**AI BASED PROGNOSIS AND DIAGNOSIS NEGATION DETECTION IN HEALTH RECORDS USING NATURAL LANGUAGE PROCESSING**

**INT 400 – INTERNSHIP - 3**

**PROJECT REPORT**

***Submitted by***

**ARTHI SRI S - E0421055**

**SRIKANTH B K – E0421047**

**JASWANTH KUMAR – E0421045**

***In partial fulfilment for the award of the degree of***

**BACHELOR OF TECHNOLOGY**

**in**

**COMPUTER SCIENCE AND MEDICAL ENGINEERING**

**(Artificial Intelligence and Data Analytics)**

**Sri Ramachandra Faculty of Engineering and Technology**

**Sri Ramachandra Institute of Higher Education and Research, Porur, Chennai -600116**

**OCTOBER, 2023**

**AI BASED PROGNOSIS AND DIAGNOSIS NEGATION DETECTION IN HEALTH RECORDS USING NATURAL LANGUAGE PROCESSING**

**INT 400 – INTERNSHIP - 3**

**PROJECT REPORT**

***Submitted by***

**ARTHI SRI S - E0421055**

**SRIKANTH B K – E0421047**

**JASWANTH KUMAR – E0421045**

***In partial fulfilment for the award of the degree of***

**BACHELOR OF TECHNOLOGY**

**in**

**COMPUTER SCIENCE AND MEDICAL ENGINEERING**

**(Artificial Intelligence and Data Analytics)**

**Sri Ramachandra Faculty of Engineering and Technology**

**Sri Ramachandra Institute of Higher Education and Research, Porur, Chennai -600116**

**OCTOBER, 2023**

**BONAFIDE CERTIFICATE**

Certified that this project report **“AI BASED PROGNOSIS AND DIAGNOSIS NEGATION DETECTION IN HEALTH RECORDS USING NATURAL LANGUAGE PROCESSING”** is the bonafide record of work done by **“ ARTHI SRI S(E0421055), SRIKANTH B K(E0421047), JASWANTH KUMAR(E0421045)”** who carried out the internship work under my supervision.

**Signature of External Guide**

**(Signature with seal)**

**Signature of Internal Guide Signature of Head of the Department**

|  |  |
| --- | --- |
| **Dr.A.Suresh kumar**  **Assistant Professor,**  Department of Computer Science and Medical  Engineering  Sri Ramachandra Faculty of Engineering and Technology,  SRIHER, Porur, Chennai-600 116. | **Dr. A.K. Jayanthy**  **Professor and Head,**  Department of Computer Science and Medical  Engineering  Sri Ramachandra Faculty of Engineering and Technology,  SRIHER, Porur, Chennai-600 116. |

**Evaluation Date:**

**ACKNOWLEDGEMENT**

I express my sincere gratitude to our Head of the Department **Dr. A.K.Jayanthy** for their support and for providing the required facilities for this study.

I wish to thank my faculty supervisor**, Dr. A.Suresh Kumar** , Department of Computer Science and Medical Engineering, Sri Ramachandra faculty of Engineering and Technology and Project mentor, **Dr. Jayanthi G**, Cybersecurity department, Sri Ramachandra faculty of Engineering and Technology for extending help and encouragement throughout the project. Without his/her continuous guidance and persistent help, this project would not have been a success for me.

I am grateful to all the members of Sri Ramachandra Faculty of Engineering and Technology, my beloved parents and friends for extending the support, who helped us to overcome obstacles in the study.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
|  | **ABSTRACT** | 6 |
|  | **LIST OF FIGURES** | **7** |
| **1** | **INTRODUCTION** | **8** |
|  | 1.1 Objective | **9** |
|  | 1.2 Previous work  1.3 Project workflow | **11** |
| **2** | **LITERATURE REVIEW** | **13** |
| **3** | **TOOLS & TECHNOLOGIES** | **15** |
| **4** | **IMPLEMENTATION** | **29** |
| **5** | **RESULTS AND DISCUSSIONS** | **35** |
| **6** | **APPENDICIES** | **36** |
|  | Appendix-1: Code – Technical detail | **39** |
|  | Appendix-2: Screenshots | **40** |
| **7** | **REFERENCES** | **41** |
| **8** | **WORKLOG** | **42** |

**ABSTRACT**

Negation detection is a challenging task in natural language processing (NLP) and much research has been done in this field. Many researchers have developed negation detection tools, but these tools often have limitations, such as low accuracy or poor performance for specific types of negation.

We have read many research papers on negative detection and have identified some key challenges in developing a robust and accurate negative detection tool. One of the challenges faced is,

1. Ambiguity of negation: Negation can be expressed in many different ways in natural language, and it can be difficult to distinguish between negative and non-negative concepts.

We are currently developing a new negative detection tool that addresses some of the key challenges in this field.

Our tool has shown promising results in various negative detection tasks and we are working to further improve its performance.

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **S.no** | **Title** | **Page no.** |
| 1. | Work-Flow | 12 |
| 2. | Sample dataset | 29 |
| 3. | Negation detection result | 32 |
| 4. | Homepage | 33 |
| 5. | Integrated result | 33 |
| 6. | Model result | 39 |
| 7. | Final output | 39 |

**CHAPTER 1**

**INTRODUCTION**

The application of artificial intelligence (AI) to the interplay of computers and human language is known as natural language processing (NLP). Numerous industries have been transformed by it, including healthcare, where it is utilised to enhance diagnosis and prognosis. Negation detection is one of the major difficulties in NLP for healthcare. Accurately recognising instances in which patients downplay symptoms or illnesses is necessary for this. A complete picture of a patient's health may be obtained by clinicians thanks to negation detection, which is crucial for healthcare data analysis. For instance, a patient might assert, "I don't have any chest pain" or "I'm not having any breathlessness." Although these statements might appear simple, they can be challenging for computers to understand, especially when placed in the context of a complicated medical record.

Negation cue words like "not," "no," "never," and "cannot" are frequently used in negation detection systems. The breadth of the denial is then ascertained using a variety of methods, including rule-based systems, machine learning algorithms, and deep learning models. This entails identifying each concept that the negation cue word negates.

The negation cue word "not" denies the concept of "chest pain," for instance, in the statement "I don't have any chest pain." The negative does not, however, apply to the other ideas in the statement, such as "I" and "any."

Systems for detecting negatives are growing more accurate and sophisticated. This is brought on by a variety of elements, such as the creation of new machine learning algorithms and the accessibility of large datasets of labeled medical text

There are numerous crucial uses for negativity detection in the medical field. It can be utilised, for instance, to:

Boost the clinical decision support systems' (CDSSs') accuracy. Even if patients have not explicitly reported all of the pertinent symptoms, CDSSs can use negation detection to identify patients who are at risk of specific illnesses or consequences.

