The Islamia University of Bahawalpur

SOFTWARE DESIGN DESCRIPTION

BS in Artificial Intelligence

Modeling the Diffusion of Evolving Misinformation in Social Networks

Mam Quratulain Qureshi

Lecturer

Supervisor

Muhammad Aoun Cheema

F20BARIN1E02011

Fall 2020-2024

Table of Contents

Revision History

Name	Date	Reason for changes	Version

Application Evaluation History

Comments (by committee) *include the ones given at scope time both in doc and presentation	Action Taken

Supervised by

<Mam Quratulain Qureshi>

Signature_____

1 Introduction

The scope of the project encompasses the development and implementation of a Natural Language Processing (NLP)-based system with the primary objective of detecting and mitigating rumors and fake news within social networks. Focused on major modules like NLPProject, DataManager, LanguageModel, and EvaluationModule, the project emphasizes efficient data handling, robust language model training, and continuous adaptation to evolving linguistic patterns. With a user-friendly interface, the system caters to users like social media administrators and journalists. The scope extends to the responsible use of AI technologies, adhering to ethical considerations. The project's modular architecture ensures scalability and ease of maintenance, promoting continuous improvement and adaptability to emerging challenges in misinformation. Overall, the project aims to make a positive impact on information integrity by providing a sophisticated and user-centric solution for rumor and fake news detection

2 Design methodology and software process model

2.1 Design methodology:

Explanation and Justification:

Object-Oriented Programming (OOP) is chosen as the design methodology for the following reasons:

Modularity: OOP allows breaking down the system into modular, reusable components called objects. This modularity enhances maintainability and facilitates easier updates or changes to specific functionalities without affecting the entire system.

Encapsulation: OOP supports encapsulation, which means bundling data and methods that operate on the data into a single unit (class). This promotes information hiding and reduces the complexity of the system.

Inheritance: In the context of NLP projects, where there might be common functionalities or structures across different components, inheritance in OOP provides a way to reuse code and establish a hierarchy. This is particularly beneficial when dealing with different aspects of language processing.

Polymorphism: OOP allows polymorphism, enabling the use of a single interface to represent different types of objects. This can be advantageous when dealing with various data structures and algorithms commonly used in NLP.

2.2 Software Process Model:

Explanation and Justification:

The Iterative and Incremental Model is selected as the software process model for the following reasons:

Adaptability to Change: NLP projects often involve dealing with evolving datasets, models, and requirements. The iterative nature of this model allows for flexibility and accommodates changes more effectively than a rigid, sequential model.

Early Prototyping: Iterative development allows for the creation of prototypes early in the development cycle. This is particularly beneficial for an NLP project where experimenting with different algorithms, models, or pre-processing techniques may be crucial to achieving optimal results.

Feedback Loops: Incremental development facilitates regular feedback from stakeholders or end-users. This is valuable for refining the system based on user experiences, especially in the context of NLP, where the effectiveness of language models can heavily depend on real-world usage patterns.

Risk Management: Breaking down the project into smaller iterations aids in identifying and managing risks early in the development process. This is important for NLP projects, where uncertainties related to data quality, model performance, and evolving language patterns may arise.

3 System overview

3.1 Architectural design

Modular Program Structure:
Our NLP-based project is designed with a modular program structure to ensure maintainability, reusability, and flexibility. The main modules include:

Data Management Module (DataManager):

Functionality: Responsible for loading and preprocessing datasets.
Purpose: Ensures the availability of clean and formatted data for the training and fine-tuning processes.

Language Model Module (LanguageModel):

Functionality: Handles the training and fine-tuning of language models.
Purpose: Enables the development and improvement of language models based on specific datasets and tasks.

