Cleaning and Mapping Interface

* *Aakansha Dhawan*

Abstract

With the growth in data day by day, it is getting harder to collect data in a clean format. Data collected for different purposes from sources contains more noise than the data required for the purpose. This research focuses on the cleaning of data and making it meaningful to use by building an annotator which reduces manual labor and makes annotation more efficient. I built a Cleaning and Mapping Interface which detects misspelled word and corrects it, even helps in predicting the next words based on its previous knowledge. It can be used by users with no knowledge of any library or functional programming to annotate their data. It is a user-friendly software that is easy to use. The application is written in Python language with a simple user interface implemented using the Tkinter module. The program has the ability to learn new grammar. The algorithm is recursively applied to deal with the same sentences (that are matched before correcting them) in the document, making it less time consuming and efficient. Cleaning and Mapping Interface uses algorithms for recurrent neural networks and Naive Bayes for making predictions. It provides the user with correct spelling and the next word for a respective word, so annotation can be done faster.

Keywords: Long Short Term memory, Recurrent Neural Network, Naive Bayes

I. INTRODUCTION

Cleaning and Mapping Interface (CMI) is a large body of work based on manual work for grammar specification and dataset annotation. An automatic annotator is useful in removing noise from the data and cleansing the same. Algorithms like recurrent neural network [1] help in predictive analysis [2] by understanding and analyzing the data and make predictions as they are trained, we use it to make word level prediction by extending the character level prediction [2]. Furthermore, Naive Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (Naive) independence assumptions between the features [3] which can be put to several uses like spelling correction on the basis of dictionary (containing the words with correct spelling prepared manually) provided to it. These basic algorithms are put into different models to make predictions for CMI using Keras library [4] which provides better performance and less complexity to make models for these algorithms. They are bended in a graphical user interface to provide better usage, and appearance using Tkinter [5], a library which provides support for making a user interface in Python.

II. METHODOLOGY

The CMI is solely interfaced for annotating the bibliographic data which can also be extended to work with a different kind of category by accumulating and providing proper data to different modules. Intended design of CMI can be broken down into three sections: next word prediction, spelling correction, and user interface, the details of which are explained next.

A. Next Word Prediction

To make it easy for the annotator, CMI predicts the next word for current as well as the previous context. For this, LSTM (RNN) was used which is capable of having a memory to keep account of the working text so that the next word predicted can form an appropriate sentence.

To train accurate and efficient deep RNNs, it is vital to have significant amounts of clean and correctly annotated data. The data for word prediction is a text file containing names of journals and conferences with nearly 6000 lines and length of the corpus is 3,39,491. This data was scraped from the site “sjrmagojr.com”.

The model is built using Keras Library. It is a sequential model consisting of a single LSTM layer with 128 neurons which accepts input of shape (40 — the length of a sequence, 95 — the number of unique

Manuscript received January 5, 2019; revised January 20, 2019; accepted January 25, 2019. Date of publication March 6, 2019.
* A. Dhawan is a student at Chhatrapati Shahu Ji Maharaj University, Kalyanpur, Kanpur, Uttar Pradesh, India-208024.
(email : aakansha2809@gmail.com)
DOI: 10.17010/ijcs/2019/v4/i2/144272

characters in our dataset). The output layer is connected after that to form a fully connected model with 95 neurons in it. Softmax is used as the activation function for the model. Our model is trained for 20 epochs and uses 5% of the data for validation. Categorical cross entropy is considered for the loss function. The loss and the accuracy of the model can be depicted in Fig. 1 and Fig. 2 respectively.

