HarshKumar Devmurari (09)
Vedantika Ghorpade (14)
Faizan Raiba (49)

Synopsis

1. Abstract

This project presents the development of a game called "Bird Surfer" created in Unity, drawing inspiration from the classic "Flappy Bird" gameplay mechanics. The primary objective of this endeavor was to recreate the essence of the original game while introducing unique elements to enhance the gaming experience. In the process of building "Bird Surfer" in Unity, we successfully replicated the core gameplay dynamics, allowing players to control a bird character by tapping the screen to keep it afloat and navigate through a series of obstacles. Additionally, we introduced the feature of customizing bird characters and diverse obstacle themes to provide players with a personalized and immersive gaming experience. The project involved the utilization of Unity's game development capabilities, including physics-based gameplay, sprite animations, and user interface design. By integrating these elements, we were able to create an engaging and visually captivating game. In conclusion, the "Bird Surfer" project successfully achieved its goal of recreating the classic "Flappy Bird" gameplay while offering customization options for a unique player experience. This project not only showcases Unity's potential as a robust game development tool but also serves as a foundation for future enhancements and expansions within the game.

2. Methodologies

1. **Importing Libraries:**

The code begins by importing necessary libraries, including re (regular expressions), os (operating system-related functions), string, and pandas for data manipulation.

It also imports the TfidfVectorizer from scikit-learn for TF-IDF analysis.

2**. Uploading Kaggle API Token:**

The user is prompted to upload their Kaggle API token, which is essential for accessing datasets from Kaggle. This token is stored in a file named "kaggle.json."

3. Downloading Datasets:

The code uses the Kaggle API token to download two datasets: "nips-papers-1987-2019-updated" and "stopwords."

These datasets contain information about academic papers and a list of common stopwords.

4. Loading Stopwords:

The code loads the stopwords from the "stopwords" dataset and prepares them for text preprocessing.

5. Text Preprocessing:

The "clean_text" function is defined to perform text preprocessing. It involves converting text to lowercase, removing punctuation, and eliminating extra whitespace.

6. TF-IDF Vectorization:

The TF-IDF vectorizer is initialized using the TfidfVectorizer from scikit-learn.

The code prepares the text corpus by excluding the first 10 documents (for testing purposes) and applies TF-IDF vectorization to create a vocabulary.

The result is stored in the "feature_names" variable.

7. Keyword Extraction:

The code defines the "get_keywords" function, which takes a document as input and extracts the top keywords using TF-IDF analysis.

The keywords are ranked based on their TF-IDF scores.

8. User Input:

The project offers an interactive component where users can input a sentence or paragraph for keyword extraction.

9. Keyword Extraction for User Input:

The user's input is processed, and the "get_keywords" function is applied to calculate the TF-IDF scores and extract the top keywords from the input text.

10. Displaying Extracted Keywords:

The code prints and displays the extracted keywords to provide a concise summary of the input text's primary themes.

3. Technical Terms

TF-IDF (Term Frequency-Inverse Document Frequency): TF-IDF is a numerical statistic used to evaluate the importance of a word within a document relative to a collection of documents. It is commonly employed in information retrieval and text mining to determine the significance of words.

TfidfVectorizer: TfidfVectorizer is a class from scikit-learn used for text vectorization, which converts text documents into numerical representations suitable for machine learning algorithms. It calculates TF-IDF scores for each word in the corpus.

Stopwords: Stopwords are commonly used words (e.g., "and," "the," "in") that are filtered out during text preprocessing to focus on more meaningful words in natural language processing tasks.

Text Preprocessing: Text preprocessing refers to the cleaning and transformation of raw text data before analysis. This may include tasks like lowercasing, removing punctuation, and eliminating stopwords.

Kaggle API Token: A Kaggle API token is a security key used to access datasets and other resources on Kaggle, a popular platform for data science competitions and dataset sharing.

Corpus: A corpus is a collection of textual documents, often used as the dataset for natural language processing tasks.

Vocabulary: In the context of TF-IDF vectorization, a vocabulary is the set of unique words (features) present in the text corpus.

TF-IDF Score: The TF-IDF score is a numerical value that represents the importance of a word within a document. It is calculated by considering both the term frequency (TF) and inverse document frequency (IDF).

Data Preprocessing: Data preprocessing is the initial step in data analysis, which involves cleaning and organizing raw data to make it suitable for further analysis and modeling.

Vectorization: Vectorization is the process of converting text data into numerical vectors, making it compatible with machine learning algorithms.

Interactive Component: The interactive component of the project allows users to provide input and receive results in real-time, enhancing user engagement and usability.

API Token Authentication: API token authentication is a security mechanism that verifies the identity of users and grants them access to specific resources or services.

Information Retrieval: Information retrieval is the process of searching for and obtaining specific information or documents from a large collection of data.