Question Answering App Using NLP

Question Answering App Using NLP

Team Member

Dip Saha (BSSE 1001)

Ibrahim Khalil (BSSE 1009)

Supervised by

Dr. Ahmedul Kabir

Assistant Professor

Institute of Information Technology

University of Dhaka

Table of Contents

Chapter 1 **5**
1.1 Introduction 5
1.2 Purpose 5
1.3 Intended Audience 5
1.4 Conclusion 6
Chapter 2 **7**
2.1 Introduction 7
2.2 Inception of Question Answering APP 7
2.2.1 Identify the client of our project 8
2.2.2 Icebreaking 8
2.2.3 Identifying the stakeholders of the project 8
2.2.4 Identifying the multiple viewpoints of stakeholders 8
2.3 Conclusion 9
Chapter 3 **10**
3.1 Collaborative Requirements Gathering 10
3.2 Quality Function Deployment 10
3.2.1 Normal requirements 11
3.2.2 Expected requirements 11
3.3 Usage Scenario 12
3.4 Use Case Diagram 13
3.4.1 Definition of Use Case 13
3.4.2 Primary Actor 14
3.4.3 Secondary Actor 14
Level :0 14
Level :1 15
Description of use case diagram level 1: 16
Level : 1.1 17
Description of use case diagram level 1.1: 18
Level : 1.2 19
Description of use case diagram level 1.2: 20
3.5 Activity Diagram 21
3.5.1 Definition of Activity Diagram 21
Level : 1 21
Level : 1.1 22
Level : 1.2 23

Level : 1.3 24
3.6 Swimlane Diagram 25
3.6.1 Definition 25
SID(Swimlane ID) : 1 25
SID(Swimlane ID) : 1.1 27
SID(Swimlane ID) : 1.2 28
SID(Swimlane ID) : 1.3 28
3.7 Data Based Modelling 29
3.7.1 Concept 29
3.7.2 Data Objects 30
3.7.3 Data object identification 30
3.8 Class-Based Modelling 30
3.8.1 Concept 30
3.8.2 Noun list from Question Answering App 31
3.8.3 Verb list from Question Answering App 31
3.8.4 General classification 32
3.8.5 Selection Criteria 33
3.8.6 Attribute and Method Identification 35
3.8.7 Analysis 36
3.8.8 Class Cards 36
3.8.9 CRC Diagrams 40
3.9 BEHAVIORAL MODELING OF Question Answering App 41
3.9.1 STATE TRANSITION DIAGRAM : 41
3.10 Data Flow Diagram (DFD) 41
3.10.1 Definition : 41
3.10.2 0-level DFD: 41
Description: 42
3.10.3 1-level DFD: 42
Description(1-level DFD): 44
3.10.4 2-level DFD: 46
Description: Vectorization 46
Description: Modeling Layer 48
Description: Output Layer 49

Chapter 1

1.1 Introduction

This chapter is a part of our software requirement for the project “Question Answering App Using NLP”. In this chapter we will focus on the audience for this project.

1.2 Purpose

This document briefly describes the Software Requirements Analysis of Question Answering App. It contains the functional, non-functional and the supporting requirements and establishes a requirement’s baseline for the development of the system. The requirements contained in the SRS are independent, uniquely numbered and organized by topics. The SRS serves as an official means of communicating user requirements to the developer and provides a common reference point for both the developer team and the stakeholder community. The SRS will evolve over time as users and developers work together to validate, clarify and expand its contents.

1.3 Intended Audience

- This SRS is intended for all types of users.
- The user and admin will use this SRS to verify that the developer team has created a product that is acceptable to the customer.
- The project manager of the development team will use this SRS to plan milestones and a delivery date and ensure that the developing team is on track during development of the system.

- The designers will use this SRS as a basis for creating the system’s design. The designers will continually refer back to this SRS to ensure that the system they are designing will fulfill the customer’s needs.
- The developers will use this SRS as a basis for developing the system’s functionality. The developers will link the requirements defined in this SRS to the software they create to ensure that they have created a software that will fulfill all of the customer’s documented requirements .

- The tester will use this SRS to derive test plans and test cases for each documented requirement. When portions of the software are complete, teh tester will run their tests on that software to ensure that the software fulfills the requirements documented in this SRS. The tester will again run their tests on the entire system when it is complete and ensure that all requirements documented in this SRS have been fulfilled.

1.4 Conclusion

This analysis of the audience helped us to focus on the users who will be using our analysis. This overall document will help each and every person related to this project to have a better idea about the project.

Chapter 2
Inception

In this chapter, the inception part of the SRS will be discussed briefly.

2.1 Introduction

Question Answering App is an application in which an user can input a context and asks questions on the basis of that context. The system will give answers to the questions from the given context.

2.2 Inception of Question Answering APP

At the beginning of our project, we entered the inception stage. This stage includes how the project will be started and their scope and limitations. The main goal of this phase is to identify the requirements, demand and establish some sort of mutual understanding between the software team and the stakeholders of the app. In order to make this phase effective we take the following steps :

- Identify the client of our project.
- Icebreaking.
- Identifying the stakeholders of the project.
- Identifying the multiple viewpoints of stakeholders.

2.2.1 Identify the client of our project

There is no client for our project.

2.2.2 Icebreaking

Icebreaking refers to the fact that to diminish the communication barrier between two persons. It is a crucial part since it denotes the acceptance of our proposal. We started this phase by talking with the stakeholders with context free languages. Their behaviour, responding to our question impacted the whole system.

2.2.3 Identifying the stakeholders of the project

Stakeholders refers to any person or group who will be affected directly or indirectly by the system. Stakeholders include end-users who interact with the system and everyone else in an organization who may be affected by its installation. Any kind of people can be the stakeholders of our project.

2.2.4 Identifying the multiple viewpoints of stakeholders

Different stakeholders expect different benefits from the system as every person has his own point of view. So, we have to recognize the requirements from multiple viewpoints. Different viewpoints of the stakeholders about the expected software are given below :

- Easy and fast interface.
- Mobile platform based software.
- Provide best possible answers to the given questions.

- Fast response to the input.

2.3 Conclusion

The primary goal of the project is to model and design a software for all users so that they get a question answering interface. The software will be as simple as an user can easily be able to use. The software will be designed in such a way as it takes very little time to manage. To make this software project successful, collaboration with stakeholders was a main priority that what they want, how the software will work, how it can be more convenient, how it will save time and energy etc.

Chapter 3

Elicitation

We have a Question and Answer (Q & A) approach in the previous chapter where the inception phase of requirement engineering has been described. The main task of this phase is to combine the elements of problem solving, elaboration, negotiation and specification. The collaborative working approach of the stakeholders is required to elicit the requirements. We have finished the following tasks for eliciting requirements-

- Collaborative Requirements Gathering.
- Quality Function Deployment.
- Usage Scenarios.

3.1 Collaborative Requirements Gathering

We have met with many stakeholders in the inception phase. These meetings created an indecisive state for us to elicit the requirements. To solve this problem, we have met with the stakeholders (who are acting a vital role in the whole process) a few times to elicit the requirements.

3.2 Quality Function Deployment

Quality function deployment (QFD) is a technique that translates the needs of the customer into technical requirements for software. Ultimately the goal of QFD is to translate subjective quality criteria into objective ones that can be quantified and measured and which can then be used to design and manufacture the product. It is a methodology that concentrates on maximizing customer satisfaction from the software engineering process. So, we have followed this methodology to identify the requirements for the project. The requirements, which are given below, are identified successfully by the QFD.

3.2.1 Normal requirements

Normal requirements are generally the objectives and goals that are stated for a product or system during meetings with the customer, The presence of these requirements fulfills customer satisfaction. These are the normal requirements for our project -

- User will give a context as input from which he wants to ask questions.
- After inputting the context he will be able to ask questions from there.
- The system will then display the answer to the user’s question.
- Context, query and answer will be in English.

3.2.2 Expected requirements

These requirements are intrinsic to the product or system and may be so elementary that the customer does not explicitly state them. Their absence will be a cause for significant dissatisfaction. Below the expected requirements for our project are briefly described.

