ASSIGNMENT 2 –REPORT

TECHNOLOGY PARK MALAYSIA

CT052-3-M

NATURAL LANGUAGE PROCESSING

HAND IN DATE : 28th NOVEMBER 2022

GROUP NO. : 1

MEMBERS : CHA ZHAN YONG (TP066549)
LEE KEAN LIM (TP065778)
SER JIA QI (TP067540)
VIMAL MOTHI (TP067206)

LECTURER : MR. RAHEEM MAFAS

WEIGHTAGE: 50%

INSTRUCTIONS TO CANDIDATES:

1 Submit your assignment Online via Moodle

2 Students are advised to underpin their answers with the use of references (cited using the APA System of Referencing)

3 Late submission will be awarded zero (0) unless Extenuating Circumstances (EC) are upheld

4 Cases of plagiarism will be penalized

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

GUI ..... Graphical user interface

NRWE ..... Non-real word error

RWE ..... Real-word error

SECTION 1

DESIGN AND FORMULATION

1.1 Introduction

The rapid rise and evolvement of Natural Language Processing in this era have significantly affected our daily lives with its powerful application. One of the NLP's most popular applications is spell checkers for grammar and spelling mistakes which are embedded in various software, easing the workload and efficiency of language users. Spell checking is a process of detecting wrong words from input text by comparing it with a list of correctly spelled words as well as handling morphological, typological, and cognitive mistakes. Suggestions for correct spelling for the detected errors have often been returned in a spell checker by applying string metric algorithms such as the Levenshtein distance. Spelling errors have been generally categorized into two types which are non-word errors and real-word errors for textual documents (Samanta & Chaudhuri, 2013). While non-word signifies the words that do not exist in the dictionary, real-word errors refer to the misspelling of unintended words that are available in the dictionary yet apply in the wrong context such as homophones and grammatical errors. Hence, a spell checker which can detect and correct both of these errors shall be facilitating the efficiency of English language users.

1.2 Spelling Error Checker Design

A spelling error checker that is highly focused on the domain of sciences has been outlined and built. The goal of the designed checker is to detect both non-word errors and real-word errors in the English language. Hence, the designation and arrangement of the components in the graphical user interface (GUI) shall be user-friendly by respectively pointing out both non-word and real-word errors, as well as generating potential word correction for the user to alter the text in the text editor directly. As such, the proposed designation of the spelling checker has been displayed in Figure 1.1. While Table 1.1 shows the description of each component in the GUI.

Figure 1.1: GUI for the spelling error checker

Table 1.1: GUI components and their functions

Label	Components	Usage
1	Text Editor	For user to type in relevant text for spelling checking. The maximum number of words is 500.
2	Check Spelling Button	Users need to click on this button to allow the non-word spelling checking to occur. The detected wrong word will then be highlighted in red in the text editor.
3	Check Real word error Button	Users need to click on this button to allow the real word or grammar checking to occur.
4	Wrong Spelling Word	The detected wrong spelling word will be listed in this panel.
5	Potential Candidate	Potential candidates are the possible words for the detected wrong words typed in the text editor which provide alternative text suggestion to users to alter their text.
6	Real Word Error	It will display the grammatical error part of the inserted text in the text editor error.
7	Suggestion	Suggestion regarding the real word error will be reflected on this window based on the correction computed by the real-word correction model.
8	Corpus List	This is the dictionary which lists the available words in training corpus, arranged from A to Z. In other words, the list has displayed all of the known words
9	Search	To search the word that is available in the dictionary/ corpus list. The dictionary is able to return possible words even if the word in search is typed wrongly.

10	Correct Spelling Button	Users can click on the button for the selected potential candidate (correct word) to replace the wrong word in the text editor.
11	Correct Error	Users can click on the button for the real word suggestion chosen to correct the grammatical or real-word mistake in the text editor.

SECTION 2

IMPLEMENTATION

2.1 Introduction

This section documents the implementation process in the development of the spelling error detection and correction model. Two components, namely the backend which consists of the algorithms for the probabilistic model and the frontend which produces the GUI will be presented. The backend development utilizes two models where each model caters for the detection and correction of the non-word and real-word error respectively. Attached under Appendix A, shows the code snippet utilized for the development of the backend models and frontend GUI.

2.2 Python Dependencies/ Packages

Table 2.1 outlines the python packages and libraires utilized in this study for the development of the system. The composition includes both the frontend and the backend development.

Table 2.1: Python packages and libraries used in development of the system

Package/Library	Description
math	To enable efficient mathematical functions
pickle	To serialize and de-serialize a Python object structure.
operator	To provide a set of efficient functions derives from intrinsic operators
prettytable	For tabular data displaying
numpy	For powerful multidimensional array computing in Python
re	To allow regular expression
nltk	A Python package for NLP
torch	To facilitate tensor computation and development of deep neural network
spacy	For advanced Natural Language Processing
tkinter	For user interface building
csv	Allowing CSV format tabular data to be read and written.
gramformer	To detect, highlight and correct grammar errors

huggingface_hub	To download transformer models
spellchecker	Aid spellchecking in English language

2.3 Training Corpus

Kershaw and Koeling (2020) presented a collection of 40,000 scientific journals across multiple disciplines. Of the 40,000 journals, a random selection of 200 journals from the collection is adopted for the training corpus in this study. The training corpus consists of 783,448 tokens, where 29,190 are unique tokens which form the dictionary for this study. Figure 2.1 shows an example of five sentences from the corpus utilized in this study.

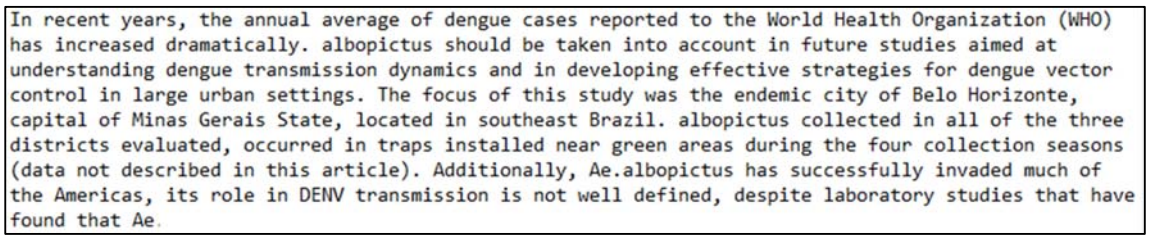


Figure 2.1: Sample view of training corpus

2.4 Non-Word Model

The detection of non-word errors in this study is performed by comparing and identifying words that are not present in the dictionary. Likely candidates are produced for the identified non-word error using a Noisy Channel model integrated with Laplace smoothing Bigrams as the choice of language model, and Damerau-Levenshtein algorithm as the edit distance-based candidate generator. The candidate with the highest combined probability computed from the Noisy Channel model is deemed the best candidate for the identified non-word spelling error.

2.4.1 Candidate Model

The Damerau-Levenshtein algorithm is utilized as the candidate model to compute all probable candidates within the edit distance of one and two. The edit operations utilized by the algorithm to produce the probable candidates include deletion, transposition, insertion, and substitution. The backend development for the candidate model is adapted from the works of Norvig (2007).

An assumption is made where the maximum allowable edit distance for the candidate generation is limited to two edits. This is due to the fact that the majority of mistakes being found are typically within two edit distance away. This confines the number of candidates generated which eases the computation at later stages.

2.4.2 Language Model

Laplace smoothing bigrams is utilized as the language model for this study. It provides some capacity of long-term dependencies from a sentence, which enhances the prediction performance as compared to a unigram model. In addition, it addresses the zero-probability issue when unknown words appear in the evaluation text. The Laplace bigram probability can be computed based on Equation 2.1.

$$P_{bigram}(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n) + 1}{C(w_{n-1}) + V} \quad 2.1$$

Where:

$V = Total\ number\ of\ vocabulary$

$w_{n-1} = Previous\ token$

$w_n = Current\ token$

The probabilities generated from the language model are represented in the logarithmic format. The multiplication of sequence of probabilities would produce a very small value which may result in numerical underflow. The use of logarithmic probabilities addressed such issues and promotes computation speed.

The bigrams model is dependent on the previous and current token to formulate the sequence probability. A challenge is present for the spelling error tokens that are located at the start of sentence and the end of sentence. Therefore, two assumptions are made to address the challenges. For tokens located at the start of the sentence, the Laplace smoothing unigram probability would be assumed using the first word. While tokens that are located at the end of the sentence, the Laplace smoothing unigram probability would be assumed for the last word.

2.4.3 Error Model

The noisy channel model is utilized as the error model for this study. Noises are introduced to the original word in the form of deletion, substitution, insertion, and transposition of letters within the word. The noisy words represent the candidates generated from the candidate model. The most probable candidate will be selected based on the highest likelihood from the product of the channel model and the prior probability which can be summarized by Equation 2.2.

$$\hat{w} = \underset{w \in C}{\operatorname{argmax}} P(x|w)\, P(w) \qquad 2.2$$

Where:

$P(x|w) = Channel\ model\ probability$

$P(w) = Prior\ probability$

When computing the channel model probability, an assumption is made where the candidates would only derive from edit distance of one or two. This facilitates the use of the four confusion matrices for counts of errors consisting of the deletion, insertion, substitution, and transposition of letters to compute the channel model probability.

The following shows an example of identifying non-word spelling error in a sentence and computation of the noisy channel probability for the candidates generated. The following sentence will be used for demonstration:

"I *haver* a dog."

It is obvious that the word "haver" is a spelling error. It is identified by comparing each word in the sentence to the dictionary. Words not found in the dictionary are deemed as non-word errors. Figure 2.2 shows the computation of the noisy channel model for each candidate suggested by the candidate model. It is observed from the figure that the candidate "have" has the highest noisy channel probability thus it will be chosen as the best candidate to replace the error word. It is to note that the computed probabilities are in the negative logarithmic format, thus a lower negative value would signify higher likelihood. Therefore, the corrected sentence would be "I have a dog.".

Correct Word Candidate (C)	Edit Distance	Edit Operations(s) [I->C]	-log[P(C)]	-log[P(SEQ)]	-log[P(I|C)]	-log[P(SEQ)P(I|C)]
have	1	[('D', 'er#', 'e#')]	2.72389	8.51443	3.74223	12.25666
gaver	1	[('S', '#ha', '#ga')]	6.91927	13.45418	4.84096	18.29514
wave	2	[('D', 'er#', 'e#'), ('S', '#ha', '#wa')]	3.96939	7.44776	7.94224	15.39
eave	2	[('D', 'er#', 'e#'), ('S', '#ha', '#ea')]	5.06801	6.65985	9.40875	16.0686
gave	2	[('D', 'er#', 'e#'), ('S', '#ha', '#ga')]	4.47056	9.67335	8.58319	18.25655
save	2	[('D', 'er#', 'e#'), ('S', '#ha', '#sa')]	5.2117	12.52476	8.83558	21.36034
hale	2	[('D', 'er#', 'e#'), ('S', 'ave', 'ale')]	6.91927	13.45418	8.64097	22.09515
ave	2	[('D', 'er#', 'e#'), ('D', '#ha', '#a')]	6.91927	13.45418	9.05869	22.51288
hare	2	[('D', 'er#', 'e#'), ('S', 'ave', 'are')]	6.91927	13.45418	9.11186	22.56604

Figure 2.2: Probability outputs of candidates from noisy channel model

2.5 Real-Word Model

The real-word model aims to identify and correct the syntax and semantics of a sentence. A pretrained sequence-to-sequence transformer by Damodaran (2021), is adopted in this study as the real-word error detection and correction model. Attached under Appendix A, shows the code snippet for loading the repository and dependencies for utilization of the transformer model. Figure 2.3 shows the code snippet initiating the pretrained model on a list of sentences filled with grammatical errors. While in Figure 2.4, shows the output from the model where each sentence is corrected.

The model is capable of rectifying syntactic and semantic errors at sentence level. In addition, the model is capable of suggesting punctuation marks to be placed in the sentence based on context. It is trained using sentences up to 64 words in length thus it is highly capable to rectify common syntactic and semantic errors found in sentences. However, the model outputs only one probable sentence for each given input, thus no selection or customization of the output is available.

```
# Sample run
# Input string sentences
influent_sentences = [
    "He are moving here.",
    "I am doing fine. How is you?",
    "How is they?",
    "Matt like fish",
    "the collection of letters was original used by the ancient Romans",
    "We enjoys horror movies",
    "Anna and Mike is going skiing",
    "I walk to the store and I bought milk",
    "what be the reason for everyone leave the company",
]

# Initiate real-word correction
for influent_sentence in influent_sentences:
    corrected_sentences = gf.correct(influent_sentence, max_candidates=1)
    print("[Input] ", influent_sentence)
    for corrected_sentence in corrected_sentences:
        print("[Correction] ",corrected_sentence)
    print("-" *100)
```

Figure 2.3: Initiate real-word error model

```
[Input]  He are moving here.
[Correction]  He is moving here.
--------------------------------------------------------------------------------
[Input]  I am doing fine. How is you?
[Correction]  I am doing fine. How are you?
--------------------------------------------------------------------------------
[Input]  How is they?
[Correction]  How are they?
--------------------------------------------------------------------------------
[Input]  Matt like fish
[Correction]  Matt likes fish.
--------------------------------------------------------------------------------
[Input]  the collection of letters was original used by the ancient Romans
[Correction]  the collection of letters was originally used by the ancient Romans
--------------------------------------------------------------------------------
[Input]  We enjoys horror movies
[Correction]  We enjoy horror movies.
--------------------------------------------------------------------------------
[Input]  Anna and Mike is going skiing
[Correction]  Anna and Mike are going skiing.
--------------------------------------------------------------------------------
[Input]  I walk to the store and I bought milk
[Correction]  I walked to the store and I bought milk.
--------------------------------------------------------------------------------
[Input]  what be the reason for everyone leave the company
[Correction]  what is the reason for everyone leaving the company?
--------------------------------------------------------------------------------
```

Figure 2.4: Output of corrected sentence from real-word model

2.6 Integration with Frontend

This section documents the development of the GUI which outlined the step and functions utilized.

