DWIT COLLEGE

DEERWALK INSTITUTE OF TECHNOLOGY

CONTEXT BASED SPELLING CORRECTION

A Mini PROJECT REPORT

Submitted to

Department of Computer Science and Information Technology

DWIT College

Submitted By

Suraj Prasai

Class of 2019

November 19, 2018

CERTIFICATION

This is to certify that this project prepared by SURAJ PRASAI entitled “Context Based Spelling Checking Using Bigram” in partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Information Technology has been well studied. In our opinion, it is satisfactory in the scope and quality as a project for the required degree.

.....

Mr. Ritu Raj Lamsal

Deerwalk Institute of Technology

ACKNOWLEDGEMENT

I would like to express my thanks to DWIT college for their continued support and encouragement for the completion of this project. I would like to thank my supervisor Mr. Ritu Raj Lamsal for his continuous support, expert advice throughout this project.

I would also like to thank COCA for providing the corpus of Bigram used in partial completion of this project.

Regards,

Suraj Prasai

ABSTRACT

Context Based Spelling Correction is a web application that corrects the spelling errors in sentences. The current version of the system does not have grammar checking function, and the initial goal of this project was to develop a standalone spell checker. But there are other types of errors that spell checkers and grammar checkers cannot detect, such as “I love eating desert”, where “dessert” is the correct spelling for “sweet snacks” but “desert” was typed. This kind of error is called a real-word error. The main focus of this project is to looks at how a context-sensitive spell checker can be built to detect real word errors by using the information probability approach. The basic framework and background necessary to develop a context-sensitive spell checker will be covered. Different approaches to context-sensitive spell checking have been investigated and the complete design and implementation of each approach will be presented.

Keyword: Spell Checker, Edit Distance,Natural Language Processing, Bigram, Levenshtein distance, Bigram, Natural Language Processing, N-gram, Corpus

OUTLINE OF DOCUMENT

The report is organized as follows:

Preliminary Section: This section consists of the title page, abstract, table of contents and list of figures and tables.

Introduction Section: In this section, the background of the project, problem statement, its objectives, scope and limitation are discussed.

Requirement and Feasibility Analysis Section: Literature review, Requirement analysis and Feasibility analysis make the bulk of this section.

System Design Section: This section consists of the methodology that were implemented in the project and the system design as well.

Implementation and Testing Section: In this section, the implementation, description of major methods and testing of the system are discussed.

Conclusion and Recommendation: This section consists of the final findings, and the recommendations that can be worked on in order to improve the project in the future.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

API:	Application Programming Interface
COCA:	Corpus of Contemporary American English
NLP:	Natural Language Processing

CHAPTER 1: INTRODUCTION

Problem Statement

Software tools are being used extensively in modern societies, including for simple tasks like preparing a word document. One of the most essential tools will be the one able to assist the user with text input. People do make mistakes when typing and there is a need to detect such errors. These errors may tend to appear more frequently for words that are not included in the dictionary of the word processor. The building of writing tools to reduce typing errors will be addressed here.

Spell Checker

The most common and simplest tool that can assist the user with writing is a spell checker. It can detect words that are not present in a dictionary. There are two types of spelling programs, spell checker and spell correctors but people often refer to a spell corrector as a spell checker. The function of a spell checker is simple, input text and incorrect words will be identified. A spell corrector, on the other hand, detects both misspelled words and finds the closest word that it deems to be spelt correctly. A spell checker may be a standalone function that corrects a block of text or part of a large application, such as a word processor. The spell checker is helpful for correcting typing errors, but it may not be helpful for logical or phonetic errors. In recent years, the functions of spell checkers have become increasingly sophisticated, and some are now capable of recognising simple grammatical errors. However, even at their best, they are rarely able to correct all the errors in a text such as homophonic errors (two or more words that are pronounced the same but differ in meaning, e.g. meat and meet) and foreign words as

misspellings. Research has been focused on developing technologies that uses the surrounding context of a word to determine the correctness of the word, even if the word itself is in the dictionary.

Objectives

The main objective of this project is to develop a spell checking mechanism using Bigram model. The use of such model adds context to our spelling corrector based on the edit distance. The goal of this project is to build a set of tools that includes the following functionalities:

- Spell checking
- Context-sensitive spell checking.