- The system will respond very quickly.
- Will answer the questions from users as soon as possible.
- Context input can be given on any topic.

3.3 Usage Scenario

Question Answering App using NLP is an automated system where a user can give a context as input from which he wants to ask questions. After inputting the context he will be able to ask questions from there. The system will then display the answer to the user's question. This App uses Bi-Directional Attention Flow (BiDAF) model to find the answers. Here is a brief description of the BiDAF model-

BiDAF is a closed-domain, extractive Q&A model that can only answer factoid questions. These characteristics imply that BiDAF requires a Context to answer a Query. The answer that BiDAF returns is always a substring of the provided Context. At the beginning, the context and queries are tokenized through Tokenization. Then the tokenized words are passed through the embedding layers. BiDAF uses 3 levels of Embedding layers to represent words as vectors.

i) **Word embedding**: BiDAF uses pre-trained GloVe embeddings to get the vector representation of words in the Query and the Context. The output of the word embedding step is two matrices — one for the Context and one for the Query.

ii) **Character embedding:** Glove can't handle OOV(out of vocabulary) words. For this reason we use Character embedding to handle OOV words properly. In this process we use "pre trained wiki-news 300d" vectors for character embedding.The output of the character embedding step is similar to the output of the word embedding step.

iii) **Contextual Embedding:** The output of word embedding and character embedding steps will be passed through Highway network. The highway network's role is to adjust the relative contribution from the word embedding and the character embedding steps. Then the output vector will be passed through 2 bidirectional LSTM layers to gather the contextual meaning of the words.

So, after completion of the embedding process, we will get a vector representation for every word in the context and query. These Query and Context representations then enter into attention and modeling layers. In the attention layer, several operations will be performed such as formation of similarity matrix, query to context attention, context to query attention and megamerge. The output of these steps is another representation of the Context that contains information from the Query. This output is referred to as the “Query-aware Context representation.”

The Query-aware Context representation is then passed into the output layer, which will transform it to a bunch of probability values. These probability values will be used to determine where the Answer starts and ends.

3.4 Use Case Diagram

3.4.1 Definition of Use Case

A use case captures a contract that describes the system under various conditions as the system responds to a request from one of its stakeholders. In essence, a Use Case tells a stylized story about how an end user interacts with the system under a specific set of circumstances. A Use Case diagram simply describes a story using corresponding actors who perform important roles in the story and makes the story understandable for the users. The first step in writing a Use Case is to define that set of actors that will be involved in the story. Actors are different people that use the system or product within the context of the function and behaviour that is to be described. Actors represent the roles that people play as the system operators. Every user has one or more goals when using the system.

3.4.2 Primary Actor

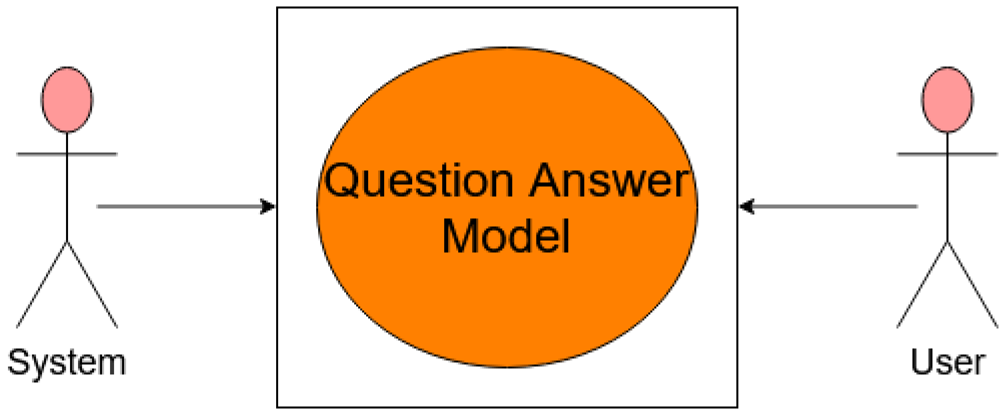
Primary actors interact directly to achieve required system function and derive the intended benefit from the system. They work directly and frequently with the software

3.4.3 Secondary Actor

Secondary actors support the system so that primary actors can do their work. They either produce or consume information.

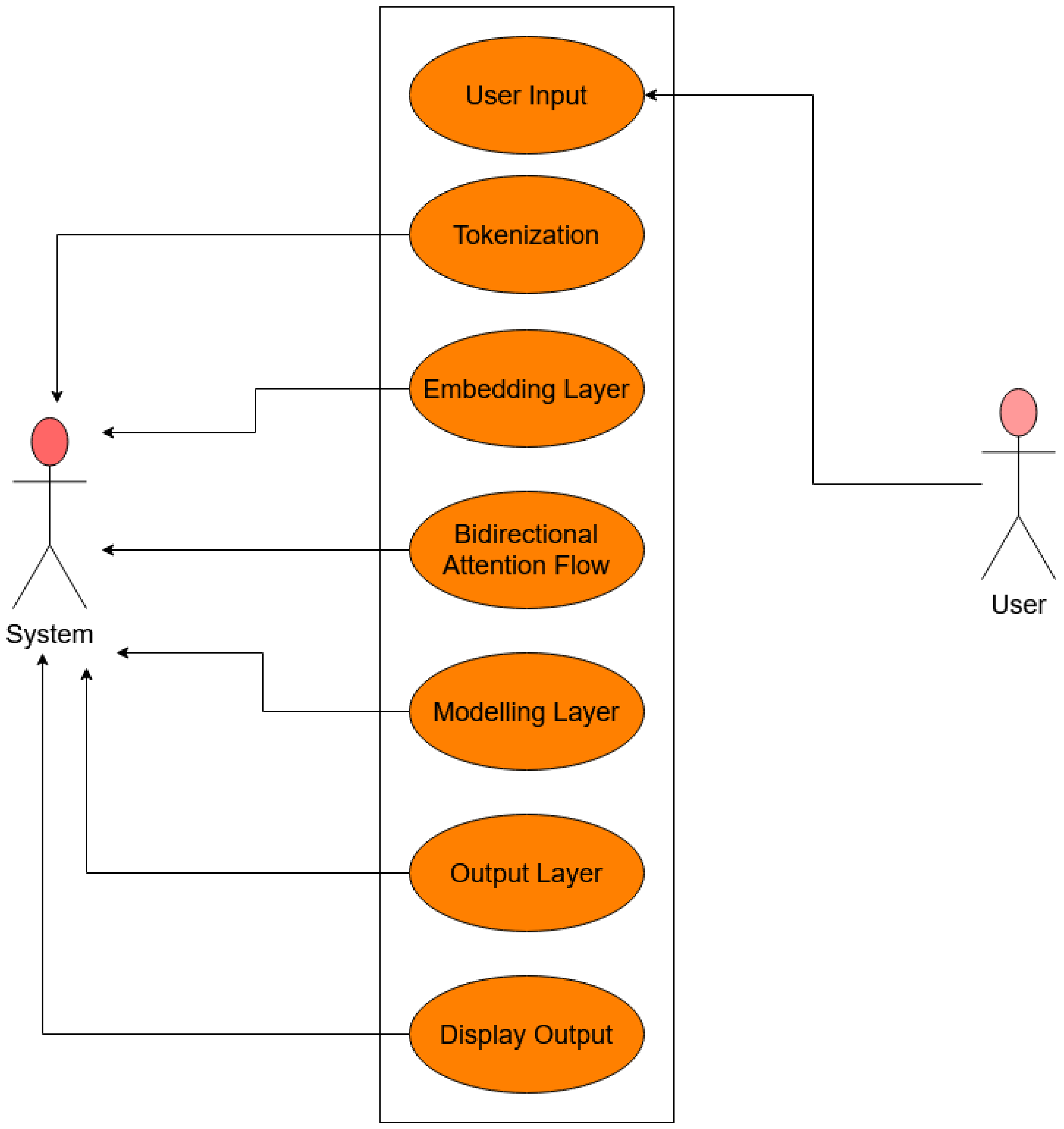
Level :0

Name : Question Answer Model
Primary actor : System
Secondary actor : User



Level :1

Name : Question Answer Model
Primary actor : System
Secondary actor : User



Description of use case diagram level 1:

User Input : User will input a context and ask questions on the basis of that context.

