Grammar Correction of Product Reviews

CS4041 NATURAL LANGUAGE PROCESSING

Mid Report

Chandralekha Yenugunti (B140269CS),
Kandhala Naveena (B140889CS),
Sumedha Birajdar (B140622CS)

February 19, 2018

Abstract

Product reviews enable the seller to know the opinions of the user and improvise the product based on the feedback. These opinions, suggestions, review of supplier etc of a product on e-commerce websites are also useful for the people who want to buy the products. However, most of the reviews may not be useful because of the poor grammar and spelling mistakes. The objective of this project is to make product reviews more clear by correcting the grammar and spellings. This will also be useful in the case when the seller decides the automate the process of review evaluation by machine learning techniques. Buyers as well as sellers will be benefited with this. The report summarizes the literature survey and design for the project.

1 Literature Survey

1.1 Types of errors in language [1]

Grammar errors can be categorized into the following types.

1.1.1 Spelling errors

This is defined as an error which can be found by a common spell checker software. Spell checkers simply compare the words of a text with a large list of known words. If a word is not in the list, it is considered incorrect. Similar words will then be suggested as alternatives.

1.1.2 Grammar errors

An error which causes a sentence not to comply with the English grammar rules. Unlike spell checking, grammar checking needs to make use of context information. Some categories of grammar errors are : Punctuation errors, Constituent agreement mistakes such as, subject verb agreement, gender, number, case, person agreement, noun-adjective agreement, phrase agreement etc.[**2**]

1.1.3 Semantic errors

A sentence which contains incorrect information which is neither a grammar error, nor a spelling error. It needs extensive world-knowledge to recognize these errors. Spelling and Grammar errors will be corrected in this project. Semantic errors are beyond the scope of this project.

1.2 Spelling errors

Spelling errors are generally divided into two types :

1.2.1 Typographic errors

These error occur when the correct spelling of the word is known but the word is mistyped by mistake. These errors are mostly related to the keyboard

and therefore do not follow any linguistic criteria. 80% of the typographic errors fall into one of the following four categories.
1. Single letter insertion; e.g. typing acress for cress.
2. Single letter deletion, e.g. typing acress for actress.
3. Single letter substitution, e.g. typing acress for across.
4. Transposition of two adjacent letters, e.g. typing acress for caress.

1.2.2 Cognitive errors

The errors produced by any one of the above editing operations are also called single-errors. These are errors occurring when the correct spellings of the word are not known. In the case of cognitive errors, the pronunciation of misspelled word is the same or similar to the pronunciation of the intended correct word. e.g. receive -¿ recieve, abyss -¿ abiss.

1.3 Error Detection in spellings

A string of characters separated by space bar or punctuation marks may be called candidate words. A candidate word is a valid if it has a meaning else it is a non word. The error detection process usually consists of checking to see if an input string is a valid index or dictionary word. Efficient techniques have been devised for detecting such types of errors. The two techniques used for error detection is N-gram analysis and dictionary look up. Spellcheckers rely mostly on dictionary look up and text recognition systems rely on n-gram techniques.

1.3.1 Dictionary look up

A dictionary is a list of words that are assumed to be correct. Dictionaries are represented in many ways, each with their own characteristics like speed and storage requirements. Large dictionary is a dictionary with most common words combined with a set of additional dictionaries for specific topics such as computer science or economy. Large dictionary also uses more space and may take longer time to search. The non-word errors can be detected as mentioned above by checking each word against a dictionary. Hash tables are the most common used technique to gain fast access to a dictionary.

1.3.2 N-gram analysis [3]

N-gram analysis is described as a method to find incorrectly spelled words in a mass of text. Instead of comparing each entire word in a text to a dictionary, just n-grams are controlled. A check is done by using an n-dimensional matrix where real n-gram frequencies are stored. If a non-existent or rare n-gram is found the word is flagged as a misspelling, otherwise not. An n-gram is a set of consecutive characters taken from a string with a length of whatever n is set to. If n is set to one then the term used is a uni-gram, if n is two then the term is a Bi-gram, if n is three then the term is tri-gram. The n-gram algorithm was developed as one of the benefits is that it allows strings that have differing prefixes to match and the algorithm is also tolerant of misspellings. Each string that is involved in the comparison process is split up into sets of adjacent n-grams. The n-grams algorithms have the major advantage that they require no knowledge of the language that it is used with and so it is often called language independent or a neutral string matching algorithm.

