and is available on the MSE servers. However, you will need to consider what a suitable regular expression "query" will look like in this context.

   (b) Use a Local Edit Distance matching strategy for each location name (for example). Note that this isn't how approximate string search was framed in this subject, but it is a legitimate strategy to attempt to find one (or more) instances of the location name in the tweet collection.

3. You might also consider Phonetic 语音的 methods like Soundex 探测法, either to direct compare to the location names, or as a pre-processing step before one of the other methods.

The above is not an exhaustive list of possibilities, other strategies are possible as well; you might find some of them by reading through relevant published literature.

It is recommended that you use pre-implemented versions of these methods where possible; please don't spend all of your time in software development, as the main focus of the assessment is the report (below). If you do use external packages (or supplied code), its source should be clearly indicated in your `README`.

In general, the purpose of the software is to inform your analysis; other technical details of the system (e.g. output format) are up to you. You should briefly (in a couple of sentence) describe the basic mechanisms of your system and how to use it in your `README`. (Note: **not** in your report.)

Task 2: Evaluation

You should attempt to ascertain 查明 the effectiveness of your system, preferably by estimating its **Precision**, but you may use other evaluation mechanisms as you find suitable. To do this, you will need to take a look at a few of your systems' results (by hand), and decide whether the match represents an actual instance of the location name (to the best of your ability).

Note that a large number of matches will be **exact** matches; this should not comprise the main focus of your evaluation — in fact, we would prefer that you exclude exact matches from your evaluation.

For example, `Seattle` appears in the gazetteer as a location name. Let's say we observed approximate matches in the following three tweets:

```
I'm leaving for Seattle tomorrow #EmeraldCity
Seatle Mariners gonna win the World Series!!!
MY SEAT LEGS ARE TOO HIGH AND I CANT REACH THE GROUND
```

The first instance is an exact match[2] and it should not be counted. The second is arguable: the "Seattle Mariners" are a sports team based in this location, but it isn't the location exactly — one could reasonably argue that this is indeed an instance of the location name, or that it isn't. Note that this problem is too generally too difficult to solve using the methods that we've discussed in the subject to this point. The third instance is clearly not an incidence of the location name, howver, even though many of the characters are present.

You should **choose a few illustrative examples of the behaviour of your system(s)** to demonstrate the performance of the method(s) for your report.

Task 3: Report

- **COMP30018** students will write 750–1250 words.
- **COMP90049** students will write 1000–1500 words.

The report should discuss:

1. A basic description of the problem and the 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 location names of more than one word (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. Some conclusions about the problem of searching for misspelled location names within a collection of tweets

Note that we will be looking for evidence that you have thought about the task and have determined reasons for the performance of the method.

While 1000 words might seem long at first glance, it isn't: being concise will be invaluable, and overlong reports will be penalised. You will not have space to discuss the technical elements of your implementation; you should focus primarily on the knowledge you have gained in your experiments. Spending more time on your report will allow you to use the word limit more effectively.

[2] Of course, observing exactly the same string is no guarantee that it actually corresponds to the location in question! However, it tells us little about the problem of **approximate** string search regardless.

Submission

Submission will entail two parts:

1. An archive which contains your code and a `README` file, which you will upload to the same directory on the MSE servers where you downloaded your tweet file, namely: `/home/subjects/comp90049/submission/your-login/`

2. Your written report, as a single file in Portable Document Format (PDF); this should be uploaded to Turnitin via the corresponding link on the LMS. This will be made available one week before the deadline.

Your software can be implemented in any programming language or combination of programming languages. There is no requirement that it runs on the MSE servers; please be considerate of other users if you run your system there — the task may be very memory–intensive.

If your report is submitted in any format other than PDF, we reserve the right to return it with a mark of 0.

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.