**PROJECT SUBMISSION: CREATE A CHATBOT USING PYTHON**

**LEADER: ROHINEE G - 211521104127**

**MEMBERS:**

**SHRUTHI P – 211521104146**

**UMA MEENAKSHI R – 211521104173**

**VASUMATHI D – 211521104175**

**YUVASREE G - 211521104189**

**Objective:**

This documentation provides a complete guide to create and deploy a chatbot using Python. A chatbot is a software application or web interface that mimics human conversations through text or voice interactions. This documentation provides an extensive overview of the project which covers all phases and aspects of the project’s implementation and deployment.

**Project Description:**

The project aims to create a chatbot using Python. The primary goal is to create a chatbot with user-friendly interface. The users can post a query to which the chatbot will provide a solution. AI chatbots are used in social media messaging apps or as an assistant which has the ability to make a phone call, set alarm and all the other activity that the user requests.

**Dataset:**

Link: <https://www.kaggle.com/datasets/grafstor/simple-dialogs-for-chatbot>

# Chatbot:

Chatbot, short for chat robot, is a computer program designed to simulate conversation with human users, especially over the internet. These artificial intelligence (AI) applications are increasingly becoming integral components of various industries, offering automated and interactive communication.

Purpose and Benefits: Chatbots serve diverse purposes, ranging from customer support and information retrieval to entertainment and task automation. They operate in messaging apps, websites, and other platforms, providing users with a conversational interface to interact with services and obtain information. The benefits of chatbots include improved efficiency, 24/7 availability, and enhanced user experiences.

Examples of Industries: Chatbots find applications across a wide array of industries. In customer service, they assist in answering frequently asked questions and troubleshooting. E-commerce platforms employ chatbots for product recommendations and order tracking. Healthcare chatbots can provide initial medical information, and educational chatbots assist in learning processes. As technology advances, the versatility of chatbots continues to expand, making them a valuable asset in modern digital ecosystems

**Programming Language - Python:**

Python is used for creating a chatbot because:

* It has a simple syntax that’s easy to understand.
* It has an extensive set of libraries for Machine Learning and Natural Language processing which makes the chatbot creation very easy and efficient.
* It makes it possible for developers to build chatbots that have conversational nuances and empathetic tones using python.
* It is versatile.
* It also has a database of multiple languages, making it possible for developers to deploy multilingual chatbots.
* Ability to handle multiple queries.
* It supports cross platform compatibility.

**Characteristics of Chatbot:**

* It should support omnichannel support. Omnichannel messaging involves communicating with the audience seamlessly across all the messaging channel.
* The chatbot should be used to analyse the sentiment of the user and provide the necessary feedbacks.
* Create chatbots to help users find products, make purchase recommendations, track orders, and answer product-related questions.
* Ability to provide appropriate solutions for the users’ queries.

**Key Features:**

* Natural Language Processing (NLP): Allows the chatbot to understand and interpret human language and provide responses in a human-like conversational manner.
* Machine learning: Enables chatbots to learn from their interactions with users and improve over time.
* Predictive analytics: AI algorithms can enable chatbots to predict a user’s next prompt. This makes them more effective and improves customer satisfaction.
* Intent recognition: AI algorithms enable chatbots to understand the intent behind a user’s query in order to provide an appropriate response.

**Steps to develop a chatbot:**

1. Define the purpose and goals

The objective of the chatbot should be defined so that it can be developed as needed.

1. Identify the target audience

Understanding the needs and preferences of the target audience will help us in developing a user-friendly and efficient chatbot.

1. Selecting a platform

Choose the platform in which the chatbot will be deployed (eg. Facebook, Instagram etc). The selected platform should also match the preference of the target audience.

1. Design the conversation flow

Create a flowchart outlining the conversation structure. Define various user inputs and the chatbot responses. Consider various user scenarios and extreme cases for seamless interaction.

1. Choose the technology stack

Depending upon the type of chatbot, the technology stack and tools must be used. The necessary libraries must also be implemented.

1. Create a knowledge base

The chatbot should contain a knowledge base or database of information it can learn from to answer user queries. Regular maintenance and updating of the database is mandatory.

1. Develop user interface

Design user interface for the chatbot including chatbot window and graphical elements. The interface should be visually appealing and user-friendly.

1. Security and privacy measures

The user data should be securely handled with relevant data protection regulations. The authentication and authorization is mandatory.

**Implementation:**

1. Importing dataset

The necessary dataset is imported from a database or repository.

1. Importing the necessary libraries

The necessary libraries in Python like numpy, tensorflow, chatterbot and natural language toolkit must be imported.

1. Text processing

Once you have collected the data, you will need to pre-process it. This includes cleaning and normalizing the data, removing irrelevant information, and tokenizing the text into smaller pieces. Once data is collected for training a chatbot, it’s important to pre-process it to ensure it’s clean and ready for use. Here are a few steps involved in pre-processing:

* [Data Cleaning](https://www.analyticsvidhya.com/blog/2022/08/template-for-data-cleaning-using-python/): Remove irrelevant or duplicate data, correct errors, and standardize the data format.
* Text Normalization: Convert text to lowercase, remove punctuation, and expand contractions to ensure consistency in the data.
* [Tokenization](https://www.analyticsvidhya.com/blog/2022/01/guide-for-tokenization-in-a-nutshell-tools-types/): Break the text down into smaller units, such as words or phrases, to make it easier for them to understand and process.
* Stop Words Removal: Remove common words such as “the,” “is,” and “and” which don’t add much meaning to the text.
* [Lemmatization](https://www.analyticsvidhya.com/blog/2022/06/stemming-vs-lemmatization-in-nlp-must-know-differences/): Group together different forms of the same word, such as “running” and “ran,” to reduce the dimensionality of the data.
* Part-of-speech Tagging: Identify the grammatical role of each word in the text, such as a noun, verb, or adjective.

1. Building a model

After completing the data preprocessing, the model must be built and the pre-processed data must be fed into the model. The architecture and the number of layers required can vary accordingly.

1. Calling the relevant functions and interacting with the chatbot

This step involves giving input to the chatbot and calling the user-defined functions.

**Selecting NLP techniques:**

Various [NLP techniques](https://www.analyticsvidhya.com/blog/2022/12/top-10-blogs-on-nlp-in-analytics-vidhya-2022/) can be used to build a chatbot, including rule-based, keyword-based, and machine learning-based systems. Each technique has strengths and weaknesses, so selecting the appropriate technique for your chatbot is important. Machine Learning for NLP such as BERT can be used in chatbot creation.

Various natural language processing (NLP) techniques can be used to build a chatbot, each with its strengths and weaknesses. Here are a few examples of NLP techniques that can be used to build it:

Rule-based Systems: These systems rely on predefined rules to understand and respond to user inputs. They are simple to implement and effective for simple tasks, but they may struggle with more complex inputs.

1. Keyword-based Systems: These systems rely on matching keywords in the user input to predefined responses. They are easy to implement but can be limited in their ability to understand the context and handle more complex inputs.
2. Machine Learning-based Systems: These systems rely on machine learning algorithms to understand and respond to user inputs. They are more complex to implement but can handle complex inputs and improve over time as they learn from more data.
3. Intent Recognition: Identifying the intent behind the user’s input, for example, booking a flight or asking a question, using techniques such as supervised learning, unsupervised learning, or deep learning.
4. Language Model: These models are pre-trained on a large dataset and can be fine-tuned for specific tasks such as language translation, question answering, and text summarization.
5. Sentiment Analysis: Identifying the sentiment or emotion behind a text, such as positive, negative, or neutral, using techniques such as supervised learning or deep learning.

**BERT (Bidirectional Encoder Representations from Transformers):**

Advantages:

1. Contextual Understanding: BERT captures contextual relationships between words, allowing it to understand the nuanced meaning of words based on their surrounding words. This contextual understanding is crucial in sentiment analysis, where the meaning of a word can change based on the context in which it's used.
2. Pre-Trained Representations: BERT is pre-trained on massive amounts of text data, allowing it to learn rich language representations. Leveraging these pre-trained representations often leads to better performance, especially with smaller datasets like the one provided.
3. Attention Mechanism: BERT uses attention mechanisms to weigh the importance of different words in a sentence. This mechanism enables BERT to focus on relevant words, capturing intricate relationships and sentiments within the text.
4. Fine-Tuning Capabilities: BERT models can be fine-tuned on specific tasks, allowing them to adapt and specialize for sentiment analysis while retaining the general language understanding learned during pre-training.

**Implementing and training the model:**

After selecting the appropriate NLP techniques, you can start building the chatbot. This includes implementing the NLP techniques, training the chatbot using the data collected earlier, and fine-tuning it.

Once you have selected the appropriate natural language processing (NLP) techniques, you can start building them by implementing and training them. Here are a few steps involved in this process:

1. Select a Development Platform: Choose a platform such as Dialogflow, Botkit, or Rasa to build the chatbot.
2. Implement the NLP Techniques: Use the selected platform and the NLP techniques to implement the chatbot. This includes creating the chatbot’s architecture, designing the dialogue flow, and integrating the NLP models.
3. Train the Chatbot: Use the pre-processed data to train the chatbot. This includes fine-tuning the models, testing them with different inputs, and adjusting them as needed.
4. Test the Chatbot: Test it with different inputs to evaluate its performance in terms of accuracy and user satisfaction.
5. Iterate and Improve: Based on the testing results, iterate and improve it by adjusting the models, fine-tuning the parameters, and adding new functionalities.
6. Integrate with Other Systems: Integrate it with other systems, such as databases or APIs, to access the required information and perform the intended tasks.