The main focus in this project is to find out whether context-sensitive spell checking based on the information probability approach is a valid solution to the problem of real-word errors. There are many different approaches in information probability, the one I will be investigating in this project is: Context-sensitive spell checking using Bigrams.

Scope

The project showcases the use of Bigram to detect spelling errors in sentences. The implementation can be used in editors to correct the typos, grammar and spelling mistakes that writers do very frequently while writing. Furthermore it can be used as a base for a recommender system that can recommend the use of certain word instead of another one in a sentence.

Limitation

1. Cannot recommend words that are more suitable in given context.
2. This project implementation gives high accuracy if the word is frequently used in corpus.
3. Word not used in corpus or words with wrong spelling in corpus is detected as correct by the application.

CHAPTER 2: Literature Review and Requirement Analysis

This chapter will focus on presenting the background used to develop a context-sensitive spell checker. This chapter will cover the fundamentals of the research paper in which the context-sensitive spell checking approach is discussed. The basic approaches for spell checker is covered here.

2.1 Background

2.1.1 N- Grams

An n-gram is a contiguous sequence of n items from a given sequence of text. The item can be letters or words depending on the application. N-grams are generally collected from a text corpus. An n-gram of size one is referred as a “unigram”; size two is a “bigram”; size three is a “trigram”. Larger sizes n-grams are often referred to by the value of n, e.g. “fourgram”, “five-gram”, and so on. [1] The N-grams used in this project are word trigrams and word bigrams.

2.1.2 Confusion Set

A confusion set { } means that each word is commonly confounded with any other word from the set, e.g. {“desert”, “dessert”}.

2.2 A Context Sensitive Spell Checker

There are three methodologies that can be used to work on context-sensitive spelling detection and they include: method based on semantic information, method based on machine learning and method based on probability information. The method used in this project was probability information.

2.2.1 Methods Based on Semantic Information:

The semantic information approach was first proposed by Hirst and St-Onge [2] in 1998 and later developed by Hirst and Budanitsky [3] in 2005. This approach was based on the observation that the words that a writer intends to use are semantically related to their surrounding words whereas some types of real-word errors are not, such as the example given in Hirst and Budanitsky's paper [3], "It is my sincere hole (hope) that you will recover swiftly." Such errors will result in a perturbation of the cohesion and coherence of the text. Hirst and Budanitsky use semantic distance measures in WordNet (Miller et al., 1993 [4]) to detect words that are potentially anomalous in context – that is, semantically distant from nearby words; if a variation in spelling results in a word that is semantically closer to the context, it is hypothesized that the original word is an error and the closer word is its correction. [5]

2.2.2 Methods Based on Machine Learning:

The machine learning method is regarded as a lexical disambiguation task and confusion set are used to model the ambiguity between words. The machine learning and statistical approaches are often based on pre-defined confusion sets which are sets of commonly confounded words, such as {their, there} and {principle, principal}. These methods learn the characteristics of a typical context for each member of the set and detect situations in which one member occurs in context that is more typical of another. Such methods are limited to a set of common and predefined

errors, but such errors can include both content and function words. Given an occurrence of one of its confusion set members, the spellchecker's job is to predict which member of that confusion set is the most appropriate in the context. [5]

2.2.3 Methods Based on Probability Information:

Mays et al. (1991) [6] proposed a statistical method using trigram (a sequence of three words) probabilities for detecting and correcting real-word errors without the need of requiring predefined confusion sets. In this method, if the trigram deduced that the probability of an observed sentence is lower than the sentence obtained by replacing one of the words with a spelling variation, then the original is a real-word error and the variation is what the user intended to use. [5]

2.2.4 Proposed Method:

The probability information approach has been chosen for this project because it is not limited by any pre-defined confusion set of words and research has shown that the accuracy of this method approach is relatively high [5]. As mentioned in the previous chapter, this project will focus on probability information approaches of Context-sensitive spell checking using bigrams.

2.2 Requirement Analysis

2.2.1 Functional Requirements

The functional requirements of this project are:

Context Based spell Checker shall be implemented to detect various errors in a sentence such as typos, spellings.

a. A text box area shall be provided for users to input incorrect text.
b. The users is provided immediately with the correct sentence predicted by the model.

2.2.1 Non-Functional Requirements

The non-functional requirements of this project is that the application must have user-friendly interface.

