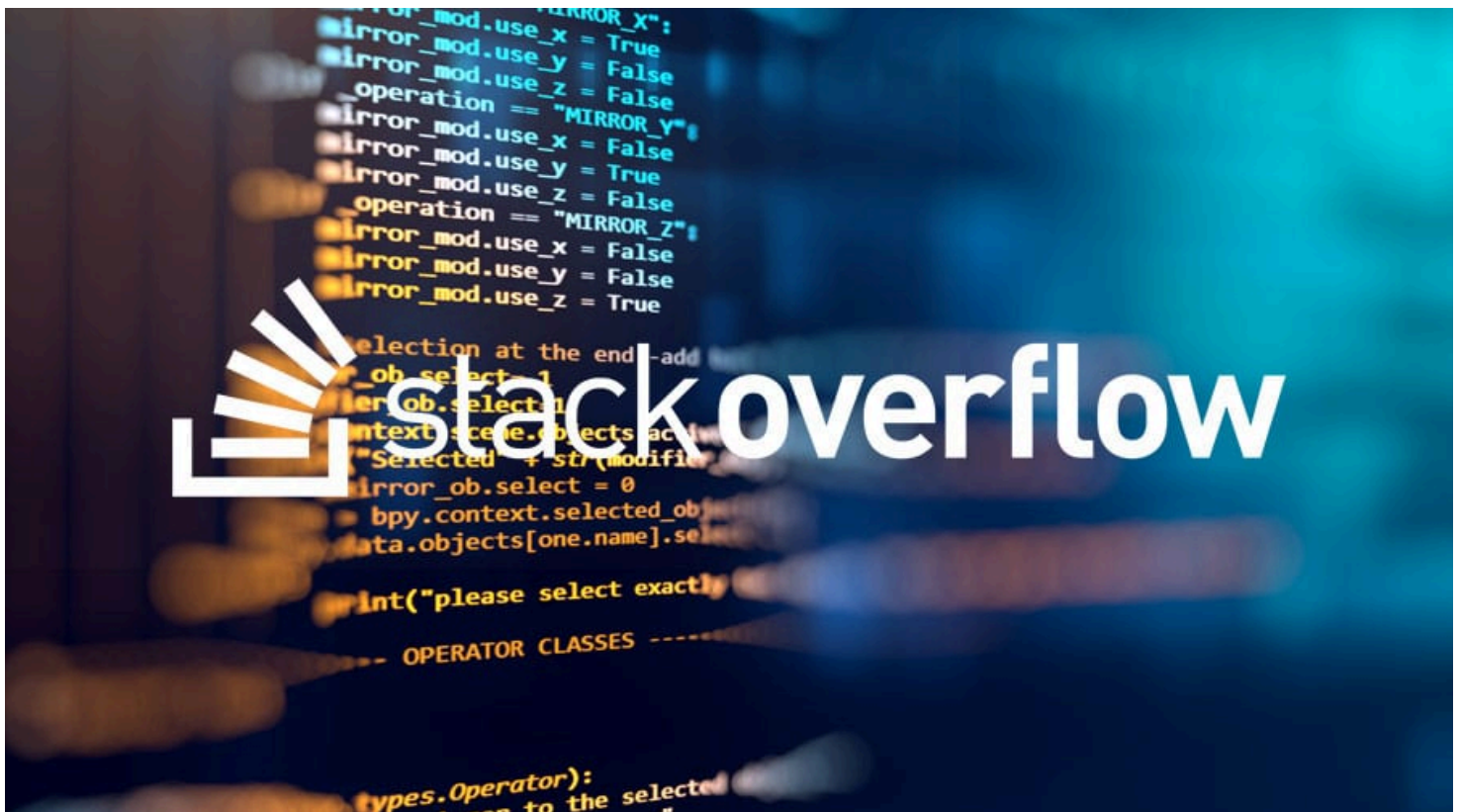


Natural language processing:

similar questions predictor on StackOverflow



Houda Ait Abdeslam

1- Description:

In this project, my task is to develop a tool designed to predict similar questions on StackOverflow, a popular platform for programmers to ask and answer technical questions. My approach will involve analyzing a dataset of StackOverflow questions, leveraging natural language processing (NLP) techniques, and implementing machine learning algorithms to achieve my goal.