Step 1: Integration of non-word checker

The code snippets for model integration with the GUI have been demonstrated in Figure A.16, A.17, and A.18 under Appendix A. Outlined in Table 2.2. the functions that integrate with the formerly built non-word error model to facilitate further GUI creations.

Table 2.2: User defined functions created for non-word error checking

Function	Description
checkSpelling	This function is to check the input text from the text editor and identify the real word error and initialize the spelling word and its candidate based on key-value pair.
highLightText	This function will get the corpus word from the corpus list and based on this value it will highlight the word in the text editor
ChooseCandidate	This function will get the wrong spelling word from the list and based on the wrong spelling word it will highlight the exact text in the text editor
correctSpelling	This will replace the wrong spelling word with the selected candidate word and change the text in the text editor

Step 2: Integration of real-word checker

In the current step, the user-defined function outlined in Table 2.3 has been utilized to facilitate the presentation and functionality of the spelling checker in the GUI window.

Table 2.3: User defined functions created for real-word error checking

Function	Description
highLightRealTextError	This function will highlight both the difference between the original sentence and the corrected sentence in the text editor and the suggestion text box.
checkError	This function will get the input from the text editor and perform correct() function of real-word model. Based on the return it will compile the difference of both original and corrected sentence.
correctError	This function will perform correction of the sentence in the text editor and replace it with the corrected sentence from real-word model.

Step 3: Complete system compilation with GUI

The codes are focused on creating windows and widgets in the GUI using the tkinter library. The integration with the spelling-checking commands defined in the last step has been compiled at this stage to allow users to interact with the system in the GUI windows generated for spelling checking. Outlined in Table 2.4, the main functions used from tkinter library.

Table 2.4: Tkinter library function

Function	Description
tk()	To create a GUI window.
text()	To allow multi-line text entry
label()	To display a single line of text on GUI window.
place()	To provide exact position to place the widget.
title()	To name the GUI window.
Listbox()	To display a list of options for the user to choose from.
button ()	To provide button function on the application.

SECTION 3

RESULTS AND EVALUATION

3.1 Introduction

This section provides a sequential demonstration of the system in detecting and correcting the real word and non-real word errors. In addition, the latter part of the section evaluates the model performance and identifies the strengths and weaknesses of the system.

3.2 Non-Word Detection and Correction

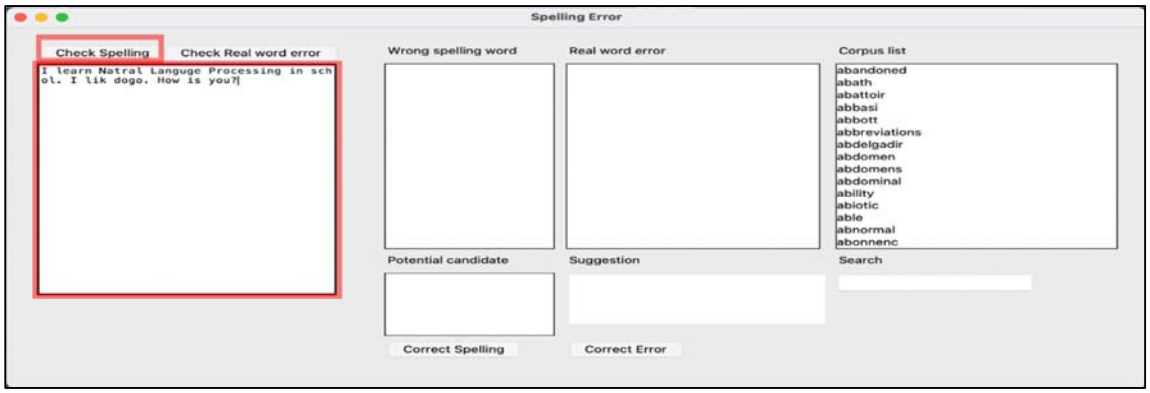


Figure 3.1: Input to the text editor for spelling checking

Illustrated in Figure 3.1, the user is able to input text into the text editor window to perform spelling checks. To check for the non-word error, user is required to click on the ‘check spelling’ button once the text is inserted.

After the user clicked on the ‘check spelling’ button, the system will automatically return the detected non-word spelling errors which are highlighted in red in the text editor as illustrated in Figure 3.2. Additionally, the wrong spelling words will also be displayed in the ‘wrong spelling word’ window individually for reference.

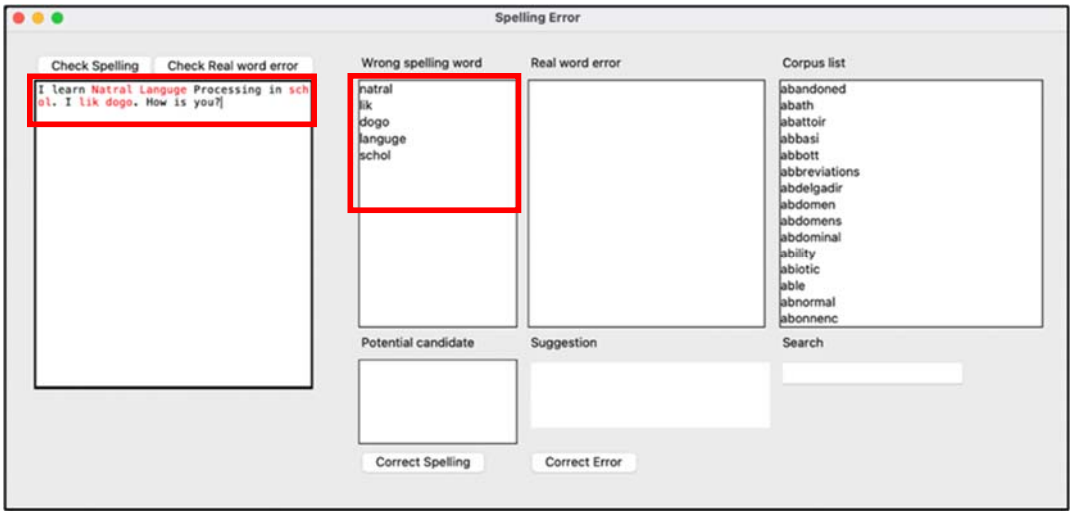


Figure 3.2: Wrong spelling words highlighted in red and listed in checker

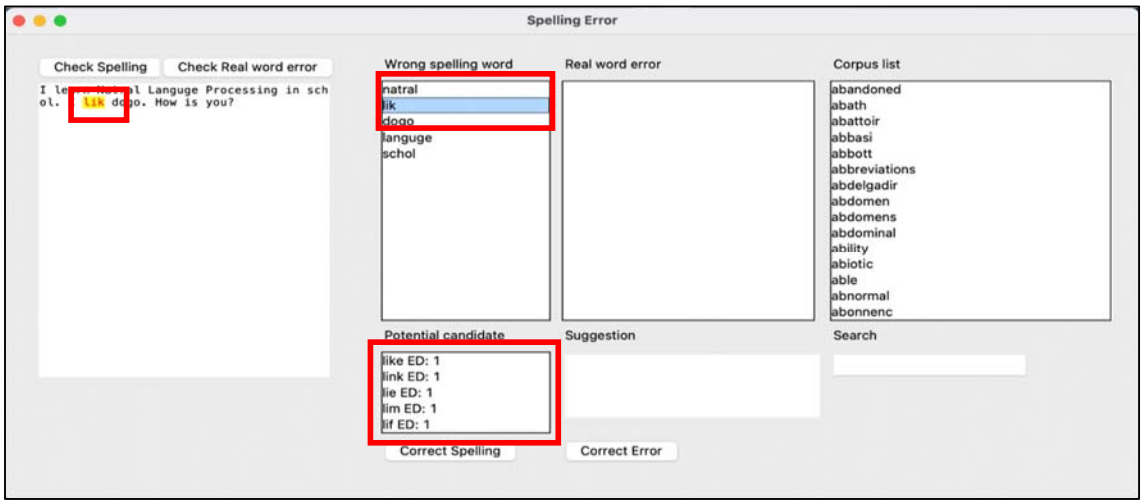


Figure 3.3: Click on the listed wrong spelling word to check for potential candidate

The users are able to click on one of the specific detected wrong words to check for any potential candidates that are available to further correct the mistake as illustrated in Figure 3.3. In this example, the user selects the detected error word ‘lik’. The potential candidate will then return the possible replacement word based on the edit distance algorithm generated from the backend. In addition, the list is sorted based on the probability generated from the noisy channel model. Where, the most probable candidate is listed at a higher position. The edit distance has been displayed for reference. When the specific wrong word is selected, the wrong word typed in the text editor will be highlighted in yellow to prompt user the word they are selecting and wanted to replace.

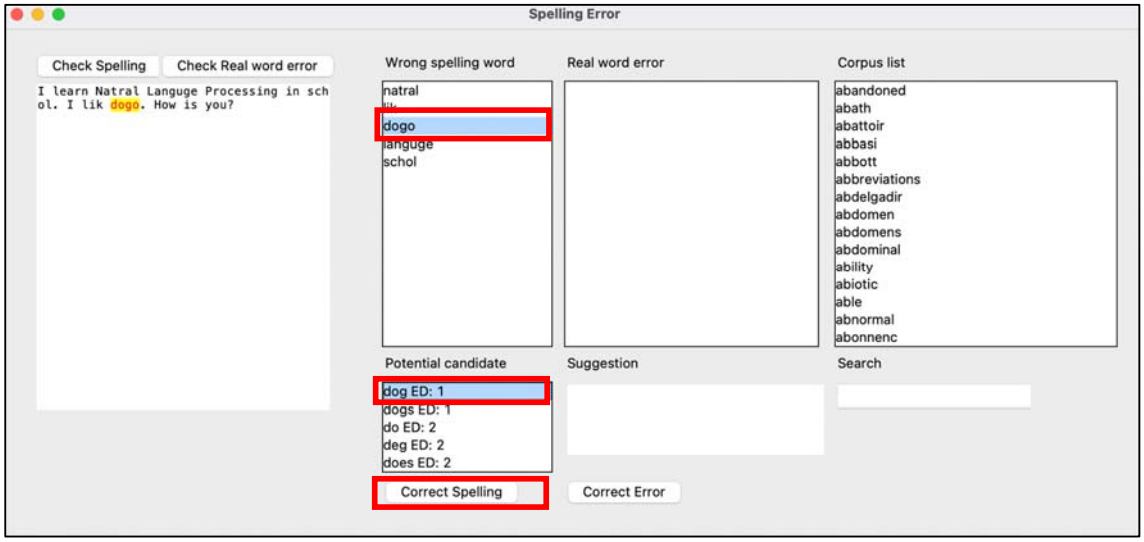


Figure 3.4: Choosing desired candidate from potential candidate list

Next, the user selects the detected error word ‘dogo’ as shown in Figure 3.4. The potential candidate window returns the possible candidates such as ‘dog’, ‘deg’, ‘does’ and ‘dogs’. Based on the candidates displayed, the user can be informed that the word ‘dog’ has the lowest edit distance and ranked highest in the list. The user can then decide to select on the candidate ‘dog’, which is more probable to be the correct word to replace the detected error ‘dogo’ word highlighted in the text editor. To proceed with the correction, the user can click on the correct the non-word spelling to correct specifically for the error ‘dogo’ word. However, if the user wishes, other candidates can be selected as well, which the user deems fits the context better.