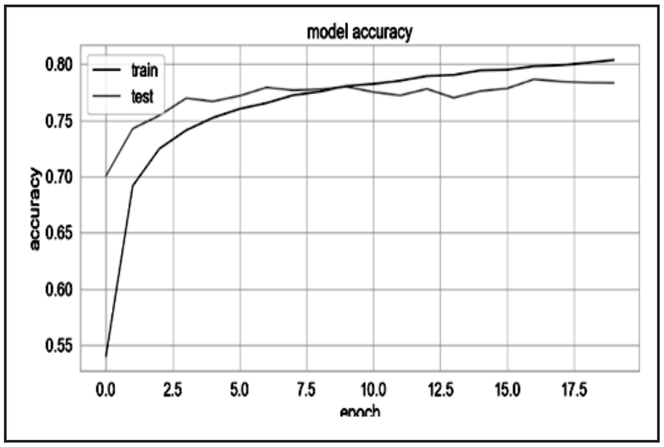


Fig. 1. Accuracy of LSTM Model

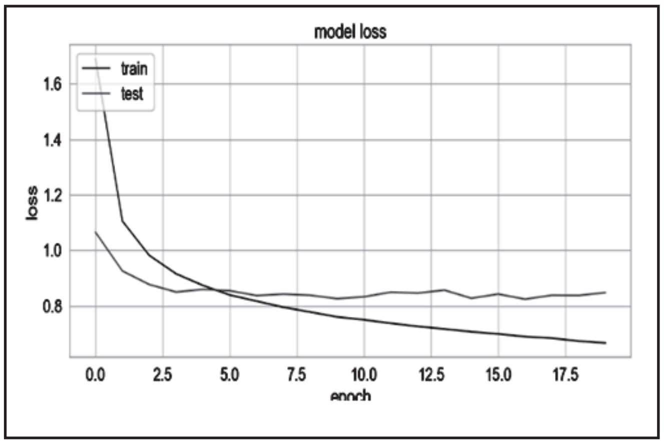


Fig. 2. Loss of LSTM Model

B. Spelling Correction

While cleaning textual data, the most challenging part is to figure out a word that is misspelled as it may have many replacements. Taking this problem into consideration, CMI has the ability to solve and apply Bayes Theorem approach to write a spell corrector and a better algorithm was built.

The dictionary made for the CMI contains nearly a thousand words which were put together from the data fed to the previous model. We are trying to find the correction c, out of all possible candidate corrections, that maximizes the probability that c is the intended correction, given the original word w:

$\text{argmax}c \in candidates\ P(c|w)$

Using Bayes' Theorem, it is seen that this is equivalent to:

$\text{argmax}c \in candidates\ P(c)\ P(w|c) / P(w)$

Since P (w) is the same for every possible candidate c, we can factor it out, giving:

$\text{argmax}c \in candidates\ P(c)\ P(w|c)$.

The model that says all known words of edit distance 1 are infinitely more probable than known words of edit distance 2, and infinitely less probable than a known word of edit distance 0. So, the model provides the first non-empty list (three suited words) of candidates in order of priority:

1. The original word, if it is known; otherwise
2. The list of known words at edit distance one away, if there are any; otherwise
3. The list of known words at edit distance two away, if there are any; otherwise
4. The original word, even though it is not known.

C. User Interface

The basic user interface was built using the Tkinter library. It consists of a single window. First, there is a drop-down menu 'File' which has options Open and Exit. The former is to open a new text file which has to be annotated, and the latter is to exit the user interface.

