CS 251: Project Report
WittyWords

Illuminati
160050040
160050068
160050107

24 October 2017

Contents

1 Abstract

WittyWords is a **proofreading rewriter** working over a web based **Django** application actualized utilizing **Natural Language Processing**. It rectifies the spelling and grammatical mistakes in the input text, giving contextual corrections to the spelling errors 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.
Also, we've made a dictionary for popular **IITB lingos** for suggesting appropriate corrections for them, to make the text look formal.

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
  - pattern3
- Django (for backend)
- HTML/CSS
- Javascript / AJAX / jQuery
- Git (for version control)
- Doxygen (for documentation)

6 Implementation

I.) **Spell Check:** First of all, we have used python's symspell module to generate the confusion set, i.e. the set of words with edit distance less than or equal to 1 (as the confusion set can be extremely large for small words) from the target word (word in question). Now we replace the target word with the candidate words one by one in the sentence, and get the frequency of the trigrams in which the candidate word occurs. Having got the frequency, we calculate the score of each candidate using the given formula

$$score(c) = \sum_{t \in T} \frac{m(t,c)}{\sum_{x \in C} m(t,x)} \tag{1}$$

If the score of any candidate is greater than a threshold, we include it in the suggestions.

II.) **Grammar check:** First of all, we assign Part Of Speech (POS) tags to each word in the sentence using Penn Treebank corpus of NLTK. For every word in a sentence, we check if the word is a verb, a determiner, or auxiliary verb. Accordingly we replace the word with all its possible alternate forms in the sentence, and get the frequency of the trigrams in which the candidate word occurs.
Having got the frequency, we calculate the score of each candidate using the formula (equation 1) and if the score of any candidate is greater than the threshold times the score of the original word, we include it in the suggestions. In the case the word to be checked is a verb, we use the module pattern3 to generate the different forms of the verb.

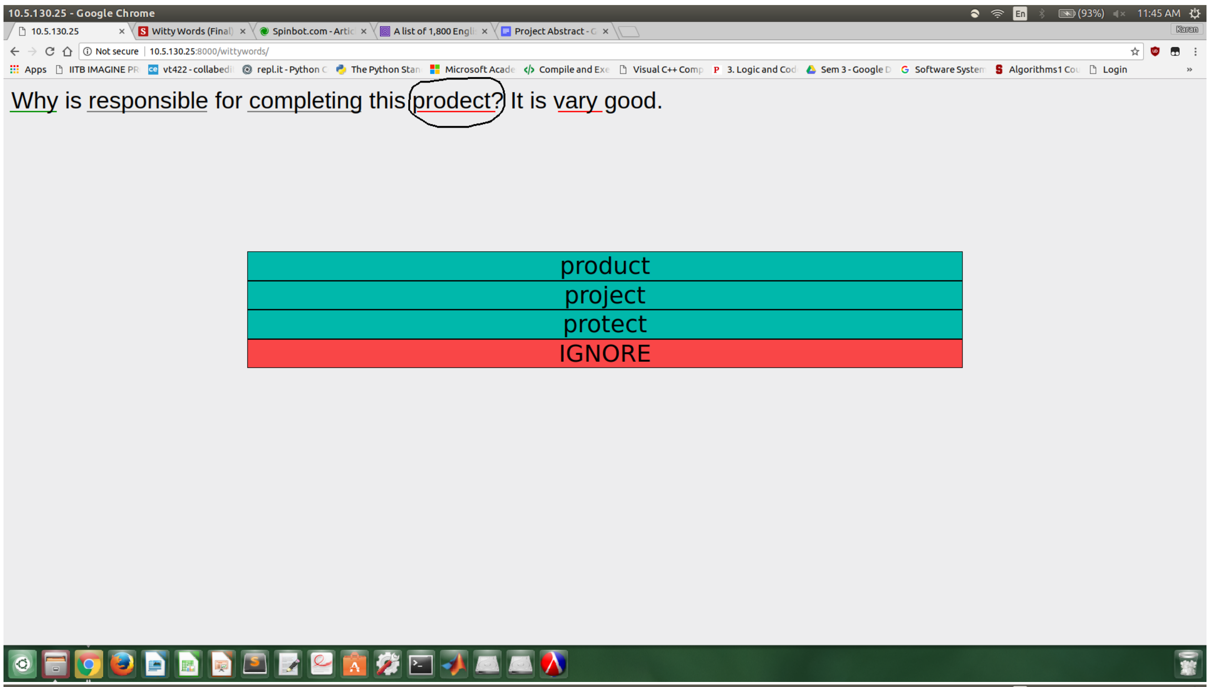
III.) **Context based synonyms:** First we generate the set of all trigrams (set of 3 consecutive words), containing the target word (upto 3). We then one by one replace the target word in the trigrams by the candidate replacement, and search for the frequency of the resultant trigram in the online API phrasefinder.io, which uses **Google Books' N-gram corpus**. Then the normalized score of each candidate is calculated using the formula (equation 1). The candidates with the highest score are shown first in the suggestions. For optimization (as getting response from the API takes significant amount of time), we have used multithreading at various stages.