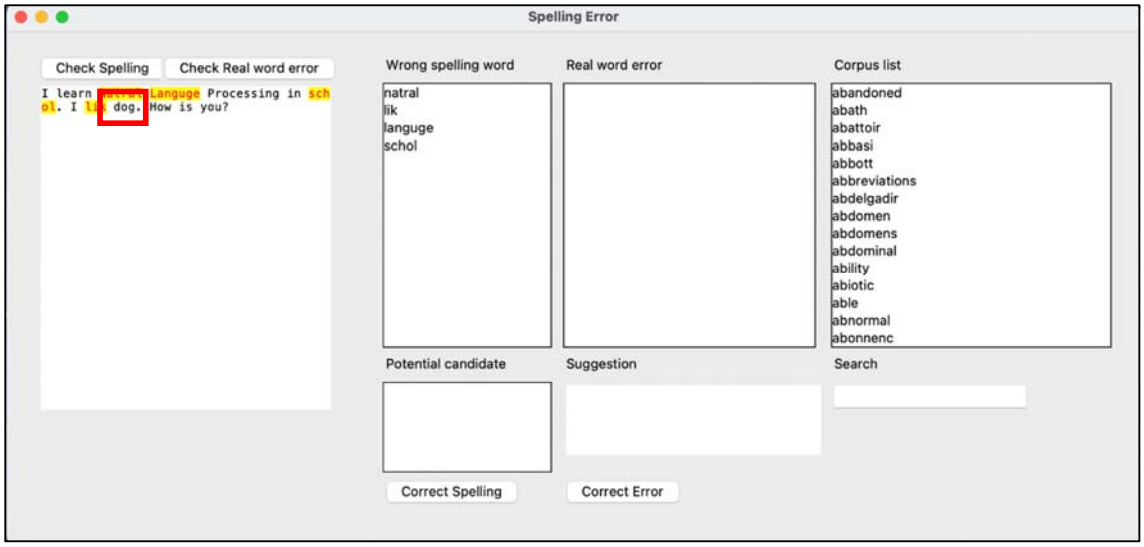


Figure 3.5: Corrected spelling after clicking ‘Correct Spelling’ button

Illustrated in Figure 3.5, it can be observed that the word ‘dogo’ in text editor has been corrected into ‘dog’ as chosen by the user. The highlights in red and yellow will both be removed after the word has been corrected. Yet, other non-word spelling mistakes will remain highlighted until they have been corrected one by one by selecting the potential candidate to replace them.

3.2 Real Word Error Correction

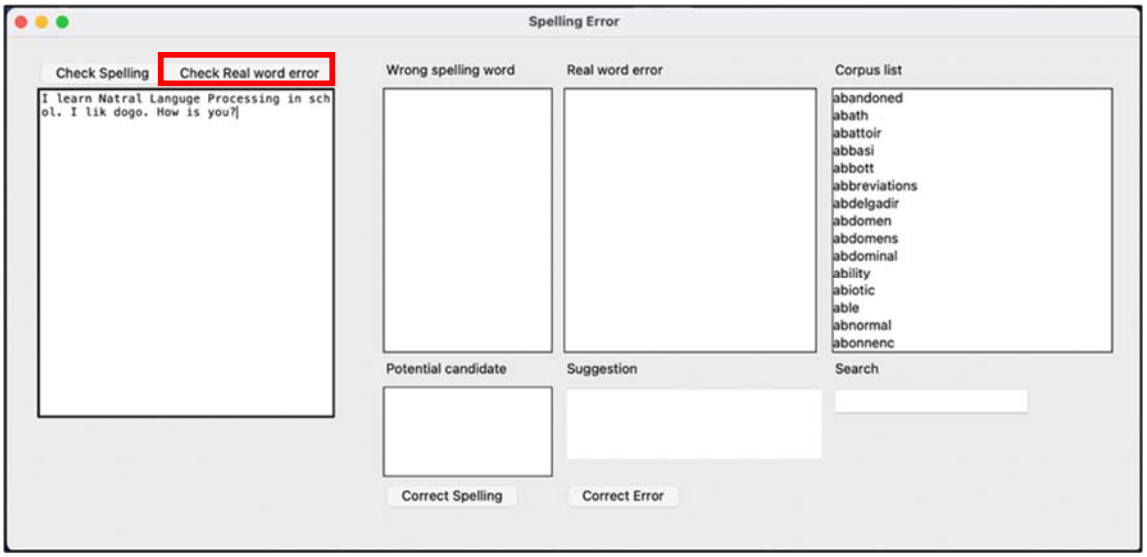


Figure 3.6: Checking real-word error with the ‘Check Real Word Error’ button

To check real word error, the user shall be able to insert a text of maximum 500 words into the text editor and click on the ‘check real word error’ button as shown in Figure 3.6.

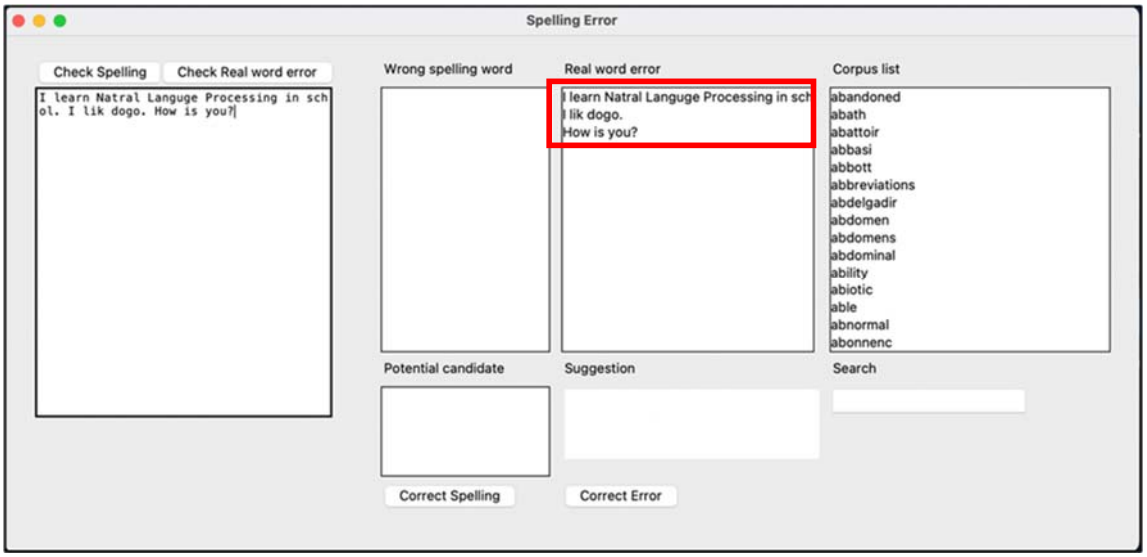


Figure 3.7: Errors listed in the ‘Real word error’ column

Once the system has detected any grammatical error, the entire sentence will show up in the 'real word error' window as illustrated in Figure 3.7.

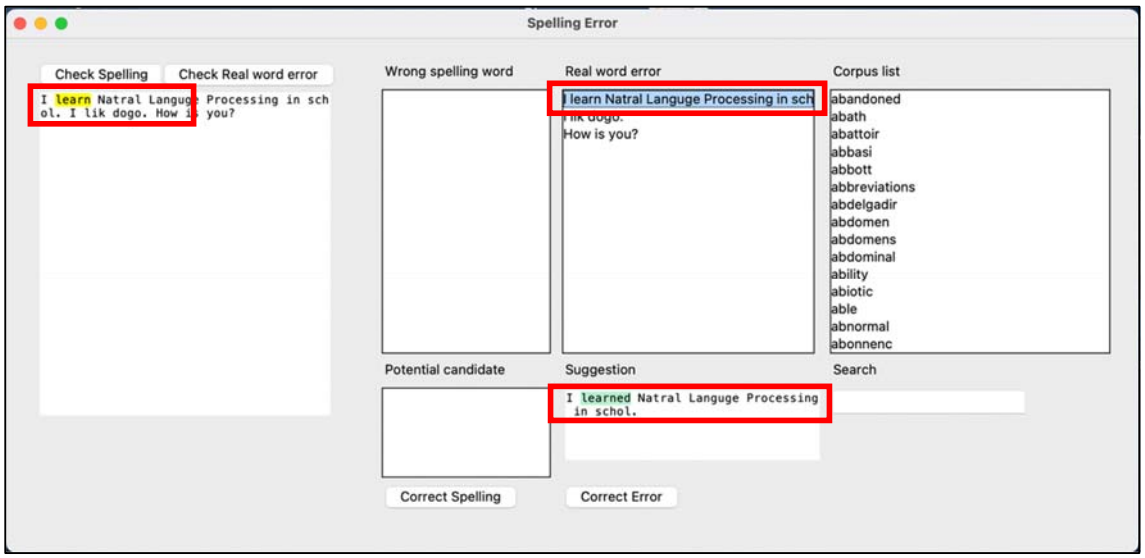


Figure 3.8: Suggested replacement for the sentence

Illustrated in Figure 3.8, as the user selects any of the error sentence displayed in the 'real-word error window', the 'suggestion' column will show the suggested correction to be applied to the mistakes made in the text editor. Furthermore, the real-word error to be corrected will also be highlighted in the text editor to prompt and emphasize the errors to the users.

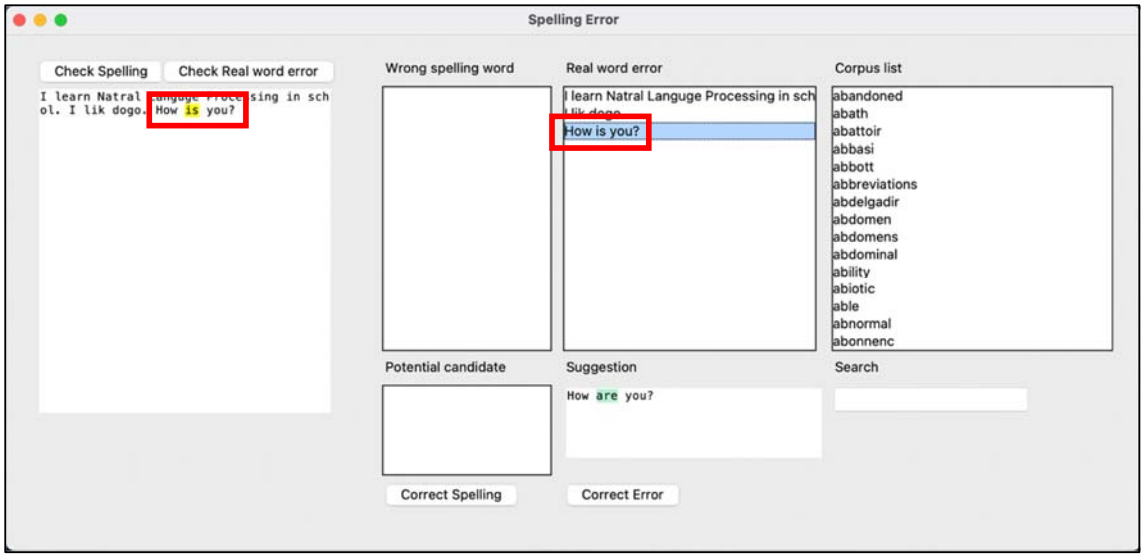


Figure 3.9: Error highlighted in text editor when error is chosen in the list

Shown in Figure 3.9, for instance, the ‘How is you?’ sentence has been detected to have grammatical error in the ‘real word error’ panel. Once the user selects the error sentence in the panel, the ‘suggestion’ column will then return the possible correction. To directly correct the mistakes in the text editor, the users can click on the ‘correct error’ button and proceed the correction.

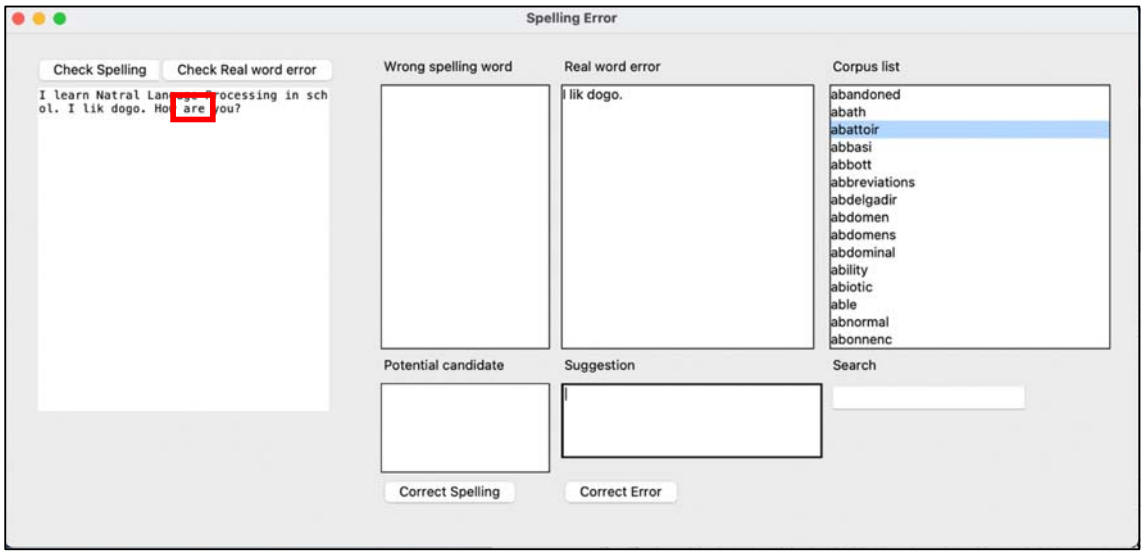


Figure 3.10: Corrected real-word error

Shown in Figure 3.10, it can be observed that the ‘How is you?’ sentence in text editor has been corrected into ‘How are you?’ as previously displayed in the ‘suggestion’ panel. The highlights in yellow will be removed as the mistakes have been amended. Nevertheless, other real-word spelling mistakes or sentences will remain displayed in the ‘real word error’ window until they have been corrected one by one by the user as suggested by the system.