Create innovative tools for clinical research. Patients who fit the requirements for clinical trials or who have suffered particular side effects from drugs can be found via negation detection.

Boost the accuracy of medical data. Healthcare data can be cleaned and normalised via negativity detection, making analysis and research easier.

In conclusion, negation detection is a crucial NLP tool for healthcare component. Better diagnosis, prognosis, and therapy are made possible by it because it gives physicians and researchers a more thorough and accurate picture of the health of their patients.

* 1. **OBJECTIVE:**

For accurate symptom identification in medical records, negation detection is a critical technique. Healthcare data frequently contain negative symptoms, and conventional symptom recognition algorithms might not be able to recognise these correctly. This may result in incorrect diagnoses and unsuitable therapies.

By precisely recognising negated symptoms in a range of circumstances, AI-driven negation detection systems can get around the limits of conventional algorithms. The use of this data can help create a more thorough and accurate profile of a patient's health, which can result in more accurate diagnoses and prognoses.

Negation detection and symptom identification together could result in a considerable improvement in patient treatment. Identifying negated symptoms precisely allows clinicians to:

1. Lessen the possibility of misdiagnosis
2. Increase the precision of predictions
3. Create better treatment strategies
4. Spot patients who are susceptible to difficulties
5. Track the development of patients throughout time.

Examples of how negation detection can be utilised to enhance patient care are given below:

A patient who has had heart disease in the past presents with new chest discomfort and shortness of breath symptoms. The patient does, however, add, "I don't have any chest pain," and "I don't feel out of breath." Despite being expressed in a complex manner, an AI-driven negation detection method would be able to recognise these negated symptoms. Even though the patient denies experiencing chest pain, this information would assist the doctor realise that the patient is at a high risk for having a heart attack.

A patient is receiving cancer treatment. An AI-driven negation detection method is being used by the patient's oncologist to track their development. In the patient's most recent medical report, the algorithm discovers a new symptom that raises the possibility that the patient's cancer is progressing. The oncologist could modify the patient's treatment regimen in light of this knowledge.

A patient's chronic ailment is being watched over. The patient's primary care physician monitors any alterations in the patient's condition utilising an AI-driven algorithm for negation detection. The system finds a negated symptom in the patient's most recent medical report, pointing to potential improvement in the patient's condition. With this knowledge, the patient's primary care physician could schedule follow-up visits less frequently.

**1.2 WORK DONE:**

An interactive dashboard analytics deployed in a website which has the capacity of rendering a complete analysis of the clinical radiology reports (i.e.) given a report either by the patient or medical professional it can scrap all the necessary details related to it, analyzing the medical history of the patient or of all the patient under that certain doctor and the specified hospital. For the general public (patient) who lack the in depth medical knowledge of their diagnosis printed on the reports , a chatbot decided to be programmed to cater to the needs of question/ answering.

**Highlights of the work done:**

1. We have developed a WebApp that is connected with a Dashboard analytics tool and ChatBot assistance.
2. We created a dataset with our own fields even though we lacked a radiology report dataset in organised format as a CSV file. We are about to create a model to convert an unstructured PDF file into a CSV file for the dataset in the near future.
3. With the aid of this WebApp, doctors may swiftly display and assess patient data without always referring to the entire report.
4. This helps to automate Healthcare record management by the doctors.

**1.3 PROJECT WORKFLOW:**

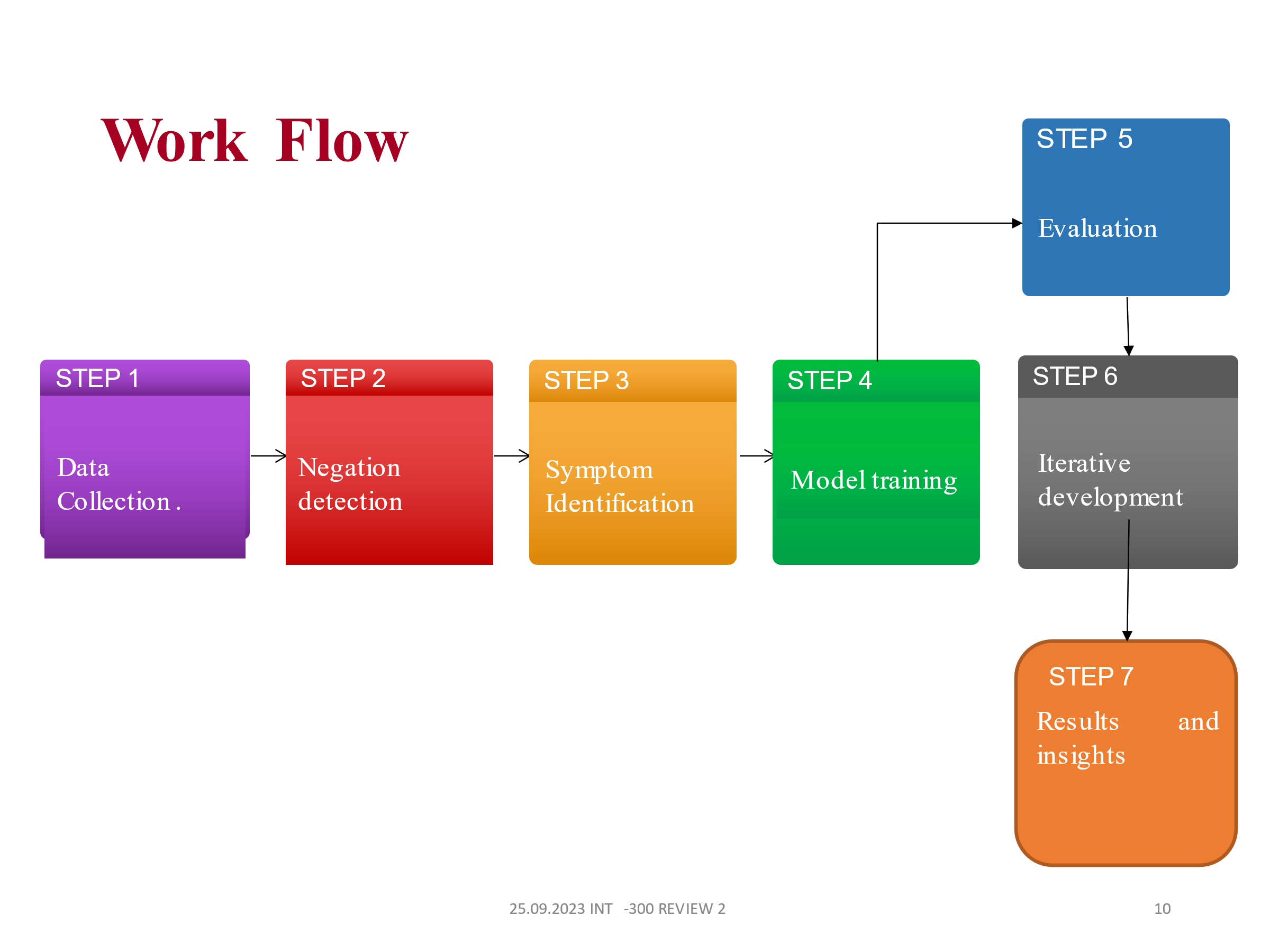


Figure 1.3.1 – Work-Flow

**CHAPTER 2**

**LITERATURE REVIEW**

There are many research papers and products that have done analysis using radiology reports trying to find negations which can facilitate the efficiency of prognosis and diagnosis in a patient’s treatment .

