Software Requirements Specification

1. Introduction

The introduction to the Software Requirements Specification (SRS) document provides an overview of the complete SRS document. This document contains all of the information needed by a software engineer to adequately design and implement the software product described by the requirements listed in this document.

1.1 Purpose

This Software Requirements Specification (SRS) document outlines the requirements for a system designed to detect hate speech in text using deep learning techniques. It is intended for developers, project managers, testers, and stakeholders involved in the development and deployment of this system.

The system aims to address the growing problem of online hate speech, which can have significant negative impacts on individuals and society. As highlighted in the YouTube video [https://youtu.be/jbexvUovHxw?si=RTpWI9nyRD0jSlf1], hate speech on platforms like YouTube can undermine community trust, propagate harmful ideologies, and incite violence. This system seeks to provide an automated solution for identifying and mitigating such content.

1.2 Scope

This project involves the development of a robust hate speech recognition system using deep learning methodologies. The scope includes:

- Collection and preprocessing of a diverse dataset of textual data, including text from YouTube comments and other online sources.
- Implementation of deep learning models, such as Recurrent Neural Networks (RNNs) and Transformers, for accurate classification of hate speech.
- Development of a modular software application.
- Rigorous testing and evaluation of the system's performance, with a focus on its ability to detect hate speech in the context of online video platforms.
- Documentation of the system design, implementation, and usage.

The system will be designed to identify and flag instances of hate speech within various online text sources, including social media posts, comments, and articles, with a specific emphasis on YouTube comments, as discussed in the referenced video.

1.3 Definitions, Acronyms, and Abbreviations

- **SRS:** Software Requirements Specification

- **RNN:** Recurrent Neural Network
- **Transformer:** A type of deep learning model
- **NLP:** Natural Language Processing
- **API:** Application Programming Interface

1.4 References

- IEEE Guide to Software Requirements Specifications
- Hate Speech Detection in social Media Using Deep Learning - IJAEM.net [http://ijaem.net/issue_dcp/Hate%20Speech%20Detection%20in%20social%20Media%20Using%20Deep%20Learning.pdf]
- A Study on Detecting Hate Speech in YouTube Videos Using Machine Learning [http://ijte.uk/archive/2023/A-Study-on-Detecting-Hate-Speech-in-YouTube-Videos-Using-Machine-Learning.pdf]

1.5 Overview

This SRS document is organized as follows:

- **Section 1** provides an introduction to the SRS, including the purpose, scope, definitions, and references.
- **Section 2** describes the general factors that affect the product and its requirements.
- **Section 3** details the specific requirements, including external interface, functional, non-functional, and design constraints.

2. General Description

This section describes the general factors that affect the product and its requirements. It clarifies the context of the system and provides a high-level overview of its functions and constraints.

2.1 Product Perspective

The Hate Speech Recognition System is a standalone software product that will be integrated into various online platforms, particularly those with video content, such as YouTube. As highlighted in the referenced video, the system will address the need for improved hate speech detection on these platforms. It will receive text data as input and provide a classification (hate speech or not hate speech) as output. The system will be designed to be scalable and adaptable to different platforms and data formats, with a focus on handling the specific characteristics of text data from video platforms.

2.2 Product Functions

The primary function of the software is to accurately classify text, especially

comments from video platforms, as either hate speech or not hate speech. This involves the following sub-functions:

1. **Data Input:** Ingest text data from various sources, including YouTube comments and other online sources.
2. **Data Preprocessing:** Clean and prepare the text data for analysis, including removing noise, normalizing text, and tokenization, with specific consideration for the nuances of online comments (e.g., abbreviations, slang, and misspellings).
3. **Feature Extraction:** Convert the text data into a numerical representation using techniques such as word embeddings, optimized for the characteristics of short-form text.
4. **Classification:** Utilize a trained deep learning model to classify the text, with a model trained on data that includes a significant portion of video platform comments.
5. **Output:** Provide a clear indication of whether the input text is classified as hate speech.
6. **API Integration:** Enable other applications, including video platform content management systems, to use the hate speech detection functionality.
7. **Model Training and Management:** Functionality to train, update, and manage the deep learning models, with the ability to incorporate new data and adapt to evolving forms of hate speech on video platforms.