Tokenization : The incoming Query and its Context are first tokenized, i.e. these two long strings are broken down into their constituent words.

Embedding Layer : The resulting words are then subjected to the embedding process, where they are converted into vectors of numbers. These vectors capture the grammatical function (syntax) and the meaning (semantics) of the words. Let the total number of words in context and query be T and J respectively. A (2d x T) and a (2d x J) matrix will be obtained after this process where d is the dimension of vector representation of a word.

BiDirectional Attention Flow : combines the information from the Context and the Query. Context matrix H (2d by T) and Query matrix U (2d by J) will enter into this process and a Giant Matrix G (8d by T) will be obtained after the completion of the process. This process combines some subprocess such as formation of similarity matrix, context to query attention, query to context attention and megamerge.

Modelling Layer : This layer consists of two layers of bi-LSTM. As mentioned above, the input to the modeling layer is G. The first bi-LSTM layer converts G into a 2d-by-T matrix called M1.

M1 then acts as an input to the second bi-LSTM layer, which converts it to another 2d-by-T matrix called M2.

Output Layer : In the output layer, M1 and M2 are first vertically concatenated with G to form [G; M1] and [G; M2]. Both [G; M1] and [G; M2] have a dimension of 10d-by-T.

We then obtain **p1**, the probability distribution of the start index over the entire Context, by the following steps:

p1= Softmax ($W^{T}_{(p1)}$ [G; M1]) where $W^{T}_{(p1)}$ is a trained 1-by-10d weight vector.

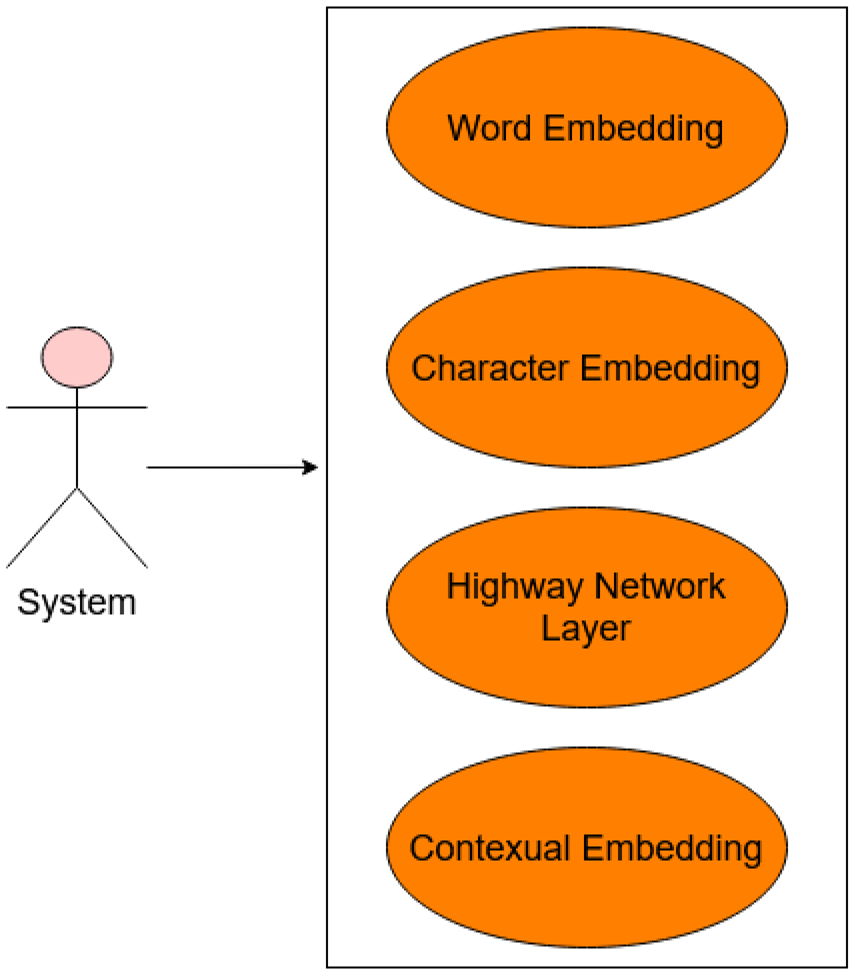
Similarly, we obtain **p2**, the probability distribution of the end index, by the following steps:

p2= Softmax ($W^{T}_{(p2)}$ [G; M2]) where $W^{T}_{(p2)}$ is a trained 1-by-10d weight vector.

View Output : p1 and p2 are then used to find the best Answer span. The best Answer span is simply a substring of the Context with the highest span score. The span score, in turn, is simply the product of the p1 score of the first word in that span and the p2 score of the last word in the span. The answer with the best span score will be displayed.

Level : 1.1

Name : Embedding Layer
Primary Actor : System



Description of use case diagram level 1.1:

Word Embedding : This process uses pre-trained GloVe embeddings to get the vector representation of words in the Query and the Context. The output of the word embedding step is two matrices — one for the Context and one for the Query.

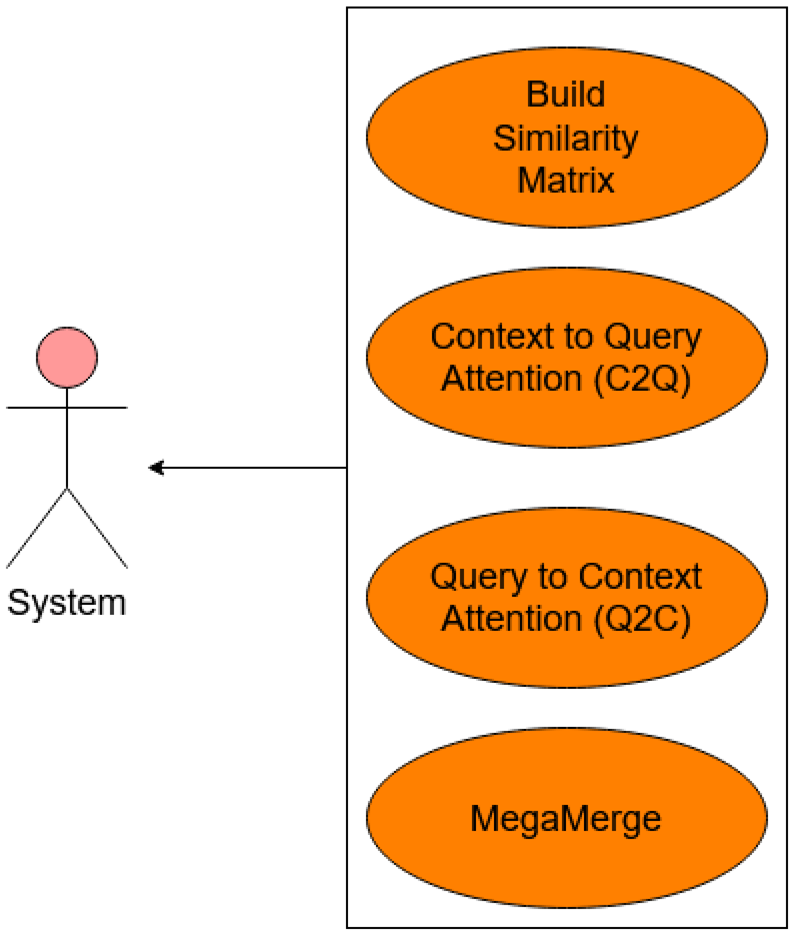
Character Embedding : Glove can't handle OOV(out of vocabulary) words. For this reason we use Character embedding to handle OOV words properly. In this process we use "pre trained wiki-news 300d" vectors for character embedding.The output of the character embedding step is similar to the output of the word embedding step.

