

BAHRIA UNIVERSITY (KARACHI CAMPUS)

Assignment-02

(Software Construction)

Class: **BSE [4]-5 (B)**

Course Instructor: **Dr. Salahuddin Shaikh**

Date: **(20/11/2023)**

Student’s Name: **Muhammad Shoaib Akhter Qadri**

**(Morning)**

Max Marks**: 5 M**

Reg. No: **79290**

**Question No. 1: [CLO#02, 5.0 marks]**

**You are tasked with creating a text processing component for a social media platform.**

**The component needs to analyze and tag user posts for specific keywords and mentions.**

**Your goal is to design grammar and implement a regular expression to identify and tag**

**these elements.**

**1. Design a Grammar:**

**Create a grammar that defines the structure of a user post on the social media**

**platform. Your grammar should account for the following elements:**

**- Text content**

**- Mentions (e.g., @username)**

**- Keywords (e.g., #keyword)**

**- Emoticons (e.g., :) or :D)**

**Solution:**

**Grammar Design:**

We need to design a grammar to represent the structure of a user post on a social media platform. The elements we want to capture include:

* Text content: Any sequence of characters.
* Mentions: Represented by the format @username.
* Keywords: Represented by the format #keyword.
* Emoticons: Examples include :), :D, ;).

The grammar rules can be defined as follows:

* post: Represents the entire user post and consists of text, mentions, keywords, and emoticons.
* text: Represents any sequence of characters.
* mention: Represents the format @username.
* keyword: Represents the format #keyword.
* emoticon: Represents emoticons such as :), :D, ;).

**2. Regular Expression Implementation:**

**Using the grammar you designed, implement a regular expression in any**

**language that can identify, and extract mentions and keywords from a user post.**

**Your regular expression should:**

**- Match mentions starting with "@" followed by a username (e.g.,**

**@john\_doe).**

**- Match keywords starting with "#" followed by a keyword (e.g.,**

**#technology).**

**Solution:**

**Regular Expression Implementation in JavaScript:**

I wrote regular expression in JavaScript Language:

      const postRegex = /(?:(?:\w+\s\*)\*[@#]\w+\s\*)+|\w+/g;

      const mentionRegex = /@(\w+)/g;

      const keywordRegex = /#(\w+)/g;

      function tagUserPost(userPost) {

          const tags = [];

          // Extract mentions

          const mentions = userPost.match(mentionRegex);

          if (mentions) {

              mentions.forEach(mention => {

                  tags.push(`[\*\*${mention}\*\*]`);

              });

          }

          // Extract keywords

          const keywords = userPost.match(keywordRegex);

          if (keywords) {

              keywords.forEach(keyword => {

                  tags.push(`[\*\*${keyword}\*\*]`);

              });

          }

          // Replace mentions and keywords in the original post

          const taggedPost = userPost.replace(postRegex, match => {

              if (match.startsWith('@')) {

                  return `[\*\*${match}\*\*]`;

              } else if (match.startsWith('#')) {

                  return `[\*\*${match}\*\*]`;

              } else {

                  return match;

              }

          });

          return taggedPost;

      }

      // Testing

      const userPost = "Great article by @john\_doe on #technology! :)";

      const taggedPost = tagUserPost(userPost);

      console.log(taggedPost);

**Explanation:**

Regular Expression (postRegex):

(?:(?:\w+\s\*)\*[@#]\w+\s\*)+: Matches mentions (@username), keywords (#keyword), or any sequence of characters.

Regular Expressions (mentionRegex and keywordRegex):

/@(\w+)/g: Matches mentions and extracts the username.

/#(\w+)/g: Matches keywords and extracts the keyword.

Function (tagUserPost):

Extracts mentions and keywords using the mention and keyword regular expressions.

Tags mentions and keywords using square brackets.

Replaces mentions and keywords in the original post.