1.4 Error correction for spelling mistakes

The most studied spelling correction algorithms, they are: edit distance, similarity keys, rule-based techniques, n-gram-based techniques, probabilistic techniques and neural networks. All of these methods can be thought of as calculating a distance between the misspelled word and each word in the dictionary or index. The shorter the distance the higher the dictionary word is ranked.

1.4.1 Edit distance

Edit distance is a simple technique. Simplest method is based on the assumption that the person usually makes few errors if ones, therefore for each dictionary word. The minimal number of the basic editing operations (insertion, deletions, substitutions) necessary to convert a dictionary word in to the non word. The lower the number, the higher the probability that the user has made such errors. Edit distance is useful for correcting errors resulting from keyboard input, since these are often of the same kind as the allowed edit operations.

1.4.2 Similarity keys

A key is assigned to each dictionary word and only dictionary keys are compared with the key computed for the non word. The word for which the keys are most similar are selected as suggestion. Such an approach is speed effective as only the words with similar keys have to be processed with a good transformation algorithm. This method can handle keyboard errors.

1.4.3 Rule-based techniques

Rule-based methods are interesting. They work by having a set of rules that capture common spelling and typographic errors and applying these rules to the misspelled words. Intuitively these rules are the inverses of common errors. Each correct word generated by this process is taken as a correction suggestion. The rules also have probabilities, making it possible to rank the suggestions by accumulating the probabilities for the applied rules. Edit distance can be viewed as a special case of a rule-based method with limitation on the possible rules.

1.4.4 N-gram-Based Techniques

N-grams can be used in two ways, either without a dictionary or together with a dictionary. Used without a dictionary, n-grams are employed to find in which position in the misspelled word the error occurs. If there is a unique way to change the misspelled word so that it contains only valid n-grams, this is taken as the correction. The performance of this method is limited. Its main virtue is that it is simple and does not require any dictionary. Together with a dictionary, n-grams are used to define the distance between words, but the words are always checked against the dictionary. This can be done in several ways, for example check how many n-grams the misspelled word and a dictionary word have in common, weighted by the length of the words.

1.4.5 Probabilistic techniques

They are, simply put, based on some statistical features of the language. Two common methods are transition probabilities and confusion probabilities. Transition probabilities are similar to n-grams. They give us the probability that a given letter or sequence of letters is followed by 16 other given letters. Transition probabilities are not very useful when we have access to a dictionary or index. Given a sentence to be corrected, the system decomposes each string in the sentence into letter n-grams and retrieves word candidates from the lexicon by comparing string n-grams with lexicon-entry n-grams. The retrieved candidates are ranked by the conditional probability of matches with the string, given character confusion probabilities. Finally, a word-bi-gram model and a certain algorithm are used to determine the best scoring word sequence for the sentence. They claim that the system can correct non-word errors as well as real word errors and achieves a 60.2% error reduction rate for real OCR text.

1.4.6 Neural Networks

Neural networks are also an interesting and promising technique, but it seems like it has to mature a bit more before it can be used generally. The current methods are based on back propagation networks, using one output node for each word in the dictionary and an input node for every possible n-gram in every position of the word, where n usually is one or two. [**4**]

1.5 Grammar checking

Grammar checker accepts sentences, paragraphs or documents for the input. These inputs are broken down into individual sentences by identifying punctuation marks. The following pre-processing must be done for checking the validity of the sentence.

1.5.1 Sentence Tokenization

This involves sentence tokenization and word segmentation. The sentence is tokenized into words followed by breaking down words into constituent morphemes and populating lexical information about the word from the lexicon.

1.5.2 Morphological Analysis

Morphological analysis returns word stems and associated affixes.

