CS6370: Natural Language Processing
Spell Check Assignment

1 Problem statement

All of you must have experienced the power of the Google Spell Checker. This is an opportunity to get a hands-on experience into building such a system. The assignment consists of three parts:

1. **Word spell check -** standalone erroneous words are given and you are supposed to suggest corrections.
2. **Phrase spell check -** words present in phrases need to be checked for spelling errors and corrected.
3. **Sentence spell check -** entire sentence needs to be checked for spelling errors.

As you might have guessed already, the code and the executable of these three tasks form the basis of the assignment.

Some points to note are the following :

- Only for *Word spell check*, you may assume that the incorrect word is not present in the dictionary.
- This may not be the case in phrases and sentences. For example, consider the following query: "peace of cake ". Here, the incorrect word *peace* would be in the dictionary. But the correct replacement, *piece*, needs to be suggested.
- Spelling errors need not always be caused by misspelt words. For example, "halloffame". Your spell checker shall be able to split them correctly.
- However, you can assume that there will be only one incorrect word in a query (phrase or sentence).

Think of the various issues that might come into play when you make your spell checker. How much of distortion would your program be able to recover from? Can you intelligently prune down the search space of candidate replacements? For the *Phrase spell check* and *Sentence spell check*, the context words are important. Does your spell checker make use of the longer contexts available in sentences to improve prediction accuracy? What is the smoothing technique you use to handle unseen ngrams?

Do use techniques like part-of-speech tags and phonetic algorithms to improve the performance of your spell checker as well as to score extra credits.

We will provide you with a sample dataset of typos for all the three tasks.

Remember that good performance on the sample dataset is a necessary but not sufficient condition for your algorithm, since the results on the test data might still surprise you. Think of inputs different from the sample dataset that might be given to a standard spell checker.

2 Resources

You may use the following resources for designing your spell check program.

- The **expected** minimal dictionary is: `http://norvig.com/ngrams/count_1w100k.txt`. You may use additional dictionaries of your choice.
- Frequencies of various bigrams and trigrams - `http://norvig.com/ngrams/`.
- Frequent n-grams data based on Corpus of Contemporary American English -`http://www.ngrams.info/`. You need to register your email to get the resources.

To improve your system, you can use any additional data or resources along with this. Some suggestions on **Additional Datasets** are :

- WordNet as a dictionary. WordNet is downloadable from `http://wordnet.princeton.edu/`
- The Brown Corpus (tagged with part-of-speech): `http://nltk.googlecode.com/svn/trunk/nltk_data/packages/corpora/brown.zip`
- Reuters dataset: `http://www.daviddlewis.com/resources/testcollections/reuters21578/reuters21578.tar.gz`

To improve your background knowledge : **Papers (present in moodle)**

- Kernighan, Mark D., Kenneth W. Church, and William A. Gale. "A spelling correction program based on a noisy channel model." *In Proceedings of the 13th conference on Computational linguistics-Volume 2*, pp. 205-210. Association for Computational Linguistics, 1990.
  (Note that the above paper deals only with single correction per word. But the input to your program can contain more than one corrections per word. For example, ocassion contains a deletion as well as an addition.)

- Golding, Andrew R. "A Bayesian hybrid method for context-sensitive spelling correction." *arXiv preprint cmp-lg/9606001* (1996).

For any other resource(s)/help that may be required, you may approach the TAs.

3 Evaluation

For **Word spell check**, you can print a maximum of 10 suggestions per query in descending order of relevance. For **Phrase spell check** and **Sentence spell**

check, you are allowed to print 3 suggestions.

The evaluation will be done using the Mean Reciprocal Rank (MRR) measure. Please go through the url `http://en.wikipedia.org/wiki/Mean_reciprocal_rank` for more information.

Consider the following example of evaluation with Mean Reciprocal Rank Measure.

Incorrect Input sentence : The departments of the institute offer corses, conducted by highly qualified staff.
Incorrect Word : corses.
Correct word : courses.

Depending on the ranking of the suggestions, Mean Reciprocal Rank will differ.

Output #1 : courses, horses, corset - correct word is at rank 1, so Mean Reciprocal Rank is 1.
Output #2 : horses, courses, corset - correct word is at rank 2, so Mean Reciprocal Rank is 1/2.
Output #3 : horses, corset, courses - correct word is at rank 3, so Mean Reciprocal Rank is 1/3.
Output #4 :horses, corset, scores - correct word not present, so Mean Reciprocal Rank is 0.

The MRR for all the test cases will be summed up to get a measure of the performance of the spell checker system you designed. **The time taken by your program to execute will also be taken into consideration.** Along with the code, you will also have to submit a report containing the details of your algorithm and the observations you have made.

4 Submission Guidelines

Note that the evaluation for the assignment is automated. You need to strictly abide by the following **guidelines**:

– **Folder structure:**

```
TeamNumber.zip (For e.g., Team10.zip)
..TeamNumber/
....Report_TeamNumber.pdf (E.g., Report_Team10.pdf)
......src/ (Contains all the source codes)
......bin/ (Contains all executables generated by the Makefile)
......data/ (Contains all data required for the programs to run)
......Makefile
......README (All packages dependencies should be mentioned
                along with the instructions to run your program
                on the terminal)
```

– **Output format:**
  The output must be in the following specific format:

  query word 1⟨tab⟩ suggestion1 ⟨tab⟩ score ⟨tab⟩ suggestion2 ⟨tab⟩ score
  query word 2 ⟨tab⟩ suggestion1 ⟨tab⟩ score

  Here, *query word* is the incorrect word and *score* refers to the values based on which you have ranked your suggestions.

Please make your programs bug free and adhere to the guidelines failing which your team will incur penalty during the assignment evaluation.

Plagiarism in any form - report or code - is intolerable. Copying contents verbatim from any paper or even references is strictly not allowed. If you are found to engage in plagiarism, it will be treated very seriously and will result in a U grade (irrespective of other course evaluations). This may also be forwarded to the Dean, Academic Courses for further action. All your project reports and codes will be verified using plagiarism detector tools.

Intermediate deadlines for the assignment are as follows:-

– Sept 10 - Word Spell Check
– Sept 19 - Phrase and Sentence Spell Check
– September 22 - Report and Code Submission

The final deadline has been set as **22nd September 2016 23:55**. This is a **hard deadline and non-negotiable**.

However you have the option of early submission :) Teams which are able to complete the assignment well ahead of deadline can make use of the time to draft your project proposal. As a good proposal can take your project a long way in earning good credits, do try to save time !