IV.) **Django:** Finally, with each of the 3 subparts, i.e. Spell check, grammar check and the synonym suggester up and running, we have developed a basic web interface using HTML/CSS and JavaScript/ jQuery, with Django for Python based backend, which gives on-the-go suggestions for spelling or grammar errors, and also some contextual synonym suggestions for some words.
On the main page, a textarea is overlayed on a div. The DOM structure and value of the div changes on changing the input in the textarea. For showing suggestions, the content in the div is added an underline, which shows up on the textarea. The div contains various span elements, which each contain a nav node, which is hidden. Now, when there is a suggestion, content is added to the corresponding nav using the CSS ::after selector, which is made visible, and it's z-index is made higher than the textarea so that the nav is clicked.
On encountering any non-alphanumeric character, an AJAX request is sent from the page, which calls a function which returns all the suggestions as a JSON response.

7 Deliverables

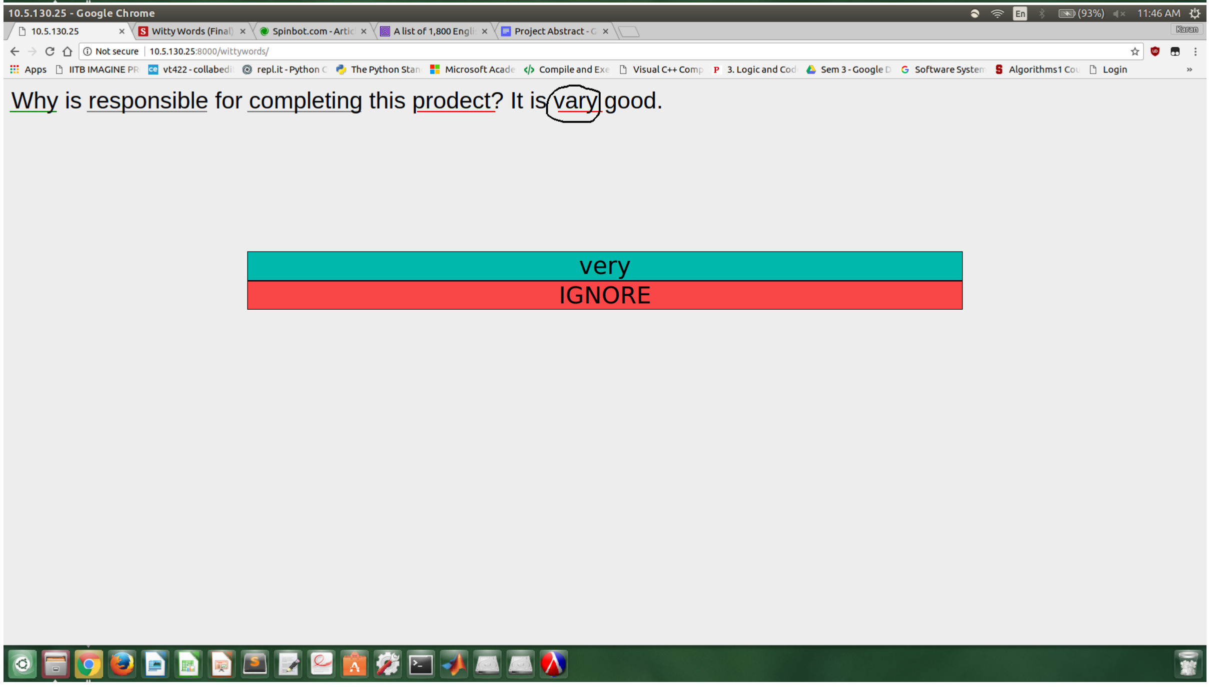
Deliverables promised:

- Context Based Spell Checker


10.5.130.25 - Google Chrome
Not secure | 10.5.130.25:8000/wittywords/

Why is responsible for completing this prodect? It is vary good.