Moreover, there is a text field which loads the

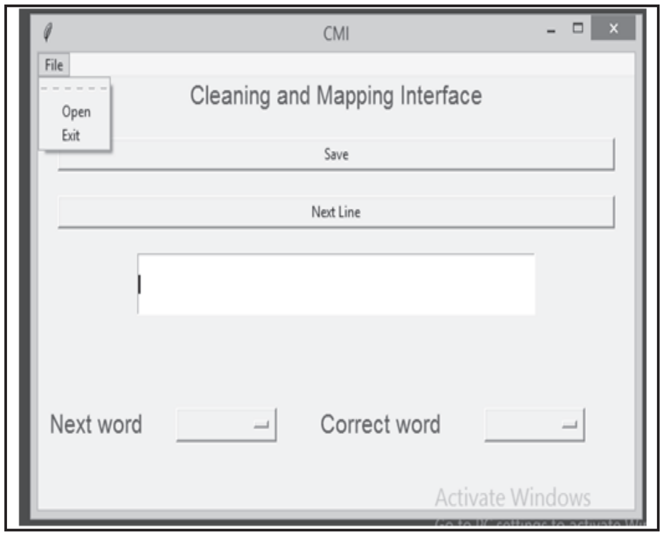
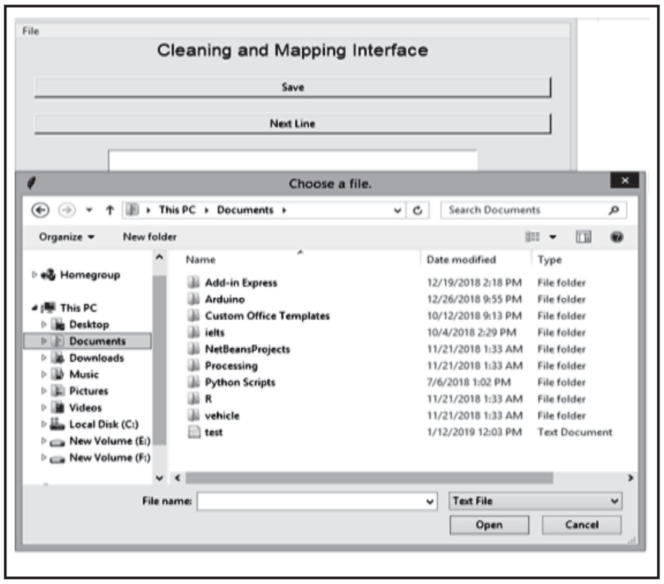


Fig. 3. Graphical User Interface With File Menu



Name	Date modified	Type
Add-in Express	12/19/2018 2:18 PM	File folder
Arduino	12/26/2018 9:55 PM	File folder
Custom Office Templates	10/12/2018 9:13 PM	File folder
ielts	10/4/2018 2:29 PM	File folder
NetBeansProjects	11/21/2018 1:33 AM	File folder
Processing	11/21/2018 1:33 AM	File folder
Python Scripts	7/6/2018 1:02 PM	File folder
R	11/21/2018 1:33 AM	File folder
vehicle	11/21/2018 1:33 AM	File folder
test	1/12/2019 12:03 PM	Text Document

Fig. 4. Interface With Open File Dialog

document line by line. User interface also has two buttons labelled 'save' and 'next' which not only saves the annotated line in the same document on the same line, but also checks for every line that has same mistakes as the current line and corrects them, which in turn, saves time and increases speed. The 'next' button loads the next line from the text file.

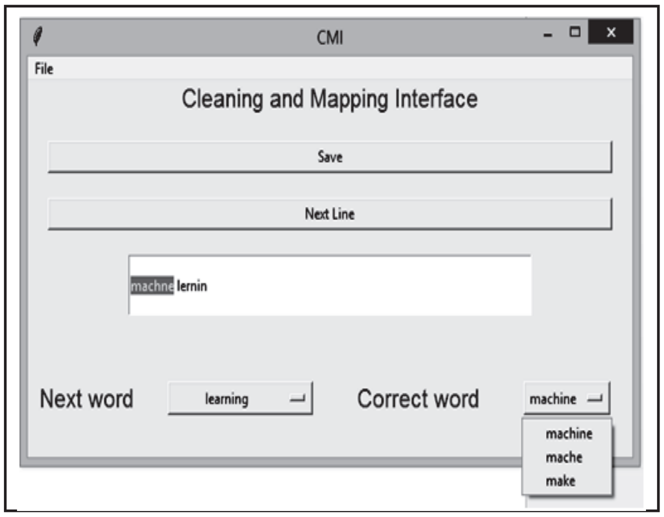


Fig. 5. Interface With Possible Correct Spellings of Selected Word

In addition, there are two other drop-down lists associated with spell checking and the next word prediction which shows the top three choices provided by the algorithms mentioned here.

