Project Report: Graph Visualization and Search System

Introduction

In this project, we aim to visualize a graph from a dataset and implement a search system. The dataset consists of a graph represented by edges and content associated with various URLs. We will create a directed graph, visualize it, process the content to remove stopwords, build an inverted index, and develop search systems based on single-word and bag-of-words queries. The ultimate goal is to rank the search results using the PageRank algorithm, enhancing the search experience by providing more relevant results.

Library Files

The following libraries were utilized in this project:

- `numpy`: For numerical operations.
- `pandas`: For data manipulation and analysis.
- `networkx`: For creating, visualizing, and analyzing graphs.
- `matplotlib`: For plotting and visualization.
- `nltk`: For natural language processing, specifically tokenization and stopword removal.
- `google.colab`: To mount Google Drive and access the dataset.

Tasks

Task 01: Create a graph from `graph.csv` and load the contents from `content.csv`.

Task 02: Create a figure of the graph using the `networkx` library.

python
```
plt.figure(figsize=(12, 8))
pos = nx.spring_layout(G)
nx.draw(G, pos, with_labels=True, node_color='lightgreen', node_size=3000, edge_color='gray', font_size=15)
plt.title("Graph Visualization")
plt.show()
```

Task 03 (Bonus): Tokenize the contents and remove words like articles, prepositions, and conjunctions using `nltk`.

python
```
nltk.download('punkt')
```

```
nltk.download('stopwords')

stop_words = set(stopwords.words('english'))
processed_content = {}

for url, content in url_to_content.items():
    tokens = word_tokenize(content.lower())
    filtered_tokens = [word for word in tokens if word.isalnum() and word not in stop_words]
    processed_content[url] = filtered_tokens
```

Task 04: Build an inverted index to map tokens to URLs.

```
python
inverted_index = defaultdict(list)

for url, tokens in processed_content.items():
    for token in tokens:
        inverted_index[token].append(url)

print(f"Inverted Index created with {len(inverted_index)} tokens.")
```

Task 05: Build a single-word query-based search system and rank the pages using the PageRank algorithm.

```
python
pagerank_scores = nx.pagerank(G)

def single_word_query(query):
    query = query.lower()
    if query in inverted_index:
        result_urls = inverted_index[query]
        ranked_urls = sorted(result_urls, key=lambda url: pagerank_scores.get(url, 0),
reverse=True)
        return ranked_urls
    else:
        return []

query = "Python"
print(f"Results for single word query '{query}':", single_word_query(query))
```

Task 06 (Bonus): Implement a bag-of-words query-based search system.

```
python
def bag_of_words_query(query):
    query_tokens = [word.lower() for word in word_tokenize(query) if word.isalnum()]
    result_urls = set()
    for token in query_tokens:
        if token in inverted_index:
            result_urls.update(inverted_index[token])
    ranked_urls = sorted(result_urls, key=lambda url: pagerank_scores.get(url, 0), reverse=True)
    return ranked_urls

query = "Python Java"
print(f"Results for bag of words query '{query}':", bag_of_words_query(query))
```

Conclusion

This project successfully demonstrates the creation and visualization of a directed graph, processing textual content, and building efficient search systems. By leveraging the PageRank algorithm, we can rank the search results and improve the relevance of the returned URLs. The implementation of both single-word and bag-of-words query systems showcases the versatility and scalability of the search functionality. This project serves as a foundational step towards developing more advanced search engines and data visualization tools.