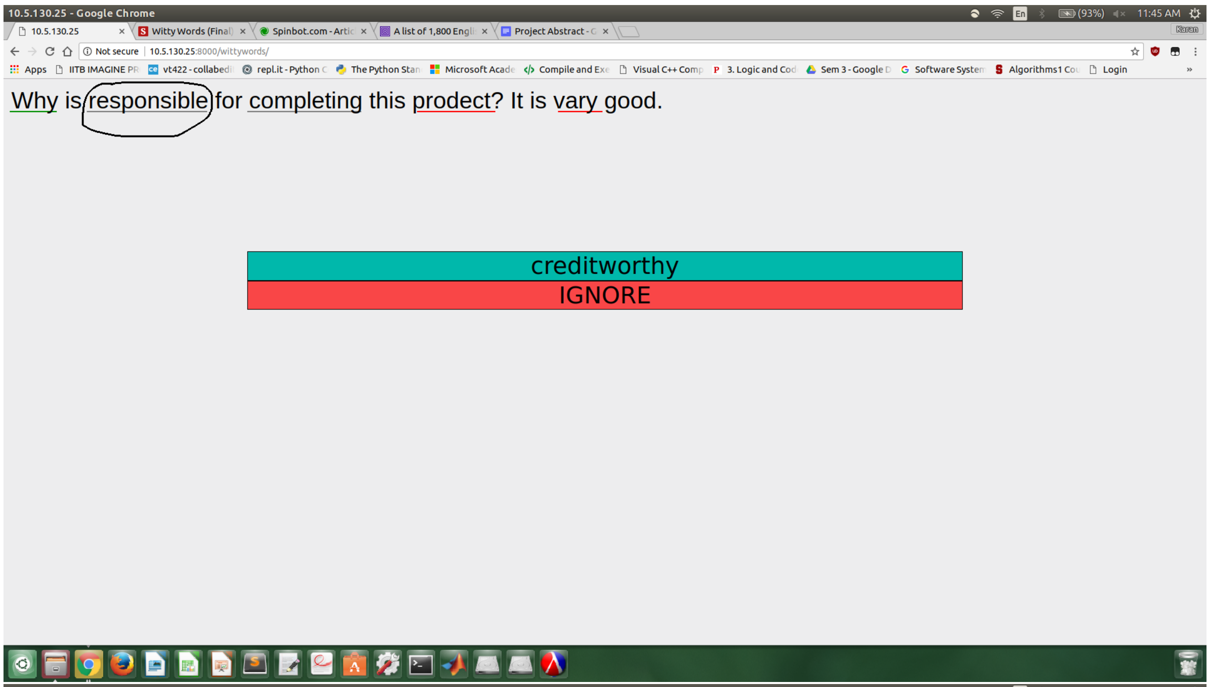
product
project
protect
IGNORE


10.5.130.25 - Google Chrome
Not secure | 10.5.130.25:8000/wittywords/

Why is responsible for completing this prodect? It is vary good.