Here are the research papers that we used for our project’s references:

**SURVEY 1:**

**Title** - Negation Scope Detection in Clinical Notes and

Scientific Abstracts.

**Author** - Elena Sergeeva

**Year** - 2019

**Methods** - CNN, LSTM

**Dataset** - Cincinnati Children’s Hospital Medical Center

**Challenge** - LSTM-based system that makes use of syntactic and does not require any human-derived cue annotation at the time of inference. When the gold cue annotation is available, the system enables its use as an extra feature and produces outcomes on the BioScope corpus that are on par with cutting-edge techniques.

**Result** – Gold negation cue independent hierarchical LSTM based model that uses local syntactic features for negation scope detection in biomedical texts.

**SURVEY 2:**

**Title** – Negation detection for sentimental analysis

**Author** - Maite Taboada

**Year** - 2020

**Methods** - NegEX

**Dataset** - Spanish SFU Review corpus

**Challenge** -Words correctly identiﬁed as scope by the Spanish negation detector that are present in the SO-CAL dictionary, but are not sentiment words in the domain understudy.

**Result** – Automatically identiﬁes negation cues and their scope in Spanish review texts and we investigate whether accurate negation detection helps to improve the results of a sentiment analysis system.

**SURVEY 3:**

**Title** – Negation detection in Dutch clinical texts

**Author** - Bram van Es

**Year** - 2023

**Methods** – biLSTM, MedCAT, RoBERT

**Dataset** - Erasmus Medical Center Dutch clinical corpus

**Challenge** -The need for more representative training data and improved context handling for negation detection in electronic health records, as well as addressing data imbalance and ambiguity in clinical language.

**Result –** From 12419 medical terms in 5365 medical records, 1748 medical terms were marked as negated by annotators. Of these, 1687 concepts were identified by at least one of the negation detection models.

**CHAPTER 3**

**TOOLS AND TECHNOLOGIES**

**3.1 Visual Studio Code:**

Visual Studio Code (VS Code) is a lightweight open source code editor, developed by Microsoft. It is available for Windows, macOS and Linux and can be used to edit many programming languages, including Python, JavaScript, C++, C# and Java.

VS Code is based on the Electron framework, which means it's built using web technologies like HTML, CSS, and JavaScript. This makes it highly portable and extensible as developers can create their own extensions to add new features.

VS Code includes a number of features that make it a popular choice for scientific computing:

1. VS Code can be used to edit and run Jupyter notebooks, which are a popular way to share and collaborate on scientific code.
2. VS Code includes built-in support for Python, and extensions are also available for other scientific computing libraries such as NumPy, Pandas, and Matplotlib.
3. VS Code offers code completion and syntax highlighting for many programming languages, making it easier to write and debug code.
4. VS Code includes a built-in debugger that allows developers to step through code, inspect variables, and view the call stack.
5. VS Code integrates with Git and other version control systems, making it easy to track code changes and collaborate with others.

In addition to these features, VS Code is also highly customizable. Users can change the editor's theme, layout, and keyboard shortcuts to suit their own preferences. There are also a large number of extensions available that can be used to add new functionality.

Here are some examples of how to use VS Code for scientific computing:

1. Develop and execute Python scripts:
2. VS Code can be used to develop and run Python scripts for a variety of scientific tasks, such as data analysis, machine learning, and numerical simulations.
3. Create and edit Jupyter notebooks:
4. VS Code can be used to create and edit Jupyter notebooks, which are a popular way to share and collaborate on scientific code.
5. Debugging scientific code:
6. VS Code's built-in debugger can be used to debug scientific code written in Python or other programming languages.
7. Overall, VS Code is a powerful and flexible code editor that is well-suited for scientific computing. It is lightweight, extensible, and includes several features that make it easier to write, debug, and run scientific code.

**3.2 Flask and Flask API:**

Flask is a lightweight open source web framework written in Python. It is based on the Werkzeug WSGI toolkit and the Jinja2 templating engine. Flask is designed to be easy to learn and use, and is a good choice for building simple web applications and APIs.

Flask is a microframework, which means it doesn't include much built-in functionality.

Flask is a Python web framework that can be used to deploy deep learning models.

Flask is a good choice for deploying deep learning models because it is easy to learn and use, and is scalable and performant.

To deploy a deep learning model with Flask:

1. Export the deep learning model from the framework you used to develop it.
2. Create a Flask application.
3. Load the deep learning model into the Flask application.
4. Create a route that processes requests to the deep learning model.
5. Deploy Flask application to web server.
6. The Flask application creates a route named /predict. This route accepts a JSON request with input data for the deep learning model.
7. The Flask app then makes a prediction from the deep learning model and returns it as a JSON response.

This is just a simple example but it shows how Flask can be used to deploy deep learning models. Flask can be used to deploy more complex deep learning models as well as deploy deep learning models into production environments.

**3.3 Python:**

Python is a general-purpose programming language that is widely used in deep learning. It is a popular choice for deep learning because it is easy to learn and use, and it has a number of powerful libraries and frameworks available, such as TensorFlow, PyTorch, and Keras.

Python is also a good choice for deep learning because it is flexible and extensible. This means that it can be used to implement a wide variety of deep learning algorithms, and it can be used to build complex deep learning models.

Python is used to develop and train deep learning models for a variety of tasks, such as image classification, object detection, and natural language processing.

Python is used to build deep learning applications for a variety of purposes, such as developing self-driving cars, creating medical diagnostic tools, and building chatbots.

Python is widely used in deep learning research to develop new deep learning algorithms and to evaluate the performance of deep learning models.

* 1. **Python Libraries:**

1. **Negex:**

Negex is a rules-based denier identification system developed by the National Library of Medicine (NLM). It is used to identify rejected concepts in clinical writing.

Negex uses a set of rules to define negative concepts. The rules are based on the following:

1. Negative words:

Negex defines a set of negative words, such as “don't,” “don't,” and “don't.”

1. Negative scope:

Negex determines the scope of negation by identifying words affected by the negation.