3.3 Search and Dictionary

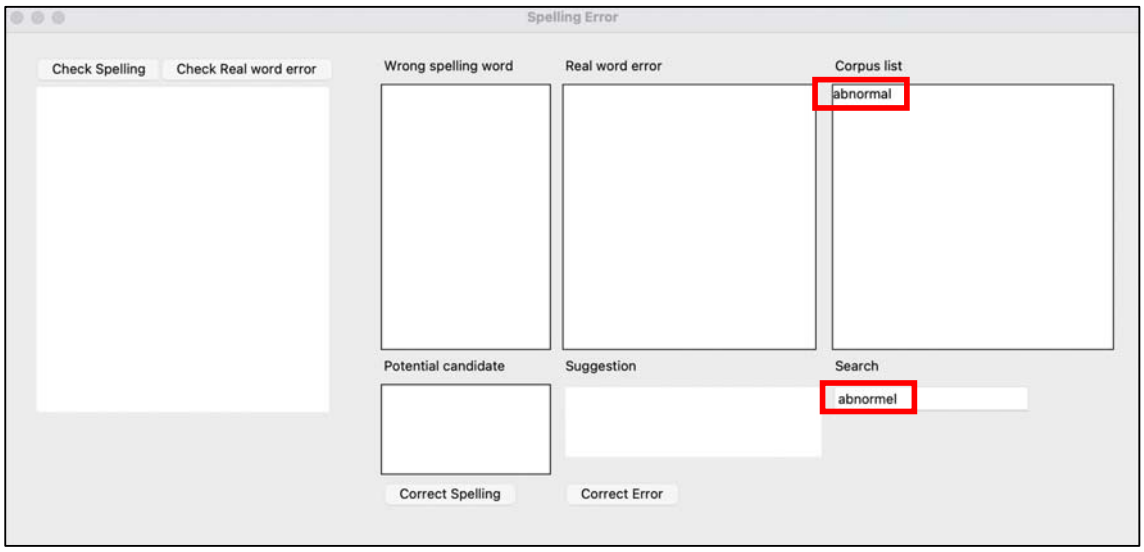


Figure 3.11: Search function and dictionary

Shown in Figure 3.11, the ‘search’ panel allows users to type the word that they are unsure of in the window to check upon the corpus dictionary. The dictionary returns the closest vocabulary even when the user has inserted a word with slightly different spelling in the search bar. This is due to the integration of candidate generation from the edit distance algorithm and noisy channel model. Thus, the user does not need to type in the exact spelling to identify a word from the dictionary.

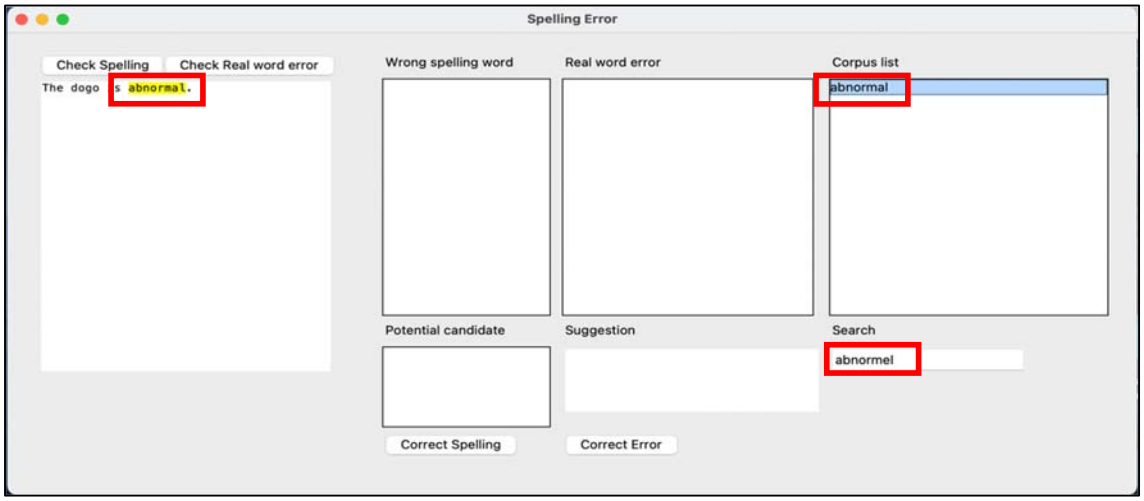


Figure 3.12: Searching for words in text editor

The user may select a word listed in the ‘corpus list’ column, and if a word is present in the evaluation text, the word in the text editor will be highlighted in yellow as shown in Figure 3.12.

This function allows the user to check for any available word from the dictionary even when they accidentally encounter typographical errors which makes the system more user-friendly.

3.4 Model Performance

The compiled model was evaluated across a range of sentences embedded with non-real words and real words error. Each sentence is evaluated using the GUI mentioned in previous sections. A total of 6 evaluation sets are used to gauge model performance. Each evaluation set consists of 10 sentences with errors randomly allocated throughout the text. The general experimental procedure follows the steps below:

Non-Real Word Error:

1) Enter text into the input window
2) Click on check spelling
3) Review candidate words
4) Select candidate words that match

   Note: Model evaluation is given +1 if correct word is generated amongst candidate words.
5) Compute Non-Real Words Accuracy

Real Word Error:

1) Click on check real word error after non-real word error has been fixed if present
2) Review candidate sentences
3) Select candidate sentences that match

   Note: Model evaluation is given +1 if correct grammar is generated amongst candidate words.
4) Compute Real Words Accuracy

Tabulated in Table 3.1 are the results obtained from evaluating the model across 6 evaluation sets. In total there are 50 non-real word errors and 45 real word errors. From evaluation results, the developed model produced reliable results with 79.81% accuracy for non-real word error detection and 83.37% accuracy for real word error detection. Details regarding each sentence and its non-real word error and real word error are provided under Appendix B. In addition to that, the results of each sentence evaluation are attached as screenshots and documented in Appendix B.

Table 3.1: Model Evaluation Results

	Non-Real Words Error			Real Words Error		
Set No.	**Quantity**	**Detected**	**Accuracy NRW (%)**	**Quantity**	**Detected**	**Accuracy RWE (%)**
1	8	6	75.00	9	8	88.89
2	6	5	83.33	9	6	66.67
3	9	5	55.56	8	7	87.50
4	8	6	75.00	6	6	100.00
5	10	9	90.00	7	4	57.14
6	9	9	100.00	6	6	100.00
		Average Accuracy	**79.81**		**Average Accuracy**	**83.37**

3.5 Limitation of the System

- The training corpus is relevant to scientific journals only which leads to undertraining for words related to other domains.
- Only 200 journals have been used to generate the training corpus in which the spelling check developed is highly insensitive to many other words.
- The system is unable to show both grammar mistakes (real-word error) and non-word error at the same time in terms of highlighting them in another color.
- Does not work with spelling errors with an edit distance greater than 2.
- Does not work when the word is not found in the dictionary in addition no available candidates are generated for that word.
- Does not work on multi-word phrases.
- As the learning corpus is extracted from open-source journals, there are the possibility of informal English such as abbreviations and terms of other languages to be included in the corpus.
- The real-word error model is highly focused on grammar mistakes but is less sensitive in detecting homophones and other morphological mistakes.

3.6 Strength of the System

- The model has taken both the bigram model and unigram prior probability into account which calculates the probability of the existence of specific words in the training corpus while taking the context into account. This is a good approach for a higher chance of correct words occurring for the non-word error.
- It works for both sentence level and word level of correction.

- Smoothing have been done for both unigram and bigram which eliminates the effect of zero probability.

CONCLUSION

To sum up, the current study has demonstrated a system that is sufficient to identify non-word errors and real-word errors by utilizing various models and framework in Natural Language Processing such as edit distance, noisy channel, etc. Additionally, the backend logic has been integrated to the graphical user interface for users to interact and provide input for spelling checking. Nevertheless, there are still some limitations of the model that need to be modified to better detect and correct spelling errors which shall be further investigated in future studies.

REFERENCES

Damodaran, P. (2021). Gramformer version 1.4. Retrieved from https://github.com/PrithivirajDamodaran/Gramformer

Kershaw, D., & Koeling, R. (2020). Elsevier oa cc-by corpus. arXiv preprint arXiv:2008.00774.

Norvig, P. (2007). How to Write a Spelling Corrector. Retrieved from https://norvig.com/spell-correct.html

Samanta, P., & Chaudhuri, B. B. (2013, October). A simple real-word error detection and correction using local word bigram and trigram. In *Proceedings of the 25th conference on computational linguistics and speech processing (ROCLING 2013)* (pp. 211-220).

APPENDIX A

CODE SNIPPETS

A.1 Non-Word Error Model

Candidate Model

Figure A.1 shows the code snippet for the edit distance matrix generator given two strings as input. While Figure A.2 shows the code snippet for candidate suggestion derived from the deletion operation.

```
# Generate edit distance matrix
def getEditDistance( x , y ):
    m = len(x)
    n = len(y)
    arr = np.array([0]*((m+1)*(n+1))).reshape( m+1 , n+1 )
    for a in range(m):
        arr[a+1][0] = a+1
    for a in range(n):
        arr[0][a+1] = a+1
    for i in range(1,m+1):
        for j in range(1,n+1):
            v1 = arr[i-1][j] + 1
            v2 = arr[i][j-1] + 1
            v3 = arr[i-1][j-1]
            if x[i-1] != y[j-1]:
                v3 += 1
            v4 = INFINITY
            if ( i > 1 and j > 1 and x[i-1] == y[j-2] and x[i-2] == y[j-1] ):
                v4 = arr[i-2][j-2] + 1
            v = [ v1 , v2 , v3 , v4 ]
            arr[i][j] = INFINITY
            for each in v:
                if each < arr[i][j]:
                    arr[i][j] = each
    return arr
```

Figure A.1: Edit distance matrix generator

The getEditDistance() function generates the edit distance matrix based on input of two strings.

```
def getCandidates( word , root , s , lev ):
    letters    = 'abcdefghijklmnopqrstuvwxyz'
    store_1 = []
    # Candidates with 1 edit distance
    if ( not len(word) ):
        return
    splits      = [(word[:i], word[i:])     for i in range(len(word) + 1)]
    deletes     = [L + R[1:]                for L, R in splits if R]
    transposes  = [L + R[1] + R[0] + R[2:]  for L, R in splits if len(R)>1]
    replaces    = [L + c + R[1:]            for L, R in splits if R for c in letters]
    inserts     = [L + c + R                for L, R in splits for c in letters]
    for w in deletes:
        if (w in VOCAB):
            store_1.append(w)
    for w in transposes:
        if (w in VOCAB):
            store_1.append(w)
    for w in replaces:
        if (w in VOCAB):
            store_1.append(w)
    for w in inserts:
        if (w in VOCAB):
            store_1.append(w)
    print('Candidates: ', store_1)
    # Candidates with 2 edit distance
    if lev == 1:
        for cand_1 in store_1:
            splits      = [(cand_1[:i], cand_1[i:])      for i in range(len(cand_1) + 1)]
            deletes     = [L + R[1:]                     for L, R in splits if R]
            transposes  = [L + R[1] + R[0] + R[2:]       for L, R in splits if len(R)>1]
            replaces    = [L + c + R[1:]                 for L, R in splits if R for c in letters]
            inserts     = [L + c + R                     for L, R in splits for c in letters]
            for w in deletes:
                if len(w) > 1:
                    if (w in VOCAB) and (w not in s):
                        s.append(w)
            for w in transposes:
                if len(w) > 0:
                    if (w in VOCAB) and (w not in s):
                        s.append(w)
            for w in replaces:
                if len(w) > 0:
                    if (w in VOCAB) and (w not in s):
                        s.append(w)
            #for w in inserts:
                if len(w) > 0:
                    if (w in VOCAB) and (w not in s):
                        s.append(w)
```

Figure A.2: Candidates generator for edit distance one and two

The getCandidates() function generates probable candidates for a given word based on the four operations namely insertion, deletion, transposition, and substitution. A given non-word would undergo the first iteration to generate the probable real-word candidates which has an edit distance of one. Following, each real-word candidate would undergo second operation to generate more probable candidates which represent candidates with edit distance of two.