2- Introduction:

2.1- Aim:

The aim of our project is to develop a tool that predicts similar questions on StackOverflow using NLP algorithms. By automating the process of identifying similar questions,

2.2- scope:

Our project focuses on implementing NLP techniques to analyze the textual content of questions posted on StackOverflow. We aim to create a tool that can accurately identify similarities between questions based on their semantic meaning and context, rather than relying solely on keyword matching or manual tagging.

2.3- methodology and tools:

Dataset: The dataset is sourced from GitHub and contains StackOverflow questions. We load and preprocess the data using Python.

Google Colab: We use Google Colab as our development environment for implementing the solution.

Natural Language Processing (NLP):

- **Text Preprocessing**
- **Word2Vec Model**
- **Similarity Calculation**

I. Theoretical Part

3. Theory Used to Solve the Problem

3.1- Preprocessing:

- **Tokenization:** This process involves splitting text into smaller units called tokens, which can be words, phrases, or symbols. Tokenization is a crucial step in text preprocessing.
- **Stop Words Removal:** Stop words are common words that carry little semantic meaning (e.g., "and", "the", "is"). Removing these words helps in focusing on the meaningful words in the text.
- **Lemmatization:** This process reduces words to their base or root form, ensuring that different forms of a word are treated as a single entity (e.g., "running" becomes "run").
- **Synonym Replacement:** This technique involves replacing words with their synonyms to enhance the text representation and capture semantic similarities.

3.2 Word Embeddings

- **Word2Vec Model:** Word2Vec uses neural networks to learn word associations from a large corpus of text. It produces word vectors such that words with similar meanings have similar vector representations. We trained a Word2Vec model on our StackOverflow dataset to capture the semantic relationships between words.

3.3 Similarity Metrics

To measure the similarity between questions, we use cosine similarity, a metric that measures the cosine of the angle between two vectors. It is widely used in text analysis to compare document similarity.

II. Practical Part

4. Solution to the Problem

4.1 Data Preprocessing

We begin by loading and preprocessing the dataset, which includes titles, bodies, and tags of StackOverflow questions. The preprocessing steps are as follows:

- **Loading the Dataset:** The dataset is sourced from GitHub and loaded into a pandas DataFrame.
- **Removing HTML Tags:** Using BeautifulSoup, we strip HTML tags from the text to ensure clean text data.
- **Tokenization and Lemmatization:** We tokenize the text into words and lemmatize them to their root forms.
- **Stop Words Removal:** Stop words are removed to focus on meaningful content.
- **Synonym Replacement:** Words are replaced with their synonyms to capture semantic nuances.

4.2 Results

The results of our approach are presented in a new column in the DataFrame called `Word2Vec_Similarity`. This column contains the similarity scores (as percentages) between original and duplicate questions. These scores indicate how similar the questions are based on their semantic content.

The model was trained with an extensive corpus including titles, bodies, and tags to capture a wide range of contexts and meanings. This comprehensive training helps in achieving better similarity detection.

4.3 Discussion

Our approach successfully identifies similar questions based on their semantic content, leveraging the power of Word2Vec embeddings and cosine similarity. However, the performance of the model can still be improved by:

- **Increasing the dataset size:** A larger dataset would provide more training examples, leading to better word embeddings.
- **Fine-tuning hyperparameters:** Adjusting parameters such as vector size, window size, and epochs for the Word2Vec model could improve performance.
- **Incorporating additional features:** Including more contextual information or meta-data, such as user interactions or timestamps, could enhance similarity detection.