Project Management Module (NLPProject):

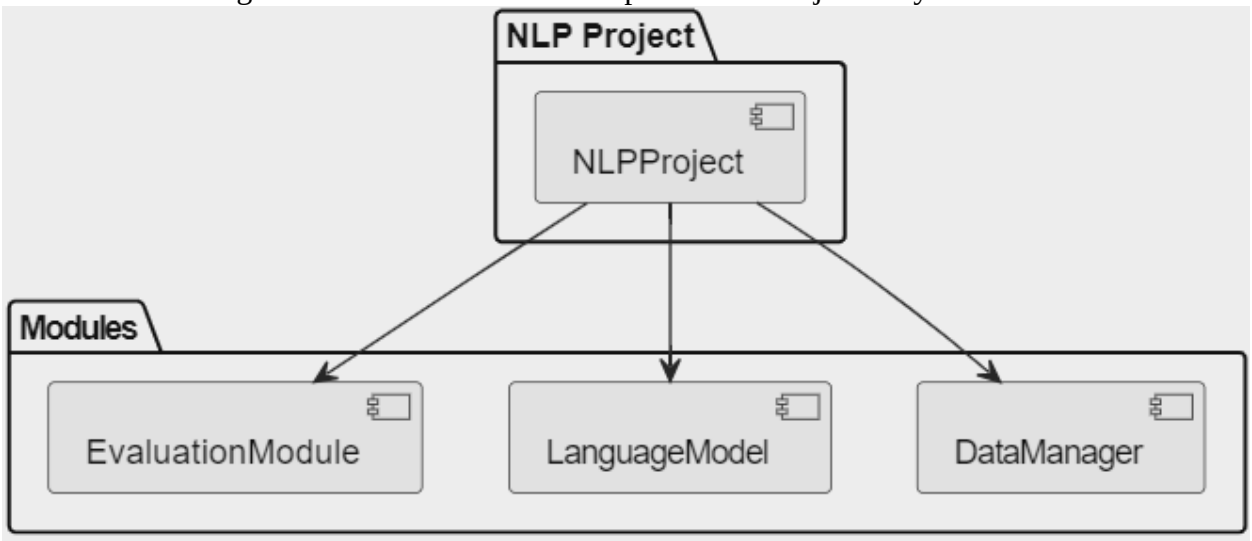
Functionality: Orchestrates the overall functionality of the system.
Purpose: Coordinates interactions between the DataManager, LanguageModel, and other modules. Initiates training and evaluation processes.

Evaluation Module (EvaluationModule):

Functionality: Evaluates the performance of the language models.
Purpose: Generates metrics to assess the effectiveness of the models in real-world scenarios.

Relationships Between Modules:
The NLPProject module acts as the central coordinator, interacting with the DataManager, LanguageModel, and EvaluationModule to achieve end-to-end functionality.
The DataManager provides necessary data to the LanguageModel for training and fine-tuning.
The EvaluationModule receives outputs from the LanguageModel and assesses its performance.

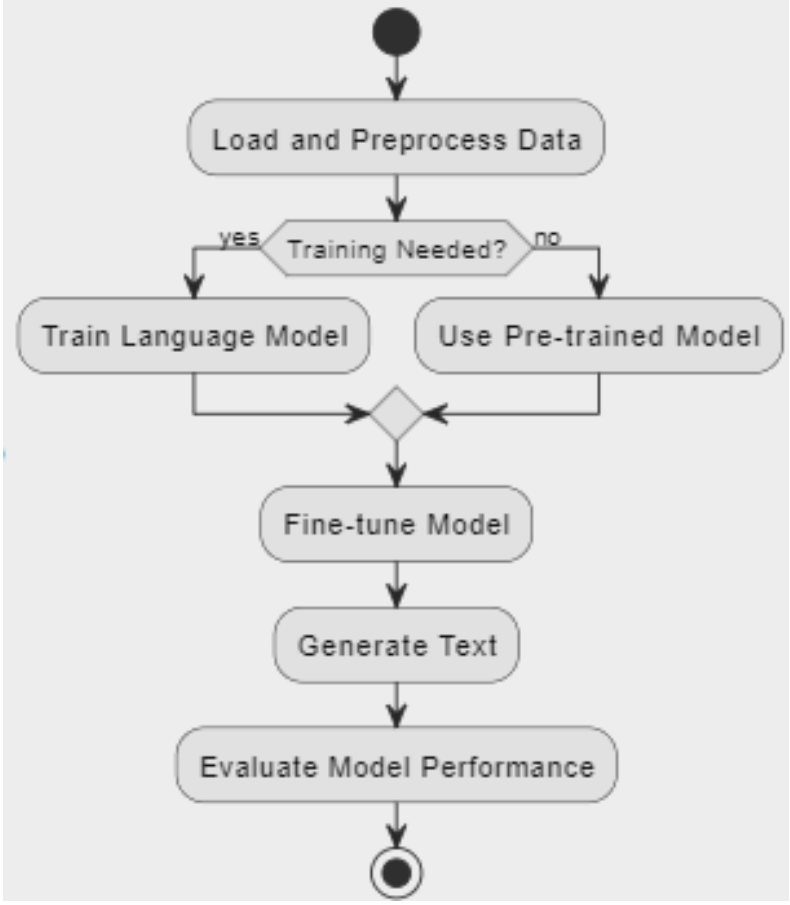
Diagram:
The Line-Box Diagram illustrates the relationships between major subsystems:



.

3.2 Process flow/Representation

Activity Diagram:
The activity diagram illustrates the flow of major processes in the system. Below is a simplified example:

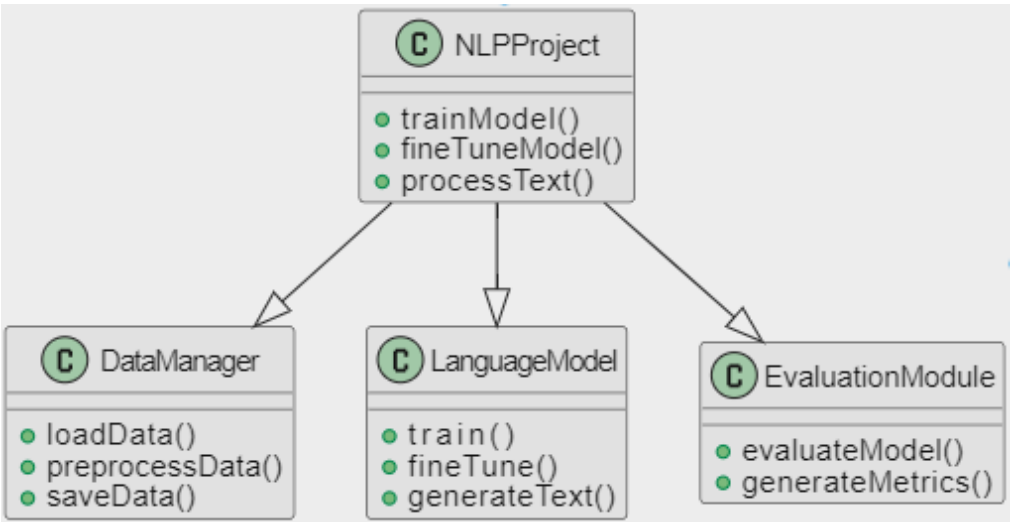


This activity diagram represents the sequence of activities, starting with data loading and preprocessing, followed by training or using a pre-trained language model, fine-tuning, text generation, and evaluation of model performance.

4 Design models [along with descriptions]

1. Class Diagram:

The Class Diagram provides a high-level view of the system's structure, depicting the classes, their attributes, methods, and relationships.



Class Diagram Description:

NLPProject orchestrates the project and interacts with other modules.

DataManager manages data loading, preprocessing, and storage.

LanguageModel handles training, fine-tuning, and text generation.