**Testing and Evaluating:**

Once the chatbot is built, it’s important to test and evaluate its performance to ensure it meets the target audience’s needs and reaches its goals. Here are a few steps involved in testing and evaluating a chatbot:

1. User Acceptance Testing: Test the chatbot with a group of users to gather feedback on its performance and user experience.
2. Functional Testing: Test the chatbot’s ability to perform specific tasks, such as answering questions or providing information.
3. Performance Testing: Measure the chatbot’s response time, accuracy, and scalability.
4. A/B Testing: Compare the chatbot’s performance against a control group or a different chatbot version.
5. Error Handling: Test the chatbot’s ability to handle unexpected inputs or error conditions.
6. Usability Testing: Evaluate the chatbot’s user interface and how easily users interact.

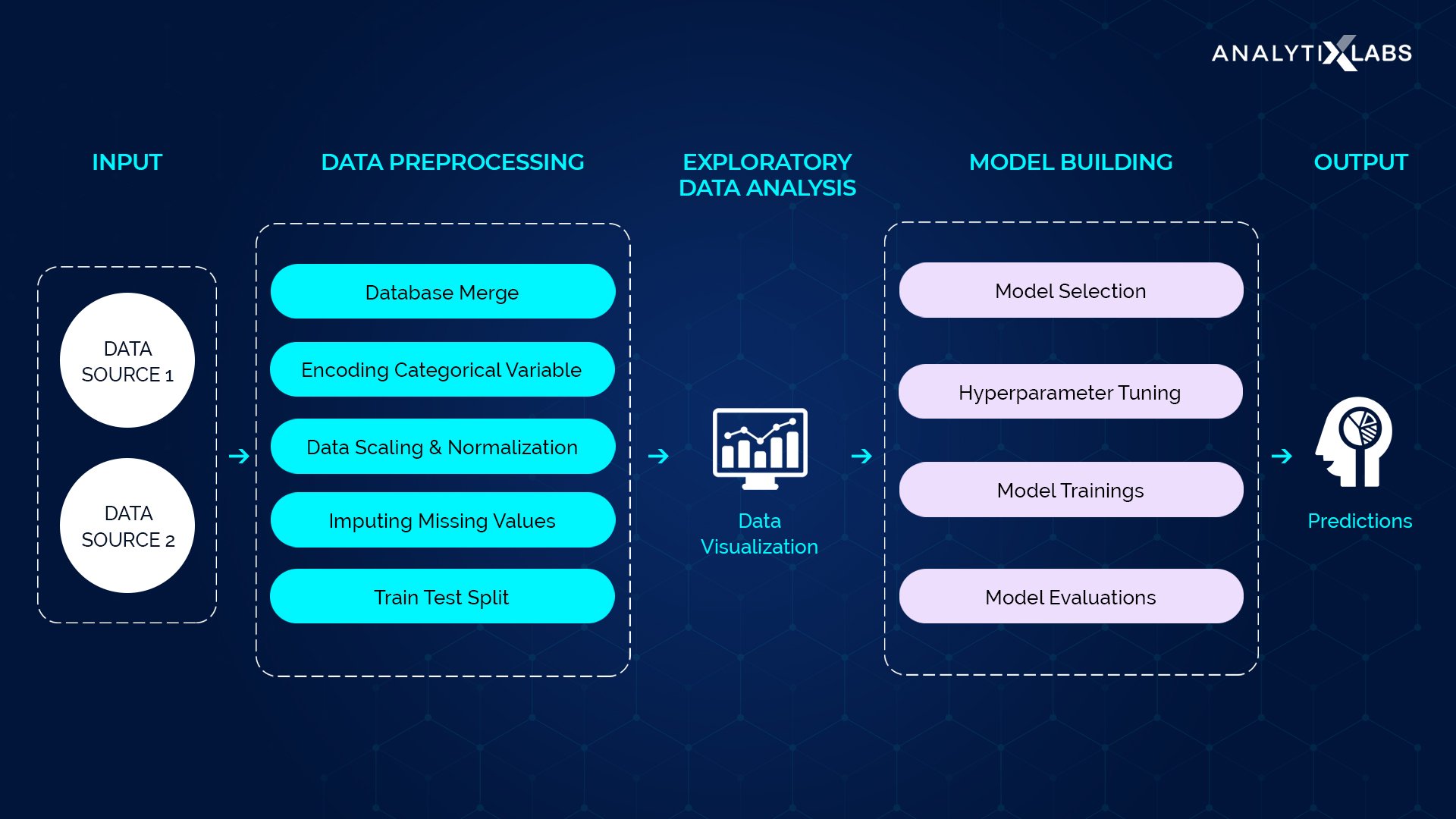
DATA PREPROCESSING

Introduction:

A chatbot is a computer program that simulates and processes human conversations, allowing humans to interact with digital devices as if they are communicating with a real person. Chatbots, like other AI tools, will be used to further enhance human capabilities and makes our world a more efficient and connected place to live and work.

DATA PREPROCESSING

Data Preprocessing is the process of preparing the raw data and making it suitable for the machine learning model. It is the first and the most crucial step in machine learning. The data that we have are not proper, clean and formatted. For the machine learning model to perform efficiently, the data must be cleaned and put in a formatted way before performing any other process.



WHY DO WE NEED DATA PREPROCESSING?

Data is not always pure and clean. Data usually contains noise, missing values or sometimes the data might be unformatted. This cannot be used by the machine learning model as they might reduce the accuracy of the model. Data preprocessing involves the following steps:

* Importing the necessary libraries
* Importing the dataset
* Finding missing data
* Encoding categorical data
* Splitting the dataset into training and testing dataset
* Feature scaling

IMPORTING THE NECESSARY LIBRARIES:

Python has many libraries which can perform specific data preprocessing functions. Importing all these libraries is essential for data preprocessing. Python libraries like numpy, pandas, matplotlib, sklearn are essential libraries required for data preprocessing.

IMPORTING THE DATASET

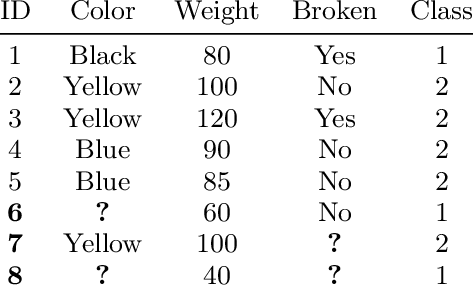
The dataset that should be used in the model must be imported. Importing the dataset is done using the read\_csv() function in Pandas library. This function can read a CSV (Comma Separated File) file. We can also import a HTML file or an Excel file. After importing the dataset, the dependent and independent variables should be extracted and labelled separately. For this, we use the iloc[] function from the Pandas library. This function can extract selected rows and columns from the dataset.

FINDING MISSING VALUES

The data that we use may have missing values which when used in the machine learning model would reduce the accuracy and efficiency of the model. The model might draw faulty conclusions from this dataset. So, to avoid these problems, missing values are identified and corrected during the data preprocessing phase. There are two ways to handle missing values:

* Deleting a particular row - In this method, you remove a specific row that has a null value for a feature or a particular column where more than 75% of the values are missing. However, this method is not 100% efficient, and it is recommended that you use it only when the dataset has adequate samples. You must ensure that after deleting the data, there remains no addition of bias.
* Calculating the mean - This method is useful for features having numeric data like age, salary, year, etc. Here, you can calculate the mean, median, or mode of a particular feature or column or row that contains a missing value and replace the result for the missing value. This method can add variance to the dataset, and any loss of data can be efficiently negated. Hence, it yields better results compared to the first method (omission of rows/columns).

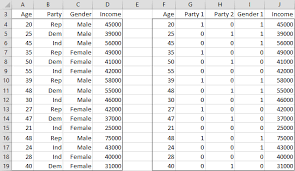
To handle missing values, Scikit-learn library has many libraries used for building machine learning models. Sklearn.preprocessing library has Imputer class which replaces the missing data with mean, median or mode. Median value is always preferred because mean value is influenced by outliers and skewness in the data. There are many types of imputers like SimpleImputer, IterativeImputer, KNNImputer etc.



ENCODING A CATEGORICAL DATA

Categorical data refers to the information that has specific categories within the dataset. Age, gender, educational qualification are examples of categorical data. Categorical data must be encoded because the machine learning model cannot use categorical data in raw form. Categorical data are of two types:

* Ordinal categorical variable: Ordinal categorical variables can be ordered. For example, grades’ can be arranged in order from A to E. For ordinal categorical variable, LabelEncoder is used.
* Nominal categorical variable: Nominal categorical variables can never be ordered. For example, the color of the eye lens can never be arranged in an order. For nominal categorical variable, OneHotEncoder is used.



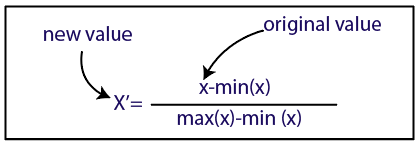
SPLITTING THE DATASET INTO TRAINING AND TEST SET

After data preparation, the dataset is divided into two parts – training set, and the test set. The training data is used to build the model. It identifies the patterns inside the dataset and generates model parameters. The training helps to identify the overfitting and underfitting in the data. The test set is used to predict the output by calculating the accuracy. Usually the split is in the ratio 80:20 or 70:30. To implement training and test set splitting, we have to import train\_test\_split function from sklearn.model\_selection library. Thus, the train\_test\_split() function includes four parameters, the first two of which are for arrays of data. The test\_size function specifies the size of the test set. The test\_size maybe .5, .3, or .2 – this specifies the dividing ratio between the training and test sets. The last parameter, “random\_state” sets seed for a random generator so that the output is always the same.

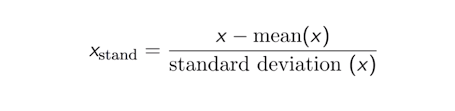
FEATURE SCALING

Feature Scaling is the process of standardizing the independent variables of a dataset within a specific range. Feature scaling limits the range of variables so that you can compare them. There are two ways of feature scaling:

* Normalisation: Normalization is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1. It is also known as Min-Max scaling.