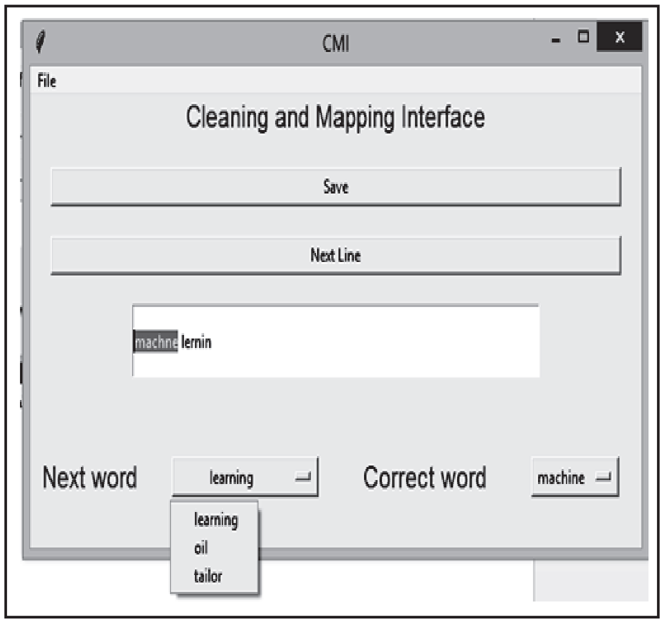


Fig. 6. Interface Showing Possible Next Words of the Selected Word

III. ALGORITHM

The algorithm for Cleaning and Mapping Interface can be stated as:

1. Open the text file that has to be annotated using "Open" in File menu.
2. The first line of the file is read in the text field.
3.Select the word from the text field that has to be corrected.
4.The selected word passes through two different algorithms for spelling correction and next word prediction.
5. From each of the algorithms mentioned in the previous step, top three words are returned which are displayed into the corresponding drop-down list, most accurate being on top.
6. The word is selected from the list and is replaced in the text field.
7. By clicking on "Save" the current line is saved in the original document at the same place.
8. Then the next line is loaded by clicking on the "Next" button.
9. Steps 3 to 8 are repeated until the end of the file.

IV. CONCLUSION

We implemented most of the aims of CMI successfully. The program can successfully predict the words in the text file. Both algorithms are also able to

