**User Guide**

**Introduction**

**The problem**

Several databases contain identical information using similar spelling (e.g. Bogota Vs. Bogotá, San Luis de Potosi Vs S. Luis Potosí). This makes it difficult to integrate them. This software aims to address this problem, identifying the best match for a word. The standard spell check algorithm (Dynamic Programming) is not computationally efficient with big databases.

**Usage**

[**Video (Agrosavia Internal Only)**](https://corpoicaorg-my.sharepoint.com/:v:/g/personal/erubagotti_agrosavia_co/EZXtjGq4yyRKg6umjV1MlvYBodvIIQvu7IdcLrrkhB2cJA?e=heWDbC)

*Step 1 Index preparation*

To create the Index: perl rotateAndReverse.pl dictionary

The file dictionary contains the list of all allowed words

It outputs dictionary.bwt.sorted

*Step 2 Query*

Perl 1.spell.check.pl file2correct

Un video

**The algorithm**

[Video Explanation(Spanish only)](https://corpoicaorg-my.sharepoint.com/:f:/g/personal/erubagotti_agrosavia_co/EsaQKBDucX9PsKMZjnDECWABeDviAUof08SvzpwdcVyRzQ?e=hbvVBB)

Step 1- Indexing of the database

1. Compile an exhaustive list of labels in the database;
2. All suffixes and prefixes of each single word are inserted in an array ([circular shift](https://en.wikipedia.org/wiki/Circular_shift)) with 2 columns. The first column contains the original word, the second column contains the suffix or prefix;
3. The array is sorted alphanumerically.

Step 2-The “query” step

1. Each suffix and prefix of the substrings of the “query” is located in the array (if existed) and an hash containing the corresponding word is increased of 1;
2. Use binary search to locate the point in the array;
3. The word with the highest value is chosen as match.

**Circular shift**

It is an operation applied to a string. It consists in appending the last char of a string to the first one. E.g. the circular shift of the string 0123456789^ is 9^012345678. To identify where a word starts it is used the EOL char ‘$’. As a consequence the circular string of 0123456789$ would be 9$012345678

**Ordering Circular Shift**

By producing all circular shift of a set of words and ordering them in alphabetical order we will identify matches between subwords.

**The data structure**

Here I should describe the circular alphabet file.

**Experiments**

Here I should describe the experiment.

The final version is in a private repository at <https://github.com/enricorubagotti/normalize>