* Standardisation: Standardization is another scaling method where the values are centered around the mean with a unit standard deviation. This means that the mean of the attribute becomes zero, and the resultant distribution has a unit standard deviation.



To implement feature scaling, we import StandardScaler class from sklearn.preprocessing library.

TEXT PREPROCESSING

In Natural Language Processing, text preprocessing is the practice of cleaning and preparing text data. NLTK and re are the usual python libraries used for text preprocessing.

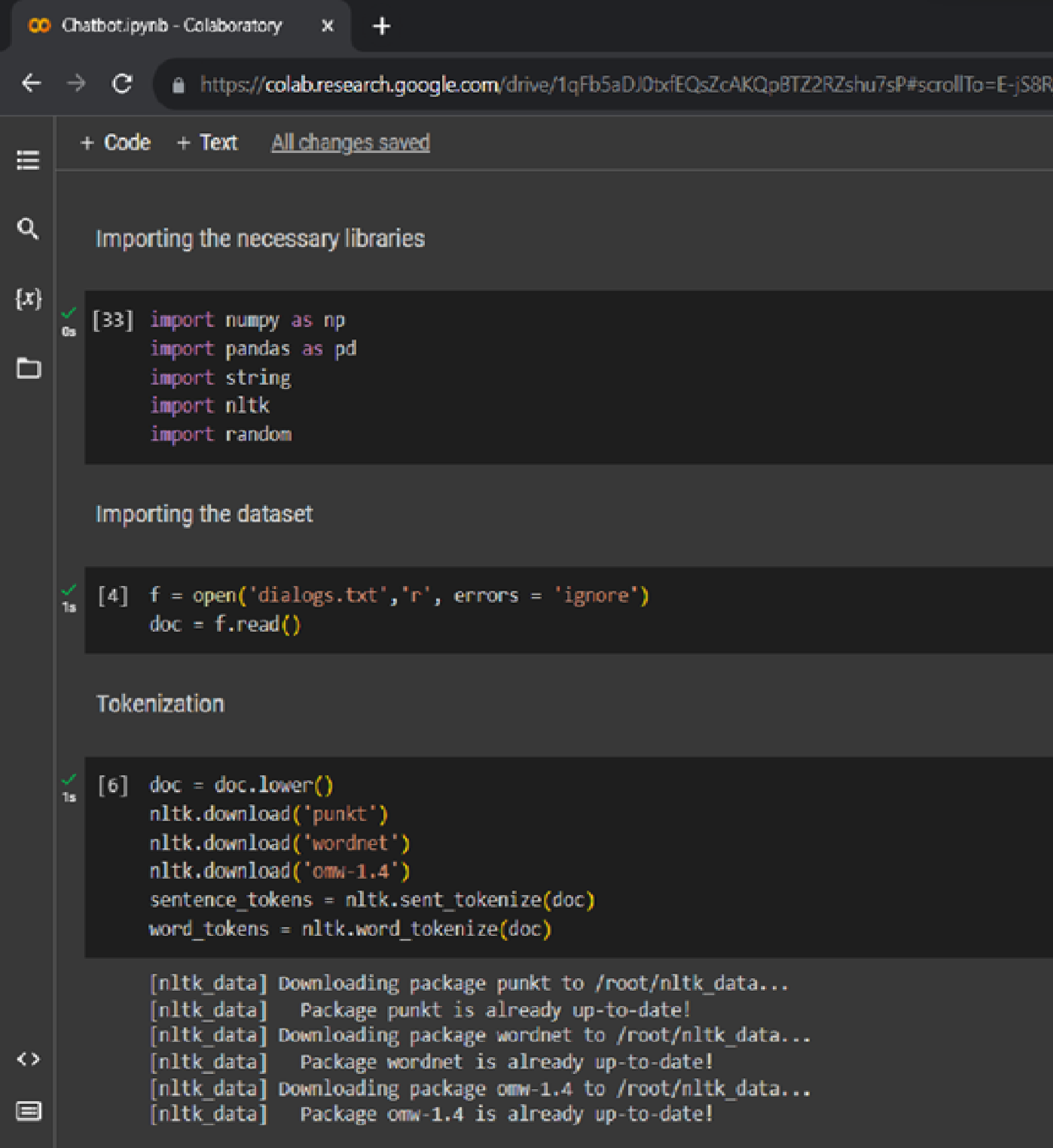
STEPS INVOLVED IN TEXT PREPROCESSING

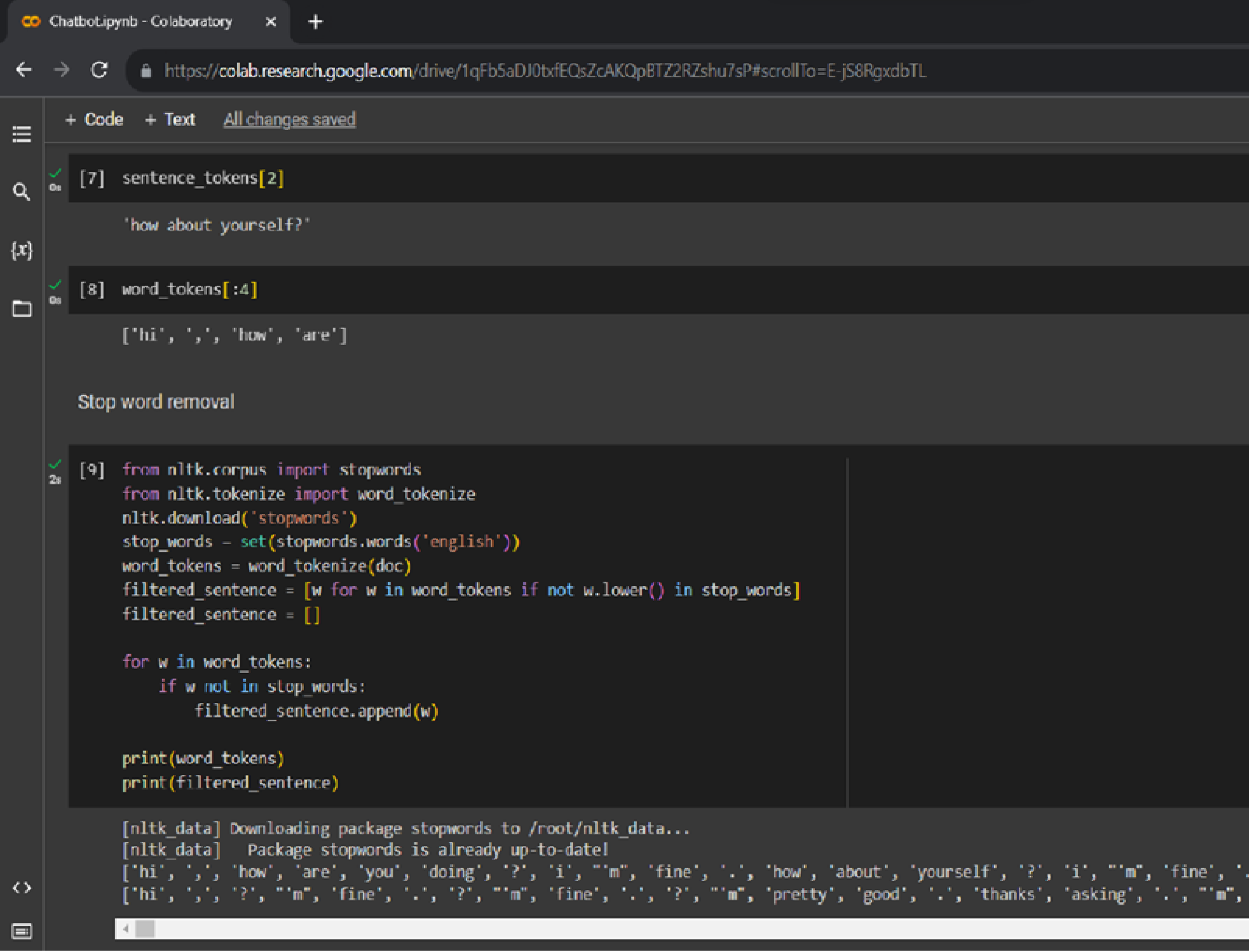
* Noise removal: It is a text preprocessing task that removes characters, digits and pieces of text that can interfere with the text analysis.
* Tokenisation: It is the text preprocessing task of breaking up text into smaller components of text (known as tokens).
* Text normalisation: Normalization encompasses many text preprocessing tasks including stemming, lemmatization, upper or lowercasing, and stopwords removal.
* Stemming: It is the text preprocessing normalization task concerned with bluntly removing word affixes (prefixes and suffixes).
* Lemmatization: It is the text preprocessing normalization task concerned with bringing words down to their root forms.
* Stopword removal: It is the process of removing words from a string that don’t provide any information about the tone of a statement.
* Part-of-speech tagging: It is the process of assigning a part of speech to every word in a string. Using the part of speech can improve the results of lemmatization.

