GRAMFIX

MINOR PROJECT-I (15B19CI591)

MEMBERS:
HARSHIT KAWATRA 19103153 B5
GAURAV DHINGRA 19103164 B5
PRABHAT RANJAN 19103183 B6

INDEX

Students' Self Declaration for Open Source libraries and other source code usage in Minor Project

We hereby declare the following usage of the open source code and prebuilt libraries in our minor project in **Vth** Semester with the consent of our supervisor. We also measure the similarity percentage of pre written source code and our source code and the same is mentioned below. This measurement is true with best of our knowledge and abilities.

1. List of pre build libraries - happytransformers, tesseract.js,express.js, bootstrap
2. List of pre build features in libraries or in source code- T5 model
3. Percentage of pre written source code and source written by us. 95% source code written by us.

Student ID	Student Name	Student signature
19103153	Harshit Kawatra	Harshit
19103163	Gaurav Dhingra	Gaurav
19103183	Prabhat Ranjan	Prabhat

Declaration by Supervisor (To be filled by Supervisor only)

I, ...(Name of Supervisor) declares that I above submitted project with Titled .... was conducted in my supervision. The project is original and neither the project was copied from External sources not it was submitted earlier in JIIT. I authenticate this project.

(Any Remarks by Supervisor)

Signature (Supervisor)

Problem Statement:

Problem: Correct the grammar of sentence provided by user and display the output

Solution: The sentence provided by the user is sent to the backend of the project using an API,

The backend consists of a machine learning model that runs through the sentence and corrects the grammar based on its training and returns the output to API which transmits the corrected sentence to front-end of the program.

Abstract of the Project:

In this day and age of social media, people are using the incorrect form of grammar and even incorrect English in their messages, so when they have to write a formal letter, mail, message to someone they face issues. People often use some software like Grammarly to ensure that the message they send is absolutely correct.

Our project also has the same motive and we have tried to make an alternative to Grammarly, without the hassle of logging in to a website and having a possible risk of data leak.

Tools and languages used:

- **JavaScript**

  JavaScript is a text-based programming language used both on the client-side and server-side that allows you to make web pages interactive. Where HTML and CSS are languages that give structure and style to web pages, JavaScript gives web pages interactive elements that engage a user.

  We have used different libraries of JS to create the UI and API of our project

- **React JS**
  React JS is a free and open-source front-end JavaScript library for building user interfaces based on UI Components. It has been used in our project to create the frontend of the project. The web-app the user interacts with and provides their grammatically incorrect sentences.
- **Node.js**
  Node.js is an event-driven programming to web servers, enabling development of fast web servers in Java Script and is based on Google's V8 engine.

- **Tesseract.js**

  IT is a pure JavaScript port of the popular Tesseract OCR engine. This library supports more than 100 languages, automatic text orientation and script detection, a simple interface for reading paragraph, word, and character bounding boxes.

  We have used this in our project for obtaining text from images.

- **HTML**

The HyperText Markup Language, or HTML is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets and scripting languages such as JavaScript.

- **CSS**

Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language such as HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript

- **Bootstrap**

Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains CSS- and JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components.

- **Python**

Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. It is used in our project to create the backend of our project i.e., the machine learning model.

- **Google Colab**

Colaboratory, or “Colab” for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education.

We have used Colab for training our ML model.

Machine Learning Model Used:

T5-base model:

The Text-to-Text Transfer Transformer(T5) model is based on transfer learning techniques for the NLP, it aims to explore what works best and how far can we push the tools we already have.

This model generates a revised version of inputted text with the goal of containing a fewer grammatical error. It was trained with Happy Transformer* (Happy Transformer is a package built on top of Hugging Face's transformer library that makes it easy to utilize state-of-the-art NLP models.) using a dataset called **JFLEG****. This model outperforms the human baseline on the General Language Understanding Evaluation (GLUE) benchmark- making it one of the most powerful NLP models in existence.