For example, the following sentence contains a negative concept:

The patient has no fever.

In this sentence, the negative word is “full” and the scope of the negative is “fever”. This means the patient does not have a fever. Negex is a useful tool for identifying negative concepts in clinical writing. This information can be used for a variety of tasks, such as developing clinical decision support systems and improving the accuracy of natural language processing systems.

Here are some examples of how to use Negex:

1. Identify canceled drugs in prescriptions.
2. 2.Identify exempt symptoms in the patient's medical record.
3. Identify invalid results in medical imaging reports. Negex is a powerful tool for identifying denied concepts in clinical texts. It can be used to improve the accuracy of natural language processing systems and develop clinical decision support systems that can help healthcare professionals make better decisions.
4. **UMLS:**

The Unified Medical Language System “UMLS” is a comprehensive biomedical dictionary and thesaurus developed by the National Library of Medicine (NLM). It is used to normalize and standardize biomedical terminology. UMLS includes more than 3 million concepts and 11 million terms from a variety of biomedical sources, including medical dictionaries, textbooks, and clinical databases.

UMLS can be used to analyze medical reports in many ways, including:

1. **Identify medical concepts:**

UMLS can be used to identify medical concepts in medical reports. This could be useful for developing clinical decision support systems and improving the accuracy of natural language processing information.

1. **Extract information from medical reports:**

UMLS can be used to extract information from medical reports, such as medications, symptoms, and findings. This information can be used to develop clinical decision support systems and improve the efficiency of the healthcare process.

1. **Standardization of biomedical terminology:**

UMLS can be used to standardize biomedical terminology in medical reports and hence this can be useful for improving the interoperability of medical records systems and facilitating communication between healthcare professionals.

Here are some examples of how UMLS can be used to analyze medical reports:

A clinical decision support system can use UMLS to identify canceled drugs in a prescription and alert a healthcare professional if a patient is allergic to any drug. Natural language processing systems can use UMLS to extract information from a patient's medical record, such as medications, symptoms, and results.

This information can then be used to create a summary of the patient's medical history or to identify possible drug interactions.

Medical records systems can use UMLS to standardize biomedical terminology in medical reporting.

This will make it easier to search for medical records and compare medical records from different healthcare organizations.

UMLS is a powerful tool that can be used to improve the accuracy and efficiency of medical report analysis. It is a valuable resource for healthcare professionals, researchers, and developers of clinical decision support systems and natural language processing systems.

1. **SciSpacy:**

SciSpacy is a Python library for natural language processing (NLP) tasks in the scientific domain. It is built on top of the Spacy library. SciSpacy provides a number of features for NLP tasks such as tokenization, lemmatization, part-of-speech tagging, named entity recognition, and dependency parsing.

SciSpacy also includes a number of features that are specifically designed for NLP tasks in the scientific domain, such as:

1. Support for biomedical terminology:

SciSpacy includes support for biomedical terminology from the Unified Medical Language System (UMLS). This makes it easier to identify and process biomedical concepts in scientific text.

1. Support for scientific abbreviations:

SciSpacy includes support for scientific abbreviations. This makes it easier to identify and process scientific abbreviations in scientific text.

1. Support for scientific symbols:

SciSpacy includes support for scientific symbols. This makes it easier to identify and process scientific symbols in scientific text.

SciSpacy can be used for a variety of tasks in the scientific domain, such as:

1. Extracting information from scientific papers:

SciSpacy can be used to extract information from scientific papers, such as the abstract, introduction, methods, results, and discussion sections. This information can then be used to generate summaries of scientific papers or to develop tools that can help researchers find relevant information in the scientific literature.

1. Developing clinical decision support systems:

SciSpacy can be used to develop clinical decision support systems that can help healthcare professionals make better decisions. For example, a clinical decision support system could use SciSpacy to identify negated medications in a prescription order and warn the healthcare professional if the patient is allergic to any of the medications.

1. Building chatbots that can answer questions about scientific topics:

SciSpacy can be used to build chatbots that can answer questions about scientific topics. This could be useful for developing educational tools for students or for developing tools that can help patients learn more about their health conditions.

1. **NegSpacy:**

NegSpacy is a Python library for negator identification and negation scope resolution. It is built on top of the Spacy library. NegSpacy uses a rule-based approach to identify negators and to determine the scope of negation.

NegSpacy can be used for a variety of tasks, such as:

1. Identifying negated concepts in clinical text:

NegSpacy can be used to identify negated concepts in clinical text, such as negated medications, negated symptoms, and negated findings. This information can be used to develop clinical decision support systems and to improve the accuracy of natural language processing systems.

1. Identifying negated concepts in scientific text:

NegSpacy can be used to identify negated concepts in scientific text, such as negated results and negated conclusions. This information can be used to develop tools that can help researchers find relevant information in the scientific literature or to develop tools that can help students learn about scientific concepts.

1. Identifying negated concepts in news articles and social media posts:

NegSpacy can be used to identify negated concepts in news articles and social media posts. This information can be used to develop tools that can help people understand the news and social media more accurately.

1. **SpaCy:**

SpaCy is a Python library for general-purpose natural language processing (NLP) tasks. It can be used for tasks such as tokenization, lemmatization, part-of-speech tagging, named entity recognition, and dependency parsing.

SpaCy is a powerful and versatile library that can be used for a variety of tasks in different domains, including:

1. Text classification:

SpaCy can be used to classify text into different categories, such as news articles, blog posts, and social media posts.

1. Question answering:

SpaCy can be used to develop question answering systems that can answer questions about a given text.

1. Machine translation:

SpaCy can be used to develop machine translation systems that can translate text from one language to another.

1. Chatbots:

SpaCy can be used to develop chatbots that can interact with users in a natural language.

SpaCy is a popular choice for NLP tasks because it is easy to use and has a wide range of features. It is also well-documented and has a large and active community.

Overall, SciSpacy, NegSpacy, and SpaCy are all powerful libraries for natural language processing tasks. SciSpacy is specifically designed for tasks in the scientific domain, NegSpacy is specifically designed for negator identification and negation scope resolution, and SpaCy is a general-purpose NLP library.

1. **NLTK:**

Natural Language Toolkit (NLTK) is a Python library for natural language processing (NLP). It provides a variety of tools for tasks such as tokenization, rooting, lexicalization, parsing, and sentiment analysis.

NLTK is a popular choice for NLP because it is easy to use and has many features. It is also well documented and has a large and active community.

Here are some benefits of using NLTK for NLP:

1. Easy to use:

NLTK is a Python library, making it easy to use and learn. It also has a simple and intuitive API.

1. Many features:

NLTK offers a variety of tools for different NLP tasks. This makes it a one-stop shop for many NLP needs.

1. Good documentation:

NLTK is fully documented, with comprehensive instructions and manuals. This makes it easier for you to get started with NLTK and learn its various features.