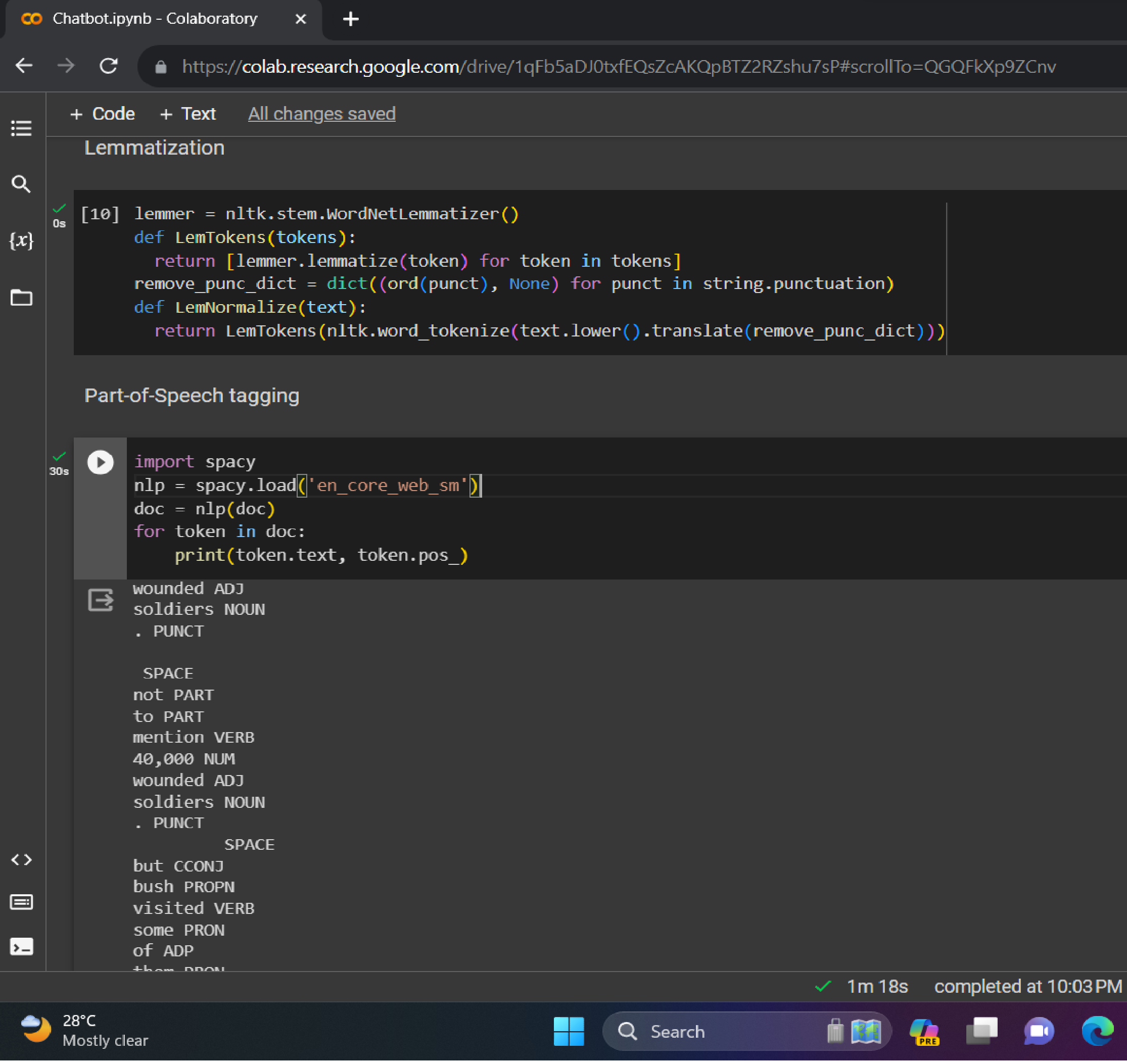
CODE:

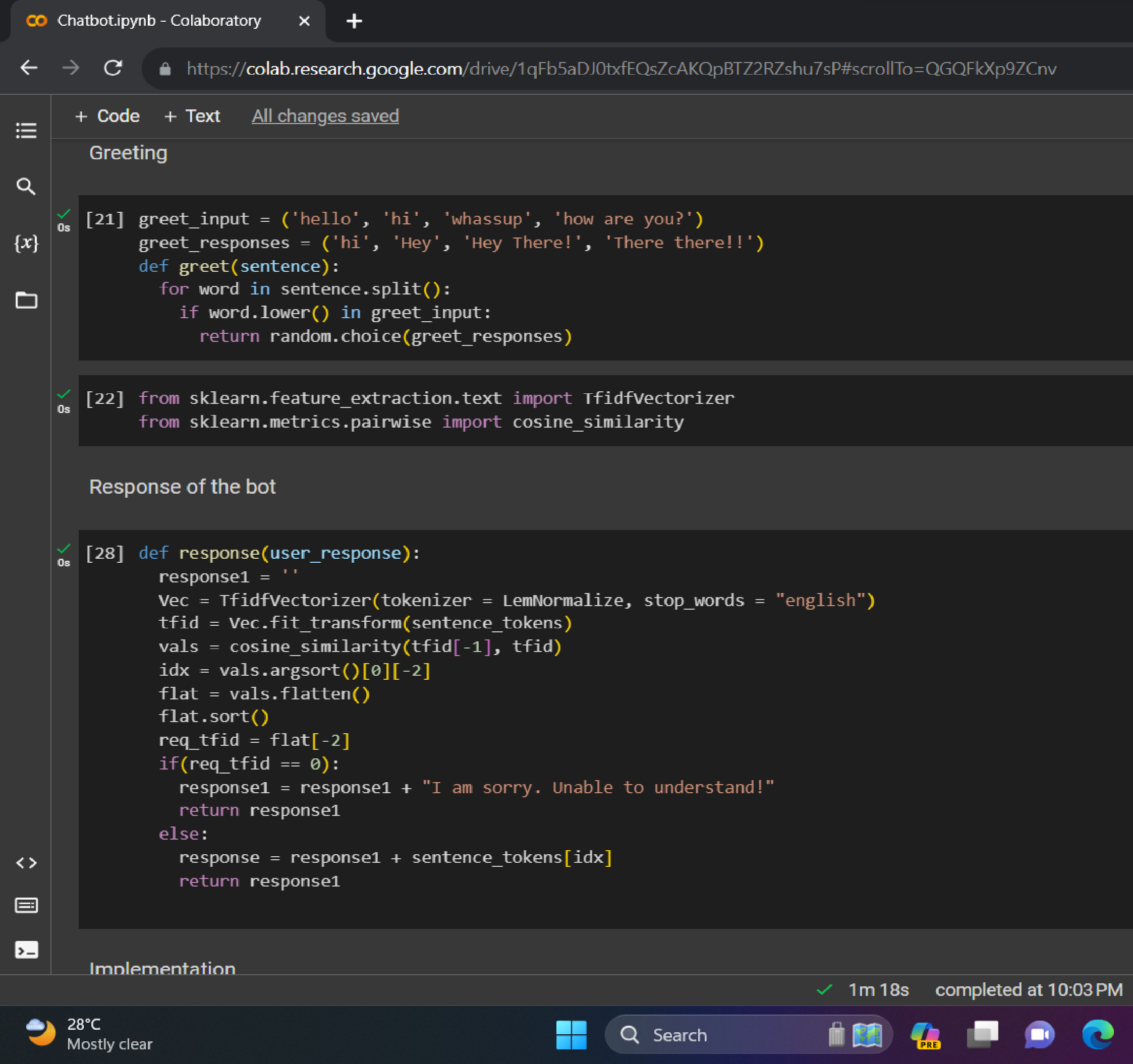
WORKING ENVIRONMENT: Google collaboratory notebook

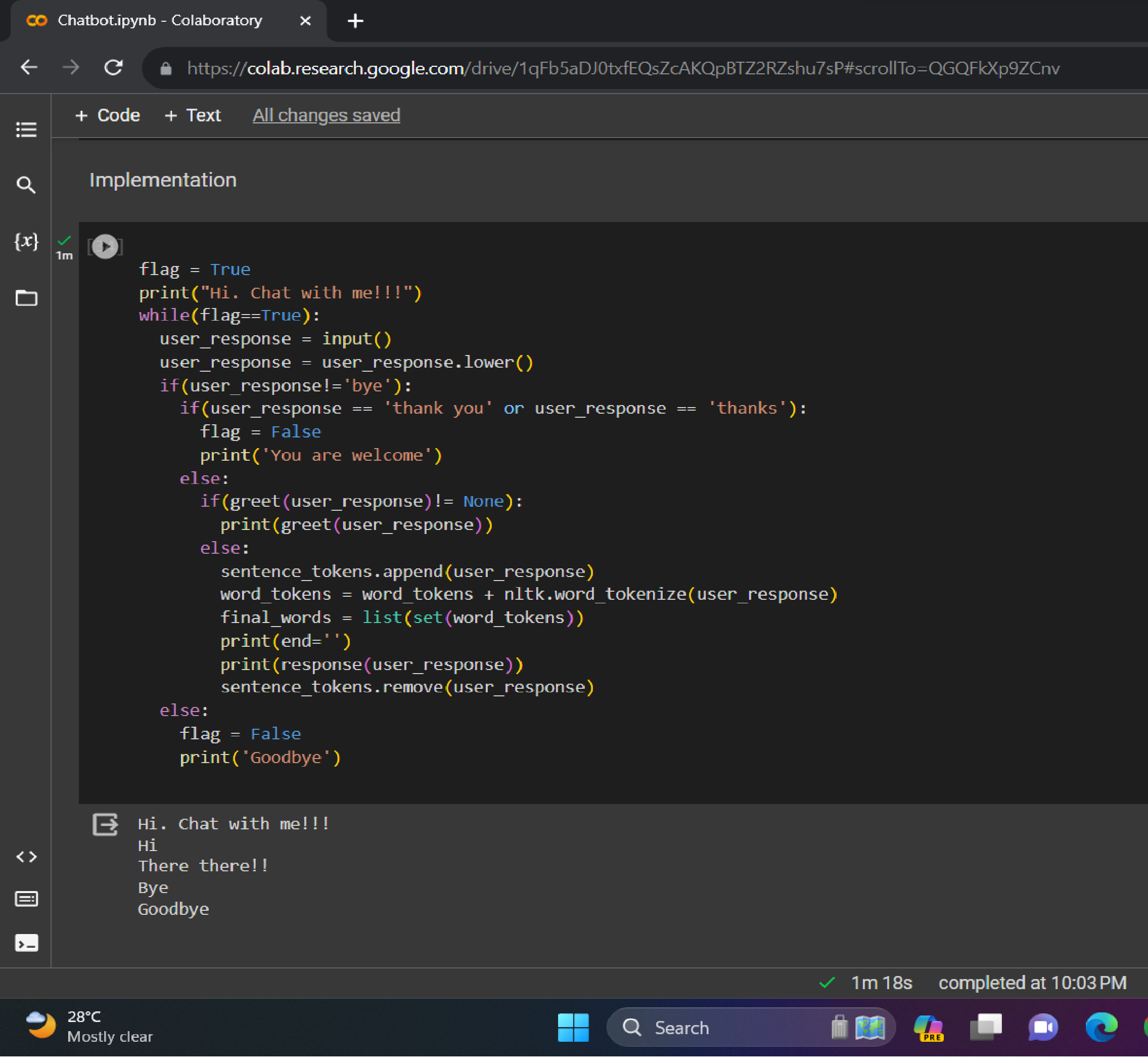
CODE:

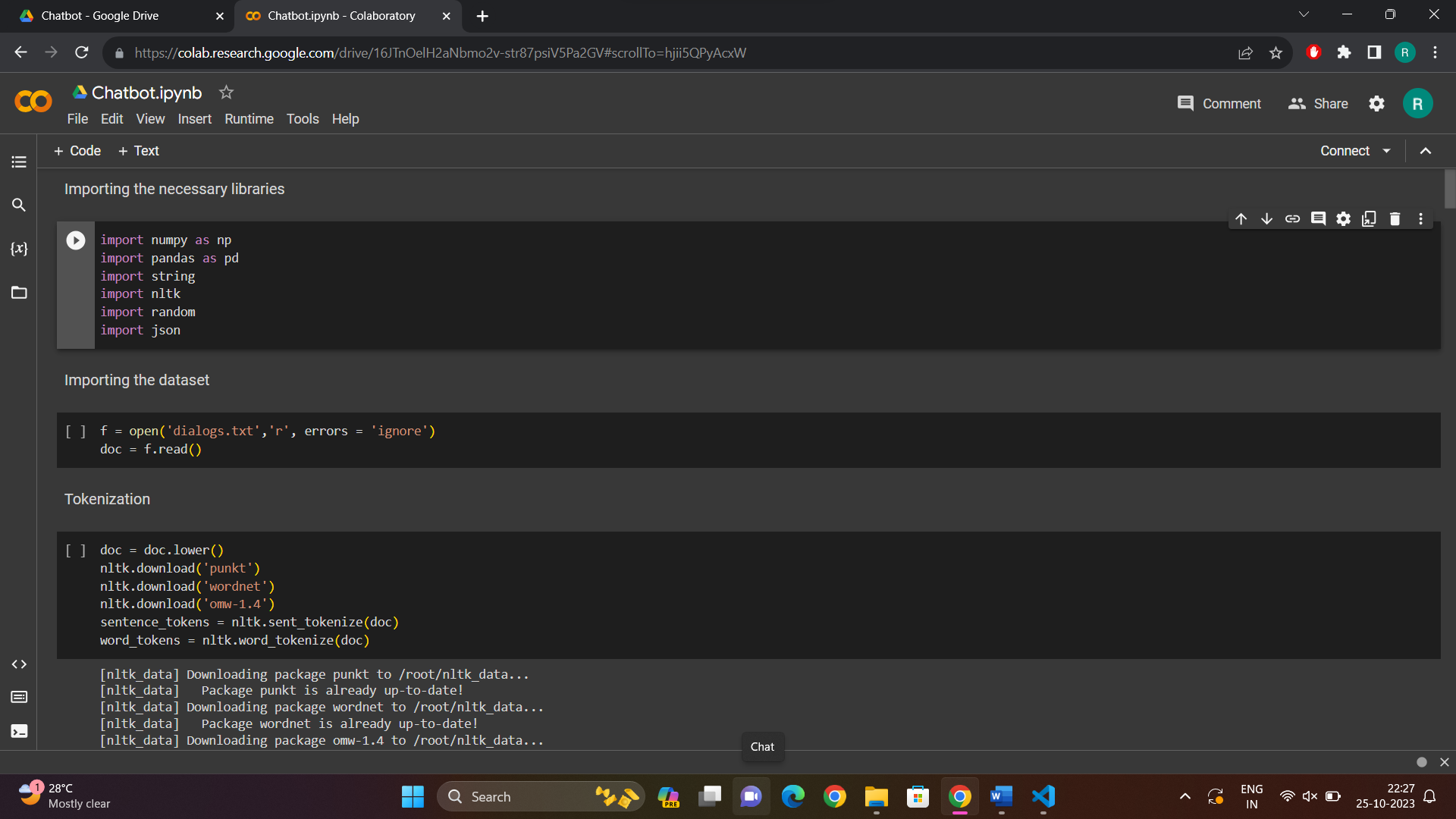


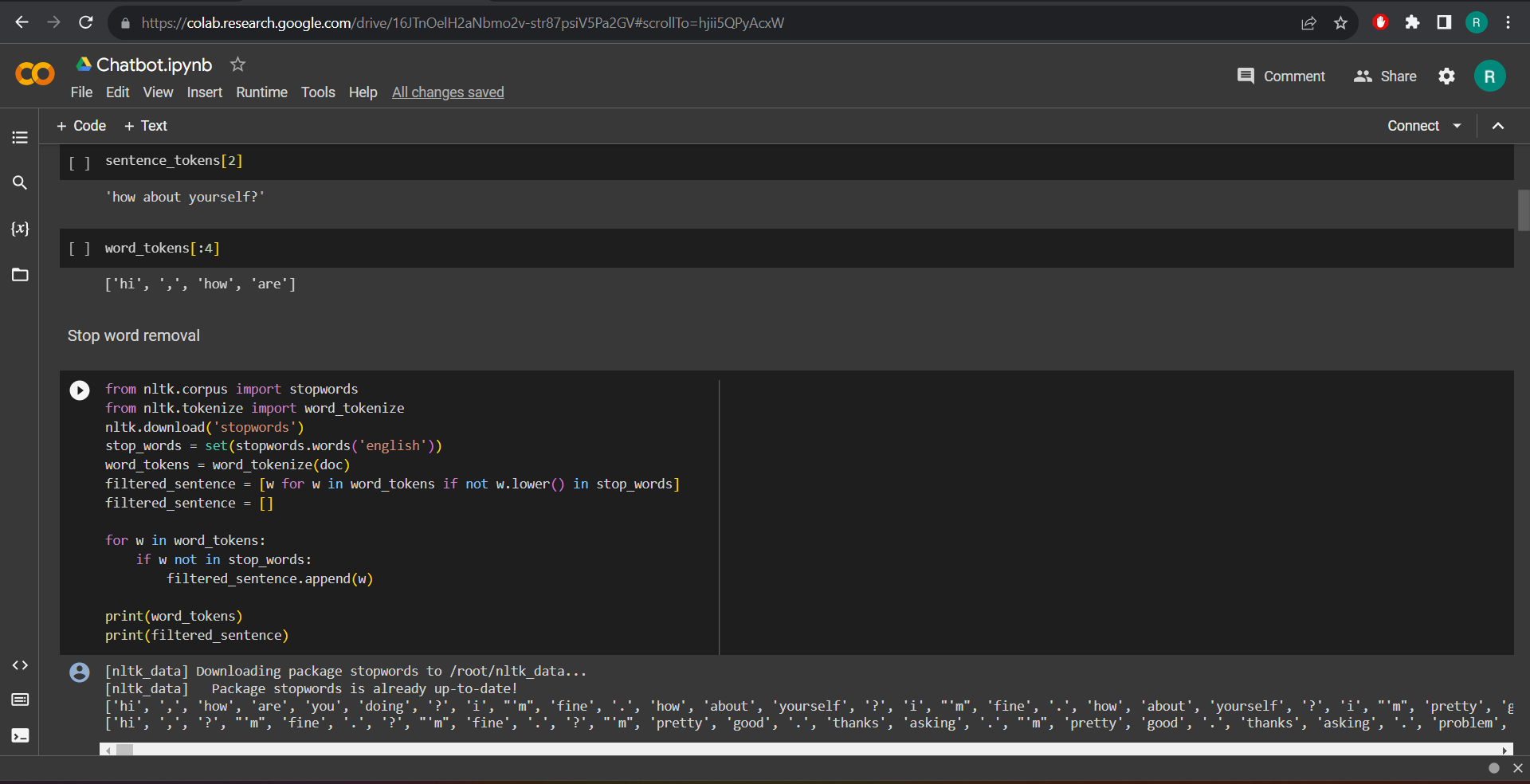


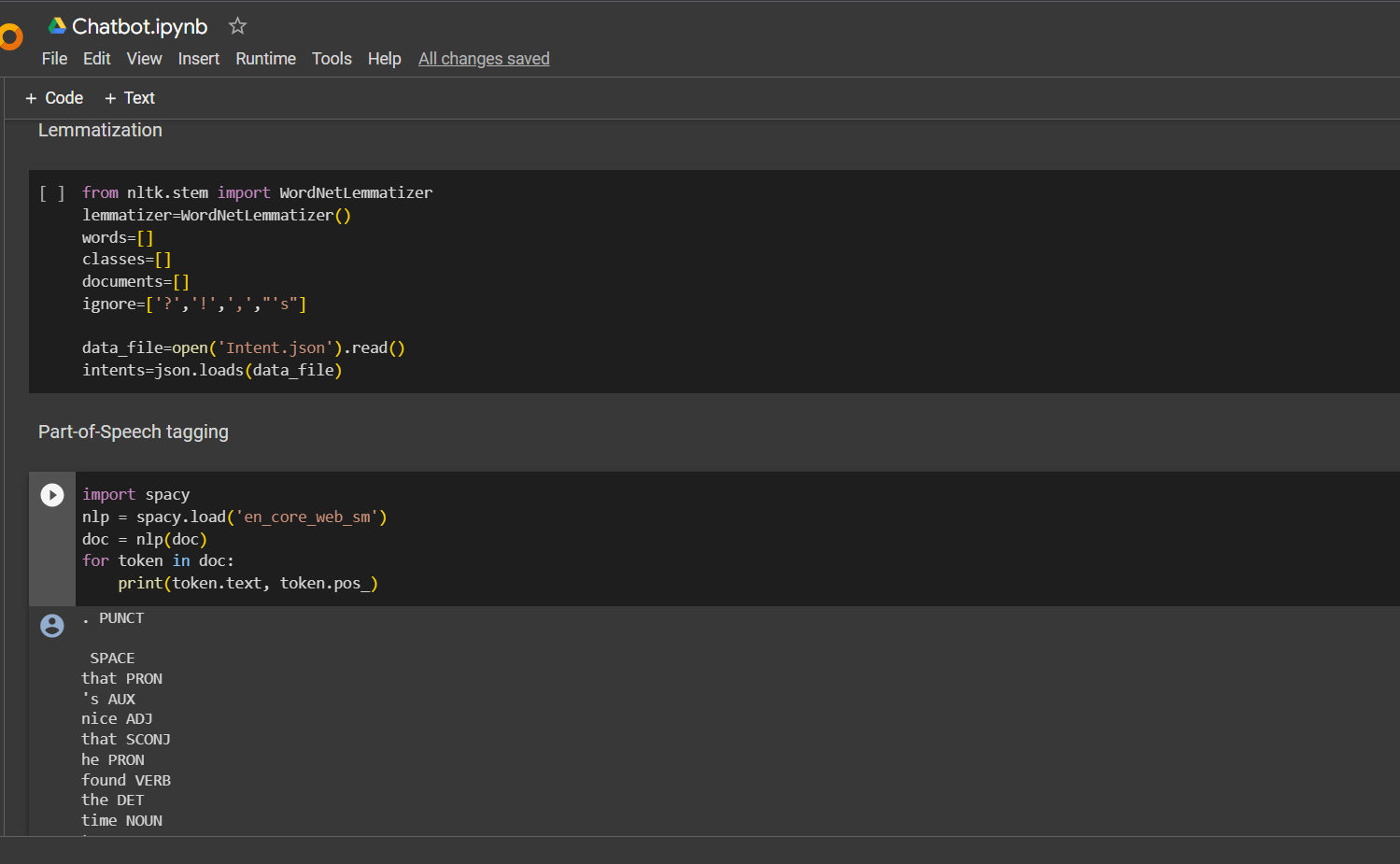


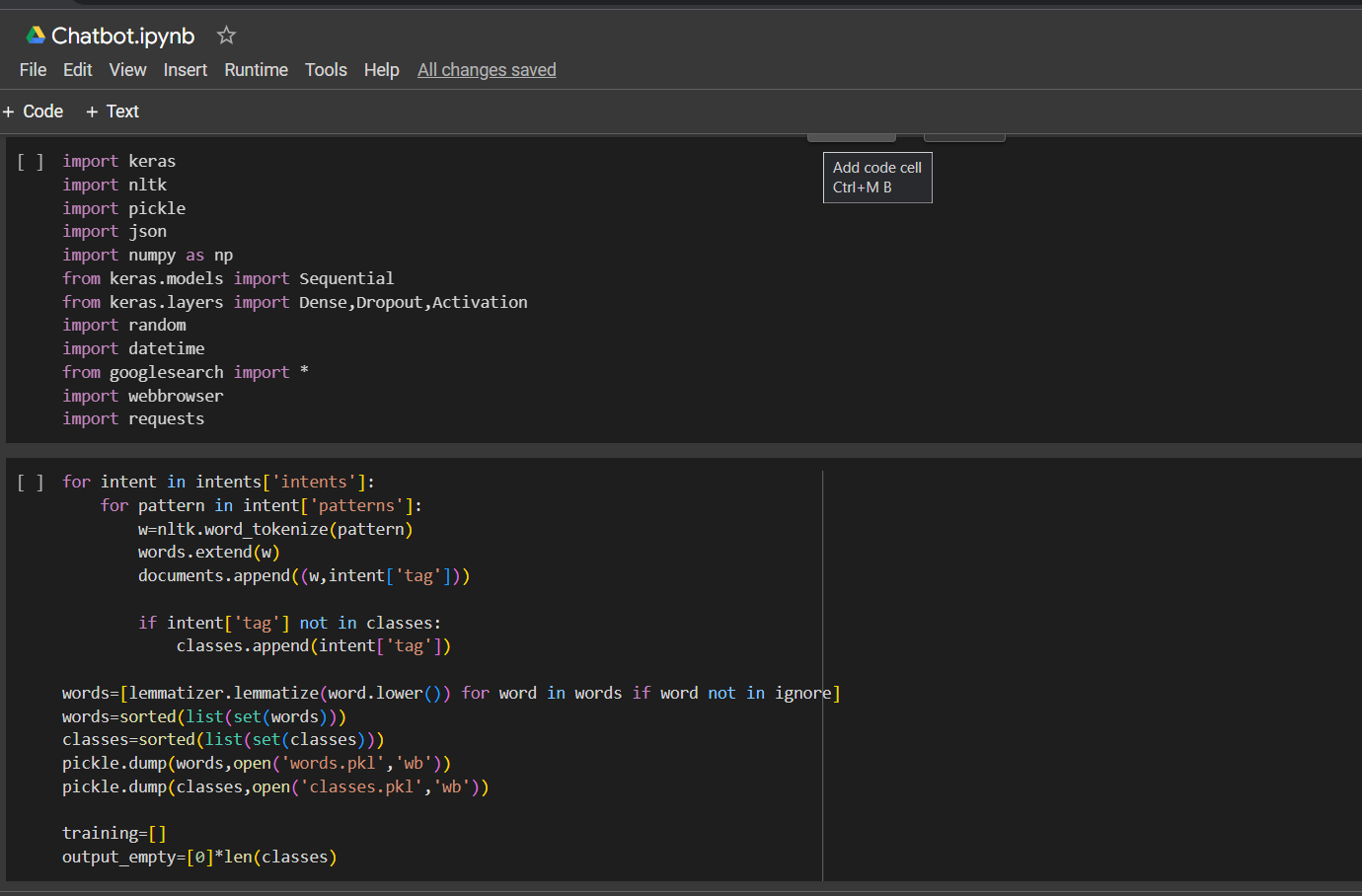


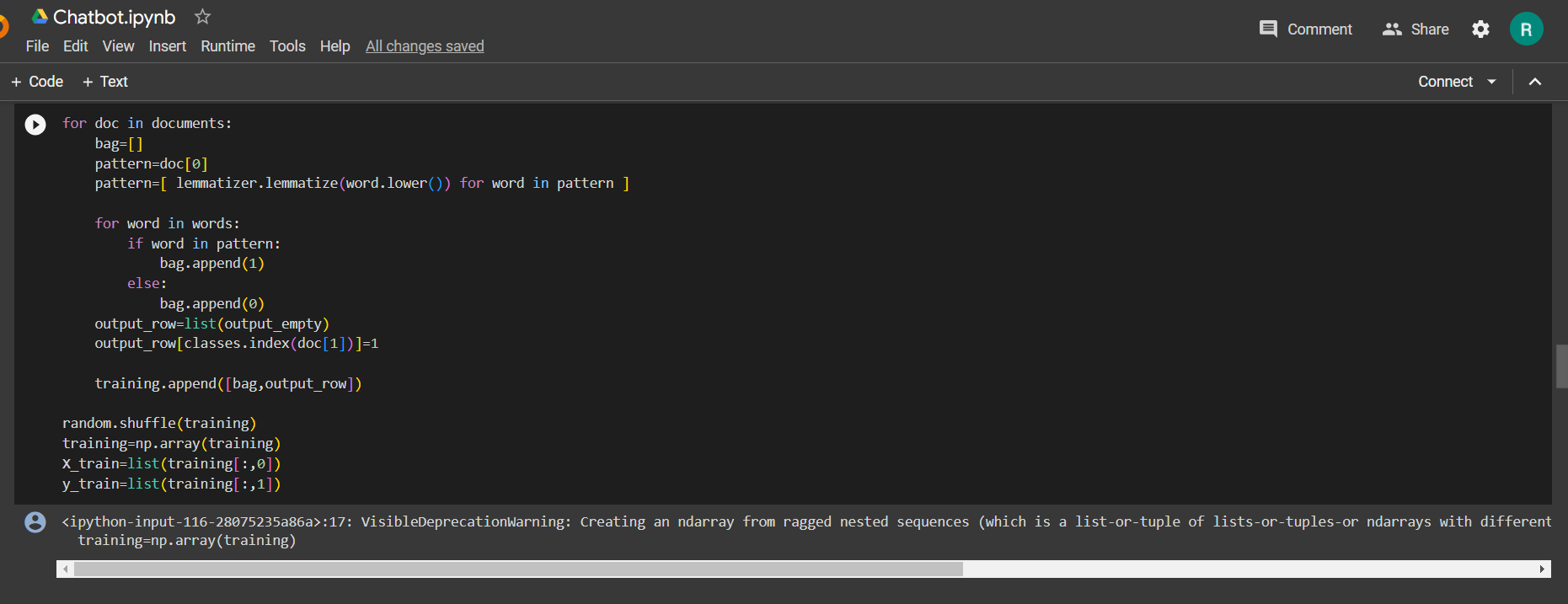


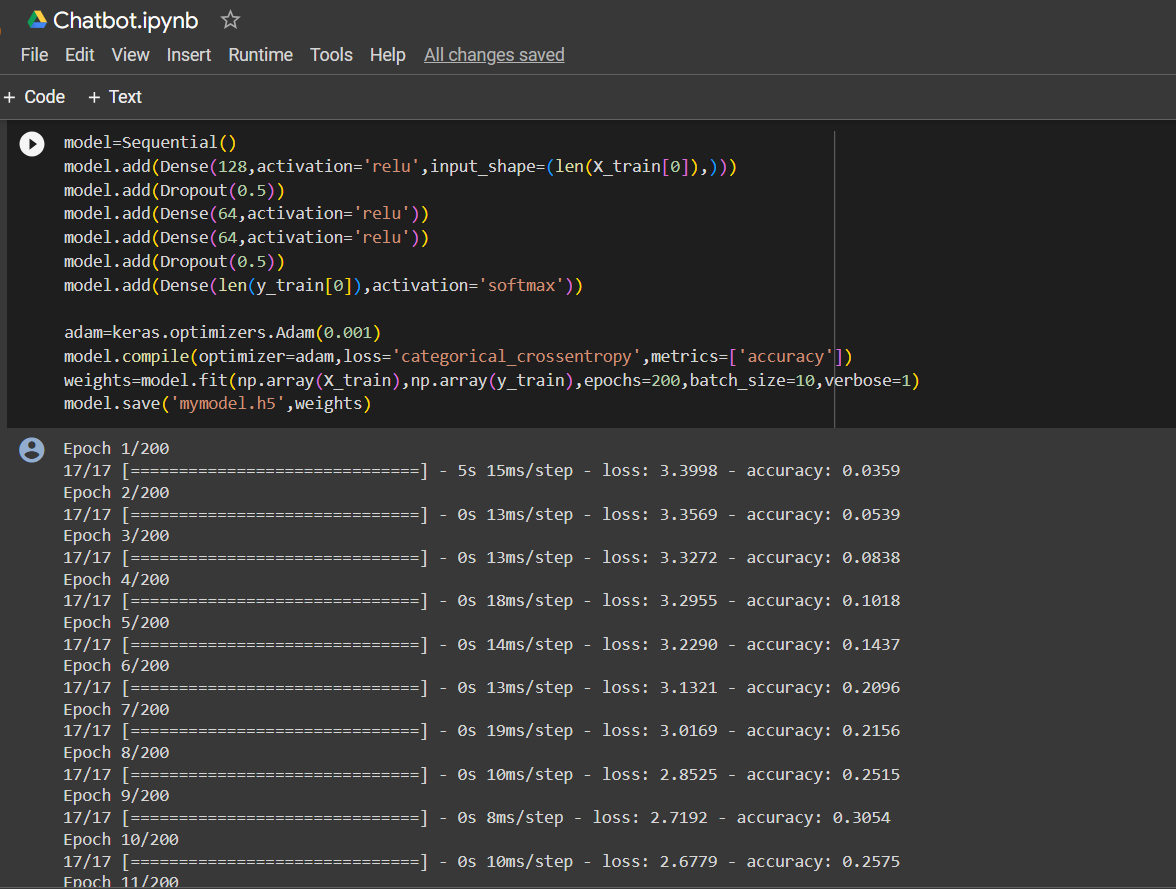


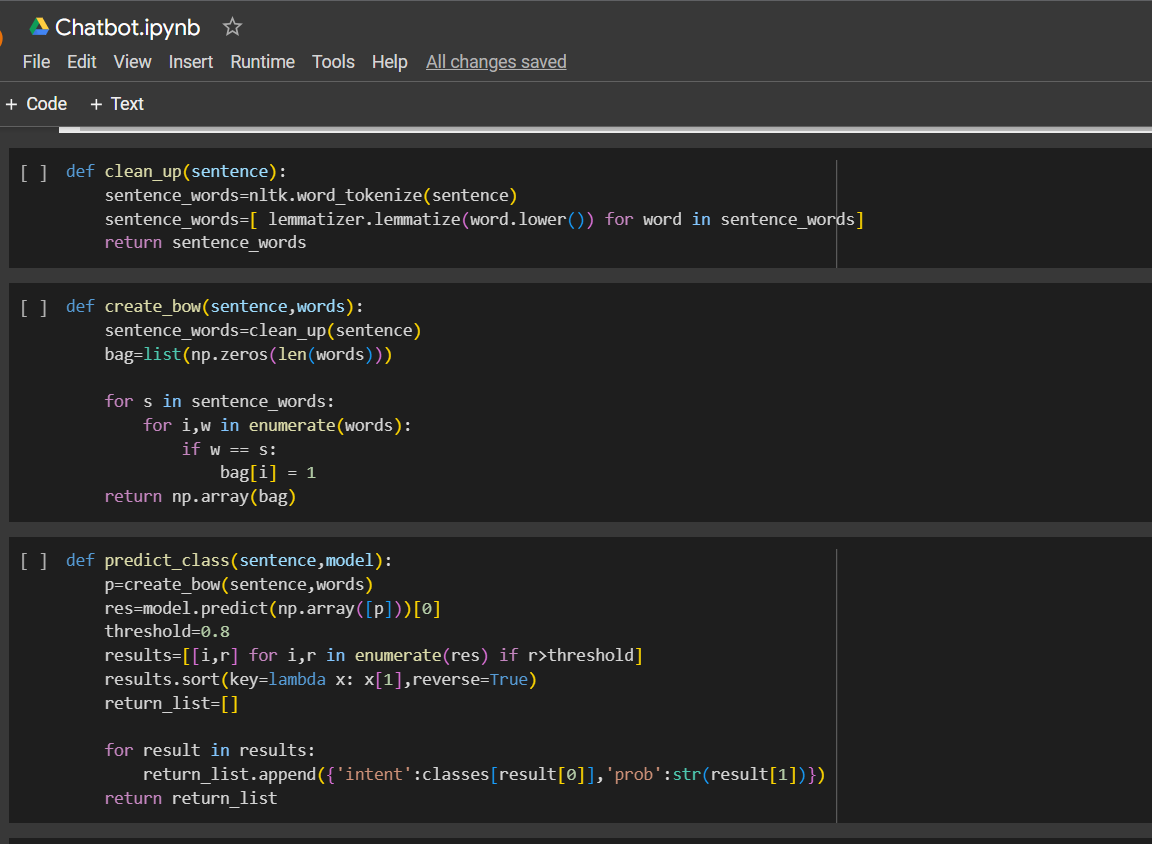