2.3 Feasibility Analysis

After gathering of the required resource, whether the completion of the project with the gathered resource is feasible is checked using the following feasibility analysis.

2.3.1 Technical Feasibility

The project can be implemented using any high-level programming language along with an in-memory database. In this project, Flask framework for Python programming language is used for the web implementation of the project. The frontend module shall be implemented using HTML, CSS, and JavaScript.

2.3.2 Economic Feasibility

All the tools and technologies used for the project shall be open-source and freely available. The project required knowledge about the domain of N-gram and Levenshtein distance. With this knowledge, the project can be easily completed with a dedicated time allocation of two hours per day for one week.

2.2.1 Operational Feasibility

The project meets all the requirements identified in the requirement analysis phase. The project can be implemented in any application, editors that needs the use of spelling corrector.

CHAPTER 3: SYSTEM DESIGN

3.1 Development

Methodology implemented while carrying out the project are discussed below:

3.1.1 Development Methodology

The development process used during this project is ‘agile’. This is an iterative and incremental approach. The agile method breaks tasks into small iterations that typically last from one to four weeks. Each iteration involves working through a full software development cycle that includes planning, requirements analysis, design, coding and testing etc. This minimises the overall risk and allows the project to adapt to changes more easily. This is essential in a research project because it is likely that the code will have to be modified to fulfil the various requirements. [9]

3.1.2 Programming Language

The programming language used in this project is Python. Python is used because of the programmer’s familiarity with the language.

3.2 Building a Context Sensitive Spell Checker

The design of the context-sensitive spell checking with the algorithm used will be discussed in this section.

3.2.1 Context Sensitive Spell Checking Algorithm

As mentioned in the previous chapter, Bigram approach is followed to implement the context-sensitive spell checker. This algorithm can be broken down into three main steps. In this section, only the general concept of the algorithm is presented, while the detailed implementation of each algorithm is presented in the implementation chapter. First, the text to be spell checked is split up into n-grams. Every word is the start of a new ngram, resulting in a number of n-grams equal to the number of words in the text minus (n-1). For example, the five word sentence 'Please fill in the form' is split up into four bigrams: 'please fill', 'fill in', 'in the' and 'the form'.

Second, a check for each n-gram is made to verify whether all three words are real words. In this case, the lexicon spell checker will perform non-word error detection and correction before the context-sensitive spell checking takes place.

Third, every n-gram is looked up in a precompiled database containing a list of n-grams and their number of occurrences in the corpus used for compiling the database. If the n-gram is inside the database, the n-gram is regarded as correct and will not be considered further. If the n-gram is not in the database, it is considered to be unlikely and is hence identified as an erroneous n-gram containing a real-word error.

3.2.2 Building a N-gram Database

To build an n-gram database that contains a large amount of English n-grams, a corpus of written text is needed. As the lexicon spell checker uses a lexicon of English, the corpus for extracting n-grams will also be in English. For the dictionary lookup spell checker the corpus “Big.txt” provided by Peter Norvig is used. The Bigram dataset is taken from Corpus of Contemporary American English which is freely available on the web.

In the algorithm for extracting n-grams, a word is defined as a character sequence bordered by white spaces. The algorithm removes all punctuation and capitals, so the bigrams ‘Is it?’ and ‘is it’ are the same. As the algorithm removes punctuation, sentence borders are removed too. Moreover, text borders are also removed in order to facilitate the extracting process. As a result, some bigram may consist of one or two words from one text and one or two words from another text, but this is a relatively small number of bigrams. The detailed implementation of this algorithm will be presented in the Implementation chapter

3.3 System Architecture

The system can be split into two major parts: the lexicon spell checker, and the context sensitive spell checker.

The lexicon spell checker checks for the possible errors in every wrongly written words in the the input box. It provides the candidates that could possibly replace the wrong word to make it right. This corrector checks for the word with nearest edit distance and checks for its frequency in the corpus. It then replaces the word by the word with highest frequency in the corpus.

The context sensitive spell checker is provided with the candidates from the lexicon spell checker. It checks for the pair and its frequency in bigram corpus. It provides us with the

probability of a word given another word. The bigrams with highest frequency will be most likely the correct word.

