# CS6370: Natural Language Processing
# Spell Check Assignment

## Word spell check

**Problem**

The task is to predict possible corrections to the stand-alone erroneous words that are given.

**Dataset**

Natural language corpus as a dictionary.

**Approach**

We tried three approaches:

**Approach 1: The Naïve Spell checker**

This is based on Peter Norvig's approach. It generates all possible words at the edit distance of at most two from the error word and then pruning those who are not present in the dictionary. But when we increased the edit distance to incorporate more words, the number of irrelevant candidate generation just blew up exponentially. The generation of candidates that were at higher edit distance and had some relevance to the error word became difficult.

**Approach 2: Based on - "A Spelling Correction Program Based on a Noisy Channel Model".**

- Instead of taking the words that are at a fixed edit distance from the error word we used bigrams and trigrams of the word to come up with candidates as follows.
- Created the inverted index of bigrams and trigrams of all the words.
    - For each bigram or trigram, it has set of words associated with it. These are stored according to their length.
    - Confusion matrices of insertion, deletion, substitution and transposition are taken from the Noisy channel paper.
- The bigrams and trigrams are computed for the given error word.
- All the words that occur in the range of ±k length of the error word for each of its bigram and trigram are taken as the candidate word.
- The prior for these candidates is taken from the natural language corpus provided.
- The edit distance is computed using standard Wagner-Fisher's algorithm and all the letters that are inserted, deleted, replaced and transposed are stored. These are used for getting the conditional probability from the confusion matrices. This is done as follows:

$$\Pr(t|c) = \begin{cases} \frac{del[c_{p-1}, c_p]}{chars[c_{p-1}, c_p]} \ if\ deletion \\ \frac{add[c_{p-1}, t_p]}{chars[c_{p-1}]} \ if\ addition \\ \frac{sub[t_p c_p]}{chars[c_p]} \ if\ substitution \\ \frac{del[c_p, c_{p+1}]}{chars[c_p, c_{p+1}]} \ if\ reversal \end{cases}$$

where t is error word and c is the correction word.

- The candidate correction score is given using the Bayesian model as:

$$\Pr(c|t) = \Pr(c) \Pr(t|c)$$

- Add one smoothing has been used, to avoid the cases when the given word is not present in the corpus.

**Approach 3: Incorporating phonetics**

Computed the phonetic score using the "Double Metaphone" algorithm. The “Double Metaphone” algorithm generates two keys a primary and a secondary key. The scoring is based on the rule if primary matches primary then it's a strong match, if primary matches secondary then it is weak, when secondary matches secondary then it is minimal otherwise no match. So we gave a score of 10^4, 10^2, 10 and 1 respectively, and multiplied it to the previous score.

ERROR WORD BELIVE

CANDIDATE GENERATION

INVERTED BI-GRAM MATRIX

BELIVE

BE

4 — {BEEN, BEST, ...}

5 — {BELOW, BEACH, ...}

9 — {DESCRIBES, ...}

INVERTED TRI-GRAM MATRIX

BELIVE

BEL

4 — {BELL, BELT, ...}

5 — {LABEL, BELLY, ...}

9 — {BELONGING, ...}

CANDIDATE SCORING

PROIR SCORE

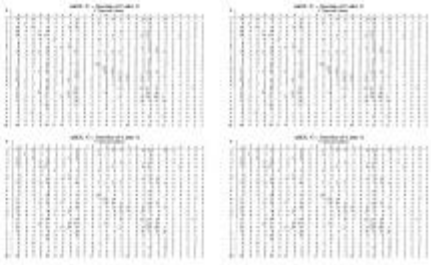
$$\text{Score} = \frac{\text{Frequency in Dictionary}}{\text{Word Count of Dictionary}}$$

CHANNEL NOISE SCORE

Formula for score

$$Pr(t|c) = \begin{cases} \frac{del[c_{p-1}, c_p]}{chars[c_{p-1}, c_p]}, & \text{if deletion} \\ \frac{add[c_{p-1}, t_p]}{chars[c_{p-1}]}, & \text{if insertion} \\ \frac{sub[t_p, c_p]}{chars[c_p]}, & \text{if substitution} \\ \frac{rev[c_p, c_{p+1}]}{chars[c_p, c_{p+1}]}, & \text{if reversal} \end{cases}$$

Probabilities calculated from confusion matrices



PHOENETIC SCORE

Primary key & Secondary key Generated by the Double Metaphone Algorithm

**Strongest Match**
Primary Key (1) = Primary Key (2) } 1000

**Normal Match**
Secondary Key (1) = Primary Key (2)
Primary Key (1) = Secondary Key (2) } 100

**Minimal Match**
Secondary Key (1) = Secondary Key (2) } 10

**No Match** } 1

**Implementation Notes**

We parallelized the execution of the program using the “joblib library”, to compute candidate scores providing a major boost in the execution time of the algorithm.