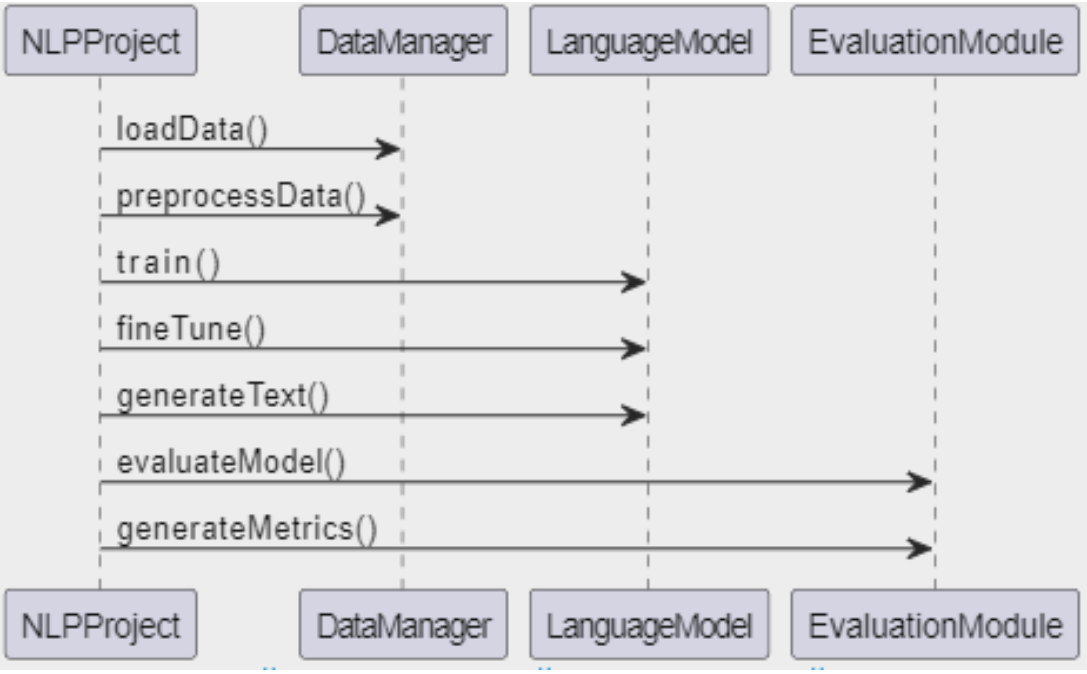
EvaluationModule evaluates the performance of the language models.

2. Sequence Diagram:

The Sequence Diagram illustrates the interaction between different components over time.

Sequence Diagram Description:

Describes the sequence of interactions between NLPProject, DataManager, LanguageModel, and EvaluationModule during the system's operation.