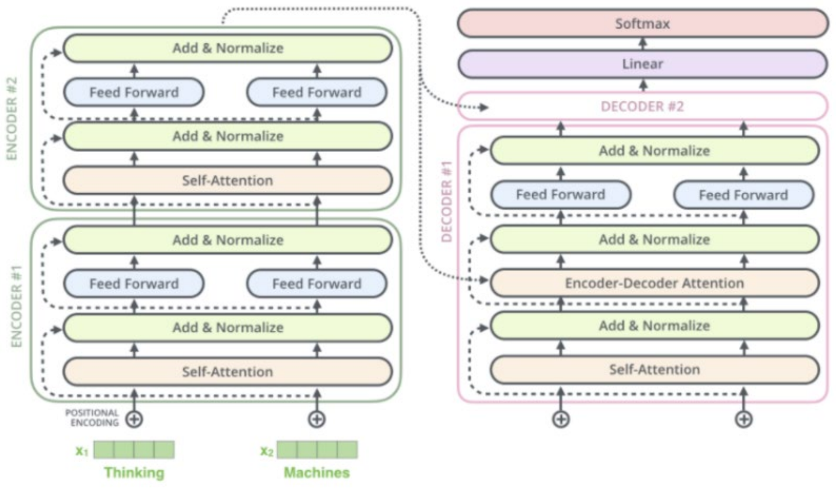


Fig: T5 Model Structure

T5 model was created by Google AI and released as open-source for anyone to download and use. Google had released several sizes for their model, the largest one contained 11 billion parameters and the smallest one contained 60 mil. parameters, we have used the base model which has around 220 mil. parameters.

T5 is a text-to-text model, meaning given text, it generated a standalone piece of text based on import.

The JFLEG database is split into two parts with 50-50 ratio i.e., train and eval. The data is trained on the train portion of dataset to make it fine-tuned for error correction to provide more accurate results. Once the model gets the input of incorrect sentence it runs a beam search (Beam search is an optimization of breadth-first search that reduces its memory requirements) for generation of grammatically correct sentence.

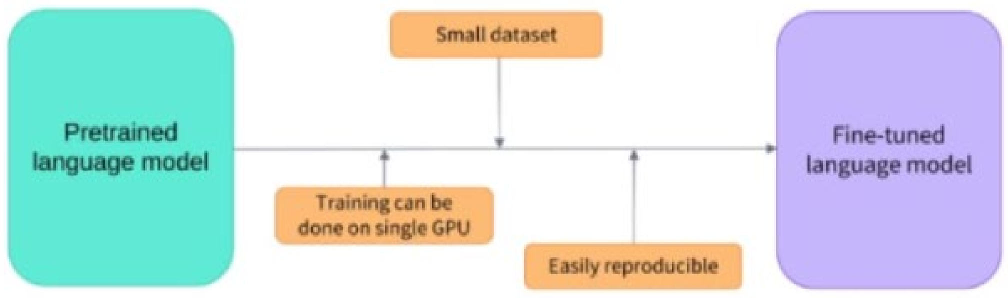

Small dataset
Pretrained language model
Fine-tuned language model
Training can be done on single GPU
Easily reproducible

Fig: Fine Tuning a LM

Detailed Design:

The program is loaded on the chrome browser and the user interface is displayed which is created on the ReactJS platform with the help of some Bootstrap functions.

The program loads the home page on launching the program which contains two text boxes. One gets the input from the user for the text that needs to be corrected. And the other text box provides the corrected text as output.

The user can provide input using two methods, i.e. writing/pasting the text or by uploading a image in the specified portion of the page.

By clicking on the "Fix it" button, the program sends the input text to the API which initializes the model and waits for the model to provide a result, which is then sent back to the Interface to provide a output to the user.