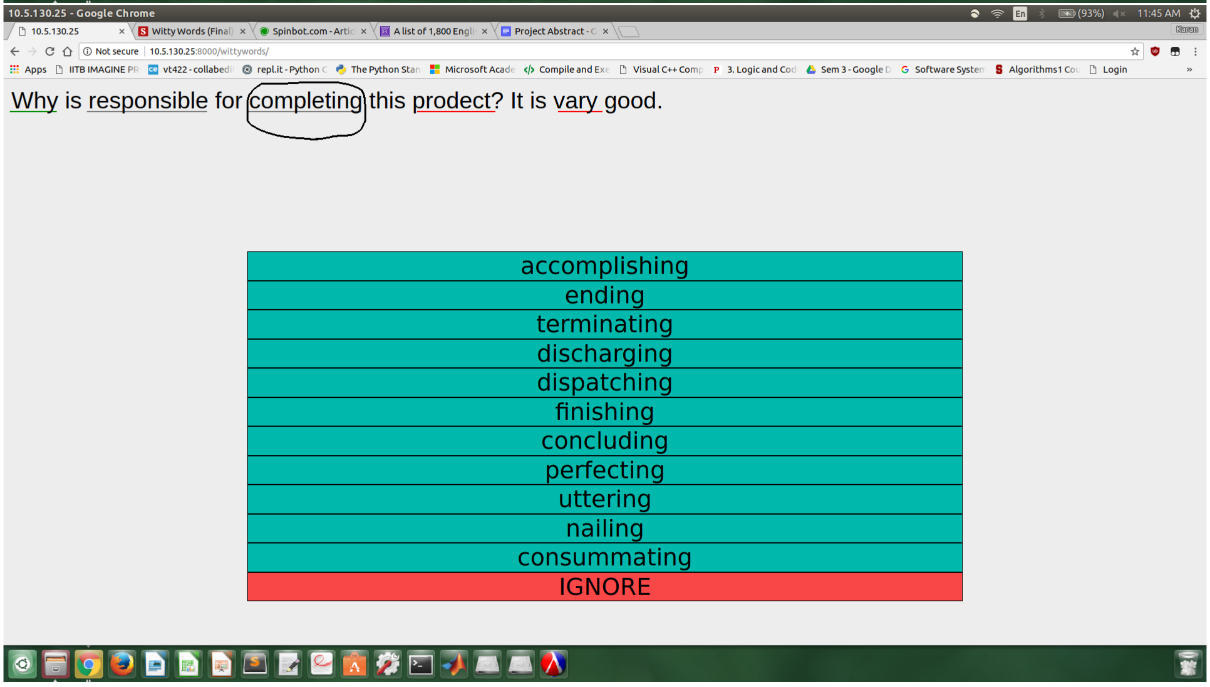
very
IGNORE

- Context Based Vocabulary Enhancer (Synonym Suggester)



Why is responsible for completing this prodect? It is vary good.

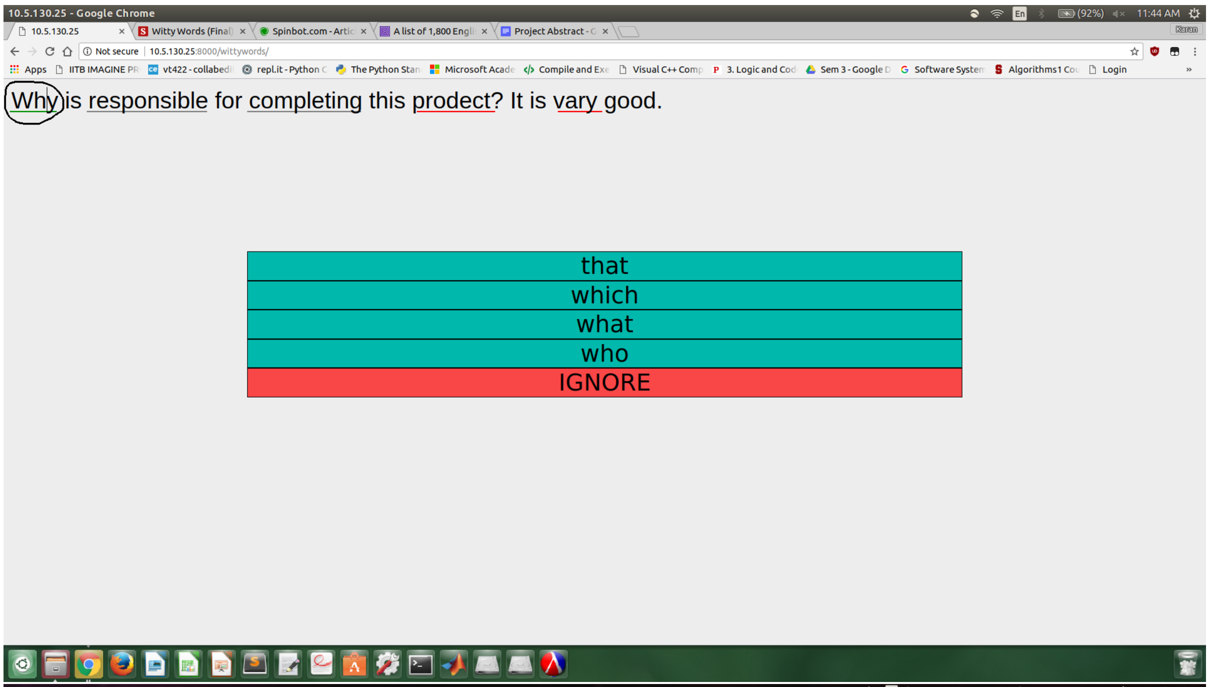
creditworthy
IGNORE



Why is responsible for completing this prodect? It is vary good.

