Clothing Item Scraper

Goal

The goal of this code is to scrape any e-commerce websites based on an input keyword, and return N-number of relevant links based on similarity on product descriptions.

Summary:

The provided code implements an Ajio clothing item scraper that allows users to input a description and retrieves relevant links to clothing items from the Ajio website. It leverages web scraping, natural language processing, and machine learning techniques to achieve this.

The code follows the following steps:

1. Imports necessary libraries and modules, including Selenium for web browsing, JSON for data manipulation, and TensorFlow and its associated modules for machine learning tasks.

2. Sets up the RAKE algorithm and BERT models. RAKE is used for keyword extraction for both input string and product descriptions, while BERT is used for text preprocessing and encoding.

3. Defines a function to get the sentence embedding using BERT. This function takes a list of sentences as input and returns their corresponding embedding.

4. Implements a function to extract features from the input text using RAKE. It extracts keywords from the text and returns the top 10 keywords as a single string.

5. Implements a function to scrape clothing items. It performs web scraping on the Ajio search page, retrieves the item links and small descriptions, and follows each link to extract additional information such as brand, color, and detailed description.

6. Defines a function to remove unnecessary keys/info from the scraped results.

7. Implements a function to clean the descriptions. It adds brand and color information to the beginning of each description, converts the descriptions to lowercase, and removes specific lines containing irrelevant information.

8. Defines a function to calculate the similarity between the input description and the cleaned descriptions of the suggested items. It uses BERT embedding's and cosine similarity to calculate the similarity scores.

Methodology:

The series of steps below are confined to extract features from the input string, then pass the cleaned input text to search for items in website. We now extract for each product its description, clean its description. After that we extract features from each description and generate embeddings for both descriptions and input string. Then we rank few links based on cosine similarity between the embeddings.

Detailed view of each step:

1. Import necessary libraries and modules:
   - `selenium`: Provides automated web browsing capabilities.
   - `json`: Enables JSON data manipulation.
   - `time`: Allows for adding delays in the code execution.
   - `multi_rake`: Implements the RAKE algorithm for keyword extraction.
   - `numpy`: Supports mathematical operations on arrays and matrices.
   - `sklearn`: Provides various machine learning algorithms and utilities.
   - `tensorflow`: Enables building and training machine learning models.
   - `tensorflow_hub`: Allows for the use of pre-trained models from TensorFlow Hub.
   - `tensorflow_text`: Provides text preprocessing capabilities using TensorFlow.
2. Set up RAKE and BERT models:
   - Initialize the RAKE model (`rake`) for keyword extraction.
   - Load the BERT models for text preprocessing and encoding:
     - `bert_preprocess`: Preprocesses text using BERT.

- `bert_encoder`: Encodes preprocessed text using BERT.

3. Define a function to extract features from the input text:

- `extract_features_from_text(full_text)`: Takes the full text as input and extracts keywords using RAKE.

- Returns the top 10 keywords concatenated into a single string.

4. Define a function to scrape Ajio clothing items:

- `scrape_ajio_clothing_items(description)`: Takes a description as input and performs web scraping on the Ajio search page.

- Initializes a headless Selenium WebDriver using Chrome.

- Encodes the description for the URL query parameter.

- Loads the Ajio search page and finds all the search result items.

- Extracts the item link and small description for each result and stores them in a list of dictionaries.

- For each result, follows the item link and extracts the item brand, color, and detailed description.

- Appends the extracted data to the corresponding dictionary in the results list.

- Quits the WebDriver.

- Returns the results as a JSON response.

5. Define a function to remove unnecessary data from the result:

- `remove_key(x)`: Removes unnecessary keys (brand, color, and small description) from a dictionary.

- Returns the modified dictionary.

6. Define a function to clean Ajio data:

- `clean_ajio_data(suggested_items)`: Takes a list of suggested items as input and cleans their descriptions.

- Adds the item brand and color information to the beginning of each description.

- Converts the descriptions to lowercase.

- Removes specific lines containing 'package' or 'other information' from the description.

- Updates the description in each item's dictionary.

- Removes unnecessary keys and transforms the list of dictionaries into a list of lists.

- Returns the cleaned suggested items.

7. Define a function to extract features and get the sentence embedding using BERT:

- from RAKE we extract important words/features from list of descriptions

- `get_sentence_embedding(sentences)`: Takes a list of processed sentences as input and returns the sentence embeddings using BERT.

8. Define a function to calculate similarity between the input and descriptions:

- `calculate_similarity(input_string, string_list)`: Takes the input string and a list of string items as input.

- Calculates the sentence embedding for the input string and the string items using BERT.

- Calculates the cosine similarity between the input embedding and the string embeddings.

- Combines the descriptions, links, and similarities into a list of tuples

9) Define a function which return N-number of relevant links of products

- The ultimate function which return us required number of relevant link of products which we desire based on ranking them.

Conclusion:

The clothing item scraper code provides a convenient way to retrieve relevant clothing item links based on a given description. By leveraging web scraping, keyword extraction, and similarity calculations, it enhances the user experience by automating the search process. The code can be further expanded and customized to meet specific requirements, such as integrating with a user interface or incorporating additional features for personalized recommendations.