2.3 User Characteristics

The intended users of this system include:

- **Content Moderators:** To identify and remove hate speech from online platforms, particularly video-sharing platforms.
- **Platform Administrators:** To integrate the system into their platforms, such as YouTube, and manage its settings.
- **Researchers:** To analyze hate speech patterns and trends, including the spread of hate speech in video comments.
- **Application Developers:** To integrate the API into their applications.

These users are expected to have varying levels of technical expertise. The system should be user-friendly and provide clear and concise results, with specific features tailored to the needs of video platform moderators.

2.4 General Constraints

The development of this system is subject to the following constraints:

- **Data Availability:** The availability and quality of labeled training data, especially data from video-sharing platforms, may impact the system's performance.

- **Computational Resources:** Training deep learning models requires significant computational resources.
- **Real-time Processing:** The system may need to process text data in real-time, which imposes performance constraints, especially for platforms with high comment volumes.
- **Evolving Nature of Hate Speech:** Hate speech is constantly evolving, requiring the system to be adaptable and regularly updated, with a focus on detecting new forms of hate speech that emerge in online video environments.
- **Ethical Considerations:** The system must be developed and used in an ethical manner, with careful consideration of potential biases and impacts on free speech, particularly in the context of user-generated content on video platforms.

2.5 Assumptions and Dependencies

The following assumptions and dependencies exist:

- A sufficient amount of labeled data, including data from video-sharing platforms, will be available for training the deep learning models.
- The necessary hardware and software infrastructure will be available for development and deployment.
- Users will have access to the internet for accessing the system or its API.
- Third-party libraries and frameworks, such as TensorFlow or PyTorch, will be used.

3. Specific Requirements

This section details the specific requirements for the Hate Speech Recognition System. It covers external interface, functional, non-functional, and design constraints.

3.1 External Interface Requirements

3.1.1 User Interfaces

- The system shall provide a user-friendly web interface for administrators to manage the system, train models, and view reports.
- The web interface shall be responsive and accessible on different devices.
- The system shall provide clear and intuitive visualizations of the system's performance, including metrics specific to hate speech detection in video comments.

3.1.2 Hardware Interfaces

- The system shall be compatible with standard server hardware.

- The system may utilize GPUs for accelerated model training and inference.

3.1.3 Software Interfaces

- The system shall be compatible with a variety of operating systems, including Linux and Windows Server.
- The system shall utilize Python for development.
- The system shall use deep learning frameworks such as TensorFlow or PyTorch.
- The system shall provide an API for integration with other applications, including video platform content management systems.

3.1.4 Communications Interfaces

- The system shall support communication over HTTP/HTTPS for the API.
- The system may support other communication protocols as needed.

3.2 Functional Requirements

3.2.1 Data Input and Preprocessing

1. **Introduction**
   The system shall be able to ingest text data from various sources, including video platform comments, and preprocess it for analysis.
2. **Inputs**
   - Text data from files (e.g., CSV, JSON, text files)
   - Real-time text streams (e.g., from social media platforms, including YouTube)
   - Text data from databases
   - Text data from video platform APIs (e.g., YouTube Data API)
3. **Processing**
   The system shall perform the following preprocessing steps, with a focus on handling the characteristics of text from video platforms:
   - Text normalization (e.g., lowercasing, removing special characters, handling emojis and other platform-specific symbols)
   - Tokenization (splitting text into words or subwords, optimized for short, informal text)
   - Stop word removal (with a list tailored to online comments)
   - Handling of missing values
   - Encoding of text data into numerical representations (e.g., word embeddings, contextual embeddings)
4. **Outputs**
   - Cleaned and preprocessed text data in a numerical format suitable for deep learning models.
5. **Error Handling**

- The system shall handle invalid or malformed input data gracefully, including errors specific to video platform data formats.
- The system shall log any errors encountered during data input and preprocessing.