Highway Network Layer : At this point, we have obtained two sets of vector representations for our words — one from the GloVe (word) embedding and the other from character embedding. We then vertically concatenate these representations. So, we'll get two 2D matrices (one for context and another for query). These matrices are then passed through a so-called highway network. The highway network's role is to adjust the relative contribution from the word embedding and the character embedding steps.

Contextual Embedding : We need an embedding mechanism that can understand a word in its context. This is where the contextual embedding layer comes in. The contextual embedding layer consists of Bi-Directional Long-Short-Term-Memory (Bi-LSTM) sequences.

Level : 1.2

Name : Bidirectional Attention Flow (BiDAF)
Primary Actor : System


Build Similarity Matrix
Context to Query Attention (C2Q)
Query to Context Attention (Q2C)
MegaMerge
System

Description of use case diagram level 1.2:

Build Similarity Matrix : A similarity matrix will be built with a dimension of **T**-by-**J** (number of words in Context by number of words in the Query).
The Context matrix(H), the Query matrix(U) and their element wise multiplication are concatenated. The results of concatenation are then multiplied by a trainable weight vector and form the similarity matrix(S).

Context to Query Attention : The goal of this step is to find which Query words are most relevant to each Context word. This process gives a matrix C2Q with dimension 2d by T.

Query to Context Attention : The goal of this step is to find which Context word is most similar to either one of the Query words hence are critical for answering the Query. This process gives a matrix Q2C with dimension 2d by T.

Mega Merge : Then the matrices:context vector(H), context to query attention vector (C2Q) and query to context attention vector (Q2C) are concatenated and form (8d x T) matrix (G). The giant matrix G contains all information in H, C2Q and Q2C.