1. Large and active community:

NLTK has a large and active community of users and developers. This means there are plenty of resources available to help you learn and use NLTK, and chances are your questions will be answered if you run into any problems.

Here are some examples of how NLTK can be used for NLP:

1. Text classification:

NLTK can be used to classify text into different categories, such as articles, blog posts, and social media posts.

1. Answer the questions:

NLTK can be used to develop a question answering system capable of answering questions about a given text.

1. Automatic translation:

NLTK can be used to develop machine translation systems capable of translating text from one language to another.

1. Chatbots:

NLTK can be used to develop chatbots that can interact with users using natural language.

1. **TENSORFLOW:**

TensorFlow is an open source software library for numerical computation using data flow graphs. It is used for machine learning and deep learning applications.

TensorFlow is a popular choice for developing deep learning models because it:

1. Easy to use:

TensorFlow has a simple and intuitive API, making it easy to learn and use.

1. Strong:

TensorFlow is a powerful library that can be used to develop a variety of deep learning models, from simple to complex models with millions of parameters.

1. Can be expanded:

TensorFlow can be adapted to run on a variety of hardware platforms, including CPUs, GPUs, and TPUs. This makes it possible to train and deploy large and complex deep learning models.

Here are some benefits of using TensorFlow to develop deep learning models:

1. Flexible:

TensorFlow is a flexible framework that can be used to implement many types of deep learning architectures. This allows custom models to be developed to suit specific needs.

1. Ability of extension:

TensorFlow is an extensible framework that can be extended with custom operators and libraries. This allows new features to be added to TensorFlow.

1. Contribution to the community:

TensorFlow has a large and active community of users and developers. This means there are plenty of resources available to help you learn and use TensorFlow, and chances are your questions will be answered if you run into any problems.

TensorFlow is used by many different companies and organizations to develop deep learning models for a variety of applications, including:

1. Image classification:
   1. TensorFlow is used to develop deep learning models for image classification tasks, such as identifying objects in images and classifying images into different categories.
2. Object detection:
   1. TensorFlow is used to develop deep learning models for object detection tasks, such as detecting objects in images and videos.
3. Natural language processing:
   1. TensorFlow is used to develop deep learning models for natural language processing tasks, such as machine translation, text summarization, and question answering.
4. Voice recognition:
   1. TensorFlow is used to develop deep learning models for speech recognition tasks, such as converting audio to text.
5. HTML, CSS AND JAVASCRIPT:
   1. Develop deep learning models with HTML, CSS and JavaScript.
   2. HTML, CSS, and JavaScript are not typically used to develop deep learning models.
   3. However, they can be used to create a user interface for a deep learning model or to visualize the results of a deep learning model.
   4. For example, you can use HTML to create a form for users to input data into a deep learning model. You can use CSS to style the form and make it visually appealing.
   5. You can use JavaScript to handle user interactions, such as submitting a form or displaying the results of a deep learning model.

Here are some specific examples of how HTML, CSS, and JavaScript can be used to develop deep learning models:

1. A web application that allows users to upload images and receive predictions from a deep learning image classification model.
2. A web application that allows users to enter text and receive predictions from a deep learning text classification model.
3. A web application that visualizes the results of a deep learning model, such as a predictive heat map for a given image.

HTML, CSS, and JavaScript can be used to develop user interfaces for deep learning models or to visualize the results of deep learning models. Flask is a Python web framework that can be used to deploy deep learning models.

**3.5 Work done’s tools used:**

1. Flask API – We used Flask API to create a Web-App with Chatbot and deployed dashboard analytics.
2. Power BI – It’s a data visualisation tool for getting insights from our dataset as dashboard which provides interactive analytic charts.
3. NLP for chatbot – Numerous industry applications, including chatbots, text classification, summarization, and other use cases, use natural language processing. In our project, we created a two-person ChatBot using NLP to assist doctors in understanding the Dashboard rapidly.

**CHAPTER 4**

**IMPLEMENTATION**

**4.1 Data collection:**

The datasets that we required were radiology reports in pdf format. These included MRI reports, Ultrasound reports, CT reports and X-Ray reports. These reports were collected from various healthcare storage websites which provides sample electronic health reports.

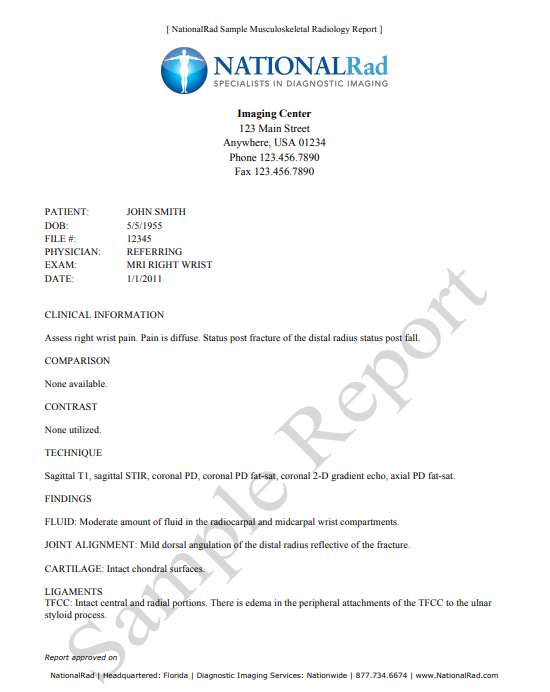
****

Figure. 4.1.1 - Sample Dataset

**4.2 Data preprocessing:**

1. **Tokenization:**

Start by breaking the text into smaller units, usually words or subwords. This process is called tokenization. Tokenization helps to identify individual words, which are essential for negation detection.

1. **Stopword Removal:**

Remove common stopwords like "a," "an," "the," "in," "on," etc. These words don't carry significant meaning for negation detection and can be safely excluded from analysis.

1. **Punctuation Removal:**

Remove punctuation marks and special characters, as they can interfere with negation detection. For example, you may want to remove commas, periods, exclamation marks, and question marks.

1. **Stemming or Lemmatization:**

Apply stemming or lemmatization to reduce words to their root form. For example, "running" and "ran" could be reduced to "run." This helps in recognizing negations consistently.

1. **Negation Handling:**

After marking negations, consider the effect of negations on the meaning of the text. You can use algorithms or rules to flip the sentiment of words within the negation scope. For example, "not good" becomes "bad," "not happy" becomes "unhappy."

**4.3 Model building:**

Deep learning models can be effectively used for negation detection in text. One popular approach is to employ Recurrent Neural Networks (RNNs)

1. **Embedding Layer:**

Use word embeddings (e.g., Word2Vec, GloVe, or pretrained embeddings like BERT) to convert words into continuous vector representations. These embeddings capture the semantic meaning of words, which is crucial for negation detection.

