CS 251: Project Proposal WittyWords

Illuminati
160050040
160050068
160050107

21 September, 2017

Contents

1 Abstract

WittyWords will be a **proofreading rewriter** working over a web based application actualized utilizing **Natural Language Processing**. It aims at rectifying the spelling and grammatical mistakes in the input text and suggesting the user with an alternative to the given text in a much more refined way with an attractive and alluring dialect keeping the meaning identical.
We'll also be aiming to suggest proposals for some equivalent phrases to replace the ones inputted. Also we'll make a dictionary for popular **IITB lingos**, and replace them if they appear in the input.

2 Motivation

In many situations, you need to utilize your **writing aptitude** the first time you contact someone new, whether this is an application letter to a college/university, an email to a potential employer about a job or even when you compose a response to an exam question.
We all know that **first impressions** count and in these situations, what you write is like a way of introducing yourself to someone you have never met before. Hence proofreading becomes a need in such a situation.
Proofreading can be a dull errand for substantial records, and troublesome even for small ones. No-one wants to read the same thing twice.
Sometimes we compose crude sentences in a rush, so there is a need to restructure it for formal documents. This will then help one iron out any little, unnecessary errors one might have made.
Additionally when generating documents focus is primarily on the matter rather than on making use of presentable vocabulary and english. So rewriting can help one gain extra credits in let say exams or placements or it may even be a speech for the post of the President :P. Hence, it would empower one to stand out from the crowd.

3 Literature Survey

Several research papers and project reports were read as part of research for the project. Most notably:

I.) Naber, D. (2003). A Rule-Based Style and Grammar Checker

II.) Frequency based spell checking and rule based grammar checking. SP Singh(2016). In International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT 2016)

III.) SW-AG: local context matching for English lexical substitution. G Dahl(2007). In Proceedings of the 4th workshop on Semantic Evaluations (SemEval-2007), Prague, Czech Republic.

IV.) Context-Sensitive Arabic Spell Checker using Context Words and N-gram Language Models. Majed M. Al-Jaferi(2013) and Sabri A. Mahmoud(2013).In Taibah University International Conference on Advances in Information Technology for the Holy Quran and its Sciences.

4 Problem Statement

To make a proofreading rewriter implemented over a webapp which would correct spelling and grammatical mistakes and recommend contextual synonyms, hence refining the input text.

5 Software Requirements

- Python
  - NLTK
  - pycorenlp (Python wrapper for Stanford CoreNLP API)
- Django (for backend)
- HTML/CSS
- Javascript
- Git (for version control)
- Doxygen (for documentation)

6 Implementation

I.) Spell Check: We intend to use python's **symspell**[5] module to generate the confusion set, i.e. the set of words with edit distance less than or equal to 2 from the target word (word in question). Now we can use the **Context Words Method**[4] to find the correct word. In this method, given the context words c_j using a 2-word window(i.e. the 2 words before and after the target) of the target word , we will find out the proper word w_i from the confusion set that is most probable to the context. The probability of each w_i in the confusion set can be calculated using Bayes' rule :

$$p(w_i|c_{-2}, c_{-1}, c_1, c_2) = \frac{p(c_{-2}, c_{-1}, c_1, c_2|w_i)p(w_i)}{p(c_{-2}, c_{-1}, c_1, c_2)}$$

The 1st term in the numerator of RHS is inappropriate to calculate due to data sparseness. So, we assume that presence of one word in the context is independent of presence of others leading us to:

$$p(c_{-2}, c_{-1}, c_1, c_2|w_i) = \prod_{j \in -2,-1,1,2} p(c_j|w_i)$$

II.) Grammar check: First of all, we assign **Part Of Speech** (POS) tags to each word in the sentence. If the sentence structure matches any rule from a predefined set of **grammar rules**, then the sentence is assumed to be grammatically correct, otherwise we find the word for which the discrepancy occurs, and the corresponding tag [2]. If the error is due to a verb, we change the form/tense of the verb. If the error occurs due to say, the presence or absence of a determiner, then we remove/add a suitable determiner to the target position. For the tagging of words in a sentence, we intend to use NLTK, and for generating the parse tree of the sentence, we intend to use pycorenlp, one of the many wrappers for the Stanford CoreNLP API in python.

III.) Context based synonyms: First we generate the set of all **trigrams** (set of 3 consecutive words), containing the target word (upto 3), and remove the trigrams which do not contain any content words, and call the resulting set context trigrams. Open class words (nouns, lexical verbs, adjectives and adverbs with the exception of the verb to be) and pronouns are to be considered content words. We then one by one replace the target word in the context trigrams by the candidate replacement, and search for the frequency of the resultant trigram in the trigram corpus obtained from the internet containing 1 million different trigrams with frequencies. Then the **normalized**

score of each candidate is calculated using the given formula [3]. The candidates with the highest score are shown first in the suggestions.

$$score(c) = \sum_{t \in T} \frac{m(t,c)}{\sum_{x \in C} m(t,x)}$$

IV.) Finally, with each of the 3 subparts, i.e. Spell check, grammar check and the synonym suggester up and running, we develop a basic **web interface** using HTML/CSS and JavaScript, with Django for Python based backend, which gives on-the-go suggestions for spelling or grammar errors, and also for some nouns and verbs

7 Feasibility

As Natural Language Processing is a vast and greatly expanding subject, there is always scope for more. The algorithms being implemented do a very basic task of correction or suggestion. Also there exist more efficient algorithms for doing the job, but are too complex to be studied and applied with our present knowledge. In grammar check, as we expand the list of rules, the efficiency of the checker increases. But again, the speed can be a issue and we need to come to a good combination of both accuracy and speed. In context based synonyms, the larger the data we have, the more probability of finding a match for the context trigrams.

References

[1] Naber, D. (2003). A Rule-Based Style and Grammar Checker.

[2] Frequency based spell checking and rule based grammar checking. SP Singh(2016). In International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT 2016)

[3] SW-AG: local context matching for English lexical substitution. G Dahl(2007). In Proceedings of the 4th workshop on Semantic Evaluations (SemEval-2007), Prague, Czech Republic.

[4] Context-Sensitive Arabic Spell Checker using Context Words and N-gram Language Models. Majed M. Al-Jaferi(2013) and Sabri A. Mah-

moud(2013).In Taibah University International Conference on Advances in Information Technology for the Holy Quran and its Sciences.

[5] https://github.com/wolfgarbe/symspell

8 Deliverables

- Context Based Spell Checker
- Context Based Synonym Suggester
- Grammar Checker (for checking Tense, Subject-Verb-Agreement and Determiners)
- Web interface for the above All the deliverables were delivered satisfactorily.