Language Model

Figure A.3 shows the code snippet for unigrams and bigrams prior probabilities calculator. While Figure A.4 shows the code snippet for the Laplace bigrams in sequence probability generator.

```
# Unigram prob in log scale
def getUnigramProb( w ):
    if ( w == '' or w == '#' ):
        return log( S + 0.1 ) - log( N + 0.1*V )
    co = len( re.findall( ' ' + w + '[.\s]' , CORPUS ) )
    ca = log( co + 1 ) - log( N + 1*V )
    return ca

# Bigram prob in log scale
def getBigramProb( w1 , w2 ):
    if ( w1 == '' or w1 == '#' ):
        c12 = len( re.findall( '. ' + w2 + '[.\s]' , CORPUS ) ) + len( re.findall( '^' + w2 + '[.\s]' , CORPUS ) )
        c1 = S
        return log( c12 + math.pow(10,getUnigramProb(w2)) ) - log( c1 + 1 )
    if ( w2 == '' or w2 == '#' ):
        c12 = len( re.findall( ' ' + w1 + '.' , CORPUS ) )
        c1 = len( re.findall( ' ' + w1 + '[.\s]' , CORPUS ) )
        return log( c12 + math.pow(10,getBigramProb(w2)) ) - log( c1 + 1 )
    c12 = len( re.findall( ' ' + w1 + ' ' + w2 + '[.\s]' , CORPUS ) )
    c1 = len( re.findall( ' ' + w1 + '[.\s]' , CORPUS ) )
    return log( c12 + math.pow(10,getUnigramProb(w2)) ) - log( c1 + 1 )
```

Figure A.3: Laplace smoothing unigrams and bigrams prior probabilities

The getUnigramProb() function generates the Laplace unigram probability for a given string. While the getBigramProb() function generates the Laplace bigram probability given two strings as input.

```
# Sequence prob using bigram
def getSequenceProb( words , pos ):
    l = len( words )
    if l == 1 and not pos:
        return getBigramProb( '#' , words[pos] ) + getBigramProb( words[pos] , '#' )
    if pos == l-1:
        if words[pos-1] in VOCAB:
            return getBigramProb( words[pos-1] , words[pos] ) + getBigramProb( words[pos] , '#' )
        return getBigramProb( words[pos] , '#' )
    if not pos:
        if words[pos+1] in VOCAB:
            return getBigramProb( '#' , words[pos] ) + getBigramProb( words[pos] , words[pos+1] )
        return getBigramProb( '#' , words[pos] )
    if words[pos-1] in VOCAB:
        if words[pos+1] in VOCAB:
            return getBigramProb( words[pos-1] , words[pos] ) + getBigramProb( words[pos] , words[pos+1] )
        return getBigramProb( words[pos-1] , words[pos] )
    if words[pos+1] in VOCAB:
        return getBigramProb( words[pos] , words[pos+1] )
    return getUnigramProb( words[pos] )
```

Figure A.4: Laplace bigrams sequence probability

The getSequenceProb() function generates the sequence probability for a given sentence based on Laplace bigram probability.

Error Model

Figure A.5 shows the code snippet for computing the edit operation between the error word and the candidate word. The edit operations comprise the deletion, insertion, substitution, and transposition of letters within the error word and the candidate word. Figure A.6 shows the code snippet for computing the channel model probability for a single candidate based on the edit operation computed from the code snippet of Figure A.5. In addition, confusion matrices for counts of errors are pre-loaded into the environment from a text file. Shown in Figure A.7, the code snippet for showing the compilation of noisy channel probability for all generated candidates.

```
# Returns edits operation
def getEditOperation( arr , x , y ):
    edits = list()
    i = len(x)
    j = len(y)
    while( i>=0 and j>=0 ):
        if ( not i and not j ):
            break
        p = 0
        if i>0 and j>0 and x[i-1] != y[j-1]:
            p = 1
        if i>0 and j>0 and arr[i][j] == arr[i-1][j-1] + p:
            if p:
                if i == 1:
                    edits.append(('S','#'+x[i-1]+x[i],'#'+y[j-1]+x[i]))
                elif i < len(x):
                    edits.append(('S',x[i-2]+x[i-1]+x[i],x[i-2]+y[j-1]+x[i]))
                else:
                    edits.append(('S',x[i-2]+x[i-1]+'#',x[i-2]+y[j-1]+'#'))
            i = i-1
            j = j-1
            continue
        elif i>0 and arr[i][j] == arr[i-1][j] + 1:
            if i == 1:
                edits.append(('D','#'+x[i-1]+x[i],'#'+x[i]))
            elif i < len(x):
                edits.append(('D',x[i-2]+x[i-1]+x[i],x[i-2]+x[i]))
            else:
                edits.append(('D',x[i-2]+x[i-1]+'#',x[i-2]+'#'))
            i = i-1
            continue
        elif j>0 and arr[i][j] == arr[i][j-1] + 1:
            if not i:
                edits.append(('I','#'+x[i],'#'+y[j-1]+x[i]))
            elif i < len(x) :
                edits.append(('I',x[i-1]+x[i],x[i-1]+y[j-1]+x[i]))
            else:
                edits.append(('I',x[i-1]+'#',x[i-1]+y[j-1]+'#'))
            j = j-1
            continue
        elif ( i > 1 and j > 1 and x[i-1] == y[j-2] and x[i-2] == y[j-1] and arr[i-2][j-2]+1==arr[i][j] ):
                edits.append(('T',x[i-2]+x[i-1],x[i-1]+x[i-2]))
                i = i-2
                j = j-2
    return edits
```

Figure A.5: Identify edit operation

```
# Likelihood of edits
def getLikelihood( edits ):
    L = 0
    for edit in edits:
        if edit[0] == 'D':
            if edit[1][0] == '#':
                length = N
            else:
                length = len( re.findall(edit[1][0],CORPUS) )
            L = L + log(CMI[h(edit[1][0],'v')][h(edit[1][1],'h')]) - log(length)
        elif edit[0] == 'I':
            if edit[1][0] == '#':
                length = len( re.findall(' '+edit[2][1],CORPUS) )
            else:
                length = len( re.findall(edit[2][0]+edit[2][1],CORPUS) )
            L = L + log(CMD[h(edit[2][0],'v')][h(edit[2][1],'h')]) - log(length)
        elif edit[0] == 'S':
            length = len( re.findall(edit[2][1],CORPUS) )
            L = L + log(CMS[h(edit[1][1],'v')][h(edit[2][1],'h')]) - log(length)
        elif edit[0] == 'T':
            length = len( re.findall(edit[2][0]+edit[2][1],CORPUS) )
            L = L + log(CMT[h(edit[2][0],'v')][h(edit[2][1],'h')]) - log(length)
    return L
```

Figure A.6: Compute channel model likelihood for single candidate

```
# Non-word model
def bestCandidate( words , pos , choice , CHANGES , sen ):
    ORG = words[pos]
    print('\tNON-WORD ERROR : ' , ORG.upper()) # Identifies the non-word error.
    cand = set()
    getCandidates( ORG , ORG , cand , 1 )
    rows = list()
    for c in cand:
        arr = getEditDistance( ORG , c )
        ed = arr[len(ORG)][len(c)]
        if ( ed > 2 ):
            continue
        log_pc = getUnigramProb(c)
        words[pos] = c
        seq_pr = getSequenceProb( words , pos )
        edits = getEditOperation( arr , ORG , c )
        log_likel = getLikelihood( edits )
        rows.append([c, ed, edits, round(-1*log_pc,5), round(-1*seq_pr,5), round(-1*log_likel,5), round(-1*seq_pr-log_likel,5)])

    rows.sort( key = lambda x: (x[1],x[6]) )
    print( '\t       [ {} CANDIDATES SHORT-LISTED ]'.format(len(rows)) ) # Candidates
    for r in range(len(rows)):
        #print('Candidates', 'Edit distance')
        print('Candidate: ', rows[r][0],'   ', '\tEdit distance: ', rows[r][1])
    words[pos] = rows[0][0]
    CHANGES[(sen,pos)] = ( ORG, words[pos] )
    print( 'BEST CANDIDATE : {}'.format(words[pos].upper()) )
    print( '\n' , end = '' )
    print( )
    if choice == 'y':
        table = PrettyTable(["Correct Word Candidate (C)", "Edit Distance",
                             "Edit Operations(s) [I->C]", "-log[P(C)]", "-log[P(SEQ)]", "-log[P(I|C)]", "-log[P(SEQ)P(I|C)]"])
        for row in rows:
            table.add_row(row)
        print(table)
        print( '\n' , end='' )
    return rows
```

Figure A.7: Noisy channel probability computation for all candidates

A.2 Real-Word Error Model

Figure A.8 shows the code snippet for loading the dependencies and the real-word error correction model from GitHub repository. While in Figure A.9, shows the code snippet to initiate the sequence-to-sequence transformer model.

```
!pip install -U git+https://github.com/PrithivirajDamodaran/Gramformer.git
!pip install spacy
!python -m spacy download en_core_web_sm
!python -m spacy download en
!pip install torch

# Dependencies for Gramformer
from gramformer import Gramformer
import torch
import spacy
```

Figure A.8: Dependencies for the transformer model

```
def set_seed(seed):
  torch.manual_seed(seed)
  if torch.cuda.is_available():
    torch.cuda.manual_seed_all(seed)

set_seed(1212)

# Initialize Gramformer
gf = Gramformer(models = 1, use_gpu=False) # 1=corrector, 2=detector
```

Figure A.9: Initialization of the transformer model

A.3 Model Combination

Figure A.10 shows the code snippet for model combination of Real Word Checker and Non-Word Checker.

```
TB = TreebankWordDetokenizer()

#--------------------------------------------------------------------------
# Non-word model
# Calling main() to initiate
#input_corpus = "I haven't bailed on writing.
# Look, I'm generating a random paragraph at this very moment in an attempt to get my writing back on track.
# I am making an effort. I will start writing consistently again."
input_corpus = "I dogo"
return_NW = main(input_corpus, 'n')
print("\nhere")
# Detokenize return_NW
print(return_NW)
detoken = []
for i in return_NW:
    print(i)
    detoken.append(TB.detokenize(i))
print("detoken")
print(detoken)
print('\n')
print("end")
#--------------------------------------------------------------------------
# Real-word model
for sent in detoken:
    corrected_sentences = gf.correct(sent, max_candidates=1)
    print("[Input] ", sent)
    for corrected_sentence in corrected_sentences:
      print("[Correction] ", corrected_sentence)
    print("\n")
```

Figure A.10: Model combination operation

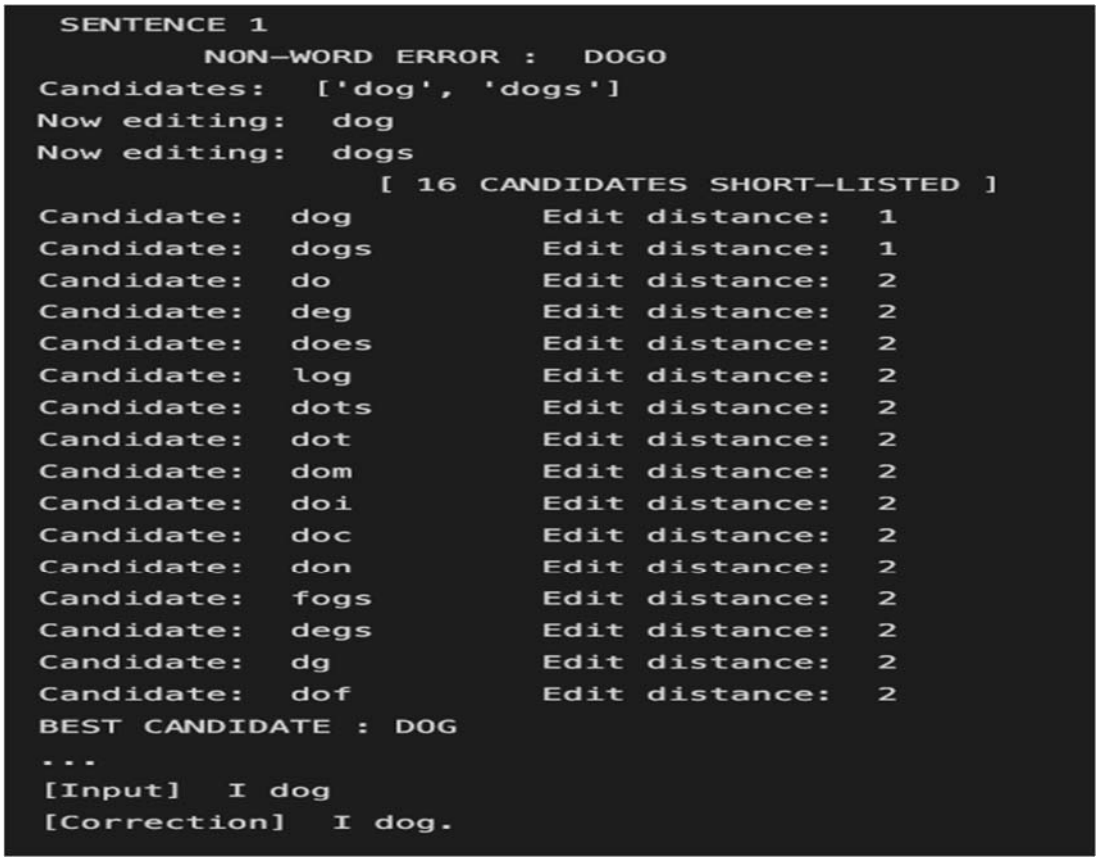


Figure A.11: Output of the model combination when error is detected and returned

A.4 Integration with Frontend

The following code snippets are demonstrating the commands for frontend and backend algorithms integration. User-defined functions are created to link the back-end algorithms with the widget of GUI windows, allowing users to interact and input to the spelling checker through GUI window created.

```
wrongWordAndCandidate = dict()

def checkSpelling():
    for tag in textEditor.tag_names():
        textEditor.tag_remove(tag, "1.0", "end")
    text = textEditor.get("1.0","end-1c")
    if(len(wrongSpellingList.get(0,'end')) > 0):
        wrongSpellingList.delete(0,'end')
    if(len(potentialCandidateList.get(0,'end')) > 0):
        potentialCandidateList.delete(0,'end')
    choice = 'n'
    query = text.replace('-',' - ').replace("'",'')
    CHANGES = dict()
    counter = 1
    print( '\n' )
    for sentence in nltk.tokenize.sent_tokenize(query):
        flag = True
        print( ' SENTENCE' , counter )
        counter = counter + 1
        words = list()
        for w in nltk.tokenize.word_tokenize(sentence):
            if re.search( '^[a-zA-Z]+$' , w ):
                words.append(w.lower())
        for x in range(len(words)):
            if not (words[x] in VOCAB or words[x] in APOS_OMITTED_WORDS):
                wrongSpellingList.insert(x,words[x])
                wrongWord = words[x]
                wrongWordAndCandidate[wrongWord] = []
                try:
                    candidate = bestCandidate( words , x , choice , CHANGES ,
counter-1 )
```