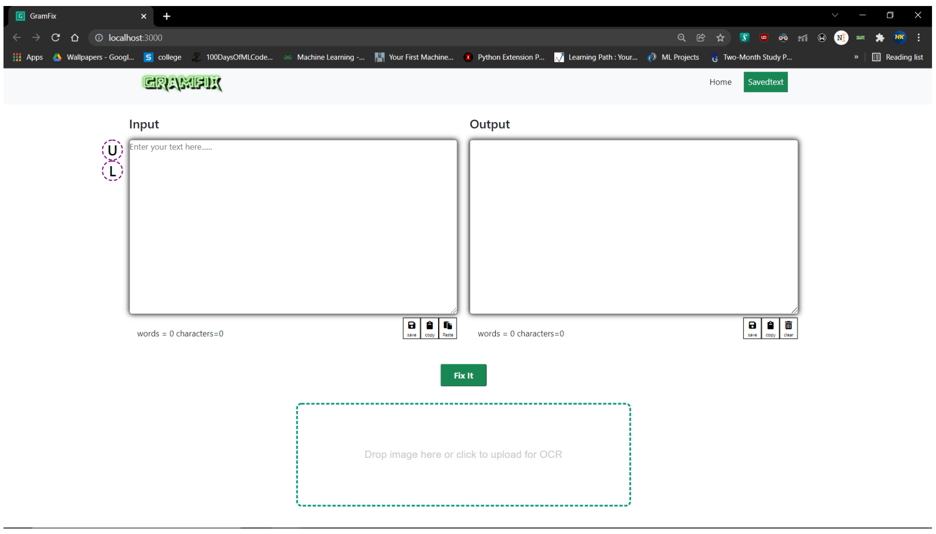
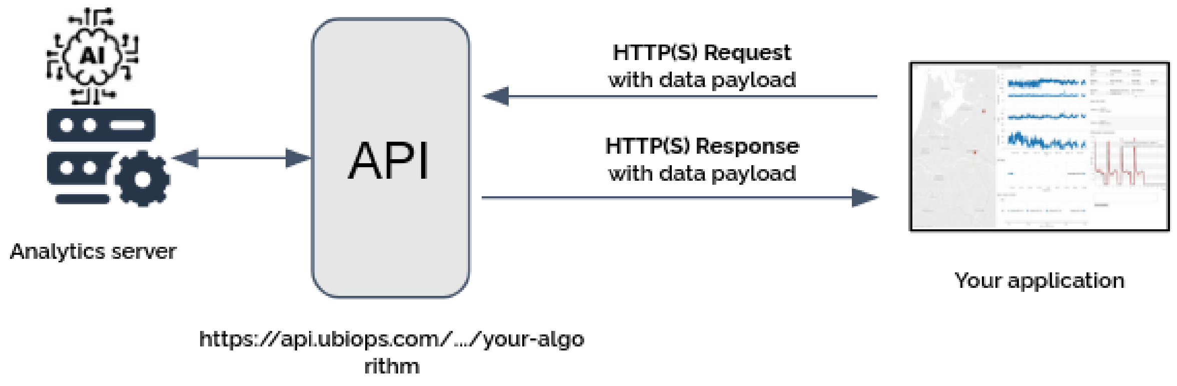


Fig: The user interface of the project

API IMPLEMENTATION:

API is the acronym for Application Programming Interface, which is a software intermediary that allows two applications to talk to each other. Each time you use an app like Facebook, send an instant message, or check the weather on your phone, you're using an API.



Analytics server
API
HTTP(S) Request with data payload
HTTP(S) Response with data payload
Your application
https://api.ubiops.com/.../your-algorithm

Fig: Working of API

In our application we have used Node.js to create REST API.

REST stands for **R**epresentational **S**tate **T**ransfer which means when a client machine places a request to obtain information about resources from a server, the server machine then transfers the current state of the resource back to the client machine.