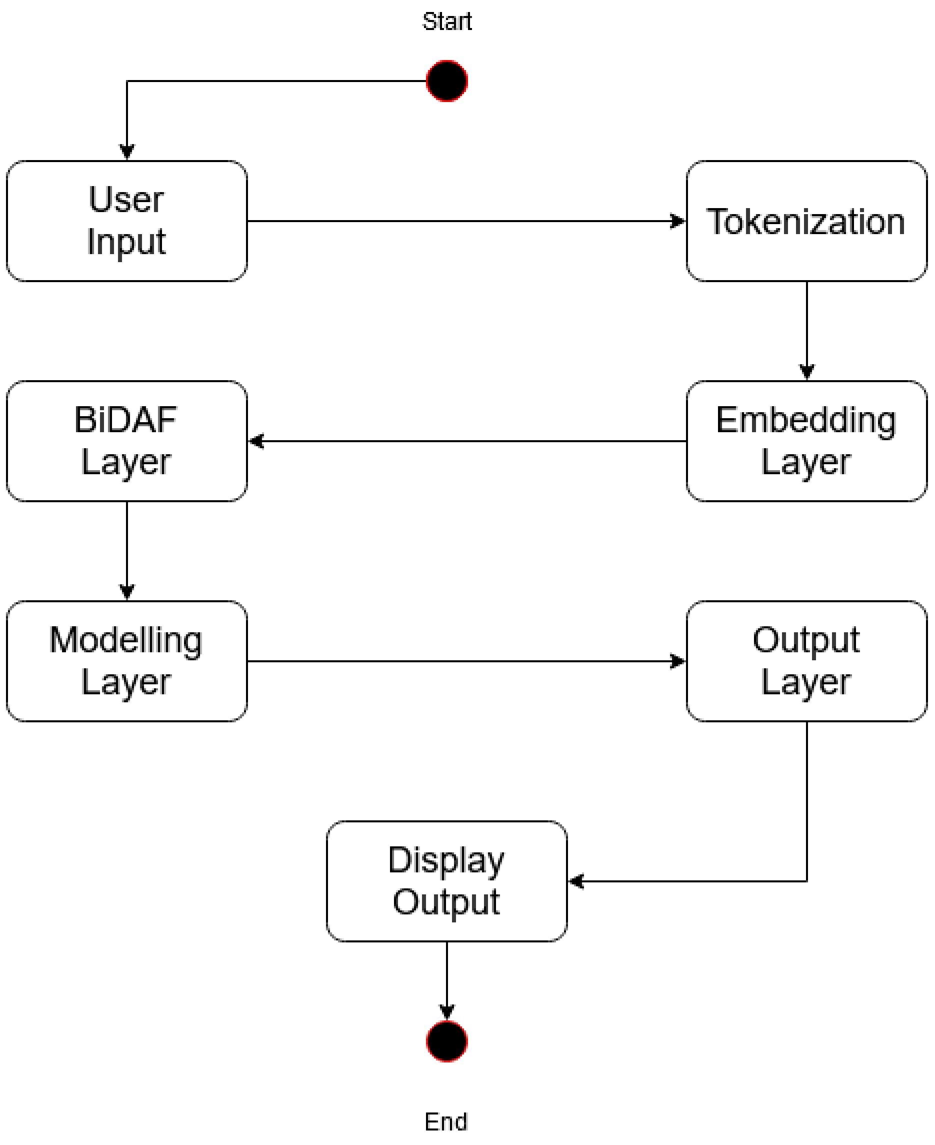
3.5 Activity Diagram

3.5.1 Definition of Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency.

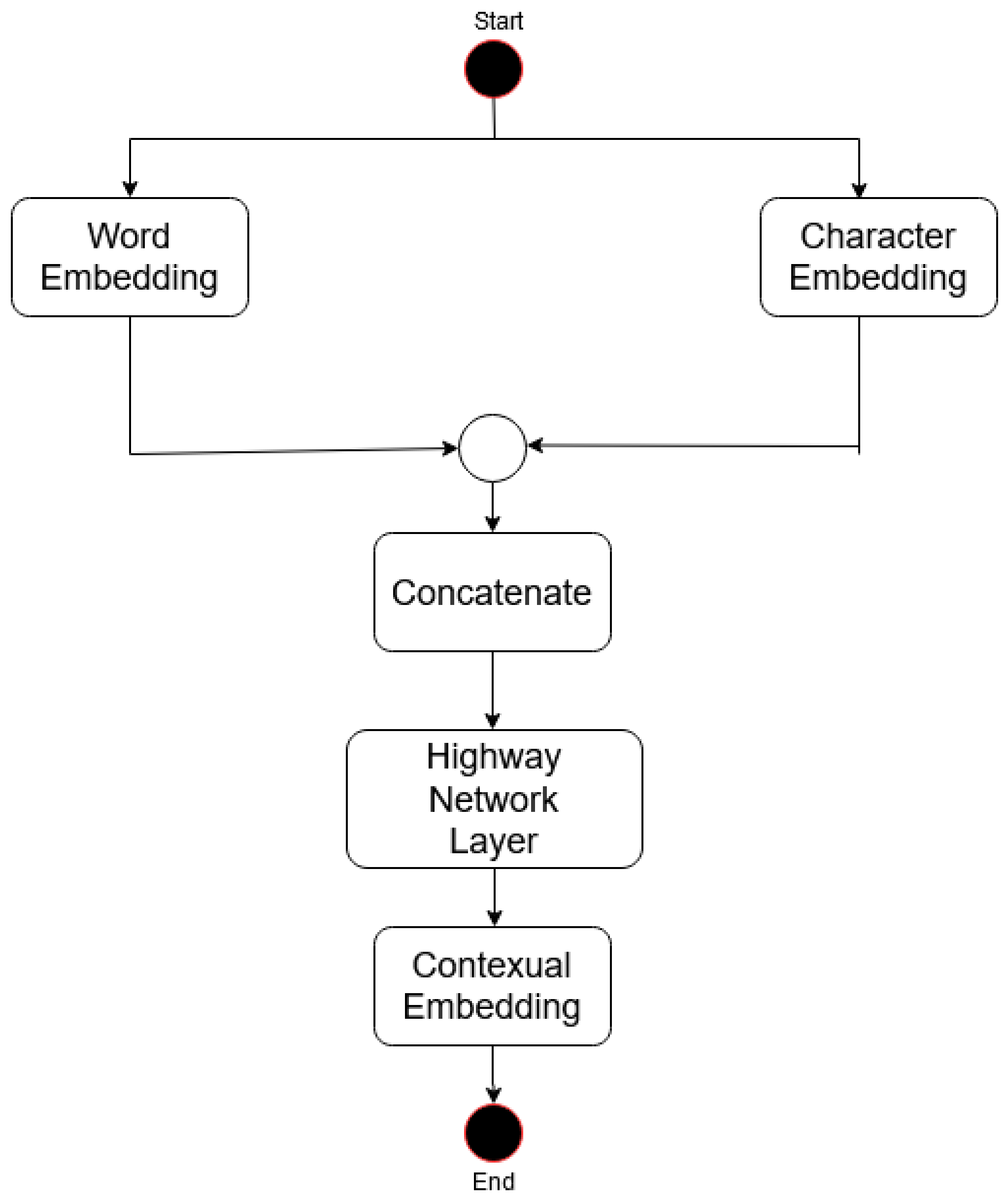
Level : 1

Name : Question Answer Model
Reference : Use Case level 1



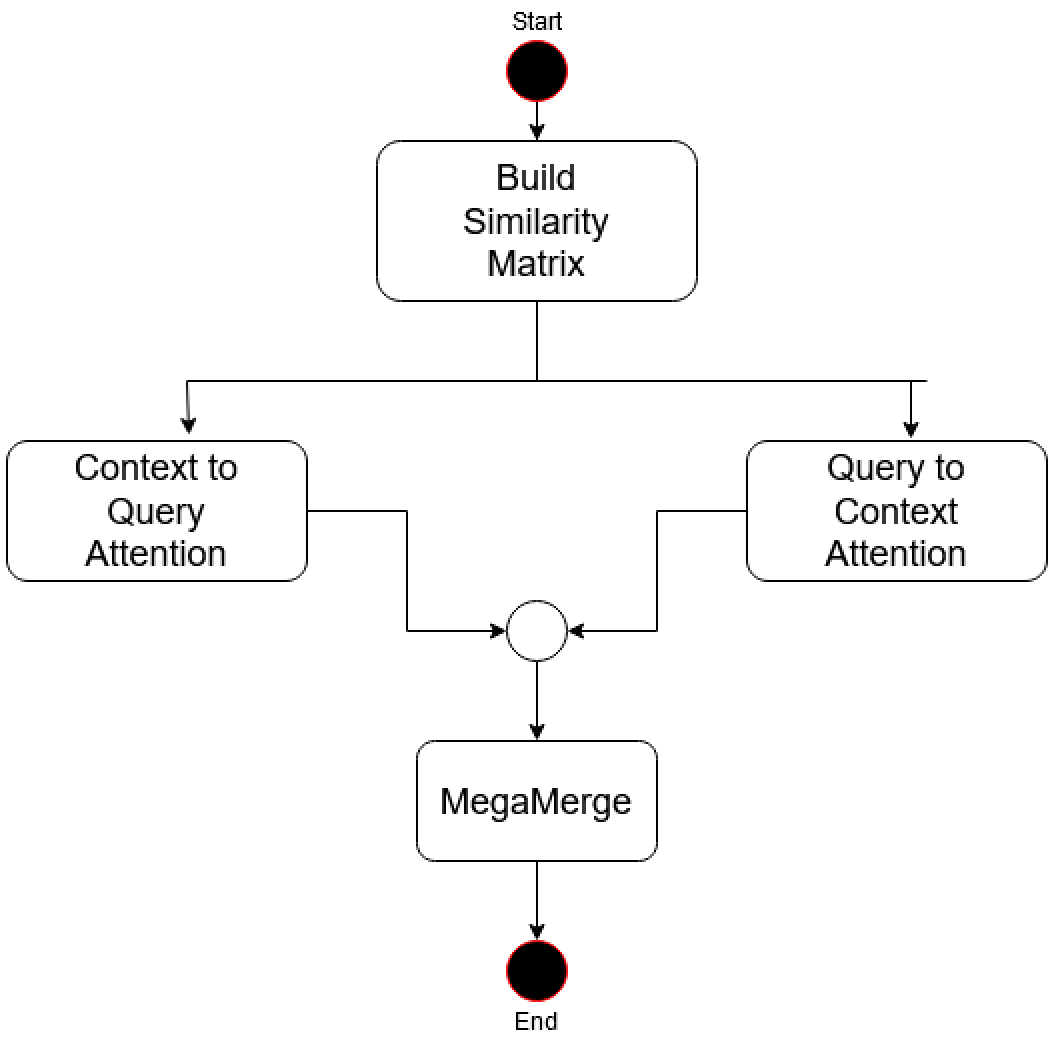
Level : 1.1

Name : Embedding Layer
Reference : Use Case level 1.1



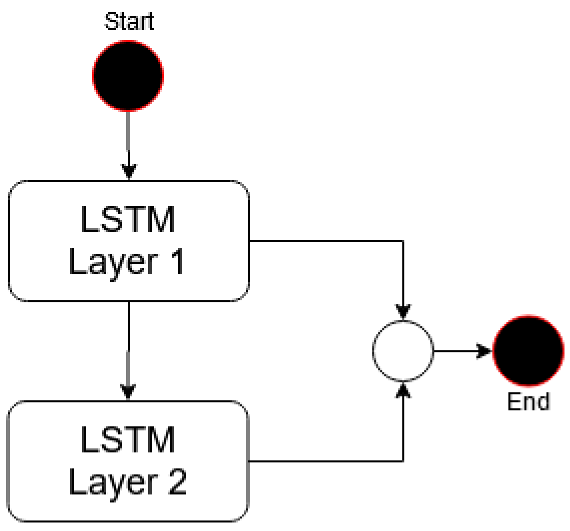
Level : 1.2

Name : BiDAF Layer
Reference : Use Case level 1.2



Level : 1.3

Name : Modelling Layer



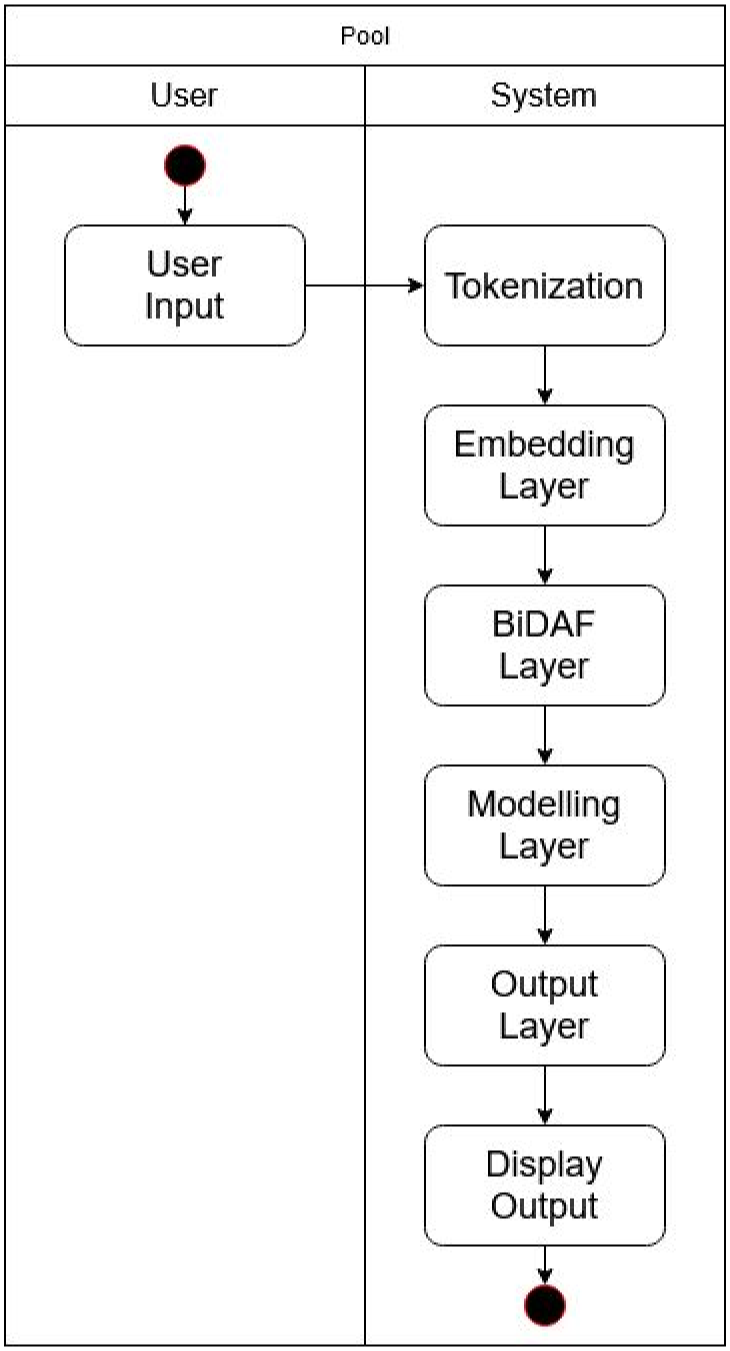
3.6 Swimlane Diagram

3.6.1 Definition