Figure A.12: Linking GUI widget with spelling checking (Part 1)

```
#if candidate == None: #----------------------------------------------
-
                except:
                    label = words[x]
                    idx = wrongSpellingList.get(0, tk.END).index(label)
                    wrongSpellingList.delete(idx)
                    break
                if wrongWord and candidate != None:
                    # start from the beginning (and when we come to the end, stop)
                    idx = '1.0'
                    while 1:
                        # find next occurrence, exit loop if no more
                        idx = textEditor.search(wrongWord, idx, nocase=1,
stopindex="end-1c")
                        if not idx: break
                        # index right after the end of the occurrence
                        lastidx = '%s+%dc' % (idx, len(wrongWord))
                        # tag the whole occurrence (start included, stop excluded)
                        textEditor.tag_add('found', idx, lastidx)
                        # prepare to search for next occurrence
                        idx = lastidx
                    # use a red foreground for all the tagged occurrences
                    textEditor.tag_config('found', foreground='red')
                for i in range (len(candidate)):
                    wrongWordAndCandidate[wrongWord].append(candidate[i][0] + " ED:
' + str(candidate[i][1]))
                flag = False
        if flag:
            print( '\n' , end='' )
    if(len(wrongSpellingList.get(0,'end')) == 0):
        print("here")
        wrongSpellingList.insert(1,"No error")
    print('Corrected text:')
    print(text)
    output, corrected_corpus = correctQuery( text , CHANGES )
    return corrected_corpus
    print('\n' , end='')
```

Figure A.13: Linking GUI widget with spelling checking (Part 2)

```
def chooseCandidate(evt):
    potentialCandidateList.delete(0,'end')
    selected_wrongSpelling = wrongSpellingList.get(wrongSpellingList.curselection())
    candidateWord = wrongWordAndCandidate.get(selected_wrongSpelling)
    for x in range(len(candidateWord)):
        potentialCandidateList.insert(x,candidateWord[x])
    for tag in textEditor.tag_names():
        textEditor.tag_remove(tag, "1.0", "end")
    selectedCorpus = wrongSpellingList.get(wrongSpellingList.curselection())
    if selectedCorpus:
        # start from the beginning (and when we come to the end, stop)
        idx = '1.0'
        while 1:
            # find next occurrence, exit loop if no more
            idx = textEditor.search(selectedCorpus, idx, nocase=1, stopindex="end-1c")
            if not idx: break
            # index right after the end of the occurrence
            lastidx = '%s+%dc' % (idx, len(selectedCorpus))
            # tag the whole occurrence (start included, stop excluded)
            textEditor.tag_add('found', idx, lastidx)
            # prepare to search for next occurrence
            idx = lastidx
        # use a red foreground for all the tagged occurrences
        textEditor.tag_config('found', background='yellow')
```

Figure A.14: Linking GUI widget with candidate displaying

```
def correctSpelling():
    selectedCandidate = potentialCandidateList.get(potentialCandidateList.curselection())
    onlyWord = selectedCandidate.split()
    onlyWord = onlyWord[0]
    print(onlyWord)
    text = textEditor.get("1.0","end-1c")
    wrongSpellingWord = wrongSpellingList.get(wrongSpellingList.curselection())
    text = text.replace(wrongSpellingWord,onlyWord)
    print(text)
    textEditor.delete("1.0","end-1c")
    textEditor.insert("1.0", text)
    potentialCandidateList.delete(0,'end')
    checkSpelling()

def highLightText(evt):
    for tag in textEditor.tag_names():
        textEditor.tag_remove(tag, "1.0", "end")
    selectedCorpus = corpusList.get(corpusList.curselection())
    if selectedCorpus:
        # start from the beginning (and when we come to the end, stop)
        idx = '1.0'
        while 1:
            # find next occurrence, exit loop if no more
            idx = textEditor.search(selectedCorpus, idx, nocase=1, stopindex="end-1c")
            if not idx: break
            # index right after the end of the occurrence
            lastidx = '%s+%dc' % (idx, len(selectedCorpus))
            # tag the whole occurrence (start included, stop excluded)
            textEditor.tag_add('found', idx, lastidx)
            # prepare to search for next occurrence
            idx = lastidx
        # use a red foreground for all the tagged occurrences
        textEditor.tag_config('found', background='yellow')
```

Figure A.15: Linking GUI widget to correct spelling and highlighting text

```
import difflib
import re

wrongRealWordAndCandidate = dict()
allOriDiffSet = []
allCorDiffSet = []
allPotentialSuggestions = []
```

Figure A.16: Importing libraries for creating user defined functions

```
def highLightRealTextError(evt):
    potentialSuggestionList.delete("1.0","end-1c")
    for item in realWordErrorList.curselection():
        index = item
    for tag in textEditor.tag_names():
        textEditor.tag_remove(tag, "1.0", "end")
    selectedCorpus = allOriDiffSet[index]
    selCorDiffSet = allCorDiffSet[index]
    potentialSuggestionList.insert("1.0", allPotentialSuggestions[index])
    string = textEditor.get("1.0","end-1c")
    for x in range(len(selectedCorpus)):
        realWordError = realWordErrorList.get(realWordErrorList.curselection())
        for match in re.finditer(realWordError, string):
            start = "1." + str(match.start())
            end = "1." + str(match.end())
        if selectedCorpus[x]:
            # start from the beginning (and when we come to the end, stop)
            idx = start
            # find next occurrence, exit loop if no more
            idx = textEditor.search(selectedCorpus[x], idx, nocase=1,
stopindex=end)
            if not idx: break
            # index right after the end of the occurrence
            lastidx = '%s+%dc' % (idx, len(selectedCorpus[x]))
            # tag the whole occurrence (start included, stop excluded)
            textEditor.tag_add('found', idx, lastidx)
            # prepare to search for next occurrence
            idx = lastidx
            # use a red foreground for all the tagged occurrences
            textEditor.tag_config('found', background='yellow')
        for x in range(len(selCorDiffSet)):
            if selCorDiffSet[x]:
                # start from the beginning (and when we come to the end, stop)
                idx = '1.0'
                while 1:
                    # find next occurrence, exit loop if no more
                    idx = potentialSuggestionList.search(selCorDiffSet[x], idx,
nocase=1, stopindex="end-1c")
                    if not idx: break
                    # index right after the end of the occurrence
                    lastidx = '%s+%dc' % (idx, len(selCorDiffSet[x]))
                    # tag the whole occurrence (start included, stop excluded)
                    potentialSuggestionList.tag_add('correct', idx, lastidx)
                    # prepare to search for next occurrence
                    idx = lastidx
                # use a red foreground for all the tagged occurrences
                potentialSuggestionList.tag_config('correct', background='#b8f2d0')
```

Figure A.17: Function to highlight error text

```
def checkError():
    allOriDiffSet.clear()
    allCorDiffSet.clear()
    allPotentialSuggestions.clear()
    realWordErrorList.delete(0,'end')
    potentialSuggestionList.delete("1.0","end-1c")
    sent = textEditor.get("1.0","end-1c")
    words = nltk.tokenize.sent_tokenize(sent)
    for x in range(len(words)):
        corrected_sentences = gf.correct(words[x], max_candidates=1)
        print("[Input] ", words[x])
        for corrected_sentence in corrected_sentences:
            print("[Correction] ", corrected_sentence)
            splitA = set(words[x].split(" "))
            splitB = set(corrected_sentence.split(" "))
            oriDiffSet = splitA.difference(splitB)
            corDiffSet = splitB.difference(splitA)
            oriDiffSet = list(oriDiffSet)
            corDiffSet = list(corDiffSet)
            if(len(oriDiffSet) != 0):
                print(corrected_sentence)
                realWordErrorList.insert(x,words[x])
                allPotentialSuggestions.append(corrected_sentence)
                allOriDiffSet.append(oriDiffSet)
                allCorDiffSet.append(corDiffSet)
            print("\n")
    print(len(realWordErrorList.get(0,'end')))
    if(len(realWordErrorList.get(0,'end')) == 0):
       realWordErrorList.insert(1,"No error")
```

Figure A.18: Function to check error

```
def correctError():
    selectedSuggestion = potentialSuggestionList.get("1.0","end-1c")
    text = textEditor.get("1.0","end-1c")
    realWordError = realWordErrorList.get(realWordErrorList.curselection())
    text = text.replace(realWordError,selectedSuggestion)
    textEditor.delete("1.0","end-1c")
    textEditor.insert("1.0", text)
    realWordErrorList.delete(realWordErrorList.curselection())
    potentialSuggestionList.delete("1.0","end-1c")
    checkError()
```

Figure A.19: Function to correct error

```
#import the tkinter,csv,spellchecker module
import tkinter as tk
import csv
import numpy as np
from  spellchecker import SpellChecker
```

Figure A.20: Importing libraries for GUI design

```
# create a simple screen and save to window variable
window = tk.Tk()

# give the GUI a name
window.title('Spelling Error')

# give a height and legnth
window.geometry('1100x500')

##column 1
#adding text editor
textEditor = tk.Text(window, width = 40)
textEditor.place(x = 30, y = 50)
#adding check spelling button
checkSpellingButton = tk.Button(window, text ="Check Spelling")
checkSpellingButton.config(command = checkSpelling)
checkSpellingButton.place(x = 30, y = 20)

checkErrorButton = tk.Button(window, text ="Check Real word error")
checkErrorButton.config(command = checkError)
checkErrorButton.place(x = 150, y = 20)
##-----------------------------##
```

Figure A.22: GUI widget and integrating with earlier predefined functions (Part 1)

```
##column 2

# Wrong spelling label
wrongSpellingLabel = tk.Label(window, text = "Wrong spelling word").place(x = 365, y = 20)
# Wrong Spelling list
wrongSpellingList = tk.Listbox(window, height = 15, width = 18, exportselection=False)

wrongSpellingList.bind('<<ListboxSelect>>', chooseCandidate)

##to be intgrated
wrongSpellingList.place(x=365, y=50)

# Potential candidate label
searchLabel = tk.Label(window, text = "Potential candidate").place(x = 365, y = 310)
# Dictionary list
potentialCandidateList = tk.Listbox(window, height = 5, width = 18)
potentialCandidateList.place(x=365, y=340)
correctSpellingButton = tk.Button(window, text ="Correct Spelling")
correctSpellingButton.config(command = correctSpelling)
correctSpellingButton.place(x = 365, y = 430)
```

Figure A.23: GUI widget and integrating with earlier predefined functions (Part 2)

```
##column 3

# Wrong spelling label
realWordErrorLabel = tk.Label(window, text = "Real word error").place(x = 540, y = 20)
# Wrong Spelling list
realWordErrorList = tk.Listbox(window, height = 15, width = 27, exportselection=False)
realWordErrorList.bind('<<ListboxSelect>>', highLightRealTextError)

##to be intgrated
realWordErrorList.place(x=540, y=50)

# Potential candidate label
suggestionLabel = tk.Label(window, text = "Suggestion").place(x = 540, y = 310)
# Dictionary list
potentialSuggestionList = tk.Text(window, height = 5, width = 35)
potentialSuggestionList.place(x=540, y=340)
correctErrorButton = tk.Button(window, text ="Correct Error")
correctErrorButton.config(command = correctError)
correctErrorButton.place(x = 540, y = 430)
```

Figure A.24: GUI widget and integrating with earlier predefined functions (Part 3)

```
##column 4
dest_file = "/Users/jiaqiser/Downloads/Code 3/Pure_Corpus.csv"
data_array = []
# Wrong spelling word label
spell = SpellChecker()
dictionaryLabel = tk.Label(window, text = "Corpus list").place(x = 800, y = 20)
#adding text editor
corpusList = tk.Listbox(window, height = 15, width = 30)
with open(dest_file,'r') as dest_f:
    data_iter = csv.reader(dest_f,
                           delimiter = ',')

    data_array = [data for data in data_iter]

data_array = data_array[0]
for i in range(len(data_array)):
  corpusList.insert(i,data_array[i])
corpusList.place(x = 800, y = 50)
corpusList.bind('<<ListboxSelect>>', highLightText)
```

Figure A.25: GUI widget and integrating with earlier predefined functions (Part 4)

```
  # Search label
  #corpuseSearchText = tk.Entry(window, height = 1, width = 20)
  searchLabel = tk.Label(window, text = "Search").place(x = 800, y = 310)
  #adding text editor
  def corpusSearching(text):
      if(text == ""):
          for i in range(len(data_array)):
              corpusList.insert(i,data_array[i])
      else:
          potentialCandidate = []
          correctWord = spell.correction(text)
          candidateWord = list(spell.candidates(text))
          #potentialCandidate.append(correctWord)
          for word in candidateWord:
              potentialCandidate.append(word)
          filterWords = []
          corpusList.delete(0, 'end')
          for word in potentialCandidate:
              if word in data_array:
                  filterWords.append(word)
          for i in range(len(filterWords)):
              corpusList.insert(i,filterWords[i])

  corpuseSearchText = tk.Entry(window, validate="focusout", validatecommand=corpusSearching)
  corpuseSearchText.bind("<KeyRelease>",lambda x:corpusSearching(corpuseSearchText.get()))
  corpuseSearchText.place(x = 800, y = 340)
```

Figure A.26: GUI widget and integrating with earlier predefined functions (Part 5)

APPENDIX B

MODEL EVALUATION

B.1 Evaluation Text

Below are the sentences used to evaluate the model performance. Highlighted in red are the **Non-Real Word** errors and highlighted in blue are the **Real Word** errors.