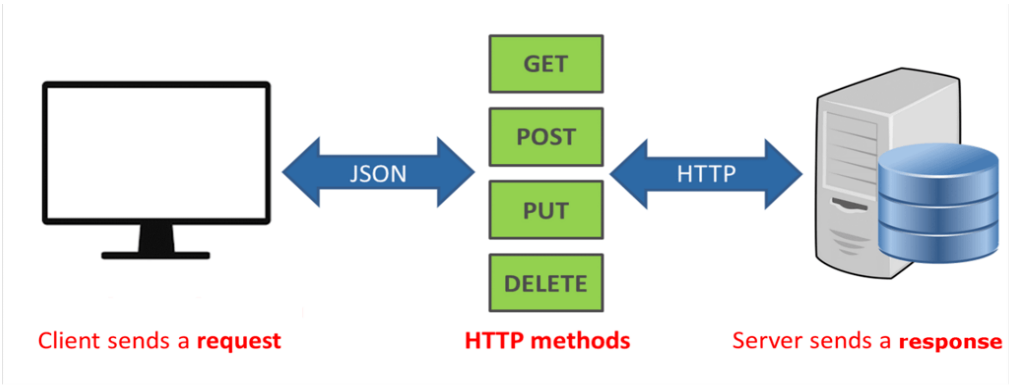


Fig: REST API architecture and usage

Following four HTTP methods are commonly used in REST based architecture.

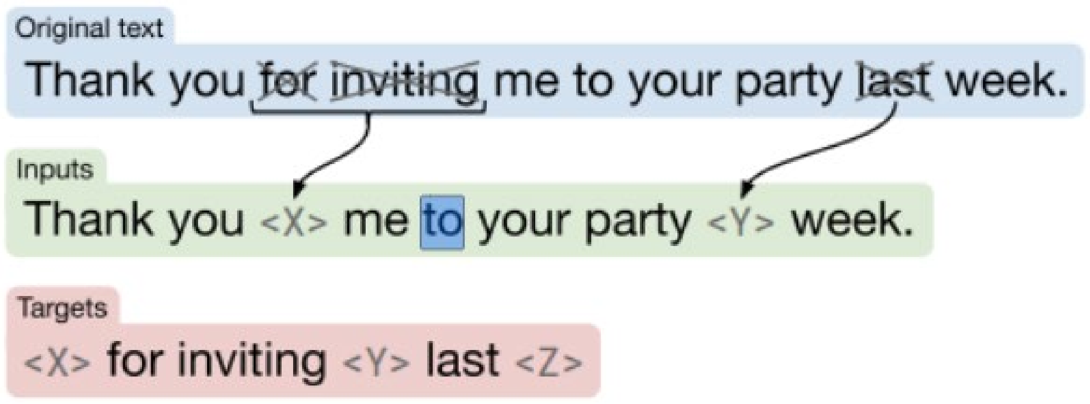
- **GET** − This is used to provide a read only access to a resource.
- **PUT** − This is used to create a new resource.
- **DELETE** − This is used to remove a resource.
- **POST** − This is used to update a existing resource or create a new resource.

Our Webapp send the user given string in form of json using **POST** method to the server and the server execute a python script which uses T5 model to improve grammar and send back to the client side as response.

ML MODEL WORKING:

The model that is Text-to-Text Transfer Transformer(T5) model is trained by a method called transfer learning, i.e., training the model on a large dataset and then finetuning it for our use

Unsupervised Objective: A sentence is inserted into the model and it independently uniformly at random removes some words from the sentence and is trained to predict basically sentinel tokens to delineate the dropped-out text.



Original text
Thank you ~~for inviting~~ me to your party ~~last~~ week.
Inputs
Thank you <X> me to your party <Y> week.
Targets
<X> for inviting <Y> last <Z>

The unsupervised objective. Source T5 paper.

Masking Technique: It was also trained using a technique which masked the data for predicting the upcoming word called casual with prefix mask this technique allows the model to look at the first bit of input sequence as it with full visuality and then it starts predicting what comes next, later on in the sentence. The result shows that the given model bi-directional context on the input is valuable. As this model is used for various purposes such as translating, predicting, generating text and much more

Features in the project

There are several features in our web-application.

- **OCR**

  Our application also uses OCR technology to get text from images given by the user in the specified area.

