Matlab Project

NAME: SOHAM KHAN
SECTION: B
ROLL NO : 60

IEM
INSTITUTE OF ENGINEERING & MANAGEMENT
Good Education, Good Jobs

TOPIC

Text Analysis and Feedback Module

Introduction

This project focuses on creating a Text Analysis and Feedback Module in MATLAB that detects and provides feedback on basic grammar errors. The goal is to develop a function capable of identifying common issues in English sentences, such as:

- Subject-Verb Agreement: Ensuring that subjects and verbs agree in number (singular/plural).
- Pluralization Errors: Checking for incorrect plural forms of nouns.

The MATLAB function will analyse input sentences, detect errors, and suggest corrections. This tool can be useful for students, writers, and anyone looking to improve their writing by receiving automated grammar feedback.

Grammar Checking

IEM
INSTITUTE OF ENGINEERING & MANAGEMENT
Good Education, Good Jobs

○What is Grammar Checking?

àGrammar checking is the process of analysing written text to identify and correct errors in syntax, punctuation, spelling, word choice, and sentence structure. It involves examining each part of a sentence —such as the subject, verb, object, and modifiers—to ensure they follow the rules of grammar. Grammar checking tools use algorithms and language models to detect patterns and inconsistencies, offering suggestions for improving sentence clarity and correctness. These tools help users refine their writing, making it more readable and professional

○Why is Grammar Checking Crucial?

à1) Clear Communication: Well-structured grammar helps convey ideas more clearly and prevents misunderstandings.

2) Professionalism: Correct grammar enhances the credibility of written work, making it more polished and professional.

3) Efficiency: Grammar checking tools save time by automating the error detection and correction process.

4) Improvement in Writing Skills: Regular use of grammar checkers helps users improve their writing by identifying recurring mistakes.

With grammar checking tools, anyone—whether a student, professional, or casual writer—can ensure their writing is accurate and effective.

Overview of MATLAB Function

The MATLAB function developed in this project is designed to detect basic grammar errors in English sentences, focusing on issues such as subject-verb agreement and pluralization. The function works by analysing input sentences and identifying common grammar mistakes, which are then flagged for review.

Key Features of the Function:

1) Error Detection: Identifies grammar mistakes, such as subject-verb mismatches and incorrect plural forms.
2) Automated Feedback: Provides suggestions for corrections, making it easier for users to improve their writing.
3) Ease of Use: Users input a sentence, and the function automatically processes and returns feedback.

This function is a simple yet powerful tool for detecting grammar issues in real-time, providing users with instant feedback to improve their written communication.

MATLAB approach

The MATLAB function uses a structured approach to detect grammar errors in sentences:

1) **Text Pre-processing**: Breaks down the input sentence into individual words and removes irrelevant words.

2) **Part-of-Speech (POS) Tagging**: Tags each word (e.g., noun, verb) to identify its grammatical role.

3) **Error Detection**: Checks for subject-verb agreement and pluralization errors by comparing tags and grammatical rules.

4) **Rule-Based** Corrections: Suggests changes when mismatches are found, e.g., adjusting verb forms to match the subject.

5) **Feedback Output**: Provides users with error notifications and correction suggestions.

This approach leverages MATLAB's text processing tools to offer clear, automated grammar feedback

Code walkthrough and Key MATLAB functions

Code Walkthrough

1) **Input & Tokenization**: Splits the input sentence into words for analysis.
2) **POS Tagging:** Tags each word (e.g., noun, verb) to identify sentence structure.
3) **Grammar Error Detection**:
a) **Subject-Verb Agreement:** Checks if subject and verb match in number.
b) **Pluralization**: Flags incorrect plural forms (e.g., “childs” vs. “children”).
4) **Correction Suggestions:** Generates fixes for errors (e.g., “She walk” ➔ “She walks”).
5) **Output Feedback**: Returns errors and suggestions for immediate user feedback.

Key MATLAB Functions Used

1) split(): Splits a sentence into individual words (tokens) for analysis.