1. **Model Architecture:**

Typically, one or more RNN layers are stacked to capture sequential information in the text. Common RNN units used in this context are Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU) cells. These units are designed to address the vanishing gradient problem and effectively capture long-range dependencies in the data.

1. **Hidden State Output:**

The hidden state of the RNN at each time step is computed and can be used for further analysis. The hidden state encodes information about the context surrounding each word.

1. **Sequence Labeling Layer:**

For negation detection, you can use a sequence labeling layer, typically implemented as a dense (fully connected) layer with a softmax activation function. This layer assigns a label to each word in the sequence, indicating whether the word is inside or outside the scope of a negation.

1. **Output Layer:**

The final output layer generates predictions. In the case of negation detection, you typically have two classes: "Inside Negation" and "Outside Negation." The softmax function is used to produce probability scores for each class.

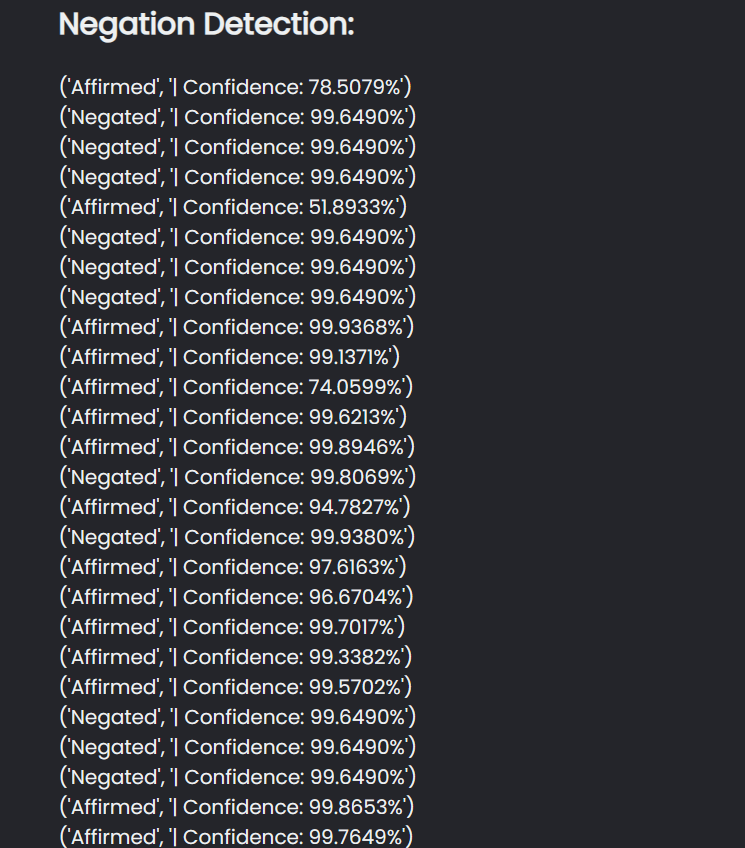


Figure 4.3.1 – Negation detection result

**4.4 Deployment:**

In our project, we deployed a negation detection model using Flask, a popular Python web framework. Through a user-friendly web interface, users can input text, and our model detects negations, allowing them to understand the impact of negative statements. The process involved creating an HTML form for user input, integrating the model into the Flask application, and defining routes for user interaction. This deployment enhances the accessibility and utility of our negation detection capabilities.



Figure 4.4.1 – Homepage

**4.5 Integration with previous work:**

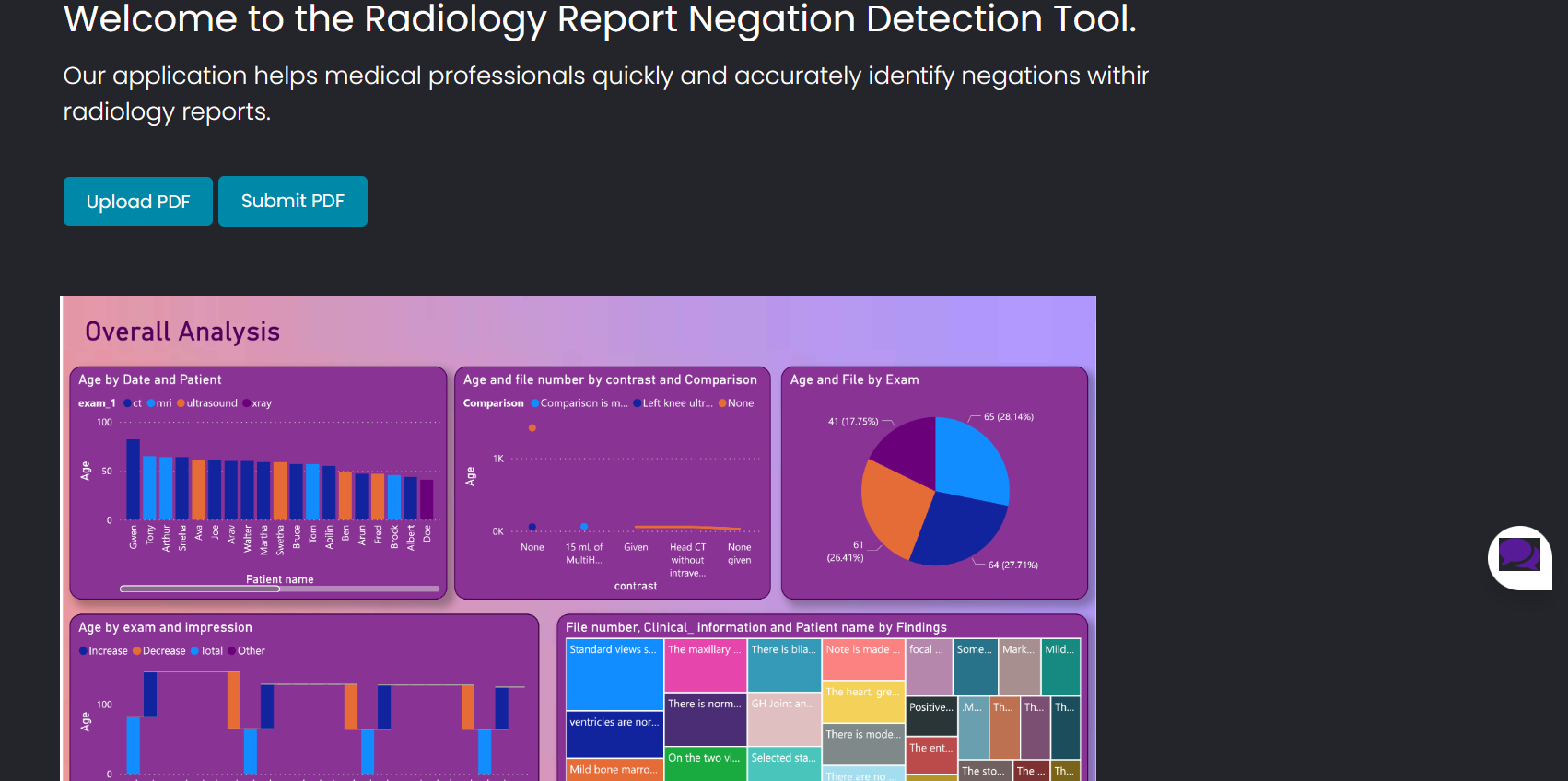
****

Figure 4.4.2 – Integrated work

**CHAPTER 5**

**RESULTS AND DISCUSSION**

Our project is to deploy a negation detection model through a Flask-based web interface represents a significant step toward facilitating the understanding of negations in Radiology reports data. We have successfully harnessed the capabilities of Deep learning to create a tool that can benefit users across various domains.