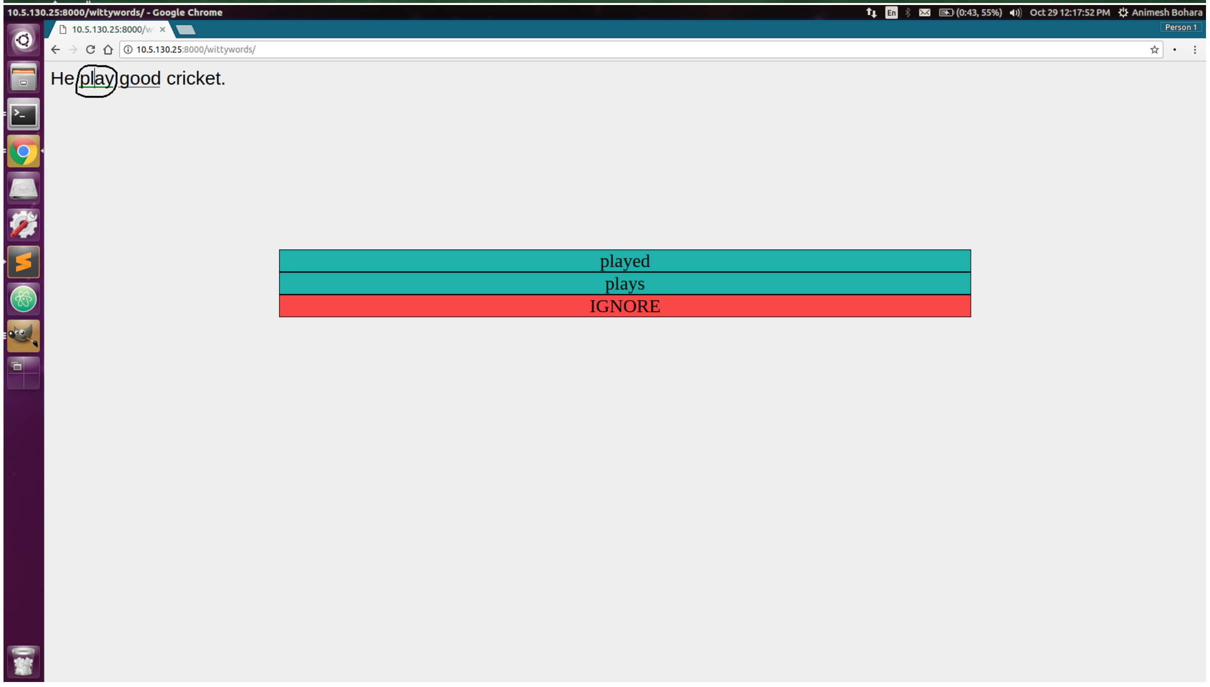
accomplishing
ending
terminating
discharging
dispatching
finishing
concluding
perfecting
uttering
nailing
consummating
IGNORE

- Grammar Checker


10.5.130.25 - Google Chrome
Not secure | 10.5.130.25:8000/wittywords/

Why is responsible for completing this prodect? It is vary good.

that
which
what
who
IGNORE


10.5.130.25:8000/wittywords/ - Google Chrome
10.5.130.25:8000/wittywords/

He play good cricket.