**3. Testing and Tagging:**

**Write code that uses your regular expression to identify mentions and keywords**

**in a sample user post. For each mention and keyword found, tag them**

**appropriately using square brackets. For example, if the post is "Great article by**

**@john\_doe on #technology!", your code should tag it as "Great article by**

**[\*\*@john\_doe\*\*] on [\*\*#technology\*\*]!"**

**Solution:**

JavaScript Code that uses a regular expression to identify mentions and keywords in a sample user post and tags them appropriately using square brackets:

      // Regular expression for mentions and keywords

      const postRegex = /(?:\w+\s\*)\*[@#]\w+\s\*|\w+/g;

      const mentionRegex = /@(\w+)/g;

      const keywordRegex = /#(\w+)/g;

      // Function to tag mentions and keywords in a user post

      function tagUserPost(userPost) {

        // Match mentions and keywords using the postRegex

        const matches = userPost.match(postRegex);

        // Tag mentions

        const taggedPost = matches.map(match => {

          const mentions = match.match(mentionRegex);

          if (mentions) {

            mentions.forEach(mention => {

              const taggedMention = `[\*\*${mention}\*\*]`;

              match = match.replace(mention, taggedMention);

            });

          }

          return match;

        });

        // Tag keywords

        taggedPost.forEach((match, index) => {

          const keywords = match.match(keywordRegex);

          if (keywords) {

            keywords.forEach(keyword => {

              const taggedKeyword = `[\*\*${keyword}\*\*]`;

              taggedPost[index] = match.replace(keyword, taggedKeyword);

            });

          }

        });

        return taggedPost.join(' ');

      }

      // Sample user post

      const userPost = "Great article by @john\_doe on #technology! :)";

      const taggedPost = tagUserPost(userPost);

      // Output the tagged post

      console.log(taggedPost);

This code defines a regular expression (postRegex) to match mentions, keywords, and other text content in a user post. It then uses this regular expression to find matches in the sample user post. The code then identifies mentions and keywords separately using mentionRegex and keywordRegex and tags them appropriately using square brackets. Finally, the code outputs the tagged user post.

**4. Validation and Efficiency:**

**Discuss how you can validate the accuracy of your regular expression and what**

**steps you would take to ensure it performs efficiently, especially on many user**

**posts.**

**You must create grammar, implement a regular expression, and provide code in**

**any language to process and tag user posts. Additionally, they need to consider**

**validation and efficiency in their solution.**

**Solution:**

**Text Processing Component: Validation and Efficiency**

**1. Grammar Design:**

* **Overview**:

The grammar design is crucial for accurately capturing various elements in a user post, such as text content, mentions, keywords, and emoticons.

* **Grammar Rules:**

post: Represents the entire user post.

text: Represents any sequence of characters.

mention: Represents the format @username.

keyword: Represents the format #keyword.

emoticon: Represents emoticons like :), :D, ;).

**2. Regular Expression Implementation:**

* **Regular Expression Overview:**

To implement the grammar, a regular expression is employed. The regular expression is designed to match and extract mentions and keywords from a user post.

Regular Expression (postRegex):

regex

Copy code

(?:(?:\w+\s\*)\*[@#]\w+\s\*)+|\w+

* **Regular Expression Components:**

**Mentions (mentionRegex):**

**regex**

/@(\w+)/g

**Keywords (keywordRegex):**

**regex**

/#(\w+)/g

**3. Code Implementation:**

* **Tagging User Posts:**

A function (tagUserPost) is implemented to tag mentions and keywords using square brackets. This function also replaces mentions and keywords in the original post.

* **Sample Testing in JavaScript Code:**

const userPost = "Great article by @john\_doe on #technology! :)";

const taggedPost = tagUserPost(userPost);

console.log(taggedPost);

**4. Validation:**

* **Importance of Validation:**

Validation is crucial to ensure the accuracy of the regular expression. Testing against various user posts with different combinations of mentions, keywords, and emoticons helps validate the correctness of the grammar and the regular expression.

* **Testing Strategies:**
* **Positive Testing:**

Test with posts containing valid mentions, keywords, and emoticons to ensure correct tagging.

* **Testing:**

Test with posts lacking mentions or keywords to ensure graceful handling of absence.

**5. Efficiency:**

* **Performance Considerations:**

Efficiency is essential, especially when processing many user posts. Several strategies are employed to enhance performance:

* **Non-Capturing Groups:**

Use non-capturing groups (?: ... ) to improve performance by avoiding unnecessary captures.

* **Compilation:**

Compile the regular expression outside the loop if processing multiple posts for improved performance.

* **Optimization:**

Optimize the regular expression based on specific platform rules to reduce unnecessary matching and enhance overall efficiency.

**6. Conclusion:**

The combination of a well-designed grammar, an accurately crafted regular expression, and thorough validation and efficiency considerations results in a robust and performant text processing component for social media platforms.