By enabling users to input text and promptly identify the presence and scope of negations, our application contributes to more accurate diagnosis of the patient's clinical data

While we have achieved a successful deployment, ongoing work is needed to fine-tune the model for higher accuracy, enhance the user interface for a better experience, and ensure robust performance as user demand grows.

In the broader context, this project demonstrates the value of combining deep learning and web development to create practical tools that address real-world challenges in detecting negations from Radiology reports. It is a foundation upon which further improvements and innovations can be built, and it underscores the relevance of NLP and AI in modern applications.

**APPENDICES**

**APPENDIX-1**

**app.py**:

def split\_text\_into\_sentences(text):

# Split text into sentences using a simple rule based on periods, question marks, and exclamation marks

sentences = re.split(r'[.!?]', text)

sentences = [s.strip() for s in sentences if s.strip()]

return sentences

def load\_from\_file(file\_path):

with open(file\_path, 'r') as file:

data = [line.strip() for line in file.readlines()]

return data

model = tf.keras.models.load\_model('Chatbot/models/negation\_model.h5')

# Rename the predict function to avoid the naming conflict

def predict\_sentiment(new\_sentence):

sentences = load\_from\_file('Chatbot/models/sen.txt')

tokenizer = Tokenizer() # Initialize a tokenizer

tokenizer.fit\_on\_texts(sentences) # Fit the tokenizer on the input sentences to build vocabulary

sequences = tokenizer.texts\_to\_sequences(sentences) # Convert input sentences to sequences of integers

max\_sequence\_length = max(len(seq) for seq in sequences) # Find the maximum sequence length

new\_sequence = tokenizer.texts\_to\_sequences([new\_sentence])

new\_X = pad\_sequences(new\_sequence, maxlen=max\_sequence\_length)

prediction = model.predict(new\_X)

if prediction[0][0] >= 0.5:

return "Affirmed", f"| Confidence: {100 \* prediction[0][0]:.4f}%"

else:

return "Negated", f"| Confidence: {100 - prediction[0][0]:.4f}%"

@app.get("/")

def index\_get():

return render\_template("base.html")

**base.html:**

<!DOCTYPE html>

<html>

<head>

<title>PDF Text Extraction & Negation Detection</title>

<link href="../static/style.css" rel="stylesheet">

</head>

<body>

<section>

<div class="container">

<div class="chatbox">

<div class="chatbox\_\_support">

<div class="chatbox\_\_header">

<div class="chatbox\_\_image--header">

<img src="https://img.icons8.com/color/48/000000/circled-user-female-skin-type-5--v1.png" alt="image">

</div>

<div class="chatbox\_\_content--header">

<h4 class="chatbox\_\_heading--header">Chat support</h4>

<p class="chatbox\_\_description--header">Hi. My name is Art. How can I help you?</p>

</div>

</div>

<div class="chatbox\_\_messages">

<div></div>

</div>

<div class="chatbox\_\_footer">

<input type="text" placeholder="Write a message...">

<button class="chatbox\_\_send--footer send\_\_button">Send</button>

</div>

</div>

<div class="chatbox\_\_button">

<button><img src="{{ url\_for('static', filename='images/chatbox-icon.svg') }}" /></button>

</div>

</div>

</div>

<div class="dashboard">

<img src="../static/images/Internship-0005.png" width="900px">

</div>

</section>

<script>

$SCRIPT\_ROOT = {{ request.script\_root|tojson }};

</script>

<script type="text/javascript" src="{{ url\_for('static', filename='app.js') }}"></script>

</body>

</html>

**model.ipynb:**

# Tokenize and preprocess the data

tokenizer = Tokenizer() # Initialize a tokenizer

tokenizer.fit\_on\_texts(sentences) # Fit the tokenizer on the input sentences to build vocabulary

sequences = tokenizer.texts\_to\_sequences(sentences) # Convert input sentences to sequences of integers

max\_sequence\_length = max(len(seq) for seq in sequences) # Find the maximum sequence length

X = pad\_sequences(sequences, maxlen=max\_sequence\_length) # Pad sequences to have a uniform length

y = np.array(labels) # Convert labels to a NumPy array

# Define the RNN model

vocab\_size = len(tokenizer.word\_index) + 1 # Get the vocabulary size

embedding\_dim = 32 # Dimensionality of word embeddings

rnn\_units = 64 # Number of units in the RNN layer

model = Sequential() # Initialize a sequential model

model.add(Embedding(input\_dim=vocab\_size, output\_dim=embedding\_dim, input\_length=max\_sequence\_length)) # Add an embedding layer

model.add(SimpleRNN(units=rnn\_units)) # Add a SimpleRNN layer

model.add(Dense(1, activation='sigmoid')) # Add a dense output layer with sigmoid activation for binary classification

# Compile the model

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

# Train the model (replace with your data)

model.fit(X, y, epochs=10, batch\_size=2) # Train the model for 10 epochs with a batch size of 2