A swimlane diagram is a type of flowchart that delineates who does what in a process of lanes in a pool, a swimlane diagram provides clarity and accountability by placing process steps within the horizontal or vertical “swimlane” of a particular employee, work group or department. It shows connections, communication and handoffs between these lanes, and it can serve to highlight waste, redundancy and inefficiency in a process.

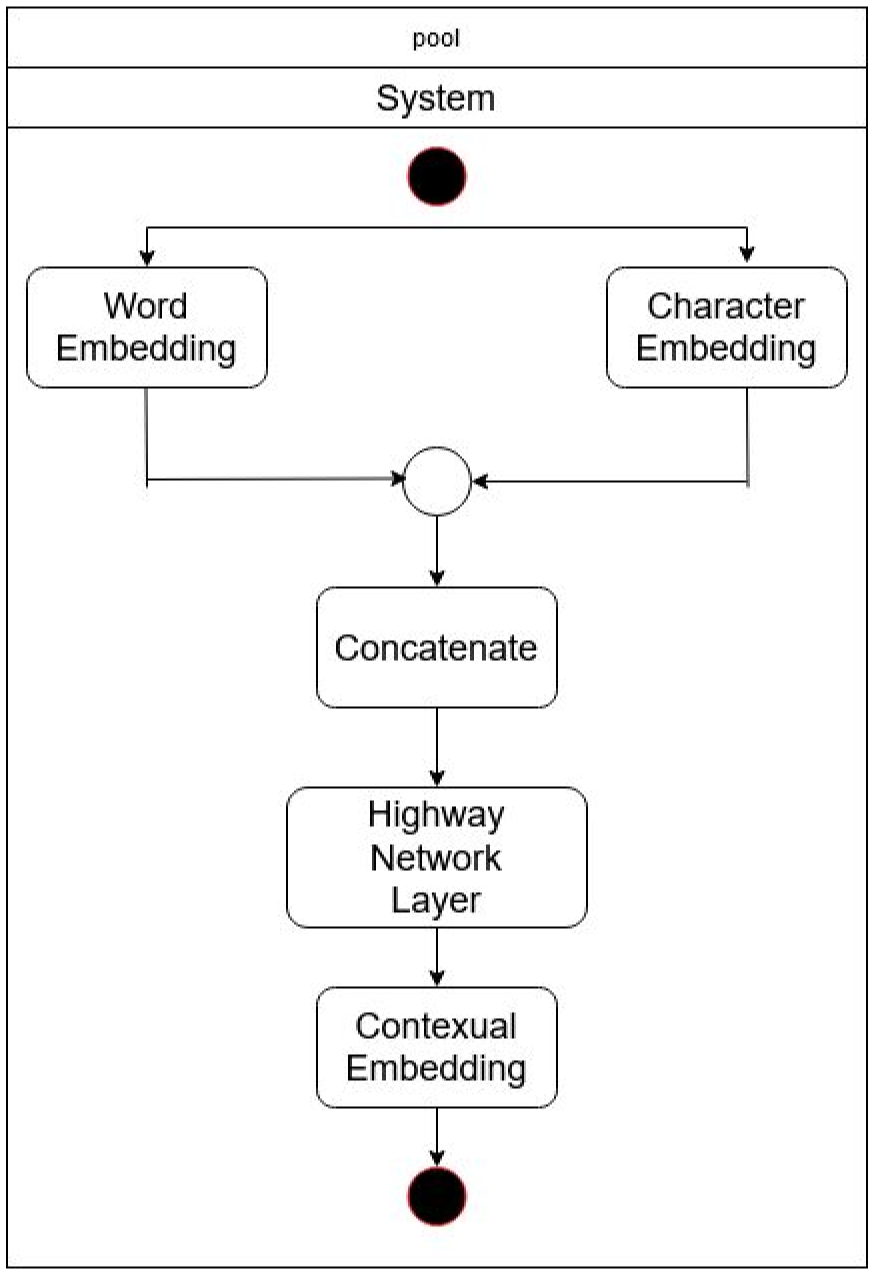
SID(Swimlane ID) : 1

Name : Question Answer Model
Reference : Use Case & Activity level 1


Pool
User
System
User Input
Tokenization
Embedding Layer
BiDAF Layer
Modelling Layer
Output Layer
Display Output

