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

Project 1: What's in a (Persian) name?

IP?

Due:	3pm (15h00 UTC+11), Fri 7 Apr 2017
Submission materials:	Source code, `README` to the CIS servers (see below); PDF Report to Turnitin (also below)
Assessment criteria:	Analysis, Method, Report Quality
Marks:	The project will be marked out of 20, and will contribute 20% of your overall mark for the subject.

Overview

Persian (or Farsi) is an Indo-European macro-language spoken by roughly 60M people, mostly in Iran, but also Afghanistan, Tajikistan, and elsewhere. Its writing system is primarily the Perso-Arabic script, which is used for numerous languages. This script is notable because it is an *abjad* — (short) vowels are often not depicted in the orthography (spelling) of a word.

One interesting consequence in this script (and many others) is **transliteration**: how to represent a word from a foreign language, which often has sounds that cannot be (or, in this case, usually are not) represented in the native orthography. This project will involve so-called **back–transliteration**: given a representation in the Persian script, what was the original foreign word being represented?

For example, the Persian راو (written right-to-left), might correspond to the name "Rao", and جرمی to "Jeremy". We will be approaching this problem primarily as spelling correction[1], by comparing a name in Persian script (but see below) to a dictionary of names in Latin script, to determine the best match.

You will be required to develop one or more programs which implement the methods and evaluation that form the technical side of this project. However, the bulk of the evaluation will be in terms of a report which details your analysis and insights about using spelling correction to solve the problem of back–transliteration in the Persian language. Consequently, you are advised to make use of available code fragments and packages wherever possible: an impeccably-coded implementation with poor analysis will not receive a strong mark.

[1]You will probably discover that this is not an optimal strategy for actually solving the back–transliteration problem, but it will hopefully be instructive nonetheless.

Resources

You will be given the following resources:

- A list of about 13K names in the Persian script, with their (lowercased) equivalent in the Latin script (`train.txt`). The format of this file is:

  ```
  PERSIAN-NAME \t latin-name \n
  ```

- A list of about 2K names in the Persian script, without their Latin equivalent (`test.txt`). The format of this file is:

  ```
  PERSIAN-NAME \t ??? \n
  ```

- A list of about 26K (lowercased) names in the Latin script (`names.txt`), one per line. All of the Persian names are in this list, but not all items in this list are present in the Persian lists.

In these files, we have **transcoded** the (right-to-left) Persian script into a (left-to-right) Latinate representation according to the DIN 31635 standard, with a few small changes[2]. For example, راو above would be represented in `train.txt` as:

```
RAV \t rao \n
```

and جرمي as:

```
JRMY \t jeremy \n
```

These alterations mean that you will not have to worry about dealing with the encoding of the strings. Additionally, it will not be necessary to read the Persian script.

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, as either (or both) of the following publications:

> Sarvnaz Karimi, Andrew Turpin, and Falk Scholer (2006) English to Persian Transliteration. In *Proceedings of the 13th Symposium on String Processing and Information Retrieval (SPIRE'06)*, Glasgow, UK, pp. 255–266.
>
> Sarvnaz Karimi, Andrew Turpin, and Falk Scholer (2007) Corpus Effects on the Evaluation of Automated Transliteration Systems. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL'07)*, Prague, Czech Republic, pp. 640–647.

[2]Namely: (i) we have removed diacritics, for example, س "s" and ش "š" are both represented as `S`; (ii) each character is always written the same way, for example و is always `V`, even though it could also be `U` or `W` and so on; (iii) we have uppercased all of the letters. For more details of the transcoder, you can examine the chart at `https://en.wikipedia.org/wiki/Persian_alphabet#Letters` — you might some discover some good hints there.

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

You have been warned.

These publications are freely available from the World Wide Web, you might like to read them to get a sense of style and structure (in fact, you can also regard the 2007 publication above as a sample paper for this project). These papers will probably not be useful in terms of content, however, as they mostly deal with the problem of building the corpus and forward-transliteration; with some effort, you might be able to find relevant publications on back–transliteration, but most of their methods will be beyond the scope of this subject (but see Extension below).

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`).

Technical Tasks

There are two sub-tasks in this project: (i) approximate matching, and (ii) evaluation.

Approximate Matching

One possible mechanism for approaching this problem is to treat the (transcoded) Persian names as "misspelled" versions of the Latin names — or *vice versa.* We have discussed a selection of Approximate Matching methods for tackling the problem of Spelling Correction: in this case, given the "correctly spelled" entries in the dictionary (of Latin names), which one (or ones) is/are the "best match" for a given "misspelled" (Persian) name?

We recommend using — at least at first — a Global Edit Distance strategy. (If you really don't want to do this, we will discuss an alternative below, but you can treat the below as a standard submission.)

Global Edit Distance

Recall that the Global Edit Distance is a score associated with the optimal set of operations (subject to some scoring parameter) by which one string is transformed into a second string.

