**NEWS VERIFICATION SYSTEM**

**“VeriFacts”**

**Software Engineering Lab**

**Mini-Project**

**GROUP MEMBERS:**

* **Srihari Srinivasan – 220905326 - 40**
* **Nakshatra B – 220905352 - 41**
* **Aryan Singh – 220905492 - 53**
* **Hima Kasaraneni – 220905520 - 55**

**Life Cycle Model:**

The **incremental development** model allows for the development of the system in manageable increments, enabling us to adapt to changing requirements while promoting error reduction through thorough testing in real-world scenarios.

This minimizes final product errors, enhancing reliability.

Additionally, the model establishes clear milestones and deliverables, making it easier to track progress and effectively manage project timelines throughout the development process.

A diagram of a software development process

Description automatically generated

**SRS DOCUMENT**

**Introduction:**

**Objective:** This paper defines the minimum requirements for designing an automated system for news verification, aimed at checking the claimed statements for accuracy by cross-referencing them with a reliable source.

**Project Scope:** The system will allow users to input URLs of news articles, extract key claims and validate these claims against information retrieved from various online sources to give a reasonable idea of the possibly validity of the target information.

**Environmental Characteristics:** NVS (News Verification System) works with existing web infrastructure to perform searches, analyze content using machine learning models, and present results to users in a user-friendly interface.

**Overall Description:**

**Product Perspective:** NVS is designed to replace manual processes involving fact verification and news analysis by working with current online environments.

**Product Features:** The NVS will enable users to input URLs of news articles and automatically extract key claims, then verify them against trustworthy sources. It will generate results with confidence scores along with links to the sources to enable the verification of news accuracy.

**User Classes:** The primary user of the News Verification System (NVS) is the customer, who plays a crucial role in utilizing the system for assessing the accuracy of news articles. Customers can be individuals, researchers, journalists, or organizations seeking to verify the credibility of information before sharing or acting upon it.

**Operating Environment:** The web-based system will run on standard web browsers and are recommended to be used only on laptops and desktops over hand-held devices.

**Design and Implementation Constraints:** The design needs to be scalable so that the increasing number of users and verification requests shall not degrade the performance of the NVS. It should also be GDPR, CCPA compliant as this ensures the safe handling of data coupled with proper user consent. Certain websites automatically block web scraping functionalities, resulting in certain URLs being invalid.

**Functional Requirements:**

**1.URL Submission**

**Description:** The system accepts a URL input from the user to verify the content of an article.

**Input:** User provides the URL of the article to be verified

**Processing:** Checks whether the URL is valid or not**.** Extracts the content of the article from the provided URL

**Output:** confirmation regarding the URL submission and initiate the verification process

**2.Fact Extraction**

**Description:** The system analyzes the article and extracts key facts or claims to be verified.

**Input:** The content of the submitted article

**Processing**: Pass the article to the first machine learning model. Relevant facts extracted from the article refer to some important claims and facts included in the article.

**Output:** List of resolved facts with key claims from the article

**3.Fact Verification**

**Description:** The system verifies the extracted facts by comparing them with other credible sources.

**Input:** Resolved facts from the article.

**Processing:** Pass the resolved facts to the second machine learning model. Google Search the relevant articles. Analyze against the resolved facts the content of the articles retrieved in step 2

**Output:** Results for each fact that is supported, contradicted, and inconclusive based on additional articles

**4.Results Presentation**

**Description:** The system displays the verification results to the user in an easy-to-understand format.

**Input:** Results of the fact verification process.

**Processing:** Presenting the original fact to be verified.

**Output:** Display the results for each of the claims extracted from the source article. Format and present the result to the user in a clear and understandable form, by displaying the status of verification (supported, contradicted, inconclusive) .