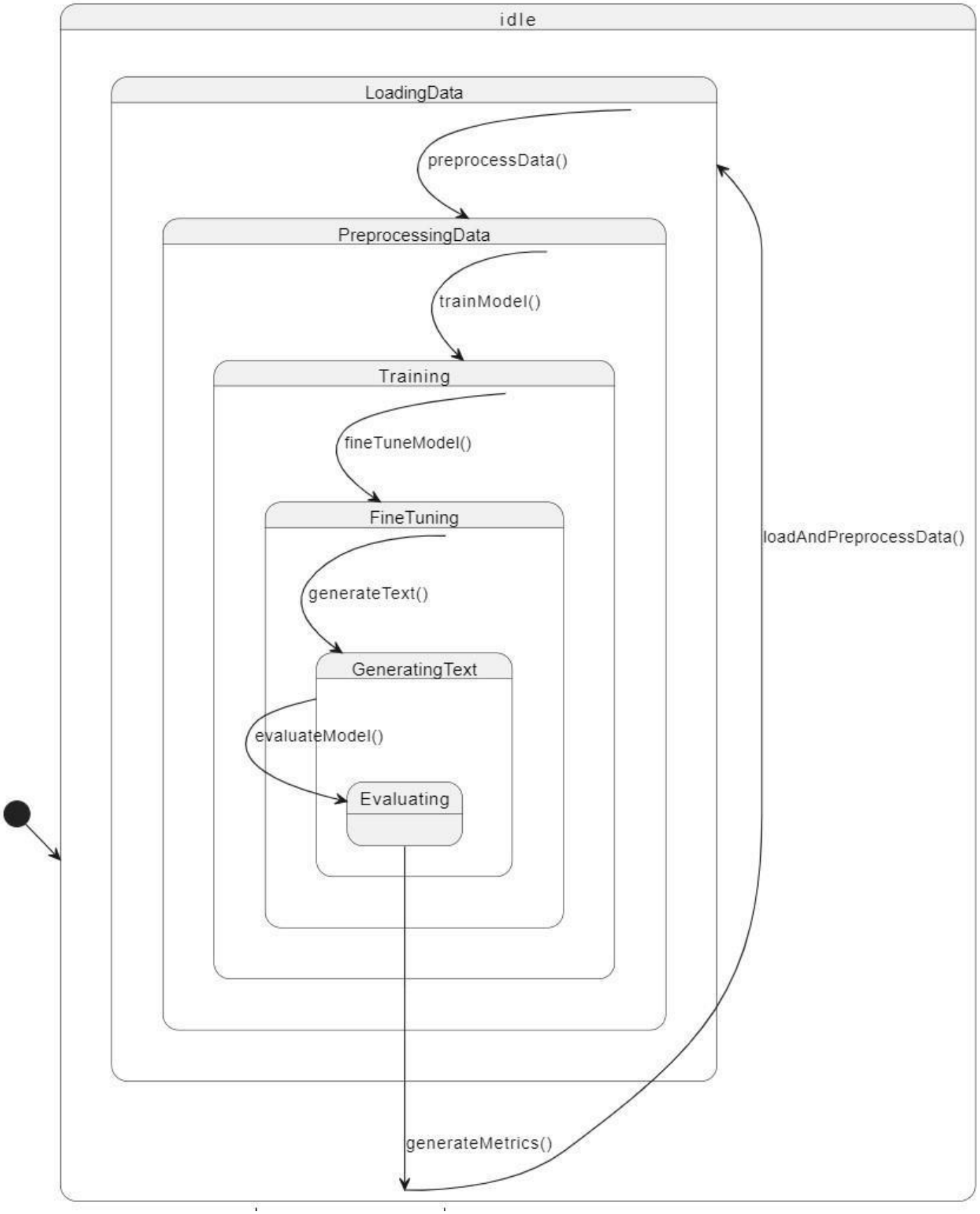


3. State Transition Diagram:

The State Transition Diagram models the different states and transitions of an object during its lifecycle.

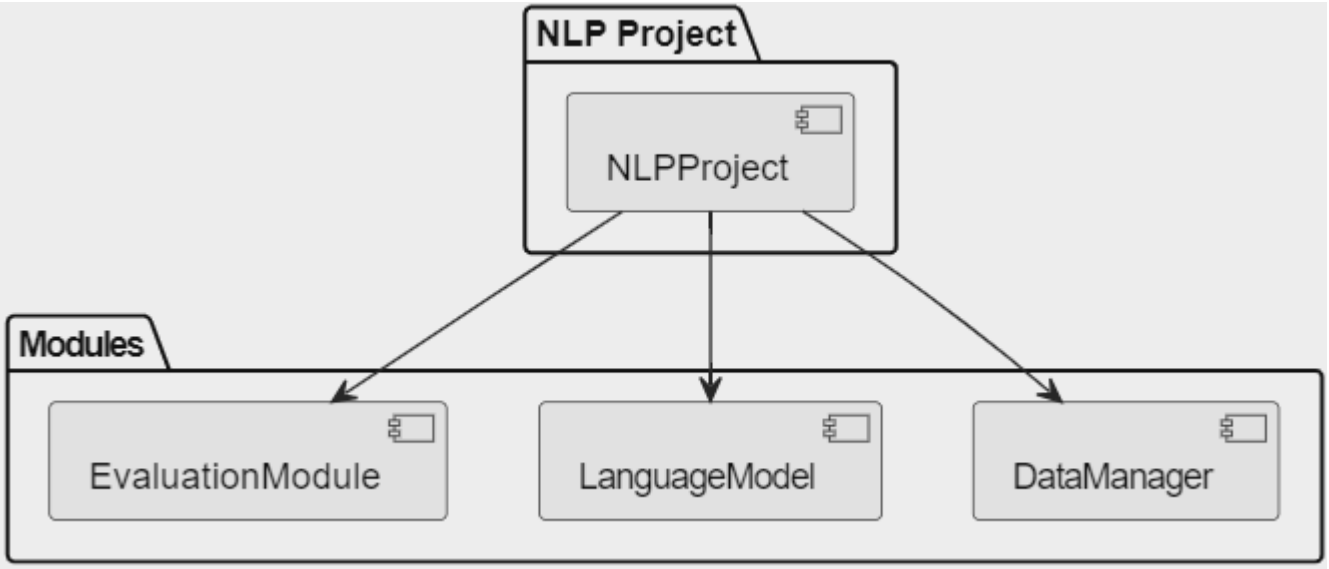
State Transition Diagram Description:

Represents the states (LoadingData, PreprocessingData, Training, Fine-tuning, Generating Text, Evaluating) and transitions between them during the execution of the NLP project.


idle
LoadingData
preprocessData()
PreprocessingData
trainModel()
Training
fineTuneModel()
FineTuning
generateText()
GeneratingText
evaluateModel()
Evaluating
generateMetrics()
loadAndPreprocessData()

4.1 Data Flow Diagram (Level 0):

The DFD Level 0 provides a high-level overview of processes, data sources/sinks, and data stores.