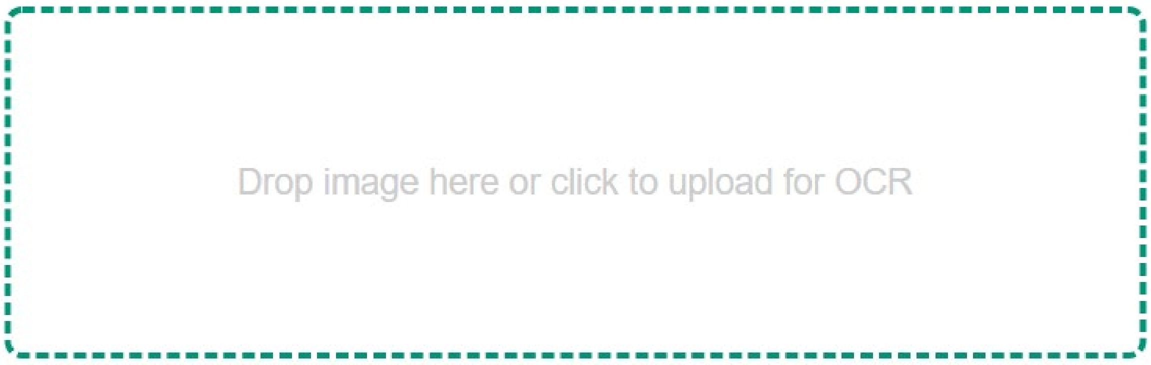


Fig: Area inside Application for OCR

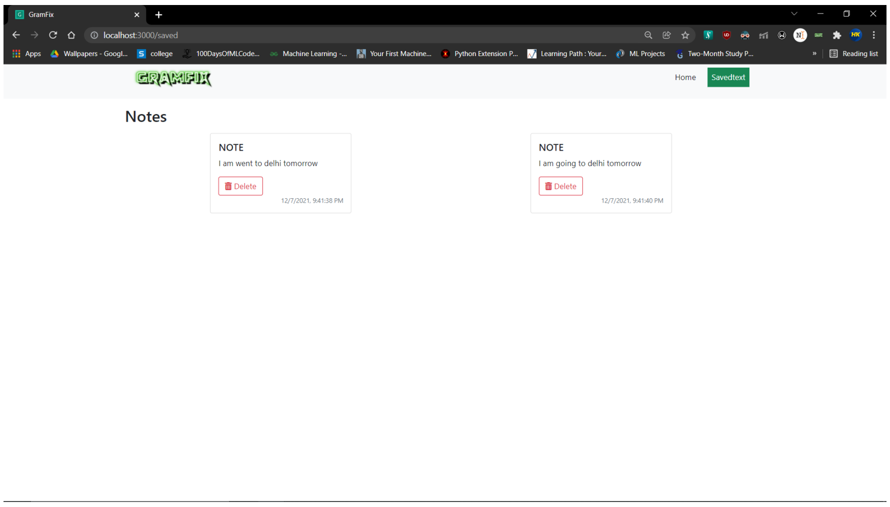
  The Optical Character Recognition (OCR) is a widespread technology to recognize text inside images such as scanned documents or photos.

  We have used Tesseract.js port in JavaScript of the popular Tesseract OCR engine for extracting the text from the image.

- **Save Text**

  The Save button saves the given text by the user in form of notes and we are storing data such that it remains even after the session is killed as we are using HTML Web Storage API to save our data whenever the page loads it checks the Web Storage for any data if the data is found it is updated in the app.

  The localStorage object stores the data with no expiration date. The data will not be deleted when the browser is closed, and will be available the next day, week, or year.



Notes

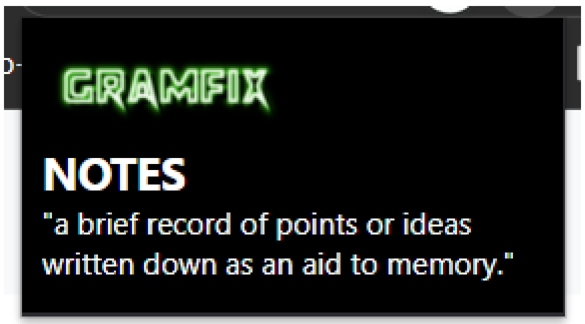
NOTE
I am went to delhi tomorrow
12/7/2021, 9:41:38 PM

NOTE
I am going to delhi tomorrow
12/7/2021, 9:41:40 PM

Fig: The Saved Text Page of Application

- <u>**Extension**</u>

We have also created an extension that can provide meanings of words that are highlighted by the user.



