NATURAL LANGUAGE PROCESSING

INTEGRATED TEXT EDITOR

|  |  |  |  |
| --- | --- | --- | --- |
| NAME | USN | PHONE | EMAIL |
| Tushar K Naik | 1RN10CS111 | +91-9449716665 | tushar.knaik@gmail.com |
| Suhas V | 1RN10CS102 | +91-9620070682 | suhas.f18@gmail.com |
| Vamanan T S | 1RN10CS114 | +91-9886512745 | vamanan.1rn10cs114@gmail.com |

Objective:

To develop an interactive editor, that implements a live spell checker (spelling error detection and error correction) using n-grams and the noisy channel model for contextual spelling errors. The editor will also include a smart version of Auto-complete, a dynamic Find option, and a theme recognizer. The GUI of the text-editor may be implemented in Java or Python. Our main objective is to concentrate on exciting base algorithms, and not on fancy implementable add-ons.

Introduction:

NLP: Natural Language Processing is a field of computer science, artificial intelligence, and linguistics concerned with the interactions between computers and human (natural) languages. As such, NLP is related to the area of human–computer interaction.  
A spell checking mechanism is a prime application of NLP. This involves scanning the document for misspelt words and words out of context typed by mistake.  
Auto-complete is a word recognition and prediction probability notion.

Problem definition:

To implement: A live text editor that has the following:

1. Spell Checker: A live automated spell checker that detects and corrects misspelt errors and also, corrects contextual errors.
2. Auto-complete: An option that continuously scans for long words and completes them automatically whenever applicable.
3. Dynamic find: That provides an option to the user to search for regular expressions
4. Context Recognition: To provide search links to the user to arrange for the privilege of easily searching for additional information on text being typed.

Design and Implementation methodologies:

1. Spell Checker: Spell checking involves 3 methods:
2. Non-word error detection: Misspelt errors, like girafe.   
   *Implementation:* The noisy channel model is used for the purpose. This can be done using a standard corpus as a reference. Words may be searched for within the corpus, if the word isn’t in the corpus, spelling error exists. An **efficient search mechanism** has to be implemented for the purpose.
3. Isolated word correction: This involves correcting misspelt words to its nearest probable correct spelling. E.g. girafe to giraffe.  
   *Implementation:* This is done using probability theory. Appropriate correction is made by choosing candidates after application of **Levenshtein edit distance algorithm**.  
   Edit distance is the total number of changes required to obtain the correct word from the incorrect word. Changes include: Addition, Deletion, Substitution, and Transposition of characters.   
   Research shows that 76% of spelling errors are because of errors that are 1 edit distance away. And 98.6% of them are within 2 edit distances from their correct spellings. After discovering appropriate candidates that are at most 2 edit distances away, the appropriate word may be predicted.
4. Word Segmentation: If the user misses out on spaces between words, the words have to be recognized and split appropriately. E.g. if the user types “ihaveahat”, he naturally means to type “I have a hat”  
   *Implementation:* Involves the usage of **dynamic programming** whereby, using recursive algorithms and a comprehensive corpus, the large word may be split into its components.
5. Contextual word correction: Involves using the context to help detect and correct spelling errors even if they accidentally result in an actual REALWORD word of English (real-word errors). This can happen from typographical errors (insertion, deletion, transposition) which accidentally produce a real word (e.g., *there* for *three*). These may be detected contextually.  
   *Implementation:* Contextual errors are detected and corrected **using n-grams and the noisy channel model**.
6. Auto-complete: This is a prediction mechanism that tries to predict long words that are being typed repeatedly by the user, in order to save time.  
   *Implementation:* This may be achieved using a **hash map** to save long words that have been typed, and using a search mechanism to search for words that may be complete-able at every instance.
7. Dynamic Find button: An innovative find option that allows the search of regular expressions in a simple way.   
   *Implementation:* This is done by collecting inputs from the user, and using the **regex library** to process the query requested.
8. Context Recognizer: An innovative concept that involves recognizing the theme/context that the user is currently attempting to type. On recognition, search links may be provided to the user to arrange for the freedom of easily searching for additional information on text being typed. The theme could be: a letter, an essay, a program, notes, etc.  
   *Implementation:* **Named entity recognition and Topic segmentation**, may be used for the purpose. Tags may be stored for each theme. After few (say 3) lines are typed, make a search for each theme, then identify the theme with most tags that are matched. The links may be provided using url retyping through java classes.

Applications:

1. This text editor basically tries to implement few features that most text/word editing processors (E.g. MS-Word) do not contain.
2. The live spell checker will cut a lot of time on evaluation for all users.
3. Auto-complete mechanisms allow users to type large essays in limited time.
4. The regular expression find option will be a very versatile search option for all users.
5. Theme recognizer will definitely assist the user at whatever he is attempting to type.

References:

1. Speech and Language Processing by Daniel Jurafsky
2. A Spelling Correction Methodology Based on a Noisy Channel Model Mark D. Kemighan Kenneth W. Church William A. Gale
3. Natural Language Corpus Data: Beautiful Data by Peter Norvig
4. http://norvig.com/spell-correct.html
5. Wikipedia
6. Java: The complete reference, 8th edition
7. Dive into Python 3 by Mark Pilgrim

Project Coordinators H.O.D. - CSE