Community Detection over StackOverflow Tags

Durga Prasad Maram*
UMass Amherst
dmaram@umass.edu

Sai Sreenivas Chintha*
UMass Amherst
saisreenivas@umass.edu

Prathik VNS*
UMass Amherst
nvunnava@umass.edu

Abhinav Sharma*
UMass Amherst
abhinavs@umass.edu

1. Motivation

As online platforms generate vast amounts of data, graph-based structures and community detection algorithms are essential for uncovering relationships and enhancing information retrieval. However, the computational demands of these algorithms can limit their scalability. **PySpark's (GraphFrames)** offers a distributed computing framework that enables efficient processing of large graphs, effectively addressing challenges encountered in real-world applications.

2. Problem Statement

StackOverflow tags play a crucial role in enhancing search, categorizing, and labeling content. Properly assigning tags to posts increases visibility and improves search performance, making it easier for users to find relevant content.

In this work, we leverage PySpark's distributed graph-based data processing capabilities to perform tag similarity-based community detection on StackOverflow data, analyzing correlations between tags. This allows us to gain insights into how different tags relate to one another and helps in recommending relevant tags as well as associated posts.

To formalize the **problem**: given a user's StackOverflow post, we mine the constructed tag correlation graph to identify the most relevant tags and recommend them to the user.

3. Data

We will use the StackOverflow Kaggle dataset (**StackOverflow_posts**) for our problem statement. This dataset provides a clear mapping of posts, post IDs, and tags, which facilitates the extraction of relevant features from the posts, the creation of tag vectors based on related posts, and the construction of a graph where each tag serves as a node and the edges represent tag similarities.

This large-scale dataset, which is publicly available via BigQuery, contains $\sim$**31M** StackOverflow posts and consists of $\sim$**55K unique tags**.

4. Approach

Our approach is structured into several key steps:

1. **Data Preprocessing:** We will begin by cleaning and preprocessing the Stack Overflow dataset to ensure the data is in a suitable format for analysis, using **Spark-NLP**. This includes tokenization, stemming, stop-word removal etc.
2. **Vectorization of Tags:** We will create vector representations for each tag based on the associated posts. We will use **TF-IDF** (Term Frequency-Inverse Document Frequency) vectors, computed with **PySpark MLLib**, for this purpose.
3. **Calculating Similarities:** Once the vector representations are generated, we will compute the similarities between tags using cosine similarity or other distances.
4. **Graph Construction:** Utilizing the similarities calculated in the previous step, we will construct a tag-correlation graph using **GraphFrames** library, where vertices represent tags, and edges denote the strength of similarity between them. This structure will facilitate further analysis and insights.
5. **Community Detection:** We will apply community detection algorithms to identify clusters of closely related tags within the graph.
6. **Subgraph Extraction and Application:** The final step involves extracting relevant subgraphs from the community detection results. These subgraphs can be used to generate tag **recommendations**.

[1] all authors contributed equally

7. **Cloud Deployment:** To ensure scalability and performance, we will investigate deploying this framework on cloud platforms such as **Google Cloud Dataproc**. This will enable distributed processing of large datasets using multiple spark clusters.

5. Evaluation

1. We will set aside some data points for testing and use **recall** as the evaluation metric. We find the node with the highest similarity to the post embedding. The associated tag, along with the tags of nearby nodes, is used as recommendations, over which recall is computed.
2. We also plan to compare the **runtimes** of single-threaded implementations using NetworkX (in Python) with GraphFrames (distributed across multi-core and multi-cluster environments).

6. Milestone goals

We will complete the first 3 steps of our approach for the milestone.

1. **Data pre-processing** - Pre-processing the text data by tokenizing, normalizing, stemming, and removing stopwords using **NLTK** - Durga Prasad and Sai Sreenivas .
2. **Vectorization of tags** - We will compute the **TF-IDF** vectors for tags based on the related posts - Prathik VNS.
3. **Similarity Computation** - Similarity matrix computation using cosine similarity - Abhinav Sharma

7. Final goals

1. Graph Construction - Sai Sreenivas & Prathik
2. Community detection. - Abhinav & Durga Prasad
3. Recommendation & Evaluation. - Durga Prasad & Sai Sreenivas
4. Hosting/Deployment. - Prathik & Abhinav