3.3.1 Model View Separation

The system is developed under the "Model-View Separation"[7]. The Model-View Separation enables a clear separation between the business logic layer (Model) and the presentation layer (View). The lexicon spell checker and context-sensitive spell checker can each be thought of as a separate "Model", while the user interface can be viewed as the "View". Keeping the Model completely separate from the rest of the system enables it to be reused in other applications and it will not be affected by changes in the presentation layer. The context-sensitive spell checker is developed using the Model-ViewController [8] design pattern and more of it will be discussed in the Implementation chapter.

The system is divided into three packages: Spell Checker based on Levenshtein Distance, Context Sensitive and GUI. The package diagram of the system is presented in Figure 1.

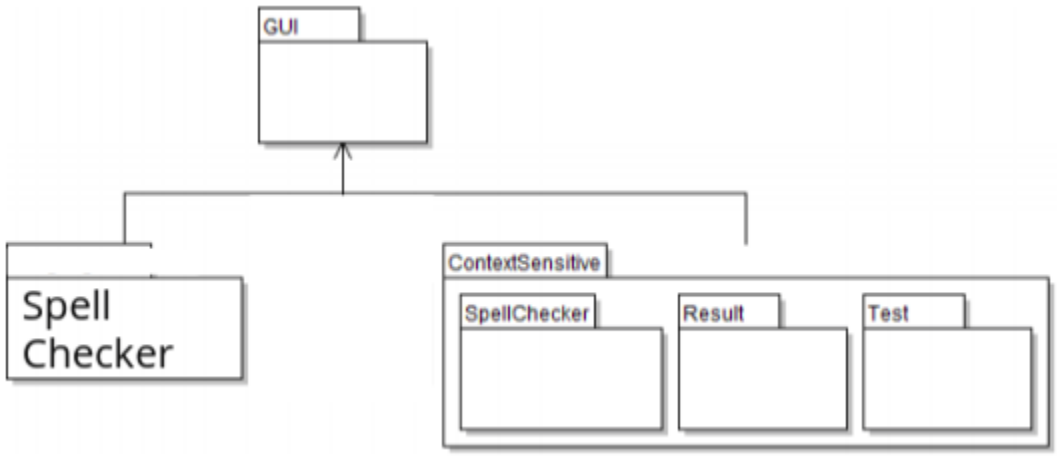

Fig 1: Package Diagram

3.5.4 High Level Design

The core of the spell checker is implemented in the SpellChecker package. It implements all the necessary algorithms required to perform context-sensitive spell checking. The SpellChecker package contains the bigram database and bigram context checking approach. With regards to the discussion from the previous chapters, the main functionalities of the context sensitive spell checker that have to be implemented are as follows: Produce the Bigram Database from training data Read and write the Bigram Database from the disk Look up a Bigram in the database. Detect any misspelling in the text input. The functionalities mentioned above are presented in the initial architecture diagram in Figure 2.