2) contains(): Checks if specific words or patterns are present in the sentence to help identify potential errors.

3) strcmp(): Compares words or tags to identify mismatches, like subject-verb agreement issues.

4) Custom Functions (e.g., posTag() and checkSubjectVerbAgreement())

a)posTag(): A custom function to assign basic POS tags (noun, verb) to each word.

b)checkSubjectVerbAgreement(): Custom function to detect subject-verb agreement errors.

5) disp() : Displays the detected errors and suggestions for easy user feedback.

Example 2

```
% Predefined sentence
sentence = 'The cat chase the mouse';
disp('Original Sentence: ');
disp(sentence);

% Call the function to check grammar and correct the sentence
corrected_sentence = checkGrammar(sentence);

% Display the corrected sentence
disp('Corrected Sentence: ');
disp(corrected_sentence);

% Function to check grammar and suggest corrections
function corrected_sentence = checkGrammar(sentence)
  % Split sentence into words
  words = strsplit(sentence);
  num_words = length(words);

  % Basic rule for subject-verb agreement correction
  for i = 1:num_words-1
    % Example: correct "chase" to "chases" for singular subjects
like "The cat"
    if strcmpi(words{i}, 'chase') && strcmpi(words{i-1}, 'cat')
      words{i} = 'chases';
    end
  end

  % Reconstruct corrected sentence
  corrected_sentence = strjoin(words);
end
```

Output:

Original Sentence:
He are playing in the park
Corrected Sentence:
He is playing in the park

Example 2

```
% Predefined sentence
sentence = 'The cat chase the mouse';
disp('Original Sentence: ');
disp(sentence);

% Call the function to check grammar and correct the sentence
corrected_sentence = checkGrammar(sentence);

% Display the corrected sentence
disp('Corrected Sentence: ');
disp(corrected_sentence);

% Function to check grammar and suggest corrections
function corrected_sentence = checkGrammar(sentence)
  % Split sentence into words
  words = strsplit(sentence);
  num_words = length(words);

  % Basic rule for subject-verb agreement correction
  for i = 1:num_words-1
    % Example: correct "chase" to "chases" for singular subjects like "The cat"
    if strcmpi(words{i}, 'chase') && strcmpi(words{i-1}, 'cat')
      words{i} = 'chases';
    end
  end

  % Reconstruct corrected sentence
  corrected_sentence = strjoin(words);
```

Output:
Original Sentence:
He are playing in the park
Corrected Sentence:
He is playing in the park

Challenges Faced

1.Complex Sentence Structures: Struggled with nested clauses and compound subjects. Simplified error checks for basic structures.

2.POS Tagging Accuracy: Limited by MATLAB's basic tagging tools. Relied on rule-based methods for simplicity.

3.Subject-Verb Agreement in Edge Cases: Issues with irregular cases like "team have." Custom rules were difficult to generalize.

4.Contextual Understanding: Couldn't handle context-sensitive errors (e.g., "there" vs. "their"). Required advanced NLP techniques

CONCLUSION

This project successfully developed a MATLAB-based grammar checking function, achieving basic error detection for subject-verb agreement and pluralization issues. The function demonstrated the ability to analyse sentences, identify common grammar mistakes, and provide useful correction suggestions. Although limited to simpler sentence structures, this tool showcases MATLAB's potential in natural language processing tasks. Applications of this project include educational tools for basic grammar practice and integration into larger text-processing systems for more robust grammar checking.

Future scope

1. **Advanced Grammar Rules**: Add support for complex structures like compound sentences and passive voice.
2. **Enhanced POS Tagging:** Use advanced tagging methods for more accurate error detection.
3. **Contextual Understanding**: Integrate semantic analysis for context-sensitive errors (e.g., “there” vs. “their”).
4. **Machine Learning**: Apply machine learning for adaptive, data-driven grammar checks.
5. **User Interface**: Develop a user-friendly interface for practical applications

Thank You!

Created by:
SOHAM KHAN
B-60

IEM
INSTITUTE OF ENGINEERING & MANAGEMENT
Good Education, Good Jobs