NOTES
"a brief record of points or ideas written down as an aid to memory."

Fig: Extension for displaying meanings

Extension Implementation:

Extensions are small software programs that customize the browsing experience. They let users tailor Chrome functionality and behaviour in many ways, providing things like:

- Productivity tools
- Web page content enrichment
- Information aggregation
- Fun and games

Extensions are built on web technologies such as HTML, JavaScript, and CSS. They run in a separate, sandboxed execution environment and interact with the Chrome browser.

Extensions let you "extend" the browser by using APIs to modify browser behaviour and access web content.

Manifest.json is the only file that must be present in every extension. It contains basic metadata such as its name, version, and the permissions it requires. It also provides pointers to other files in the extension.

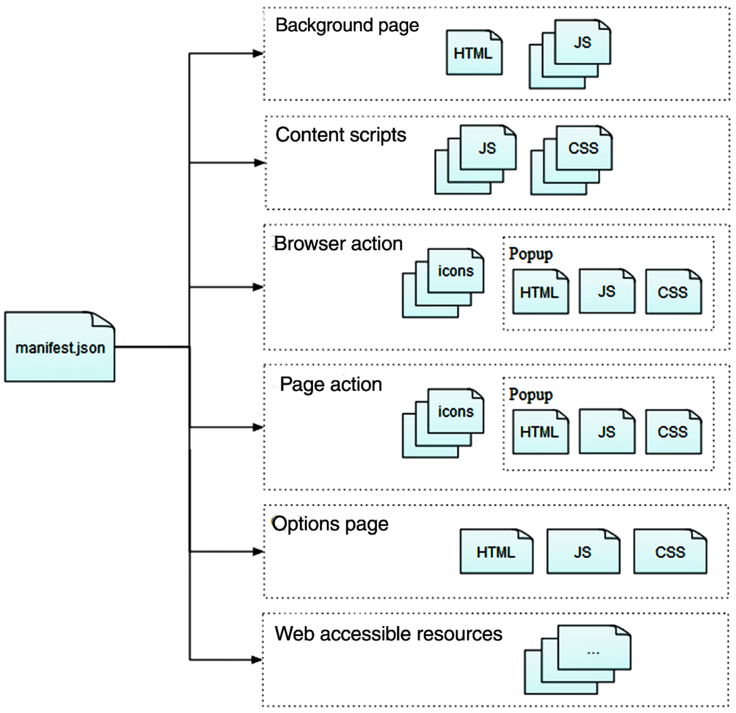


Fig: Manifest.json working

A **content script** is a part of your extension that runs in the context of a particular web page (as opposed to background scripts which are part of the extension, or scripts which are part of the web site itself, such as those loaded using the <script> element).

Extensions often need to maintain long-term state or perform long-term operations independently of the lifetime of any particular web page or browser window. That is what background scripts are for. Background scripts are loaded as soon as the extension is loaded and stay loaded until the extension is disabled or uninstalled.

Background scripts can access all the WebExtension JavaScript APIs, but they can't directly access the content of web pages. So if your extension needs to do that, you need content scripts.

Content scripts can only access a small subset of the WebExtension APIs, but they can communicate with background scripts using a messaging system, and thereby indirectly access the WebExtension APIs.

Our Extension gets data from user selection via content script and then send the data to background script. The background script uses a dictionary API (https://dictionaryapi.dev/) to get meaning of the given word and store them to local variable