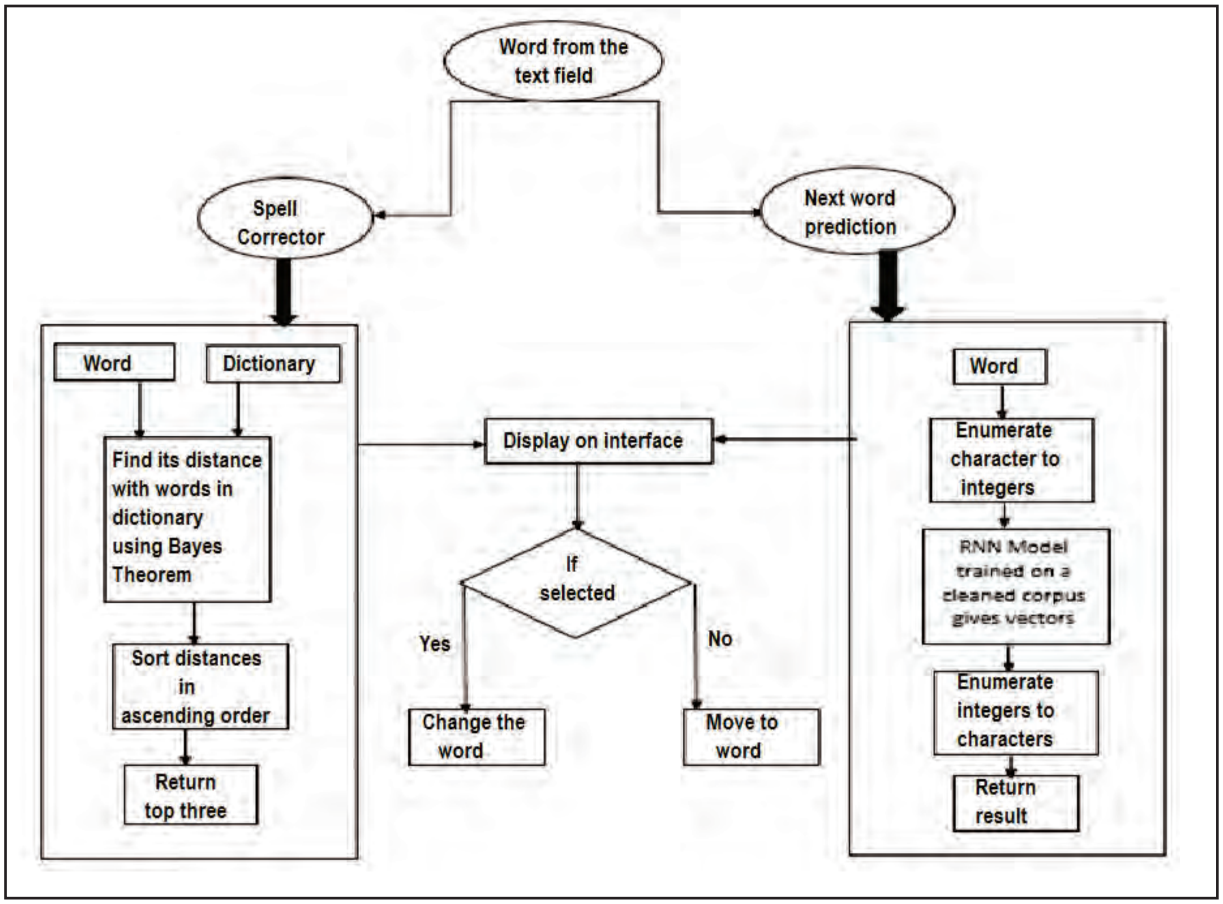

Word from the text field
Spell Corrector
Next word prediction
Word
Dictionary
Find its distance with words in dictionary using Bayes Theorem
Sort distances in ascending order
Return top three
Display on interface
If selected
Yes
No
Change the word
Move to word
Word
Enumerate character to integers
RNN Model trained on a cleaned corpus gives vectors
Enumerate integers to characters
Return result

Fig. 7. Algorithm Behind Every Word in the File

handle new and unseen words, which include understanding data and adding it to the previous. Learning capability of CMI has also been implemented with some success. Our preliminary results are encouraging as they show that our system generates the right statements in 80% of the cases.

Although, to get more precise results, it shows top three results so that the user can pick the best one.

The program successfully runs on Windows 32/64 platforms, Linux, as well as iOS environments. Multiple user access is also supported as a new parser is created for each user, and it is saved in different files.

REFERENCES

[1] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. *9*, no. *8*, pp. 1735-1780, 1997. doi: 10.1162/neco.1997.9.8.1735

[2] M. Low, "Character-level recurrent text prediction," pp. 1-6. [Online]. Available: https://cs224d.stanford.edu/reports/mwlow.pdf

[3] Naive Bayes Classifier. [Online]. Available: en.wikipedia.org/wiki/Naive_Bayes_classifier

[4] Keras: The Python Deep Learning Library. [Online]. Available: https://keras.io

[5] Tkinter - Python interface to Tcl/Tk. [Online]. Available: docs.python.org/2/library/tkinter.html

About the Author

Aakansha Dhawan is an Undergraduate Engineering student of Kanpur University pursuing Bachelors Degree in Computer Science & Engineering. She is a Software Developer and Open Source supporter. She loves to read books and watches movies and TV shows. She is interested in Artificial Intelligence and Computer Vision.

INDIAN JOURNAL OF COMPUTER SCIENCE

Statement about ownership and other particulars about the newspaper "Indian Journal of Computer Science" to be published in the 2nd issue every year after the last day of February.

FORM 1V
(see Rule 18)

1.	Place of Publication	:	NEW DELHI
2.	Periodicity of Publication	:	BI- MONTHLY
3.	4,5 Printer, Publisher and Editor's Name	:	S. GILANI
4.	Nationality	:	INDIAN
5.	Address	:	Y-21,HAUZ KHAS, NEW DELHI - 16
6.	Newspaper and Address of individual	:	ASSOCIATED MANAGEMENT
	Who owns the newspaper and partner of	:	CONSULTANTS PRIVATE LIMITED
	Shareholder holding more than one percent.	:	Y-21, HAUZ KHAS, NEW DELHI-16

I, S.Gilani, hereby declare that the particulars given above are true to the best of my knowledge and belief.

DATED : 1st March, 2019

Sd/-
S. Gilani
Signature of Publisher