3.2.2 Hate Speech Classification

1. **Introduction**
   The system shall utilize a trained deep learning model to classify text, including video platform comments, as either hate speech or not hate speech.
2. **Inputs**
   - Preprocessed text data in a numerical format.
3. **Processing**
   The system shall:
   - Load a pre-trained deep learning model.
   - Feed the preprocessed text data into the model.
   - Obtain a probability score indicating the likelihood that the text is hate speech.
   - Apply a threshold to the probability score to make a binary classification (hate speech or not hate speech).
4. **Outputs**
   - A binary classification indicating whether the input text is hate speech or not.
   - A confidence score associated with the classification.
5. **Error Handling**
   - The system shall handle cases where the model fails to make a classification.
   - The system shall log any errors encountered during the classification process.

3.2.3 Model Training and Management

1. **Introduction**
   The system shall provide functionality to train, update, and manage the deep learning models used for classification, with a focus on adapting to the evolving nature of hate speech on video platforms.
2. **Inputs**
   - Labeled text data for training, including a substantial amount of data from video platforms.
   - Configuration parameters for model training.
   - Evaluation metrics, including metrics that are robust to class imbalance and the specific challenges of hate speech detection in short-form text.
3. **Processing**
   The system shall:

    - Load and preprocess the training data.
    - Train a deep learning model using the training data and specified parameters.
    - Evaluate the model's performance using the evaluation metrics.
    - Save the trained model.
    - Provide functionality to update the model with new data, including data from video platforms, to improve accuracy and generalization.
    - Provide functionality to monitor model performance and retrain as needed.
4. **Outputs**
    - Trained deep learning models.
    - Model performance metrics (e.g., accuracy, precision, recall, F1-score).
    - Logs of training and evaluation processes.
5. **Error Handling**
    - The system shall handle errors during model training and evaluation.
    - The system shall provide informative error messages and logging.

3.3 Non-Functional Requirements

3.3.1 Performance

- The system shall be able to process text data with an average latency of less than 500ms, even for short-form text with variations in language and spelling common in video comments.
- The system shall be able to handle a throughput of 1000 requests per second.
- The system shall be scalable to handle increasing volumes of data, including the high volume of comments on popular video platforms.

3.3.2 Reliability

- The system shall have a downtime of less than 1 hour per month.
- The system shall have a Mean Time Between Failures (MTBF) of at least 1000 hours.
- The system shall provide mechanisms for fault tolerance and recovery, ensuring continuous operation even with fluctuations in data volume from video platforms.

3.3.3 Availability

- The system shall be available 99.9% of the time.

3.3.4 Security

- The system shall protect sensitive data, including user data and model parameters, from unauthorized access.
- The system shall implement authentication and authorization mechanisms.
- The system shall be regularly tested for security vulnerabilities, including those specific to web applications and APIs.

- The system shall comply with relevant data privacy regulations, including those related to handling user-generated content from video platforms.

3.3.5 Maintainability

- The system shall be designed to be modular and easy to maintain, allowing for updates to the model and preprocessing steps without disrupting other components.
- The system shall be well-documented, with clear explanations of how to adapt the system for different video platforms.
- The system shall use version control for all code and configurations.
- The system shall be designed to be easily updated with new models and features, including those that address new forms of hate speech that may emerge on video platforms.

3.3.6 Portability

- The system shall be portable across different operating systems and hardware platforms.
- The system shall be deployable in cloud environments, allowing for scalability and flexibility in handling data from various video platforms.

3.4 Design Constraints

- The system shall be developed using Python.
- The system shall use TensorFlow or PyTorch for deep learning.
- The system shall adhere to industry best practices for software development.
- The system shall be designed to be extensible and adaptable to future requirements, including the need to handle new data formats and evolving forms of hate speech on video platforms.