1.5.3 Part-of-speech Tagging

This is the method of assigning each word a POS tag. The tags are as following, Noun, Verb, Adverb, Adjective, Determiner. A word can have multiple POS tags. The more number of tags, the more difficult it becomes to assign the right POS tag for a given context in sentence by the algorithm. In English, many nouns can also be used as verbs. For example house (a building) and house (to provide shelter). POS tagging is thus a typical disambiguation problem: all possible tags of a word are known and the appropriate one for a given context needs to be chosen. Even by simply selecting the POS tag which occurs most often for a given word without taking context into account one can assign 91% of the words their correct tag. Taggers which are mostly based on statistical analysis of large corpora have an accuracy of 95-97% .

1.5.4 Phrase Chunking

Phrase chunking assigns tags to a sequence of words. Typical word phrases are Noun Phrase (NP), Verb Phrase (VP), Preposition Phrase (PP). Noun phrases typically consist of determiners, adjectives, nouns and pronouns. Verb phrases consist of single verb or auxiliary verbs plus main verbs. Phrase chunking can be done by using hand written rules or by using a probabilistic chunker which is trained. These methods can be combined to the chunking.

1.6 Implement Grammar Checker

There are three ways to implement grammar checking.

1.6.1 Syntax based checking

A text is completely parsed, i.e. the sentences are analyzed and each sentence is assigned a tree structure. The text is considered incorrect if the parsing does not succeed. There are various methods for parsing, such as top down parsing or bottom up parsing.

1.6.2 Statistics based checking

A POS-annotated corpus is used to build a list of POS tag sequences. Some sequences will be very common (for example determiner, adjective, noun as in the old man), others will probably not occur at all (for example determiner, determiner, adjective). Sequences which occur often in the corpus can be considered correct in other texts too, uncommon sequences might be errors.

1.6.3 Rule-based checking

Rule-based checking, as it is used in this project. In this approach, a set of rules is matched against a text which has at least been POS tagged.[**2**]

2 Design

The aim of our project is to improve the experience of the buyers and sellers of e-commerce websites by correcting the grammar mistakes in the product reviews. The approach to this problem is simple and is divided into two parts.

1) Correcting the spellings of wrongly spelled and vague words
2) For the sentences with no spelling mistakes find and correct the grammatical errors.
The semantics ambiguity and semantics errors are beyond the scope of this project. A given product review is first divided into sentences and further every sentence into separate words. Spelling checking for each word is applied, for each wrongly spelled word suggestions are generated. Every suggestion is replaced in the original sentence in place of wrong word to form one or more sentences. Tagged text is generated using the parts of speech knowledge. The tagged text is necessary, as it gives the division of the lexicon for provided sentence. Then the tagged text are matched against the grammar rules to find if the parse is complete or not. If the parse is complete and one or more parse trees are generated then there are no grammatical errors in the sentence. When the sentence is not correctly parsed, grammatical error is detected and suggestions for the correct grammar are generated with the help of grammar rules. We will be using programming language python as tool to work on our project. Various python programming modules such for spelling corrections and language tool module for working with grammar rule to identify the structure of sentence are used.

1) The first hurdle is finding the spelling mistakes for each word in the sentence. As this is not the primary task of the project the python module called pyEnchant is used for this purpose. This module tells whether the given word is correct or not as done in check_spelling function below and if the spelling is wrong the correct suggested words can be obtained using this module.

2) The second major task is finding if the grammar is correct and finding the correct grammar. A rule based approach is used for this purpose. Python module NLTK and Language Tool will be most handy for this purpose. The parse tree is generated for a given sentence. If no parse tree is obtained, error is returned and the position at which this error has occurred. After grammatical error is detected, appropriate suggestions are provided with the help of grammar rules and position at which error is detected.