played
plays
IGNORE

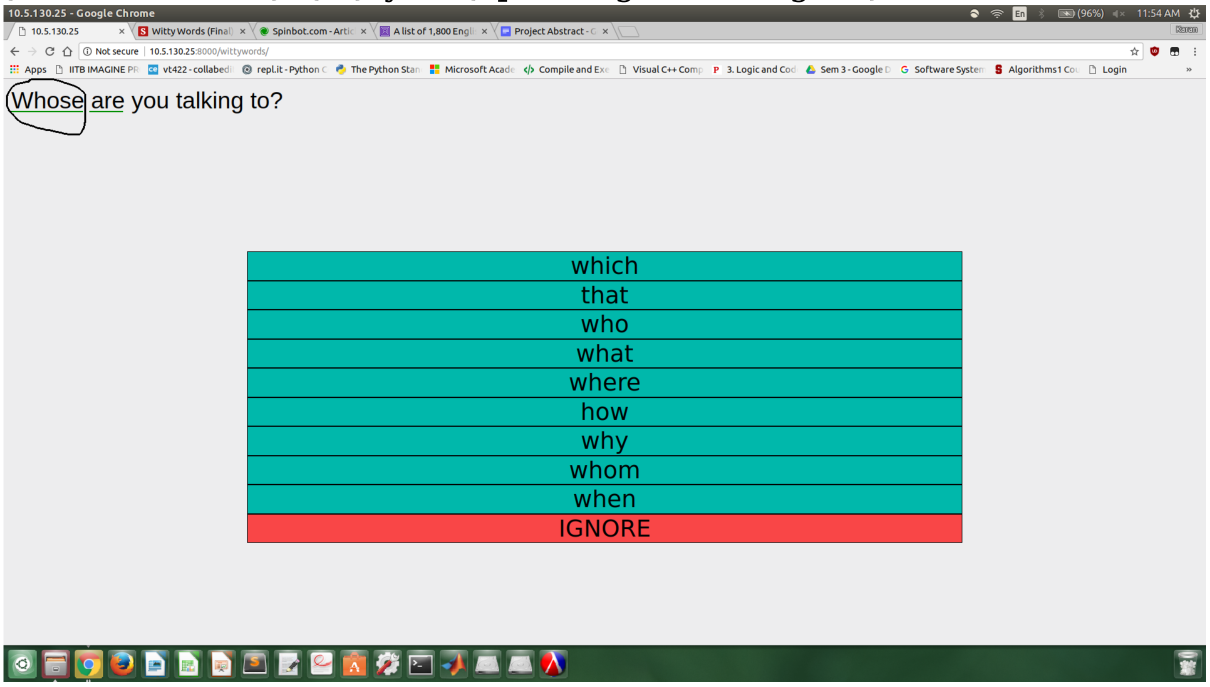
- Web interface for the above

- Dictionary for IITB Lingo Words

All the deliverables have been achieved successfully with certain changes in the implementation that was proposed earlier.

8 Limitations

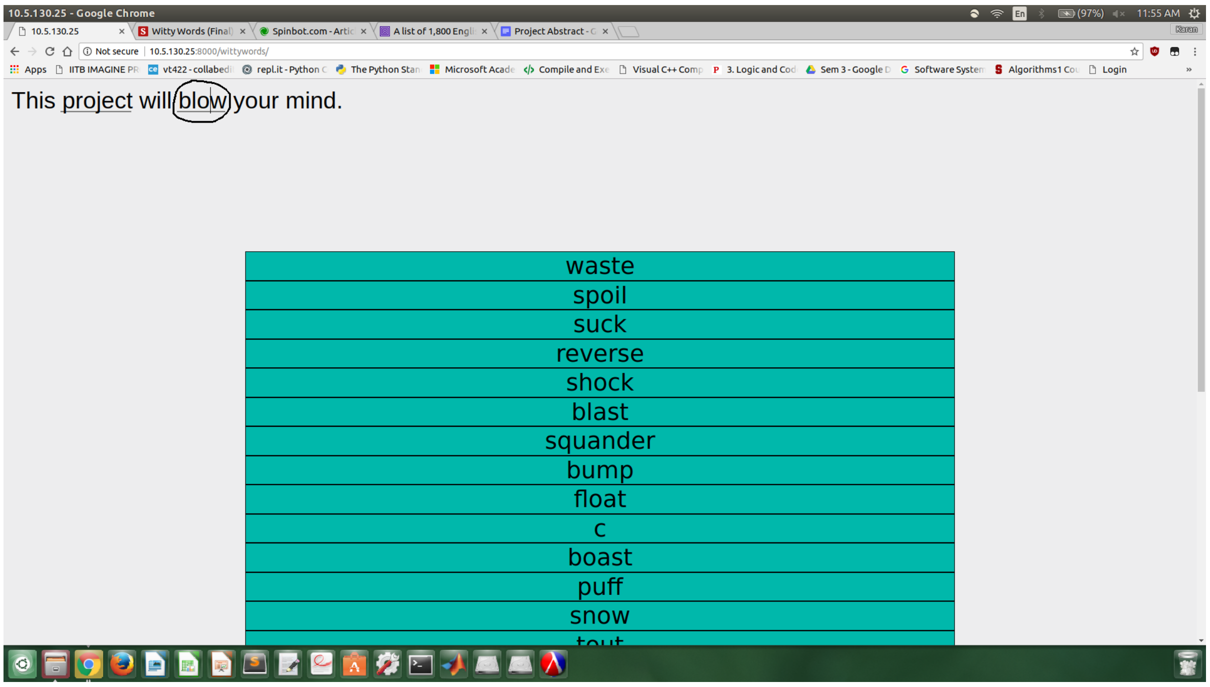
- For the suggestions in spell check and grammar check, the **threshhold** defined is not appropriate enough to give a precise correct output in certain cases. This would require training the data according to the input for those precise outputs.
  e.g. **Input:** Whose are you talking to?
  Desired Output:
  Whose are you talking to? $\Longrightarrow$
  {Whom} are you {speaking, blabbing, ...} to?
  Achieved Output:
  Whose are you talking to? $\Longrightarrow$
  {which, that, ...} {is} you {speaking, blabbing, ...} to?