**Code**

spell_checker.py

**Sample Results**

Belive

BELIVE,BELIEVE,BELEIVE,BELIEF,BELIZE,BELIEVES,RELIVE,OLIVE,BELIEVED,BOLIVIA

bouyant BUOYANT,BOUND,BOUNTY,POUND,BANTU,BRYANT,POINT,MUTANT,BUTANE,COURANT

comitte
COMMITTEE,COMITE,COMMITTE,COMTE,COMMUTE,COMMITEE,COMMATE,COMMIT,COMPETE,COMET

distarct
DISTRICT,DISTRACT,DISTANCE,DESTRUCT,DISTINCT,DISTANT,DISTANCES,DISTRICTS,DISPATCH,DISTORT

extacy

ECSTACY,EXACT,EXTRACT,EXTANT,EXODUS,ECSTASY,EXPASY,STACY,ESTATE,EXTENT

failr

FAILURE,FLR,FAIL,FAIR,FILER,FAILS,FLAIR,FAYLOR,FILE,FAVOR

hellpp

HELP,HELPU,HELLO,HELLBOY,HELPS,HELLS,HELL,HELLER,HELLEN,CELLS

gracefull
GRACEFUL,GRACEFULLY,GRATEFUL,GRATEFULLY,PEACEFUL,PEACEFULLY,USEFULL,FATEFUL,TRACEABLE,STATEFUL

liason LIASON,LIAISON,LESSON,LAWSON,LIPSON,LEESON,LESION,LITEON,SEASON,LION

ocassion
OCCASSION,OCCASION,PASSION,LOCATION,FASHION,OMISSION,SESSION,ACTION,OCCASIONI,OPTION

possable
POSSIBLE,POSEABLE,PASSABLE,POSSIBLY,POSIBLE,POTABLE,PORTABLE,NOTABLE,CONSTABLE,PEACEABLE

thruout
THROAT,THROUGHOUT,THREAT,TROUT,THRUST,TRYOUT,TURNOUT,THROUGHT,THREAD,TORTUOUS

volly

FOLLY,VOLLEY,FULLY,VOLL,VALLEY,VILLA,FOLEY,VOL,VOILA,VALLI

tatoos

TATOOS,TATTOOS,DATOS,TATS,TATES,DATES,TODOS,DADOS,TABOOS,TOTS

respe

RESP,RECIPE,RSPB,RRSP,RSP,RASP,RESET,RES,RESPECT,REST

**Analysis**

All the results gave MMR 1 except thruout, hellpp and respe they have throat, help and resp respectively as best because it had lesser edit distance less as well as phonetically also its quite close.

Correct prediction for words like failr and extacy came to top position when we made use of phoenetics. Throughout came in top 10 for thruout when we changed the max edit distance to be searched from 2 to 3.

Belive and tatoos are the actual words present in the dictionary so they are coming as the best predictions, these can be removed from the best predictions, but we gave it as it is.

# Phrase spell check

**Problem**

The task is to correct the erroneous word in a phrase/ sentence taking the context of the erroneous word into account.

**Dataset**

COCA dataset for n-gram counts.

**Approach**

Ngram Model is used for this, which approximates the probability of the nth word based on the previous n-1 words. This assumption comes from the Markovian assumption, and the likelihood of sentence say S composed of words $w_1$ $w_2$ $w_3$ … $w_n$ can be written as:

$$P(S) = P(w_1\ w_2\ w_3 \dots\ w_n) = \prod_i P(\ w_i | w_1\ w_2\ w_3 \dots\ w_{i-1})$$

But while training it's a problem as the dataset can't capture all possible sequences. So the probability is computed using the chain rail so the expression comes up to:

$$P(S) = \prod_i P(w_i | w_{i-1})$$

There can be two types of errors in the phrase or sentence:

1. Non-real words.
2. Real words occurring in the wrong context.

We need to identify the error words in the phrase/ sentence.

- First, parse through the phrase/ sentence if it's not present in the dictionary then it's the error.
- When no error is detected, then take each word as the error word. Find n-grams associated with it and give a score to it. This is done for all the words, whichever gets the lowest score is the error word.

As we are supposed to deal with only one error in the phrase /sentence so we give priority to the spelling error.

Once the error is identified all the candidates from the word spell check becomes the candidate set.

We have considered all the possible n-grams for n = 2, 3, 4 and 5 which contains the error word. The error word is replaced with the candidate word, count for all the n-grams is taken and the log sum of these counts for each n gives the n-gram score for that sentence particular candidate as the replacement.

Score from the n-gram is computed for all the candidates and is multiplied with score obtained from the spellchecker. Whichever candidate gets the max score is taken as the correct replacement.

This spellchecker handles multiple errors but we are giving output for 1 error per sentence.

**Results**

**Analysis**

Phrase check is not giving good results as the n-grams to be considered are very less which affects the output. The correct prediction shows up in top 10 predictions but not in top 3. Due to the skew in scoring of phonetics used in the spellchecker's scoring; Masking of the context based scores resulting from the n-gram. Omission of the phonetics from the spellchecker score resulted in the desired correction not appearing in the top 3 candidates.

Our spellchecker is not producing predictions for words like halloffame, this can be handled by breaking the words and searching for each word in the dictionary.

Most of the results for sentences have MMR1 except for words like ramed for which the correct output didn't showed up agiain because of phoenitics.

# References

[1] Andrew R. Golding, A Bayesian Hybrid Method for context-sensitive spelling correction.
[2] Mark D. Kemighan, Kenneth W. Church, William A. Gale, A Spelling Correction Program Based on a Noisy Channel Model.