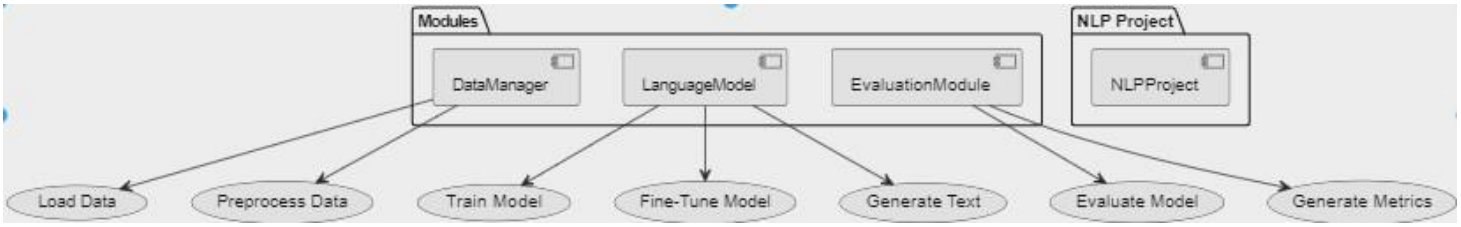


Data Flow Diagram Level 0 Description:

Illustrates the interaction between the main modules (NLPProject, DataManager, LanguageModel, EvaluationModule) in the system.

4.2 Data Flow Diagram (Level 1):

The DFD Level 1 provides more detail on the processes and their interactions.



Data Flow Diagram Level 1 Description:

Breaks down the processes in each module, including data flow between them.

5 Data design

In the context of an NLP project, the data design involves managing various types of data, including raw text data, preprocessed data, and model parameters. Here's an overview:

Raw Text Data:

Storage: Usually stored in plain text files or a structured database.
Processing: Loaded and preprocessed by the DataManager module.
Organization: Organized into datasets based on specific tasks.

Preprocessed Data:

Storage: Stored in a structured format, like tokenized sequences or feature vectors.
Processing: Generated by the DataManager module during the preprocessing step.
Organization: Structured for efficient training and model input.

Model Parameters:

Storage: Saved as weights and biases in a model file.
Processing: Handled by the LanguageModel module during training and fine-tuning.
Organization: Organized into layers and nodes within the neural network architecture.

6 Algorithm & Implementation

NLPProject Module:

Function: trainModel (rawTextData, modelParameters)
- Load rawTextData
- Preprocess rawTextData using DataManager
- Train LanguageModel with preprocessed data
- Save the trained model parameters to modelParameters

Function: fineTuneModel (rawTextData, modelParameters)
- Load rawTextData
- Preprocess rawTextData using DataManager
- Fine-tune LanguageModel with preprocessed data and existing modelParameters
- Update and save the fine-tuned model parameters to modelParameters

Function: processText(rawTextData)
- Load rawTextData
- Preprocess rawTextData using DataManager
- Generate text using the LanguageModel

DataManager Module:

Function: loadData (rawTextData)
- Load rawTextData from the source 7 Software requirements traceability matrix

Function: preprocessData (rawTextData)
- Tokenize and preprocess rawTextData
- Generate Preprocessed Data
- Save Preprocessed Data for future use

Function: save Data (preprocessedData)

- Save preprocessedData for future use

LanguageModel Module:

Function: train (preprocessedData)
- Initialize neural network model
- Train the model using preprocessedData
- Save trained model parameters

Function: fine-tune (preprocessedData, existingModelParameters)
- Load existingModelParameters
- Fine-tune the model with additional preprocessedData
- Save fine-tuned model parameters

Function: generateText (modelParameters)
- Load modelParameters
- Generate text using the trained model

EvaluationModule Module:

Function: evaluate Model (modelParameters, preprocessedData)
- Load modelParameters
- Evaluate the language model using preprocessedData
- Return evaluation results

Function: generate Metrics (modelParameters, preprocessedData)
- Load modelParameters
- Generate metrics based on evaluation results and preprocessedData
- Return metrics