- **<u>Tools</u>**

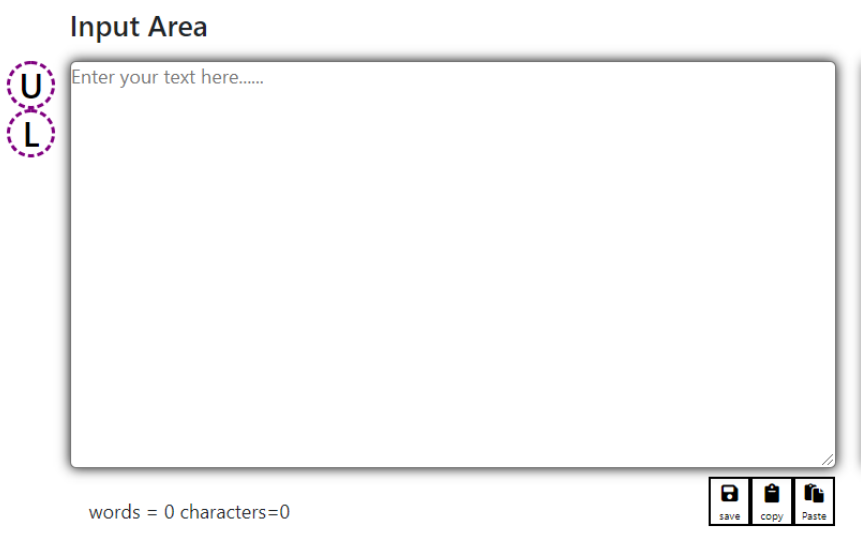


Fig: The input area of the project

The Uppercase(U), Lowercase(L) buttons on the left-hand side of the text box change the case of the text from Uppercase to Lowercase and vice versa by pressing the respective button

The words and characters are counted on the bottom of the text box that are present in it.

The paste, copy and save button on the bottom right do as they say, i.e. paste button pastes the text from the clipboard into the text-area, copy button copies the text inside the text-area to the clipboard, and the save button saves the text inside textbox in the Saved page of the application.

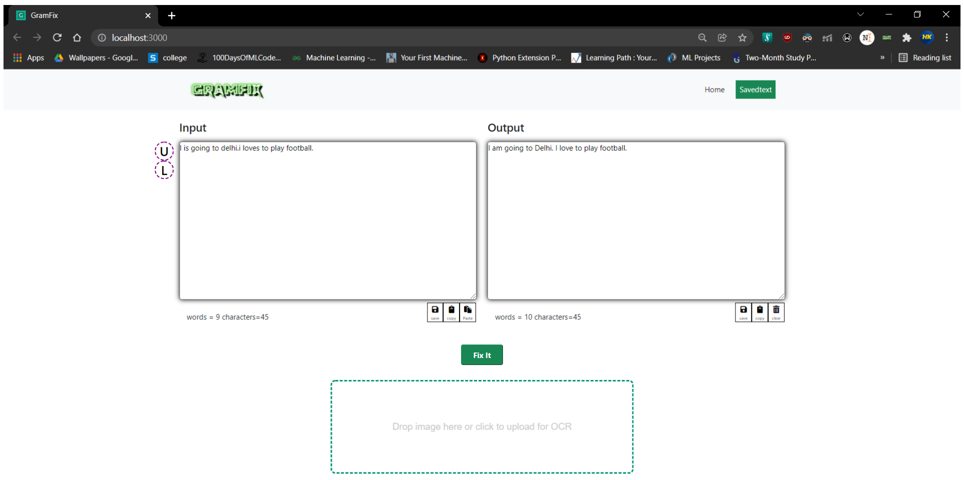
Results

Our application provided good corrections to our input sentences that we provided during testing phase, here are some examples:

1.)

Input: I is going to delhi.i loves to play football.

Output: I **am** going to Delhi. I **love** to play football.



Input

I is going to delhi.i loves to play football.