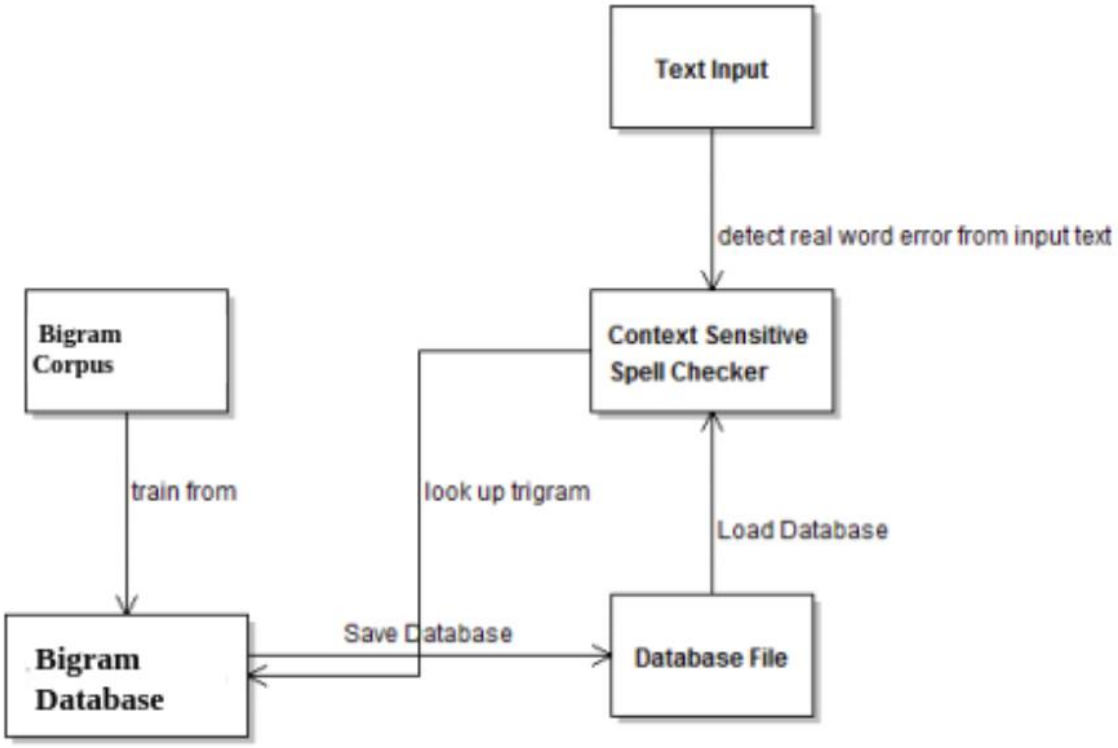

Fig 2: The initial architecture of context based spell checker

The conceptual model of a bigram spell checker is very simple. The spell checker will contain a bigram database which contains a large amount of bigrams. When the context checker performs

the spell check on the input text, it will need to look up the necessary bigram in the bigram database stored in memory to determine its correctness. The high-level design of the bigram context sensitive spell checker is presented in Figure 3.

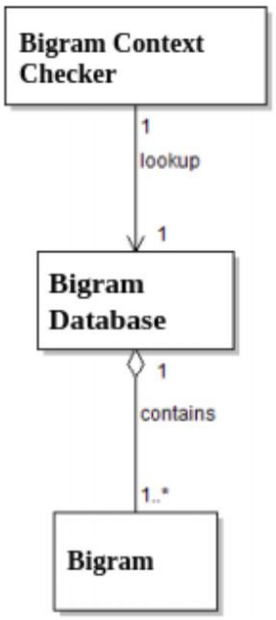

Fig 3: A high-level design model of the bigram context-sensitive spell checker

A high-level design model is only a conceptual model rather than a software class. It does not have any function associated with it or any interaction with the software classes. The software class diagram for the context-sensitive spell checker engine will be presented in the Implementation chapter. The software class diagram can only show the functionality of each class and how they are connected, but it does not show the interaction between classes in run time. A generalized version of context sensitive spell checking in run time is presented in Figure 4.

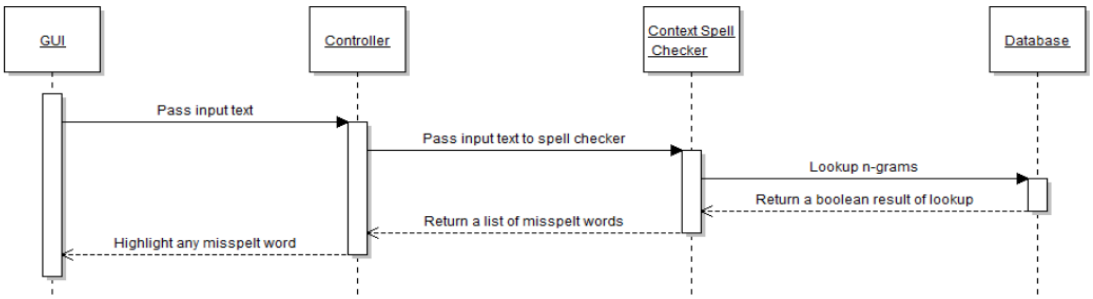

Fig 4: Sequence diagram for context sensitive spell checking process.

CHAPTER 4: IMPLEMENTATION AND TESTING

4.1 Implementation

The Context Based Spell Checker is implemented as a web application and can be accessed from the web browser. The application is developed in Python using Flask Framework.

4.1.2 Database Structure

The bigram database is the cores of the program. It stores all of the information needed for context-sensitive spell checking. During the context sensitive spell checking process, the spell checker is required to perform a look up operation in the database stored in memory. The look up process is the key process for the performance of context sensitive spell checking, and therefore a fast look up mechanism is needed. The hash data structure with a look up mechanism operates in time is one of the most efficient data structure for doing dictionary style lookups. It is a dictionary-type data structure, with a key that uniquely identifies a single element.