10.5.130.25 - Google Chrome
Not secure | 10.5.130.25:8000/wittywords/

Whose are you talking to?

which
that
who
what
where
how
why
whom
when
IGNORE

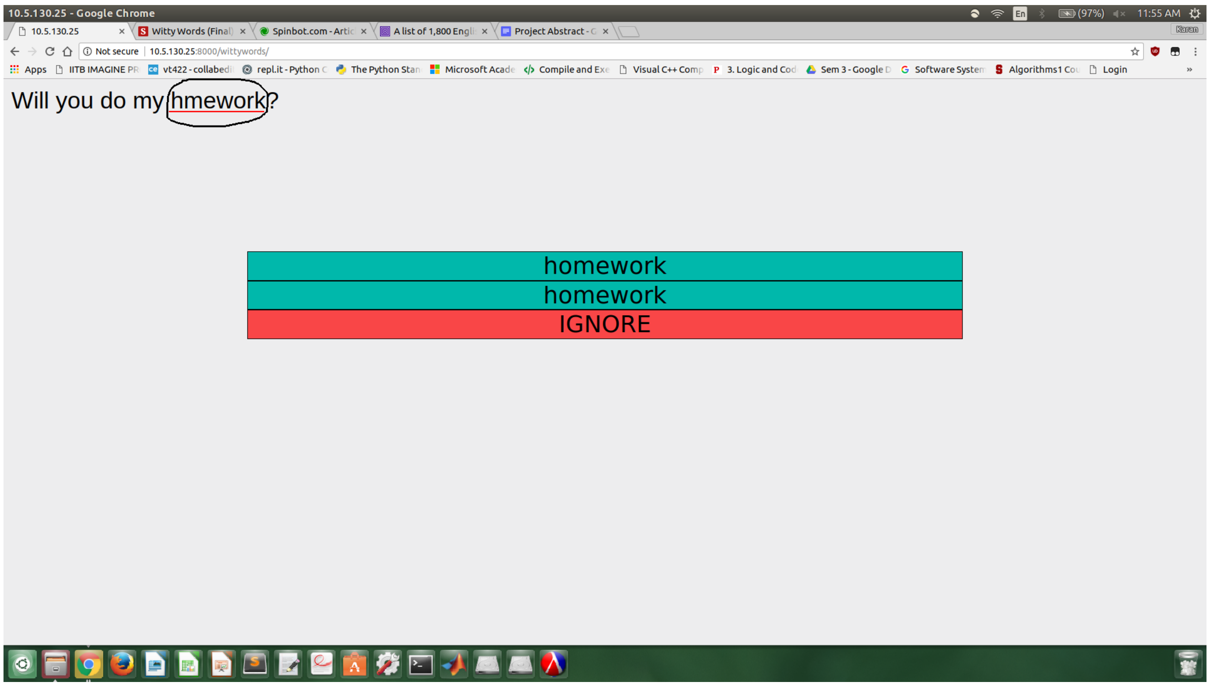
- For phrases, rather than suggesting a complete alternate phrase the synonym suggester recommends alternatives to individual words of the phrase.
  e.g. **Input:** This project will blow your mind.
  Desired Output: This task will knock your socks off.
  Achieved Output:
  This project will blow your mind. $\Longrightarrow$
  This {plan, task, ...} will {waste, spoil, suck, ...} your mind.