Table B.1: Evaluation Text

		Errors	
No.	**Sentence**	**Non-Real Words**	**Real Words**
1	Ths carpet is big enough too cover the whole floor.	1	1
2	She wasn't as productive as she was hope because she was not aleet	1	1
3	He took a big breath and flew out the candles.	0	1
4	She liked the briks used to bill the house opposite her's	1	1
5	He actually thought with undr capitalism painters would be able to find work that paid wages.	1	1
6	Child actors do they best, but they're generally vely bad at being convincing.	1	1
7	My mom when on vacation and left my dad in charge of the hiuse.	1	1
8	Her oven had a habit of seting off the fire alarm four no reason.	1	1
9	The flowers smelled beautiful and made the rom look happy.	1	0
10	You don’t have to be into the wilderness two enjoy camping.	0	1
11	Just gone ahead and prss that button.	1	1
12	In the near future, we may have a big earthquake in Japan.	0	0
13	We hsve got much bigger problems to seal with right now.	1	1
14	She was two short to see over the fence.	0	1
15	He ran out off money, so he had to stp playing poker.	1	1
16	The urgent care center was flooded with patience after the news of a new deadly virs was made public	1	1
17	Today I herd something new and unmemorable.	1	0
18	I usually use the shortened versipn of mine name because the full version is extremely long.	1	1

19	The movie we watched tomorrow was awesome.	0	1
20	They acted very child when they shoukd not have.	1	1
21	He is abset from yesterday class	1	1
22	The cafe is emty aside from and old man reading a book	1	1
23	The moms and dads all sat aroind drinking coffee and eat donuts.	1	1
24	Today is Sunday, which means tomrrow is Monday and yesterday was Saturday.	1	0
25	I made a list of people I wanted too call to my party.	0	1
26	You may take eithr the big box or the small box.	1	1
27	I is so mad that I yelled at him at the top of my lugs.	1	1
28	She had many oppotunities to give up, but chose not to.	1	0
29	My dad told me these I was his favorite persn in the whole wide world.	1	1
30	She hadn't cleaned her deskop in several year.	1	1
31	His fingers were hurting after plying the guitar for so long.	1	0
32	It's not vey likely that he when out for lunch.	1	1
33	Suddenly from under the stove thre came a squeak.	1	0
34	I didn't one to take a side in the argment.	1	1
35	You must no the sound of each letter in the English alphbet.	1	1
36	I am a hundred percent certain that eat is ging to rain later today.	1	1
37	She got stressed out whenever she saw a notification on her email.	0	0
38	They will show you the way home wen you're lost.	1	0
39	You will be surprised buy the beautiful thngs given.	1	1
40	I found a coin ate the playground after school today.	0	1
41	The door slammed dwn on my hand, and I scream.	1	1
42	She wanted be bold but ws too afraid to start.	1	1
43	Men attacked him and steal his mney.	1	1
44	He slowly begn to realize the error off his ways.	1	1
45	I am going to retun this sweater because too big.	1	1

46	I think you should stop charging so mch money.	1	0
47	There have been much big chages made around here.	1	1
48	The dog were so tired he fell asleep on the wy home	1	1
49	She is convinced her faher was a chef	1	0
50	My glass of waer broke when it fell the table.	1	1
51	I had a bad feeling the mrning after drinking too much.	1	0
52	Crime is certanly on the increase on many of our cities.	1	1
53	There is a big bttle of medicine in the cabinet.	1	0
54	To be honest you, I am older than you think.	0	1
55	I'll talk to him frst thing on the morning.	1	1
56	She lived in constnt fear of being fire.	1	1
57	You may take either the big box or the smll one.	1	0
58	Thee was a big fire near my house last night.	1	0
59	That box is too big to put in the trnk of car.	1	1
60	He was very scaed when his saw this big snake.	1	1

B.2 Evaluation Result Summary

Below is the evaluation set and its corresponding sentences used for this study. Based on the performance of the model, the prediction result is documented in the table below.

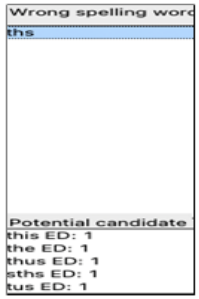
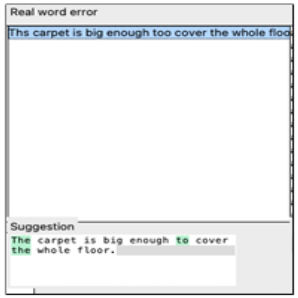
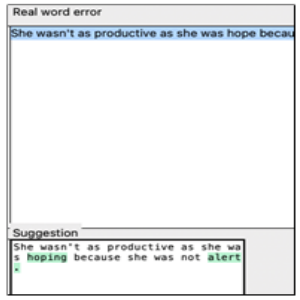
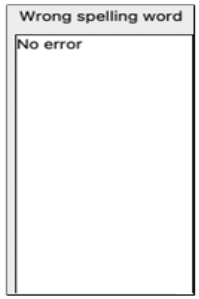
Table B.2: Evaluation Sets

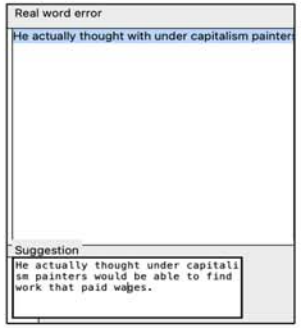
		No. of Non-Real Word Error (NRWE)		No of. Real Word Error (RWR)	
Set	Sentence No.	Actual	Predicted	Actual	Predicted
1	1	1	1	1	1
	2	1	1	1	1
	3	0	0	1	0
	4	1	1	1	0
	5	1	1	1	1
	6	1	1	1	1
	7	1	0	1	1
	8	1	1	1	1
	9	1	0	0	1
	10	0	0	1	1
2	11	1	1	1	1
	12	0	0	0	0
	13	1	1	1	1
	14	0	0	1	0
	15	1	1	1	1
	16	1	0	1	1
	17	0	0	1	1
	18	1	1	1	0
	19	0	0	1	0
	20	1	1	1	1
3	21	1	1	1	1
	22	1	1	1	1
	23	1	0	1	0
	24	1	0	0	0
	25	0	0	1	1
	26	1	1	1	1
	27	1	0	1	1
	28	1	1	0	0
	29	1	0	1	1
	30	1	1	1	1
4	31	1	1	0	0
	32	1	1	1	1

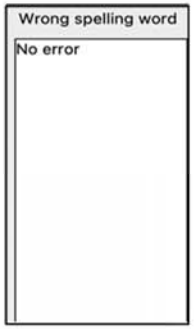
	33	1	0	0	0
	34	1	1	1	1
	35	1	0	1	1
	36	1	1	1	1
	37	0	0	0	0
	38	1	0	0	0
	39	1	1	1	1
	40	0	1	1	1
5	41	1	0	1	1
	42	1	1	1	0
	43	1	1	0	0
	44	1	1	1	1
	45	1	1	1	0
	46	1	1	0	0
	47	1	1	1	1
	48	1	1	1	1
	49	1	1	0	0
	50	1	1	1	0
6	51	1	1	0	1
	52	1	1	1	0
	53	1	1	0	0
	54	0	0	1	1
	55	1	1	1	1
	56	1	1	1	1
	57	1	1	0	0
	58	1	1	0	0
	59	1	1	1	1
	60	1	1	1	1

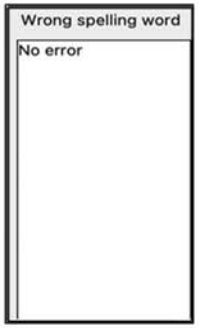
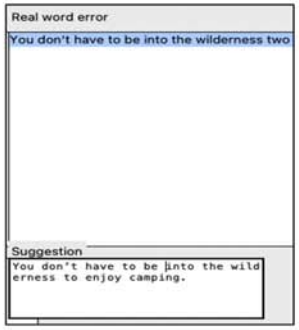
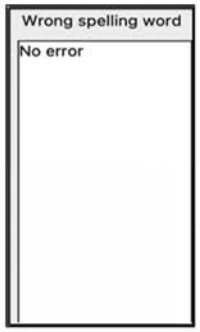
B.3 Screenshots of Model Testing

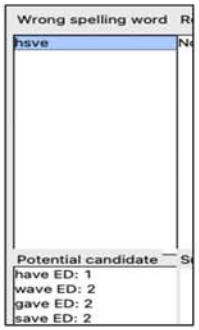
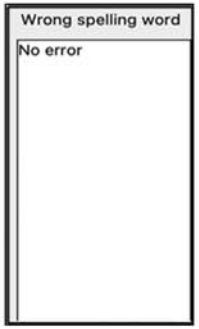
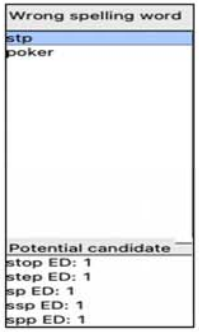
This section documents the model testing using the evaluation text from the previous section. Candidates suggested from the non-word and real-word model can be observed in the screenshot provided. In addition, a column indicating the capacity of the model to detect the error is provided.

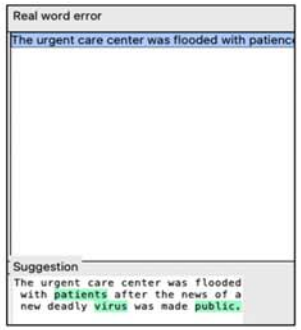
Sentence	Sentence No.	Misspelled Word	Possible Candidates	Recommendation	Detected and Corrected (Y/N)		Sentence (After NRW correction)	Possible Candidates	Recommendation	Grammar Corrected (Y/N)
Ths carpet is big enough too cover the whole floor.	1	Ths		This	Y		Ths carpet is big enough too cover the whole floor.		The carpet is big enough to cover the whole floor.	Y
She wasn't as productive as she was hope because she was not aleet	2	aleet		alert	Y		She wasn't as productive as she was hope because she was not alert		She wasn't as productive as she was hoping because she was not alert.	Y
He took a big breath and flew out the candles.	3	none		none	Y		He took a big breath and flew out the candles.		None	N

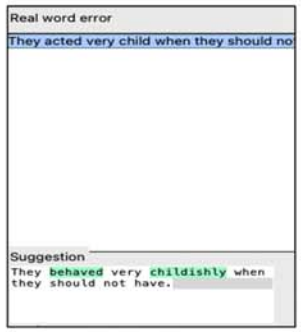
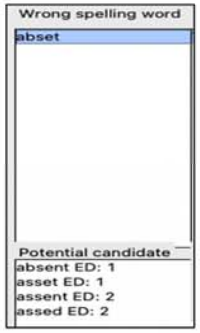
She liked the briks used to bill the house opposite her's	4	briks		bricks	Y		She liked the bricks used to bill the house opposite her's		She liked the bricks used to bill the house opposite her	N
He actually thought with undr capitalism painters would be able to find work that paid wages.	5	undr		under	Y		He actually thought with under capitalism painters would be able to find work that paid wages.		He actually thought under capitalism painters would be able to find work that paid wages.	Y
Child actors do they best, but they're generally vely bad at being convincing.	6	vely		very	Y		Child actors do they best, but they're generally very bad at being convincing.		Child actors do their best, but they're generally very bad at being convincing.	Y