**External Interface Requirements:**

1. **API Interfaces**

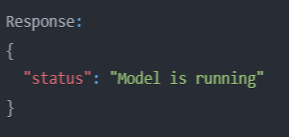
**1.1-POST/summarize**



**1.2-POST/verify**



**1.3 GET/health**



1. **Model Interfaces**

**2.1-Pegasus Model Interface**

* Uses google/Pegasus-large base model from Hugging Face’s transformer library
* Finetuned using LoRA from PEFT library
* Supports text generation with configurable parameters

**2.2-NLI Model Interface**

* Uses typeform/distilbert-base-uncased-mnli from hugging face’s transformer library (fine-tuned on Multi-Genre Natural Language Interface dataset)
* Provides Natural Language Inference (NLI) classification
* Returns probability values for entailment/neutral/contradiction

**3.External Service Dependencies**

**3.1-Google Search Integration**

* Custom web scraping implementation
* User-Agent spoofing for request authentication
* Rate limiting and timeout handling

**3.2-Web Content Extraction**

* BeautifulSoup4 for HTML parsing
* Readability for main content extraction
* Error handling for invalid URLs

**Non-Functional Requirements:**

1. **Performance Requirements**
   1. **Response Time**
      * Summary generation: < 15 seconds
      * Fact verification: < 8 seconds per fact
      * API endpoint response: < 500ms for health checks
   2. **Thoroughput**

* Support 3 concurrent fact verification threads
* Handle up to 5 article sources per fact.
* Process multiple facts in parallel using ThreadPoolExecutor
  1. **Caching**
     + Implement MD5 hash-based caching for Google search results
     + Cache article content extraction results
     + Cache NLI model predictions for repeated verifications

1. **Reliability Requirements**

**2.1-Error handling**

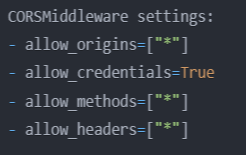
* Graceful degradation for unavailable URLs
* Fallback mechanisms for failed fact verification
* Multiple retry attempts with alternative search queries

**2.2-Logging**

* Error logging with full stack traces

**3-Security Requirements**

**3.1- CORS Configuration**



**3.2-Model Security**

* Local model path configuration
* Secure model loading and initialization
* GPU/CPU device handling

**4.Hardware Requirements**

**4.1-Compute Resources**

* GPU support for model inference (optional)
* Minimum 8GB RAM for model loading

**5.Scalability Requirements**

**5.1-Concurrent Processing**

* ThreadPoolExecutor with 3 workers
* Configurable max\_articles parameter
* Parallel fact verification

**6.Quality Requirements**

**6.1-Verification Accuracy**

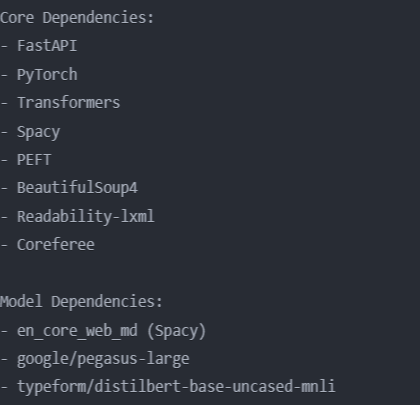
* Confidence threshold > 0.1 for meaningful results
* Multiple source verification per fact
* NLI probability threshold of 0.5 for decisive results

**7.Maintainibilty Requirements**

**7.1- Code Organization**

* Modular class structure (FactChecker, TextRequest)
* Clear separation of concerns between ML models

**8.Dependency Requirements**



**Structured Analysis / Structured Design (SA/SD)**

**Data Flow Diagram (DFD):**

Our project involves a news verification system using two machine learning models. It extracts facts from a user-provided article URL, searches for relevant facts in the article, and then analyses the content to determine if the fact is supported, contradicted, or inconclusive.

Level 0:

A black grid with white text

Description automatically generated

Level 1:

A graph with a white arrow pointing to a circle

Description automatically generated with medium confidence

Level 2:

A black grid with white text

Description automatically generated

A black grid with white text

Description automatically generated

**Data Dictionary:**

url: String

facts: {String}\*

validation: String

article-body :String

fact-summary: String

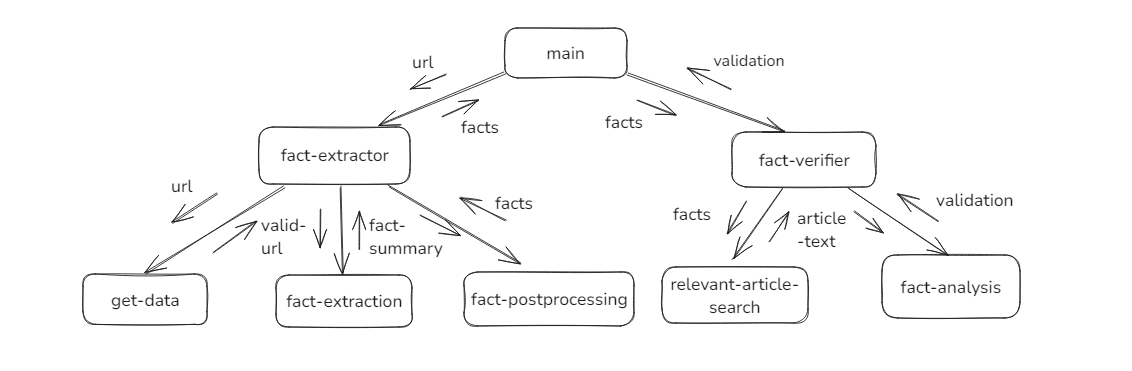
resolved facts: {String}\*

top - urls : { url }\*

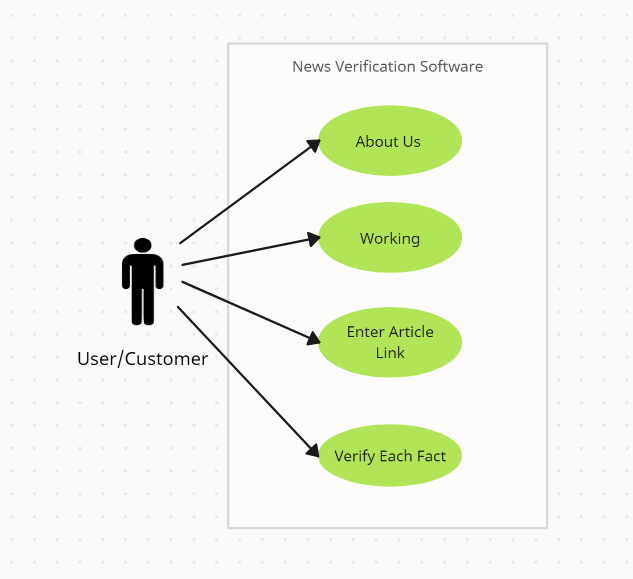
confidence: Float

nli\_result : Dict[String, Float]

**Structure Chart:**



**Unified Modeling Language (UML) Diagrams**

**Use Case Diagram**

**U1: view-about-us-page**: Using this use case, the client can view information about the team that made the project.

Scenario 1: Mainline Sequence

1.      Customer: Select the about us section

2.      System: Display a static webpage with information about the team that made the project

**U2: view-how-it-works -page:** Using this use case, the client can view information about how the project works.

Scenario 1: Mainline Sequence

1.      Customer: Select the how it works section

2.      System: Display a static webpage with information about the working of the project

**U3: enter-article-link**: Using this use case, the client can enter an article link and view the summary of the article in the form of a list of facts.

Scenario 1: Mainline Sequence

1.      Customer: Enters a URL in the search bar

2.      System: Scrolls to new page and displays the list of facts in the article

Scenario 2: At Step 2 of Mainline Sequence

2.      System displays URL not found error message

Scenario 3: At Step 2 of Mainline Sequence

2. System displays an error message saying no search term has been entered.

**U4: verify-each-fact**: Using this use case, the client can view the degree the fact matches with the top 5 results online.

Scenario 1: Mainline Sequence

1.      Customer: Selects a specific fact

2.      System: Displays a list of URLs of other news websites and whether it agrees or disagrees with the given fact.

Scenario 2: At Step 2 of Mainline Sequence

2. System: Displays a list of URLs of other news websites, and if the website can’t be accessed, the word blocked is displayed.

**Class Diagram**

**A diagram of a webpage

Description automatically generated**

**Sequence Diagram**

**A diagram of a diagram

Description automatically generated**

**Activity Diagram**

A diagram of a flowchart

Description automatically generated