2.1 Algorithm

function name : correct
Input : A raw sentence with spelling and grammatical errors.
Output : The list of suggested sentences after removing all possible spelling mistakes and grammatical errors.

```
procedure CORRECT
    $words \leftarrow$ list of word in sentence
    $final \leftarrow$ empty list
    $suggestions \leftarrow$ empty list
    $corrected\_spell \leftarrow$ empty list
    for each $word \in words$ do
        if $check\_spelling(word) = false$ then
            $suggestions = suggest(word)$
            for each $suggestion \in suggestions$ do
                $corrected\_spell.add(sentence.replace(suggestion))$

    for each $sentence \in corrected\_spell$ do
        $final.add(check\_grammar(sentence))$

    $return\ final$
```

function name : check_grammar
Input : A sentence with only grammatical errors (sentence).
Output : The list of suggested sentences after removing all possible grammatical errors.

```
procedure CHECK_GRAMMAR
    if $parse\_tree(sentence) = Error$ then
        $return\ suggest(sentence)$

    $return\ sentence$
```

function name : check_spelling
Input : A word.
Output : Boolean - true if word is correctly spelled and false if word is wrongly spelled

```
procedure CHECK_SPELLING
  if $dictionary.contain(word) = true$ then
    $return\ true$
  $return\ false$
```

2.2 Flow diagram

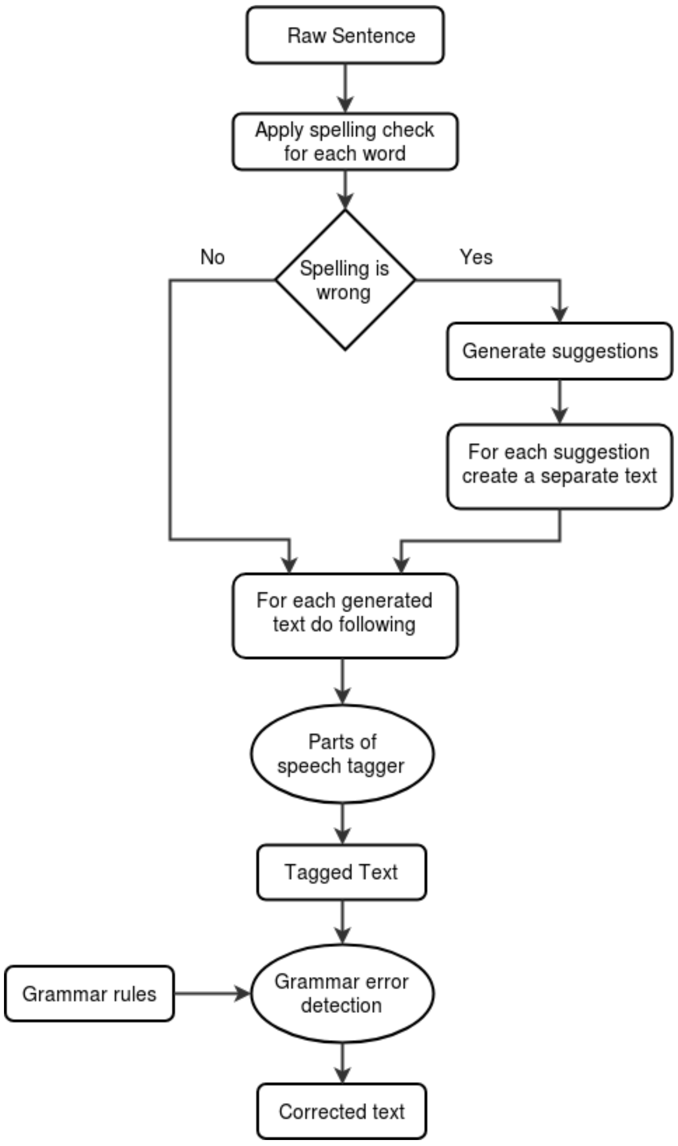

Figure 1: Flow diagram describing algorithm

References

[1] Naber, Daniel. *A rule-based style and grammar checker*, Bielefeld University Bielefeld, Germany, 2003.

[2] Bhirud, Nivedita S and Bhavsar, RP and Pawar, BV. *GRAMMAR CHECKERS FOR NATURAL LANGUAGES: AReview.*

[3] Jurafsky, Dan and Martin, James H. *Speech and language processing*, Pearson London:, 2014

[4] Gupta, Neha and Mathur, Pratistha. *Spell checking techniques in NLP: a survey*, International Journal of Advanced Research in Computer Science and Software Engineering, 2012