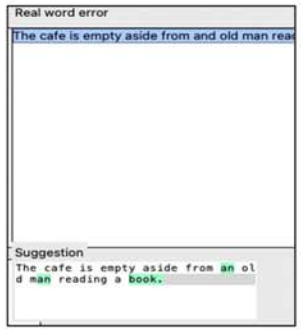
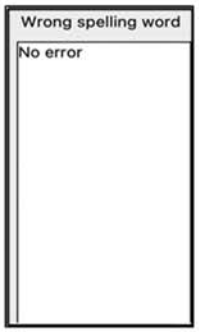
My mom when on vacation and left my dad in charge of the hiuse.	7	hiuse		Wrong detection	N	My mom when on vacation and left my dad in charge of the hiuse.		My mom was on vacation and left my dad in charge of the house	Y
Her oven had a habit of seting off the fire alarm four no reason.	8	seting		setting	Y	Her oven had a habit of setting off the fire alarm four no reason.		Her oven had a habit of setting off the fire alarm for no reason.	Y
The flowers smelled beautiful and made the rom look happy.	9	rom		No detection	N	The flowers smelled beautiful and made the rom look happy.		None	Y

You don't have to be into the wilderness two enjoy camping.	10	none		none	Y	You don't have to be into the wilderness two enjoy camping.		You don't have to be into the wilderness to enjoy camping.	Y
Just gone ahead and prss that button.	11	prss		press	Y	Just gone ahead and press that button.		Just go ahead and press that button.	Y
In the near future, we may have a big earthquake in Japan.	12	none		none	Y	In the near future, we may have a big earthquake in Japan.		None	Y

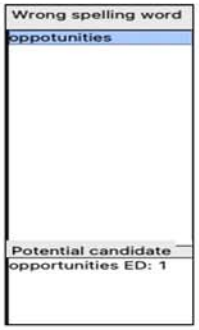
We hsve got much bigger problems to seal with right now.	13	hsve		have	Y	We have got much bigger problems to seal with right now.		We have got much bigger problems to deal with right now.	Y
She was two short to see over the fence.	14	None		None	Y	She was two short to see over the fence.		None	N
He ran out off money, so he had to stp playing poker.	15	stp		stop	Y	He ran out off money, so he had to stop playing poker.		He ran out of money, so he had to stop playing poker.	Y

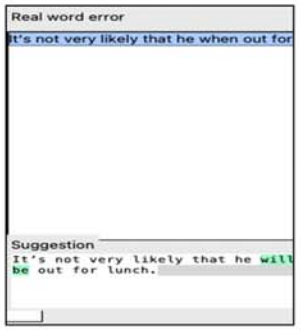
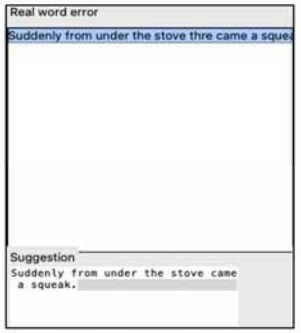
The urgent care center was flooded with patience after the news of a new deadly virs was made public	16	virs		No detection	N		The urgent care center was flooded with patience after the news of a new deadly virs was made public		The urgent care center was flooded with patients after the news of a new deadly virus was made public.	Y
Today I herd something new and unmemorable.	17	herd		heard	Y		Today I heard something new and unmemorable.		None	Y
I usually use the shortened versipn of mine name because the full version is extremely long.	18	versipn		version	Y		I usually use the shortened version of mine name because the full version is extremely long.		None	N

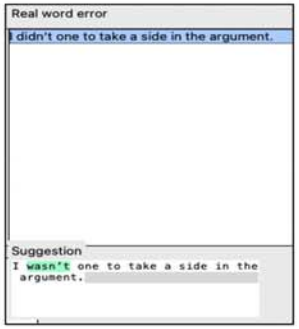
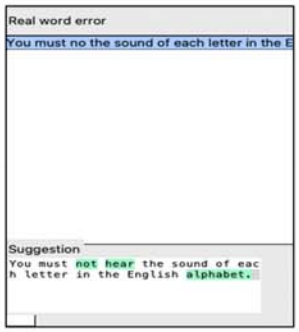
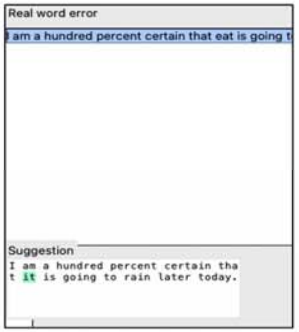
The movie we watched tomorrow was awesome.	19	none		none	Y	The movie we watched tomorrow was awesome.		None	N
They acted very child when they shoukd not have.	20	shoukd		should	Y	They acted very child when they should not have.		They behaved very childishly when they should not have.	Y
He is abset from yesterday class	21	abset		absent	Y	He is absent from yesterday class		He was absent from yesterday's class.	Y

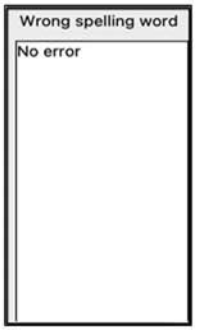
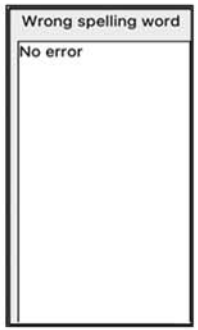
The cafe is emty aside from and old man reading a book	22	emty		empty	Y	The cafe is empty aside from and old man reading a book		The cafe is empty aside from an old man reading a book.	Y
The moms and dads all sat aroind drinking coffee and eat donuts.	23	aroind		No detection	N	The moms and dads all sat aroind drinking coffee and eat donuts.		The moms and dads all sat around drinking coffee and eat donuts.	N
Today is Sunday, which means tomrrow is Monday and yesterday was Saturday.	24	tomrrow		Wrong detection	N	Today is Sunday, which means tomrrow is Monday and yesterday was Saturday.		Today is Sunday, which means tomorrow is Monday and yesterday was Saturday.	Y

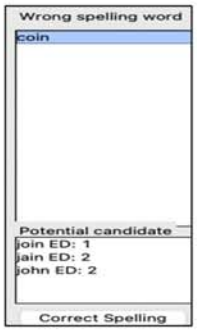
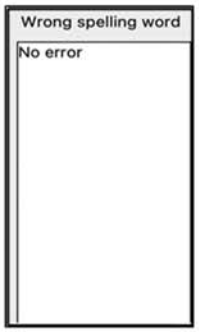
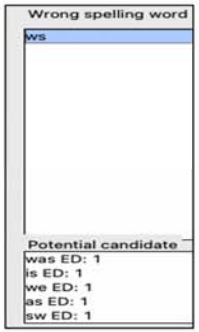
I made a list of people I wanted too call to my party.	25	none		none	Y		I made a list of people I wanted too call to my party.		I made a list of people I wanted to call to my party.	Y
You may take eithr the big box or the small box.	26	eithr		either	Y		You may take either the big box or the small box.		None	Y
I is so mad that I yelled at him at the top of my lugs.	27	lugs		Wrong Candidates	N		I is so mad that I yelled at him at the top of my lugs.		I was so mad that I yelled at him at the top of my lungs.	Y

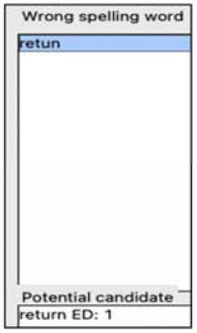
She had many oppotunities to give up, but chose not to.	28	opportunities		opportunities	Y	She had many opportunities to give up, but chose not to.		None	Y
My dad told me these I was his favorite persn in the whole wide world.	29	persn		Wrong detection	N	My dad told me these I was his favorite persn in the whole wide world.		My dad told me I was his favorite person in the whole wide world.	Y
She hadn't cleaned her deskop in several year.	30	deskop		desktop	Y	She hadn't cleaned her desktop in several year.		She hadn't cleaned her desktop in several years.	Y

His fingers were hurting after plying the guitar for so long.	31	plying		playing	Y	His fingers were hurting after playing the guitar for so long.		None	Y
It's not vey likely that he when out for lunch.	32	vey		very	Y	It's not very likely that he when out for lunch.		It's not very likely that he will be out for lunch.	Y
Suddenly from under the stove thre came a squeak.	33	thre		None	N	Suddenly from under the stove thre came a squeak.		Suddenly from under the stove came a squeak.	Y

I didn't one to take a sie in the argument.	34	sie		side	Y	I didn't one to take a side in the argument.		I wasn't one to take a side in the argument.	Y
You must no the sound of each letter in the English alphbet.	35	alphbet		Wrong detection	N	You must no the sound of each letter in the English alphbet.		You must not hear the sound of each letter in the English alphabet.	Y
I am a hundred percent certain that eat is ging to rain later today.	36	ging		going	Y	I am a hundred percent certain that eat is going to rain later today.		I am a hundred percent certain that it is going to rain later today.	Y

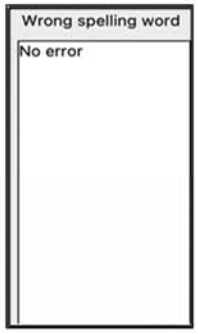
She got stressed out whenever she saw a notification on her email.	37	none		none	Y	She got stressed out whenever she saw a notification on her email.		She got stressed out whenever she saw a notification on her email.	Y
They will show you the way home wen you're lost.	38	wen		No detection	N	They will show you the way home wen you're lost.		They will show you the way home when you're lost.	Y
You will be surrised buy the beautiful things given.	39	surrised		surprised	Y	You will be surprised buy the beautiful things given.		You will be surprised by the beautiful things given.	Y

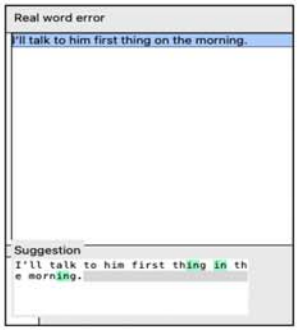
I found a coin ate the playground after school today.	40	none		Wrong detection	N		I found a coin ate the playground after school today.		I found a coin in the playground after school today.	Y
The door slammed dwn on my hand, and I scream.	41	dwn		none	N		The door slammed dwn on my hand, and I scream.		The door slammed down on my hand, and I screamed.	Y
She wanted be bold but ws too afraid to start.	42	ws		was	Y		She wanted be bold but was too afraid to start.		None	N

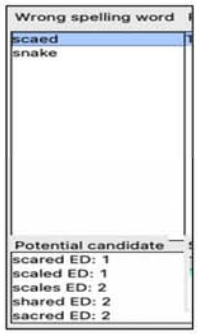
Men attacked him and steal his mney.	43	mney		money	Y	Men attacked him and steal his money.		Men attacked him and stole his money.	Y
He slowly begn to realize the error off his ways.	44	begn		begin	Y	He slowly begin to realize the error off his ways.		He slowly begins to realize the error of his ways.	Y
I am going to retun this sweater because too big.	45	retun		return	Y	I am going to return this sweater because too big.		None	N

I think you should stop charging so mch money.	46	mch		much	Y	I think you should stop charging so much money.		None	Y
There have been much big chages made around here.	47	chages		changes	Y	There have been much big changes made around here.		There have been many big changes made around here.	Y
The dog were so tired he fell asleep on the wy home	48	wy		way	Y	The dog were so tired he fell asleep on the way home		The dog was so tired he fell asleep on the way home.	Y

She is convinced her faher was a chef	49	faher		father	Y		She is convinced her father was a chef		None	Y
My glass of waer broke when it fell the table.	50	waer		water	Y		My glass of water broke when it fell table.		None	N
I had a bad feeling the mrning after drinking too much.	51	mrning		morning	Y		I had a bad feeling the morning after drinking too much.		None	Y

Crime is certanly on the increase on many of our cities.	52	certanly		certainly	Y		Crime is certainly on the increase on many of our cities.		None	N
There is a big bttle of medicine in the cabinet.	53	bttle		bottle	Y		There is a big bottle of medicine in the cabinet.		None	Y
To be honest you, I am older than you think.	54	None		None	Y		To be honest you, I am older than you think.		To be honest, I am older than you think.	Y

I'll talk to him frst thing on the morning.	55	frst		first	Y	I'll talk to him first thing on the morning.		I'll talk to him first thing in the morning.	Y
She lived in constnt fear of being fire.	56	constnt		constant	Y	She lived in constant fear of being fire.		She lived in constant fear of being burned.	Y
You may take either the big box or the smll one.	57	smll		small	Y	You may take either the big box or the small one.		None	Y

Thee was a big fire near my house last night.	58	thee		there	Y	There was a big fire near my house last night.		None	Y
That box is too big to put in the trnk of car.	59	trnk		None	Y	That box is too big to put in the trnk of car.		That box is too big to put in the trunk of a car.	Y
He was very scaed when his saw this big snake.	60	scaed		scared	Y	He was very scared when his saw this big snake.		He was very scared when he saw this big snake.	Y