The mechanism for this strategy could then be as follows:

- For each (transcoded) Persian name:
  - For each Latin name:
    - calculate the Global Edit Distance between the Persian name and the Latin name
  - Whichever Latin name has the best Global Edit Distance score, will comprise the best match for this Persian name (breaking ties somehow)

This way, each Persian name in `test.txt` can be associated with some predicted Latin name(s).

We haven't specified the $[m, i, d, r]$ parameter; you may use whichever parameter you like (e.g. Levenshtein distance). Tuning this parameter can form part of your Extension, if you wish.

For almost any programming language that you might like to work this, there is probably a package or code fragment which implements the Edit Distance algorithm. Consequently, you should be able to get a simple implementation (so you can proceed to the Evaluation step) with little effort.

Modifying the `r` parameter

In all likelihood, a basic implementation of Global Edit Distance will not work very well in finding the correct name. One type of name that you will probably discover that this system predicts correctly are the ones where the orthography is exactly the same between the two scripts; for example, `BYRD` and `byrd`. You should examine your results to observe which other sorts of names that this system predicts correctly (to discuss in your report).

There are quite a few reasons why this system might fail to find the correct names: you should consider the fact that not all replacements are equivalent. For example, substituting `K` with `c` is far more likely than substituting `K` with `m`. This phenomenon is fundamental in Computational Genomics[3] — one particular example is the BLOSUM62 matrix[4], which scores the replacement (and implicitly, matches) for various amino acids.

You should intend to modify your basic Global Edit Distance strategy to account for this phenomenon. Based on the lecture pseudocode, this would amount to altering the behaviour of the `equal` function, to account for different match/replacement scores. You may use a complete matrix, like BLOSUM, or, you may choose some particularly obvious or interesting substitutions, and create a partial matrix or a set of *ad hoc* score assignments.

Once you have modified your scoring method, you should discuss in your report how the performance changes in terms of your chosen evaluation metric, and choose some illustrative examples of names which change their behaviour due to the modified parameter. If you wish, you can consider and discuss a number of different parameter scores as your Extension, or — for more of a challenge — you can examine some systematic method of deriving the scores for the matching matrix based on the given labelled data (`train.txt`).

Evaluation

Once you have one or more systems that can assign a Latin name from the dictionary to a given Persian name, you will be required to assess the **effectiveness** of the system.

You should evaluate your system by predicting a Latin name (or in some cases, multiple names) for all (or some) of the Persian names in the first column of `train.txt`, and then comparing your predictions with the correct name given the second column.

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

- Accuracy: assuming your system predicts a single Latin name for each Persian name, what proportion of them are correct?
- Precision: out of all of the Latin names predicted by your system, what proportion of them matches the correct answer? (This is particularly meaningful if your system predicts more than one Latin name per Persian name.)
- Recall: out of all of the correct Latin names, what proportion are found by the system? (This is particularly meaningful if your system cannot predict a Latin name for some of the Persian names.)

[3]See, for example `https://en.wikipedia.org/wiki/Substitution_matrix`
[4]`https://en.wikipedia.org/wiki/BLOSUM`

Extension

A "basic" submission will correspond to:

- implementing and evaluating one Global Edit Distance system (based on a single parameter setting)
- altering the `r` to adapt the score according to which characters are being replaced, and observing how the evaluation changes
- writing a report which details some **knowledge** about this problem

Part of the "Method" marking criteria will be in terms of extending this in some way. Some suggestions (about tuning the parameters) have been made above, but there are other options you might consider:

- Implementing and evaluating a different spelling correction method, like Local Edit Distance or N-Gram Distance. You should then contrast its behaviour with your Global Edit Distance system, by evaluating and choosing some illustrative examples.
- Implementing the Soundex algorithm by transforming the English names (and adjusting your matching mechanism appropriately), or, for more of a challenge, deriving a comparable transformation table for the Persian strings. You should then contrast the behaviour of this approach with your Global Edit Distance system, by evaluating and choosing some illustrative examples.
- Find and implement a method from the literature (for example, a Finite State Transducer, a probabilistic method like the one for forward–transliteration in Karimi et al. (2006), or an approach which draws on methods from the Machine Translation community). Note that this will probably be more difficult, so you may attempt it **instead** of the Global Edit Distance described above (but you will still need to Evaluate).

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 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, including some indication of how you determined the parameters (where 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 using approximate matching methods to perform back–transliteration.

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; or have discerned properties of the Persian language based on the given data (preferably supported by citations to relevant literature); 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 the papers cited in the Terms of Use. You might also consider the bibliography from the Approximate Matching lecture slides, 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`.

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:

- The code for your approximate matching system(s) 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 following directory:
  `/home/subjects/comp90049/submission/your-login/`
  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.

- 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.

The marks available will be as follows:

Analysis	8
Method	7
Report Quality	5
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 the marks will be adjusted as follows:

- 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.