This project will blow your mind.

waste
spoil
suck
reverse
shock
blast
squander
bump
float
c
boast
puff
snow

9 Bugs

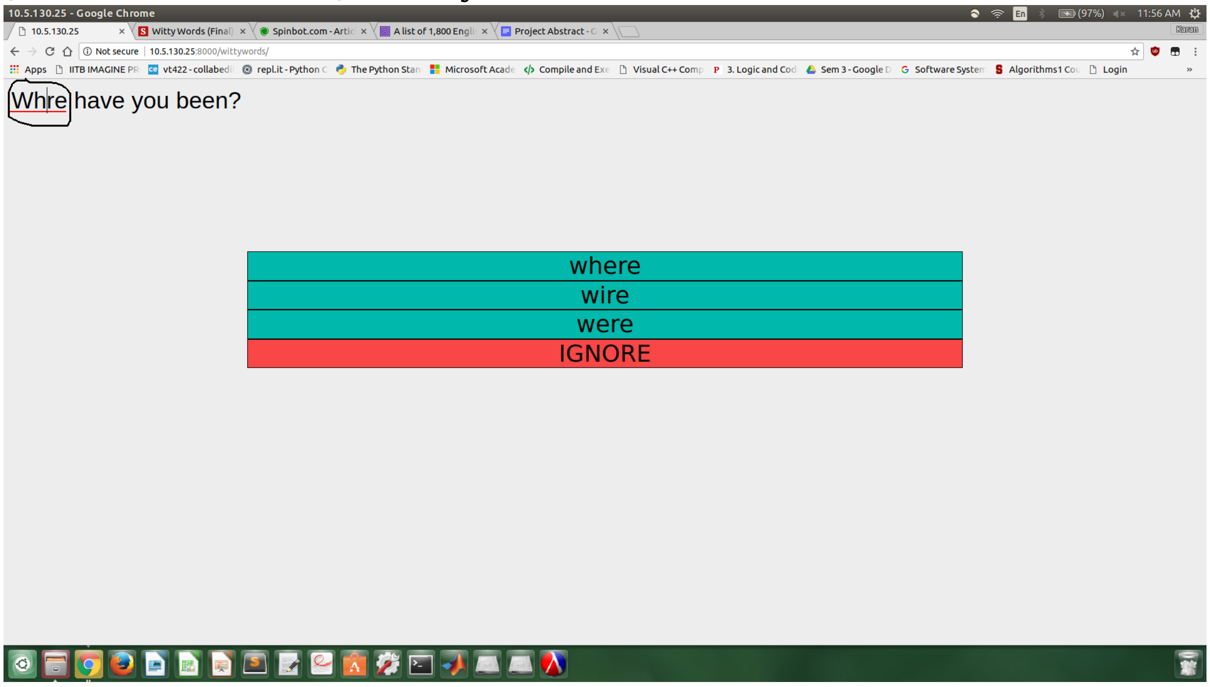
- For the presence of multiple errors, be it spelling or grammar or in case of calling multiple functions of spell check, grammar check and synonym suggester, the colorization for respective functions get randomized.
- For suggestions in specific cases, we get various duplicates of the same recommendation in the textbox.
  e.g. **Input:** Will you do my hmework?
  Desired Output:
  Will you do my <u>hmework</u>? $\Longrightarrow$
  Will you do my {homework}?
  Achieved Output:
  Will you do my <u>hmework</u>? $\Longrightarrow$
  Will you do my {homework, homework}?



Will you do my hmework?

homework
homework
IGNORE

- In case of a correction to the first letter of a line, the suggestion provided doesn't have a capital letter.
  e.g. **Input:** Whre have you been?
  Desired Output:
  <u>Whre</u> have you been? $\Longrightarrow$
  {Where} have you been?
  Achieved Output:
  <u>Whre</u> have you been? $\Longrightarrow$
  {where, wire, were} have you been?



Whre have you been?

where
wire
were
IGNORE

10 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. In grammar check, as we expand the list of rules, the efficiency of the checker increases. In context based synonyms, the larger the data we have, more the probability of finding a match for the context trigrams.

11 Citations

i. https://stackoverflow.com/questions/8951810/how-to-parse-json-data-with-jquery-javascript

ii. https://stackoverflow.com/questions/51520/how-to-get-an-absolute-file-path-in-python

iii. https://stackoverflow.com/questions/6893968/how-to-get-the-return-value-from-a-thread-in-python

iv. https://www.clips.uantwerpen.be/pages/pattern-en

v. https://www.ngrams.info/

vi. http://phrasefinder.io/api

vii. https://stackoverflow.com/questions/16332478/remove-all-classes-from-div

viii. https://developer.mozilla.org/en-US/docs/Web/API/InputEvent

ix. https://stackoverflow.com/questions/4284514/how-to-store-an-array-via-jquery-data-method-and-add-to-that-array-values?rq=1