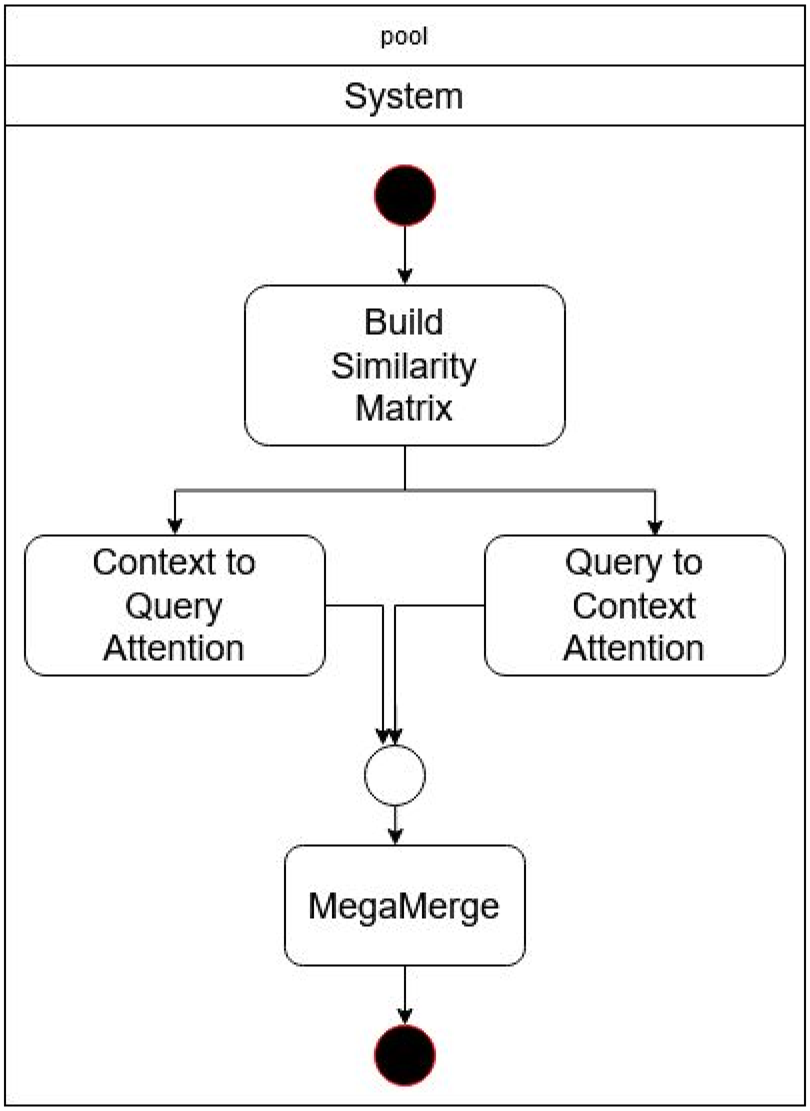
SID(Swimlane ID) : 1.1

Name : Embedding Layer
Reference : Use Case & Activity level 1.1



SID(Swimlane ID) : 1.2

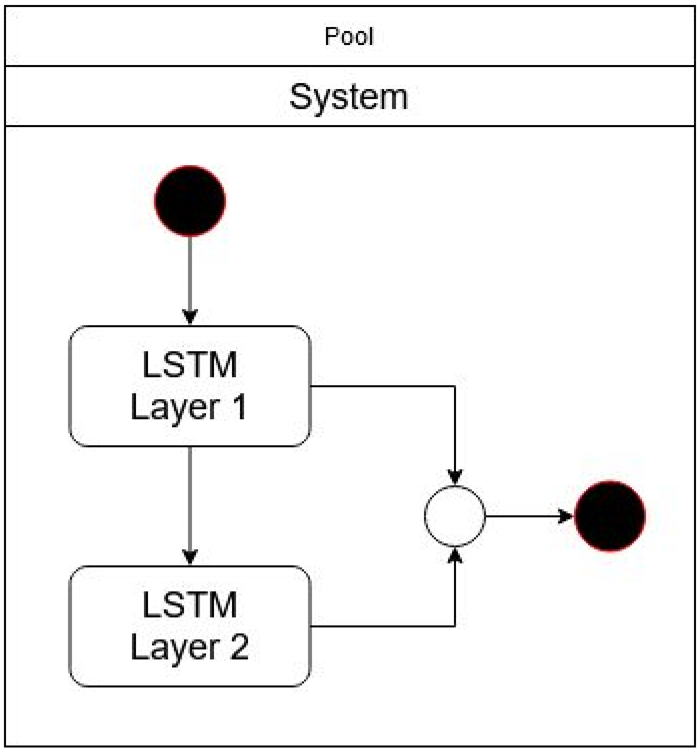
Name : BiDAF Layer
Reference : Use Case & Activity level 1.2



SID(Swimlane ID) : 1.3

Name : Modelling Layer

Reference : Activity level 1.3



3.7 Data Based Modelling

3.7.1 Concept

If software requirements include the necessity to create, extend or interact with a database or complex data structure that needs to be constructed and manipulated, then the software team chooses to create data models as part of requirements modelling. The entity-relationship diagram defines all data objects that are processed within the system, the relationships between the data objects and the information about how the data objects are entered, stored, transformed and produced within the system.

3.7.2 Data Objects

A data object is a representation of composite information that must be understood by the software. Here, composite information means information that has a number of different properties or attributes. A data object can be an external entity, a thing, an occurrence, a role, an organizational unit, a place or a structure.

3.7.3 Data object identification

In our system there is no composite information(information that has a number of different properties and attributes) which we can consider as a data object. All the vectors and matrices are generated in the runtime and take part in many mathematical operations.

As we have not any data-object and we don’t store any data in the database, we skip the step of database modelling.

3.8 Class-Based Modelling

3.8.1 Concept

Class-based modelling represents the objects that the system will manipulate, the operations that will be applied to the objects, relationships between the objects and the collaborations that occur between the classes that are defined.

3.8.2 Noun list from Question Answering App

No	Noun	No	Noun
1	System	11	Contextual Embedding
2	User	12	Vector
3	Context	13	Matrix
4	Question	14	Highway Network Layer
5	Answer	15	LSTM Layer
6	Input	16	Modelling Layer
7	Output	17	BiDAF Layer
8	Embedding Layer	18	Similarity Matrix
9	Word Embedding	19	Output Layer
10	Character Embedding	20	Word

3.8.3 Verb list from Question Answering App

No	Verb
1	give
2	ask
3	display
4	tokenize
5	embedding
6	Formation of similarity matrix
7	Context to query attention (C2Q)
8	Query to context attention (Q2C)
9	MegaMerge

3.8.4 General classification

Candidate classes were then characterized in seven general classifications. The seven general characteristics are as follows:
1. External entities
2. Things
3. Events
4. Roles
5. Organizational units
6. Places
7. Structures

Potential nouns to become a class after general classification criteria :

Noun	General Classification
System	4

User	4,5
Context	2
Question	2
Answer	2
Input	
Output	
Embedding Layer	3,4,7
Word Embedding	3,4,7
Character Embedding	3,4,7
Contextual Embedding	3,4,7
Vector	
Matrix	
Highway Network Layer	3,4,7
LSTM Layer	3,4,7
BiDAF Layer	3,4,7
Modelling Layer	3,4,7
Output Layer	3,4,7
Similarity Matrix	
Word	

3.8.5 Selection Criteria

The candidate classes are then selected as classes by six Selection Criteria. A candidate class generally becomes a class when it fulfills around three characteristics.

1. Retain information
2. Needed services
3. Multiple attributes
4. Common attributes
5. Common operations
6. Essential requirements

Potential general classified nouns to become a class after selection criteria:

Noun	**Selection Criteria**
System	2,6 (selected)
User	
Context	
Question	
Answer	
Input	
Output	
Embedding Layer	2,3
Word Embedding	2,6 (selected)
Character Embedding	2,6 (selected)
Contextual Embedding	2,3,6 (selected)
Vector	
Matrix	

Highway Network Layer	2,3,6 (selected)
LSTM Layer	2,3
BiDAF Layer	2,3,6 (selected)
Modelling Layer	2,6 (selected)
Output Layer	2,3,6 (selected)
Similarity Matrix	
Word	

3.8.6 Attribute and Method Identification

Class name	**Attribute**	**Method**
Word_embedding	-Word -vector	+tokenization() + find_word _embedding()
Character_embedding	-Word -vector	+tokenization() + find_char _embedding()

Contextual_embedding	-vector
	-matrix
		+build()
		+call()
		+compute_output_shape()

Highway_network_layer	-vector
	-matrix
		+build()
		+call()
		+compute_output_shape()

BiDAF Layer	-vector
	-matrix
	- context
	-question
	-similarity_matrix
		+build()
		+call()
		+compute_output_shape()
		+compute_similarity()
		+build_similarity_matrix()
		+Q2C_Attention()
		+C2Q_Attention()

		+megamerge()

Modelling Layer	-vector
	-matrix
		+build()
		+call()
		+compute_output_shape()

Output Layer	-vector
	-matrix
		+build()
		+call()
		+compute_output_shape()

System	-context
	-question
	-vector
	-matrix
		+initialize_model()
		+train_model()
		+main_function()
		+loss_function()
		+generate_answer()

3.8.7 Analysis

All classes included in class based diagram are selected as class for our system.

3.8.8 Class Cards

After identifying our final classes we have generated the following class cards -

Word_Embedding	
Attribute	Method
-Word -vector	+tokenization() + find_word _embedding()
Responsibilities	Collaborator
● Tokenize the word	

● Convert the word into vector	

Character_Embedding	
Attribute	Method
-Word -vector	+tokenization() + find_word _embedding()
Responsibilities	Collaborator
● Tokenize the word ● Convert the word into vector	

Highway_network_layer	
Attribute	Method

-vector
-matrix
	+build()
	+call()
	+compute_output_shape()

Responsibilities	Collaborator
● Concatenate word embedding and character embedding of a word ● adjust the relative contribution from the word embedding and the character embedding	Word_Embedding,Character_Embedding

Contexual_Embedding	
Attribute	Method
-vector	+build()

	+call()
	+compute_output_shape()

Responsibilities	Collaborator
• Build the custom layer with LSTM layer • Contextual Embedding of vector	Highway_network_layer

BiDAF _Layer	
Attribute	Method

-vector
-matrix
- context
	+build()
	+call()
	+compute_output_shape()
	+compute_similarity()
	+build_similarity_matrix()
	+Q2C_Attention()
	+C2Q_Attention()
	+megamerge()

Responsibilities	Collaborator
• Build the custom layer • Build Similarity Matrix • Compute similarity among context and query words • Calculate query to context attention • Calculate context to query attention • Megamerge	Contexual_Embedding