words = 9 characters=45

Output

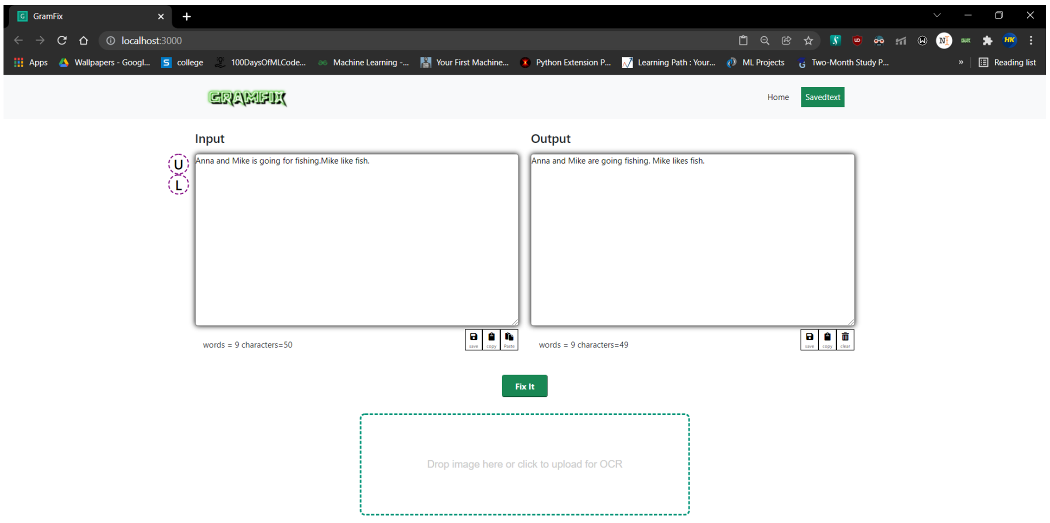
I am going to Delhi. I love to play football.

words = 10 characters=45

2.)

Input: Anna and Mike is going for fishing.Mike like fish.

Output: Anna and Mike **are** going fishing. Mike **likes** fish.



Input

Anna and Mike is going for fishing.Mike like fish.

words = 9 characters=50

Output

Anna and Mike are going fishing. Mike likes fish.

words = 9 characters=49

<u>Conclusion</u>

We believe that this is not the final stage of our project as we can improve the accuracy of the outputs by fine-tuning it on a larger dataset (that needs CPU and GPU power usage) and can give us many different cases, T5 model does a vast variety of functions, so we could add many more features in our project like translating a text, summarizing a paragraph which are just a few of its examples.

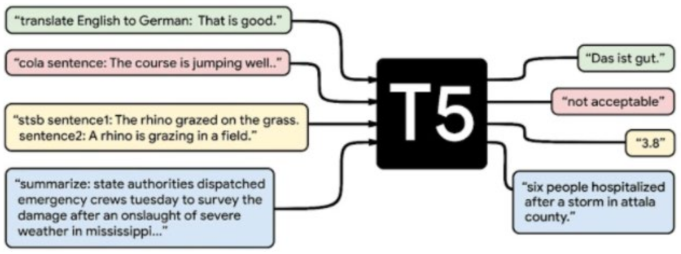

"translate English to German: That is good."
"cola sentence: The course is jumping well.."
"stsb sentence1: The rhino grazed on the grass. sentence2: A rhino is grazing in a field."
"summarize: state authorities dispatched emergency crews tuesday to survey the damage after an onslaught of severe weather in mississippi..."
T5
"Das ist gut."
"not acceptable"
"3.8"
"six people hospitalized after a storm in attala county."

Fig: Various use cases of T5 Model

Keywords Used:

*Transformer: - Transformers provides general purpose architectures for Natural Language Understanding (NLU) and Natural Language Generation (NLG) with over 32+ pretrained models in 100+ languages and deep interoperability between Jax, PyTorch and Tensorflow

*Happy Transformer: - It is a transformer built on top of Hugging Face's transformer library that makes it easy to utilize state-of-the-art NLP models

*JFLEG: - It is "a gold standard benchmark for developing and evaluating GEC systems with respect to fluency" (GEC stands for Grammar Error Correction). Its paper currently has 106 citations that is indeed respected within the NLP community.

References:

[1]Napoles, C., Sakaguchi, K., & Tetreault, J. (2017). JFLEG: A fluency corpus and benchmark for grammatical error correction. arXiv preprint arXiv:1702.04066.

[2] Napoles, C., Sakaguchi, K., Post, M., & Tetreault, J. (2015, July). Ground truth for grammatical error correction metrics. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers) (pp. 588-593).

[3] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... & Liu, P. J. (2019). Exploring the limits of transfer learning with a unified text-to-text transformer. arXiv preprint arXiv:1910.10683.