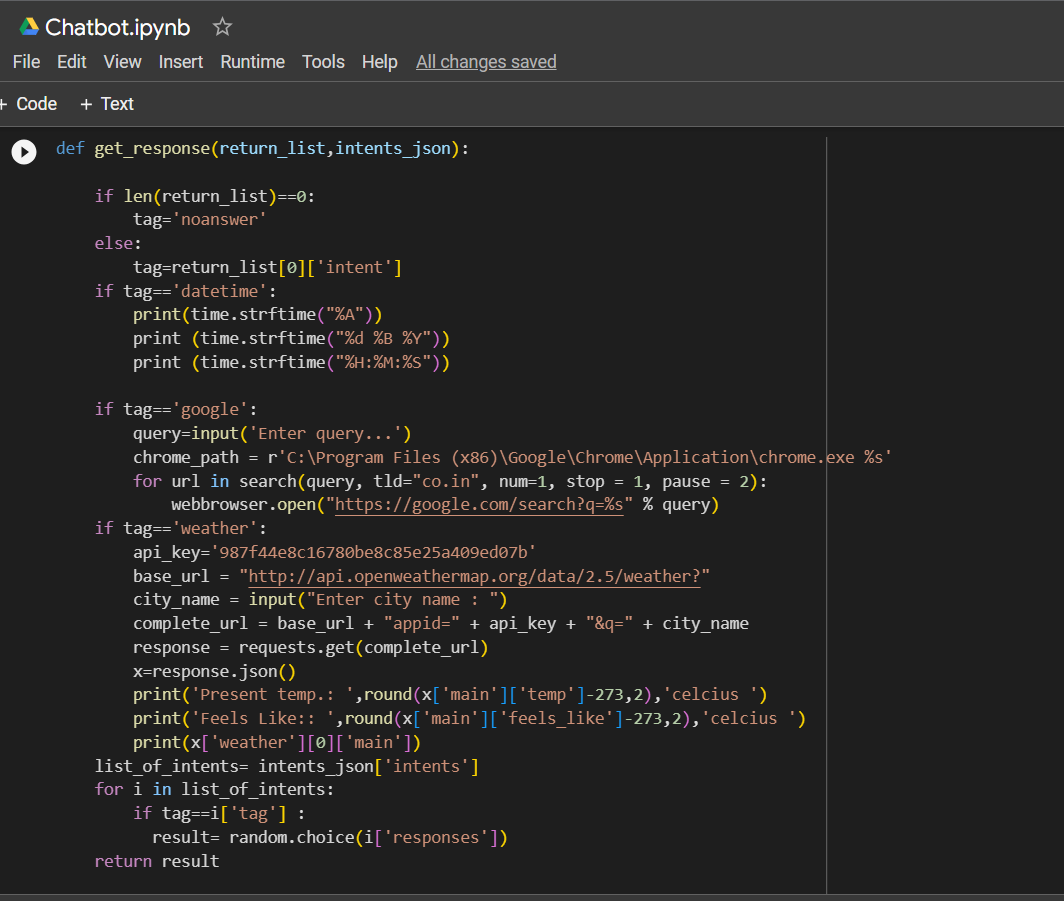


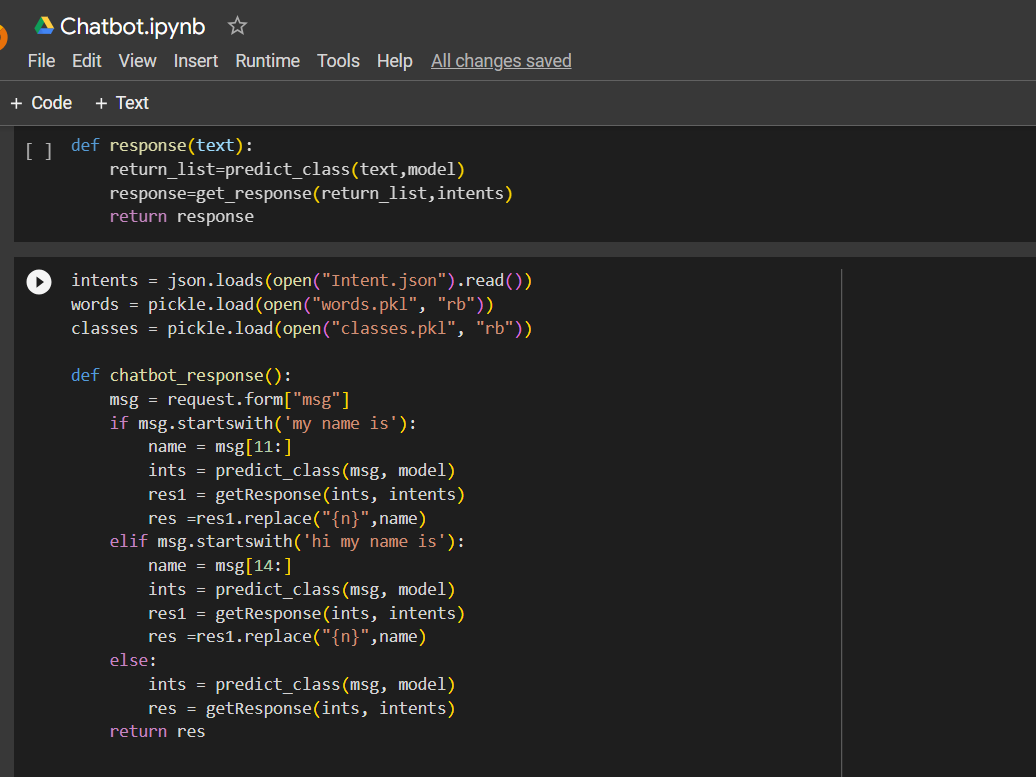


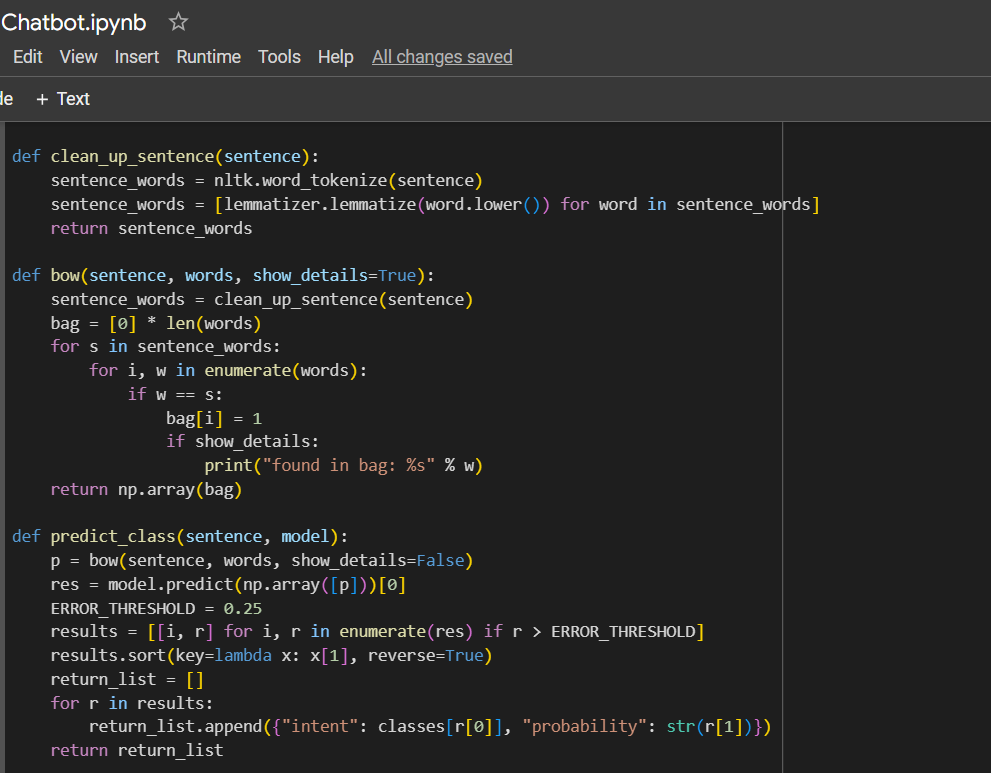


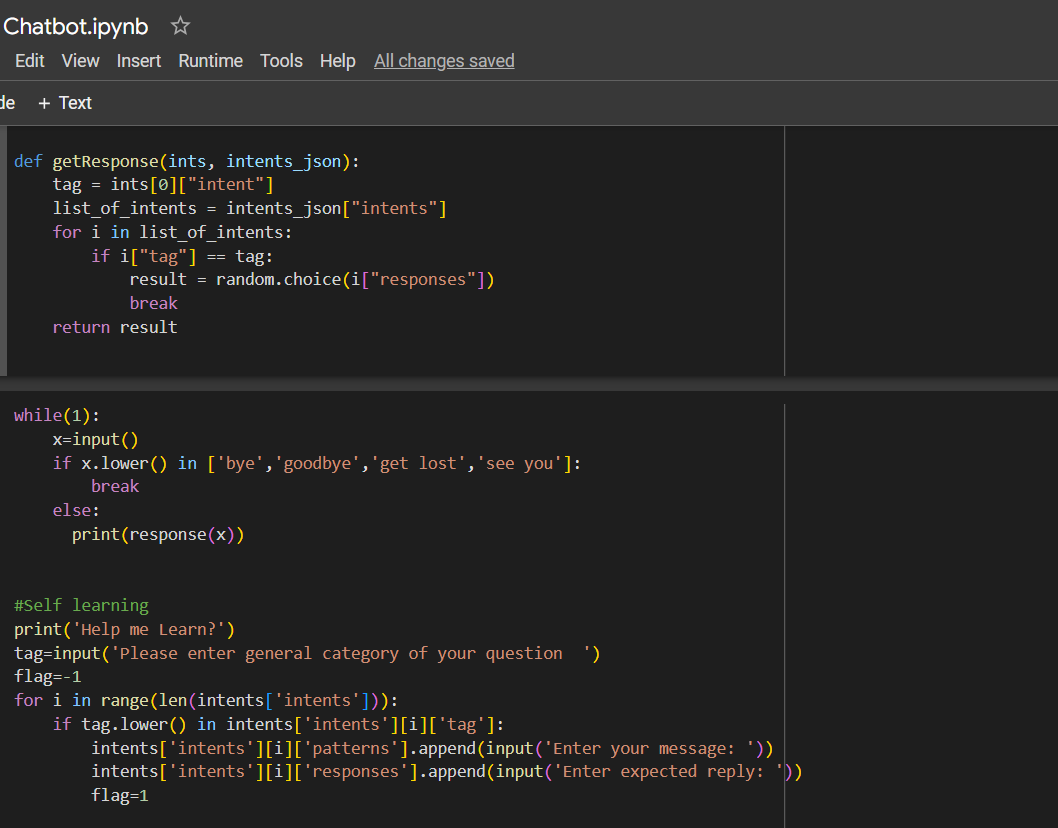


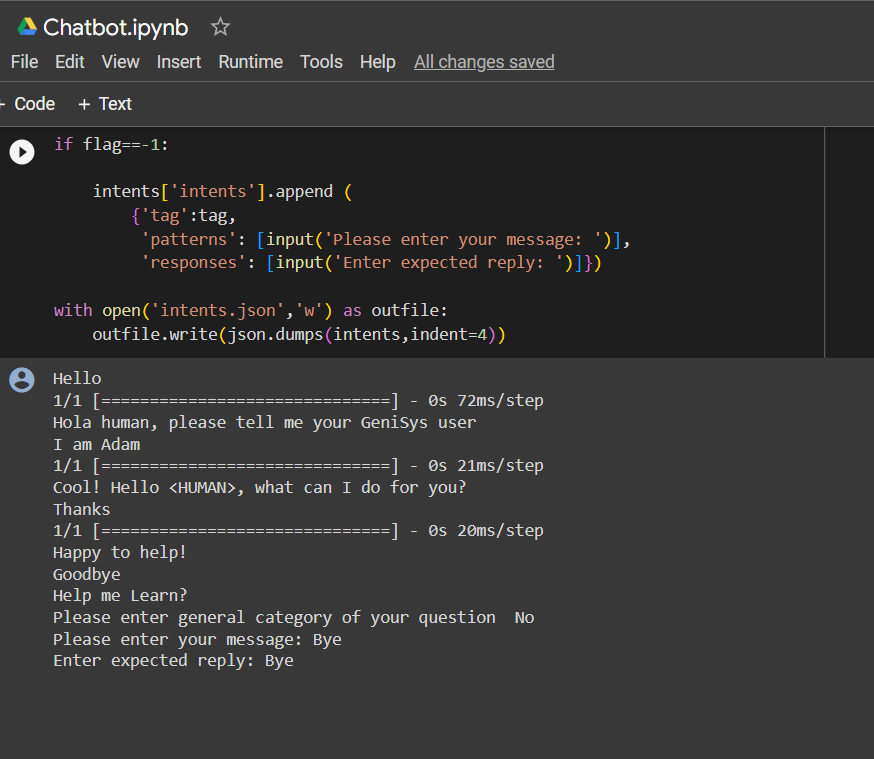












**Conclusion:**

Over the past decade, chatbots have evolved from menu-based workflows that worked through strenuous tree-based hierarchies to provide generic responses to the smart, AI-powered chatbots used today. AI continues to play a major role in chatbot development, with experts predicting that chatbots will take on more active roles in business. The world of chatbots is constantly evolving, with new techniques and tools being introduced regularly. Chatbots offer organizations, business, and individuals the opportunity to streamline processes, enhance customer experiences, and provide valuable services across various domains. So, chatbots can solve many challenges.