4.1.3 Saving and Loading the Database

An application of this kind needs to have a database containing millions or even billions of bigrams. So the application needs to decide what it needs dynamically rather than loading all the database into memory.

4.1.4 BigramContextChecker Class

The BigramContextChecker Class has the following functionalities:

contextCheck() - return an ArrayList of misspelled words for text input. calculatingFalsePositive() - return the false positive rate for text input

calculatingRecall() - return recall for the text input

calcualtePrecision() - return precision for text input

The spell checking algorithm can be broken down in two steps. The first step is to generate a list of bigrams using the generateBigram() function. The generateBigram() function generates a list of trigrams from the text input. The second step is to loop though the list of bigrams and every word is checked using the two bigram. A word is classified as correct only if all two bigrams containing the word that exists in the database, otherwise it is a misspelt word and will be added to the misspelt list.

4.2 Testing

The correctness of the algorithm under investigation is essential because it may have a significant effect on the research outcome. So the testing of the context sensitive spell checker and the database is performed. Some general methods for evaluation are:

4.2.1 Confusion Matrix

In order to measure the result of the application, we need to have a systematic way to measure the output of the program. In the case of the context-sensitive spell checker, the program will have four outputs (a confusion matrix presentation of the four outcomes is presented in Figure 16): True Positive – misspelt word detected by the spell checker False Positive – correctly spelled word detected by the spell checker False Negative - misspelt word not detected by the spell checker True Negative - correctly spelled word not detected by the spell checker

		Misspelt	
		Yes	No
Detected	Yes	True Positive	False Positive
	No	False Negative	True Negative

Fig 5: Confusion Matrix

We use precision and recall to measure accuracy of our application.

$$Precision\ , P = \frac{TP}{TP+FP} \quad Recall, R = \frac{TP}{TP+FN}$$

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. Recall is the ratio of correctly predicted positive observations to the all observations in actual class.

CHAPTER 5: CONCLUSION AND RECOMMENDATION

5.1 Conclusion

Context Based Spelling Checking is a fast and space-efficient model for checking the spelling errors in a sentence using lookup method.

However, due to lack of training data, pruned corpus the model could not be tested to its full capability to provide better accuracy. Hence, more rigorous tests must be done in order to use this system for real-world applications.

5.2 Recommendation

Training with a larger corpus can improve the accuracy of the spell checker. In the current version of the context-sensitive spell checker, the n-grams database is only trained from a small subset of Bigrams produced by the corpus COCA. It is not sufficient to provide a true accuracy of real-word error detection. For this reason, a bigger corpus is needed for training in order to provide sufficient n-grams for context-sensitive spell checking.

Context-sensitive corrector can help the user to correct real-word errors. In the current version of the context-sensitive spell checker, it does not provide any correction for misspelt words. For this reason, a spell corrector is needed to provide correction for real-word errors.

REFERENCES

[1] N-grams. http://en.wikipedia.org/wiki/N-gram, Accessed on 21/8/2011

[2] St-Onge. D. Master thesis: Detecting and correcting malapropisms with lexical chains. University of Toronto, 1995

[3] Hirst, G. J, Correcting real-word spelling errors by restoring lexical cohesion, University of Toronto, 2001

[4]. George A. Miller and Katherine Miller. Introduction to wordnet: An on-line lexical database. 1993.

[5] Aminul Islam, Diana Inkpen, Real-Word Spelling Correction using Google Web 1T 3- grams, Conference on Empirical Methods in Natural Language processing, pages 1241-1249, 2009

[6] James L. Peterson. Computer Programs for Detecting and Correcting Spelling Errors. The University of Texas at Austin. 1980

[7] Martin Fowler. Patterns of Enterprise Application Architecture. Addison-Wesley, 2002, page 330.

[8]. Eleni Katsoulotou. Master thesis: Semi-Automatic Marking of Diagrams. The University of Manchester. 2003..

[9] Martin Fowler, GUI Architectures. http://www.martinfowler.com/eaaDev/uiArchs.html#ModelViewController. Accessed on 4/4/2012.

[10] Kukich K. Techniques for automatically correcting words in text. Computing Surveys, Vol. 24, No. 4, pp. 377-439, (1992).