# CS6370: Natural Language Processing
# Spell Check Competition

February 24, 2018

## 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 spell checker consists of three parts:

1. **Word spell check** - standalone erroneous words are given and you are supposed to suggest corrections. You may assume that the erroneous word is not present in the dictionary.

2. **Phrase spell check** - words present in phrases need to be checked for spelling errors and corrected. In phrases and sentences, spelling errors need not always be caused by misspelt words. 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. However, you can assume that there will be only one incorrect word in a query (phrase or sentence).

3. **Sentence spell check** - entire sentence needs to be checked for spelling errors. In general, sentences are longer than phrases.

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? You can also make use of techniques like part-of-speech tags and phonetic algorithms to improve the performance of your spell checker and to score extra credits.

## 2 Competition

Where is the fun in spell check if there is no competition? And the best way to learn is to compete with your own classmates!

Your score (see Section 4) is going to determine your rank in the class. During the competition, there will be an intermediate evaluation where each team's score is calculated and announced in Moodle along with their rank. The teams will then have a chance to improve their scores till the final deadline.

## 3 Resources

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

- The **expected** minimal dictionary is `http://www.bragitoff.com/wp-content/uploads/2016/03/dictionary.csv` You may use additional dictionaries of your choice, in which case, you must ensure that your additional dictionaries doesn't have misspelt words.

- 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 :

- `http://norvig.com/ngrams/count_1w100k.txt`
- 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.

## 4 Evaluation

Evaluation for word spell check will be based on Jaccard Similarity between your suggestions and expected corrections.

Evaluation for phrase/sentence spell check will be based on the Mean Reciprocal Rank (MRR) measure. Please go through the article `http://en.wikipedia.org/wiki/Mean_reciprocal_rank` for more information. 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.**

## 5 Input Format

Input to your spell checker will be a text file. Each line contains a new query for word, phrase and sentence spell check.

## 6 Output Format

The output file must be in the following specific format:

**query1** ⟨tab⟩ suggestion1 ⟨tab⟩ suggestion2 ⟨tab⟩ suggestion3 . . .
**query2** ⟨tab⟩ suggestion1 ⟨tab⟩ suggestion2 ⟨tab⟩ suggestion3 . . .

Here, *query* is the incorrect word. For Word spell check, your program must output 10 suggestions for every query in **descending order of relevance.** For Phrase and Sentence spell check, your program must output only the incorrect word along with atmost 3 suggestions for every phrase/sentence. (You need not output entire phrase/sentence).

Sample input and output files are uploaded in Moodle.

## 7 Submission Guidelines

Along with the code, you will have to submit a detailed report explaining the algorithm and your analysis of results. Please make your programs bug free and adhere to the guidelines failing which your team will incur penalty during the assignment evaluation. During submission, you must follow the folder structure given below.

```
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)
```

## 8 Important Dates:

- February 24 - Assignment release date.
- March 10 - Progress meeting with TAs.
- March 12 - Intermediate deadline (code only).
- March 15 - Intermediate evaluation and announcement of scores.
- March 20 - Final deadline (code and report).

All deadlines are 11:59 PM. These are **hard deadlines and non-negotiable**.

**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.**