model.save('models/negation\_model.h5') # Save the trained model to a file

**APPENDIX -2: Screenshots**

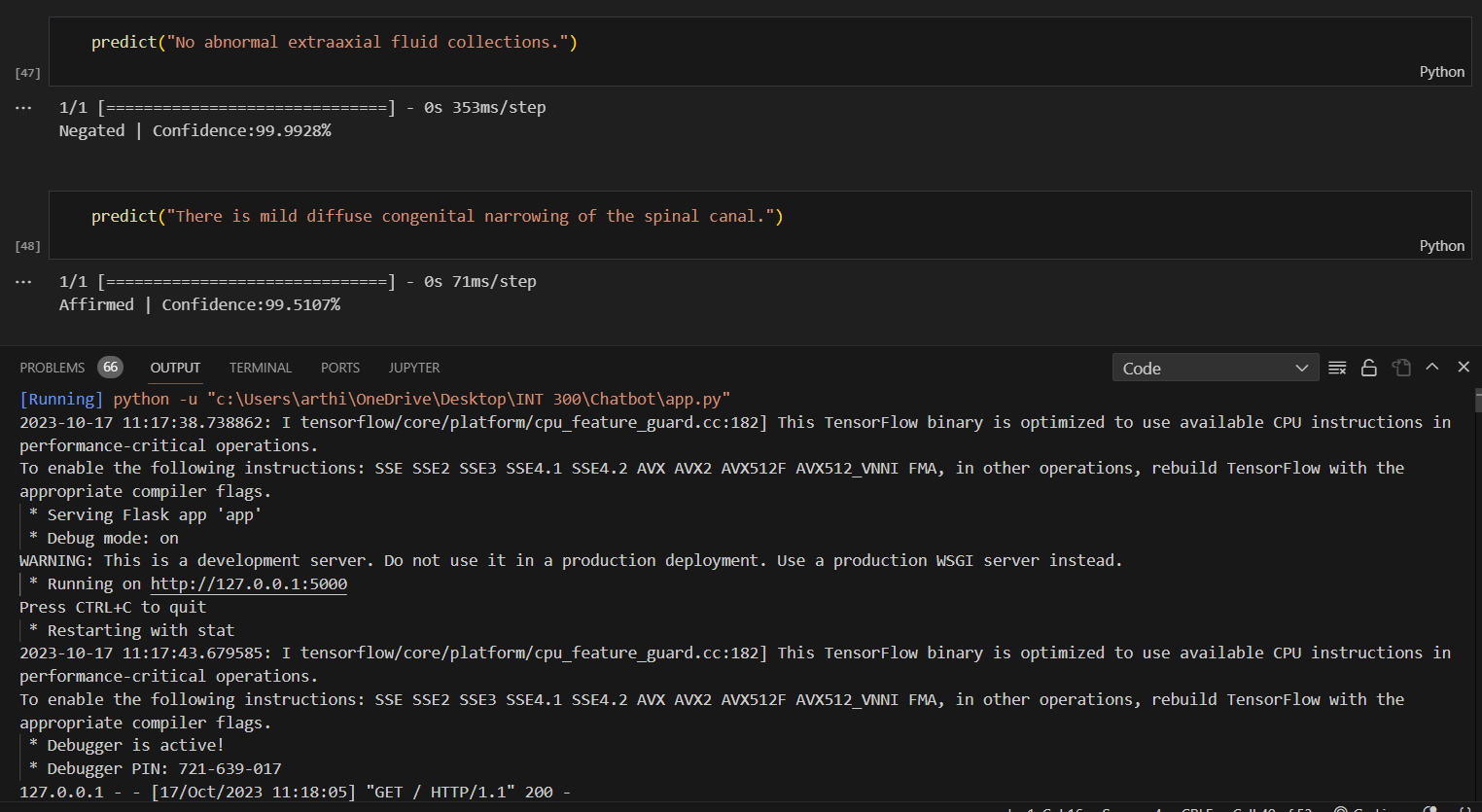
****

Figure 5.2.1 – Model result



Figure 5.2.2 – Final Output

**REFERENCES**

**Journal References:**

1. Elena Sergeeva, Henghui Zhu, Peter Prinsen, Amir Tahmasebi (2019), “Negation Scope Detection in Clinical Notes and Scientific Abstracts”, AMIA Jt Summits Translational Science Proceedings, 212-221, 25 pages
2. Bram van Es, Leon C. Reteig, Sander C. Tan, Marijn Schraagen, Myrthe M. Hemker, Sebastiaan R. S. Arends, Miguel A. R. Rios & Saskia Haitjema (2023), “Negation detection in Dutch clinical texts: an evaluation of rule-based and machine learning methods”, BMC Bioinformatics,10, 20 pages
3. Maite taboda, Maria Teresa Martín-Valdivia, Noa P. Cruz-Díaz, Salud María Jiménez Zafra (2020), “Negation detection for sentiment analysis: A case study in Spanish”, CambridgeCore, 27, 4 pages
4. Shashank Agarwal, Hong Yu (2010), “Biomedical negation scope detection with conditional random fields”, Journal of the American Medical Informatics Association, 696-701, 6 pages
5. Hao Fei , Yafeng Ren , Donghong Ji (2020), “Negation and speculation scope detection using recursive neural conditional random fields”, Neurocomputing, 374, Pages 22-29

**Web References:**

1. Negation Detection: <https://paperswithcode.com/task/negation-detection/codeless>
2. A Natural Language Processing Pipeline of Chinese Free-Text Radiology Reports for Liver Cancer Diagnosis: [IEEE Xplore Full-Text PDF:](https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9179807)
3. Qualifying Certainty in Radiology Reports through Deep Learning–Based Natural Language Processing: [Qualifying Certainty in Radiology Reports through Deep Learning–Based Natural Language Processing | American Journal of Neuroradiology (ajnr.org)](https://www.ajnr.org/content/42/10/1755)
4. NLP Negation Detection: Introduction and Python Tutorial (NegEx): <https://www.youtube.com/watch?v=nfpVBDoViRs>
5. Negation detection in Dutch clinical text: <https://github.com/umcu/negation-detection>

**WORKLOG**

|  |  |  |
| --- | --- | --- |
| **Day** | **Date** | **Task Done** |
| Day 1 | 02/08/2023 | Studying research paper and articles |
| Day 3 | 04/08/2023 | Analyzing scope of project |
| Day 6 | 07/08/2023 | Literature Review |
| Day 8 | 09/08/2023 | Product Survey |
| Day 11 | 12/08/2023 | Workflow discussion |
| Day 14 | 15/08/2023 | Project Initialization |
| Day 17 | 18/08/2023 | Data collection |
| Day 20 | 21/08/2023 | Data pre-processing |
| Day 22 | 23/08/2023 | Data annotations |
| Day 23 | 24/08/2023 | Research about Deep learning models for negation handling |
| Day 27 | 28/08/2023 | Studying about RNN model and its architecture |
| Day 31 | 01/09/2023 | Model building |
| Day 32 | 02/09/2023 | Training the model |
| Day 36 | 06/09/2023 | Testing the model with various radiology reports |
| Day 38 | 08/09/2023 | Learning Flask for the model deployment |
| Day 42 | 12/09/2023 | Creating framework and UI for the project |
| Day 48 | 18/09/2023 | Deployment of the model in the Flask API |
| Day 52 | 22/09/2023 | Integrating the model with our previous work ( Dashboard Analytics using PowerBi) |
| Day 58 | 28/09/2023 | Customizing the Chatbot and deployment in the website |
| Day 60 | 30/09/2023 | Error handling and testing the website |