Modelling_Layer	

Attribute	Method

-vector
-matrix
	+build()
	+call()
	+compute_output_shape()

Responsibilities	Collaborator
● Build the custom layer with two LSTM Layers and perform operations	BiDAF_Layer

Output_Layer	
Attribute	Method

-vector
-matrix
	+build()
	+call()
	+compute_output_shape()

Responsibilities	Collaborator
● Build the custom layer ● Calculate the probability distribution of start index and end index of answer span.	Modelling_Layer

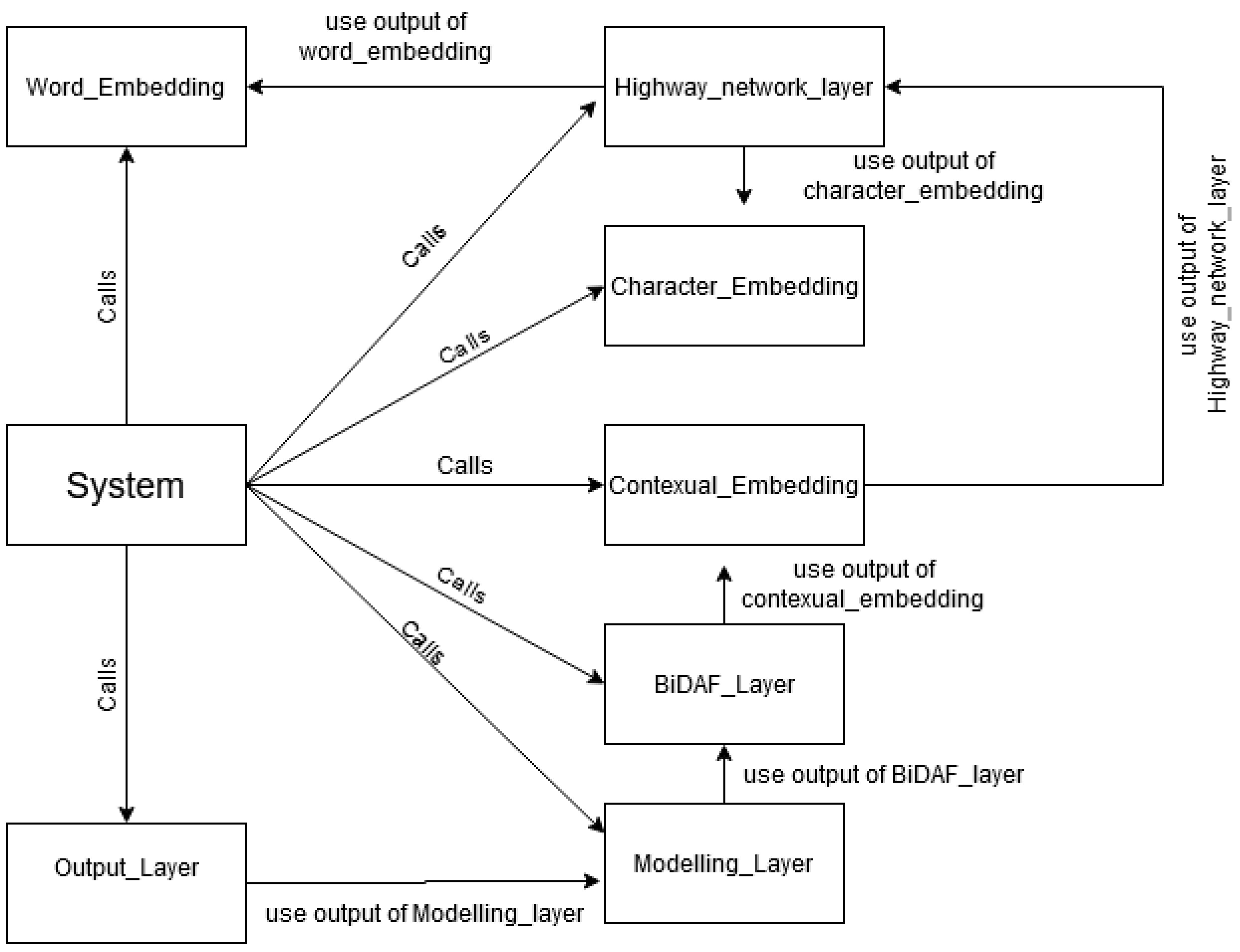
System	
Attribute	Method

-context
-question
-vector
-matrix
	+initialize_model()
	+train_model()
	+main_function()
	+loss_function()
	+generate_answer()

Responsibilities	Collaborator

● Initializing the model ● Train the model ● Receive input from user ● Generate the answer ● Calculate loss	Word_Embedding, Character_Embedding, Highway_network_layer, Contexual_Embedding, BiDAF _Layer, Modelling_Layer, Output_Layer

3.8.9 CRC Diagrams



Word_Embedding
use output of word_embedding
Highway_network_layer
use output of character_embedding
use output of Highway_network_layer
Calls
Calls
Character_Embedding
Calls
System
Calls
Contexual_Embedding
Calls
Calls
use output of contexual_embedding
BiDAF_Layer
Calls
use output of BiDAF_layer
Output_Layer
Modelling_Layer
use output of Modelling_layer

3.9 BEHAVIORAL MODELING OF Question Answering App

3.9.1 STATE TRANSITION DIAGRAM :

State diagram represents active states for each class the events (triggers).

Our App is a super simple app that has only one event initiated by the user. Without behavioral modeling we can simply define our project. Thus we skip behavioral modeling.

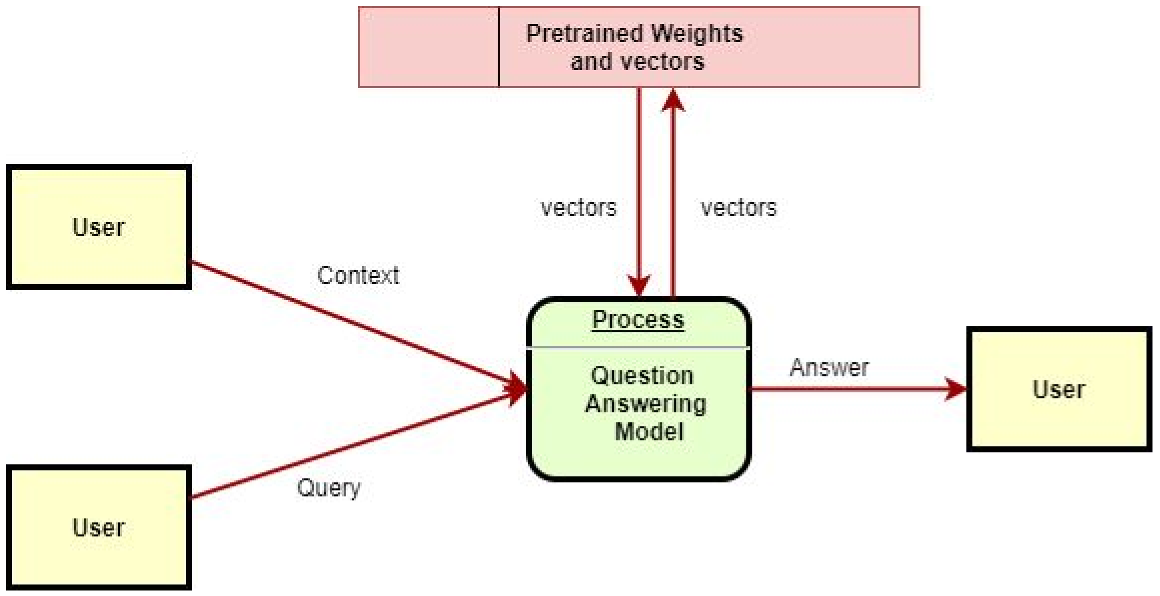
3.10 Data Flow Diagram (DFD)

3.10.1 Definition :

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system.
It shows how data enters and leaves the system, what changes the information, and where data is stored.

3.10.2 0-level DFD:

It’s designed to be an abstraction view, showing the system as a single process with its relationship to external entities.

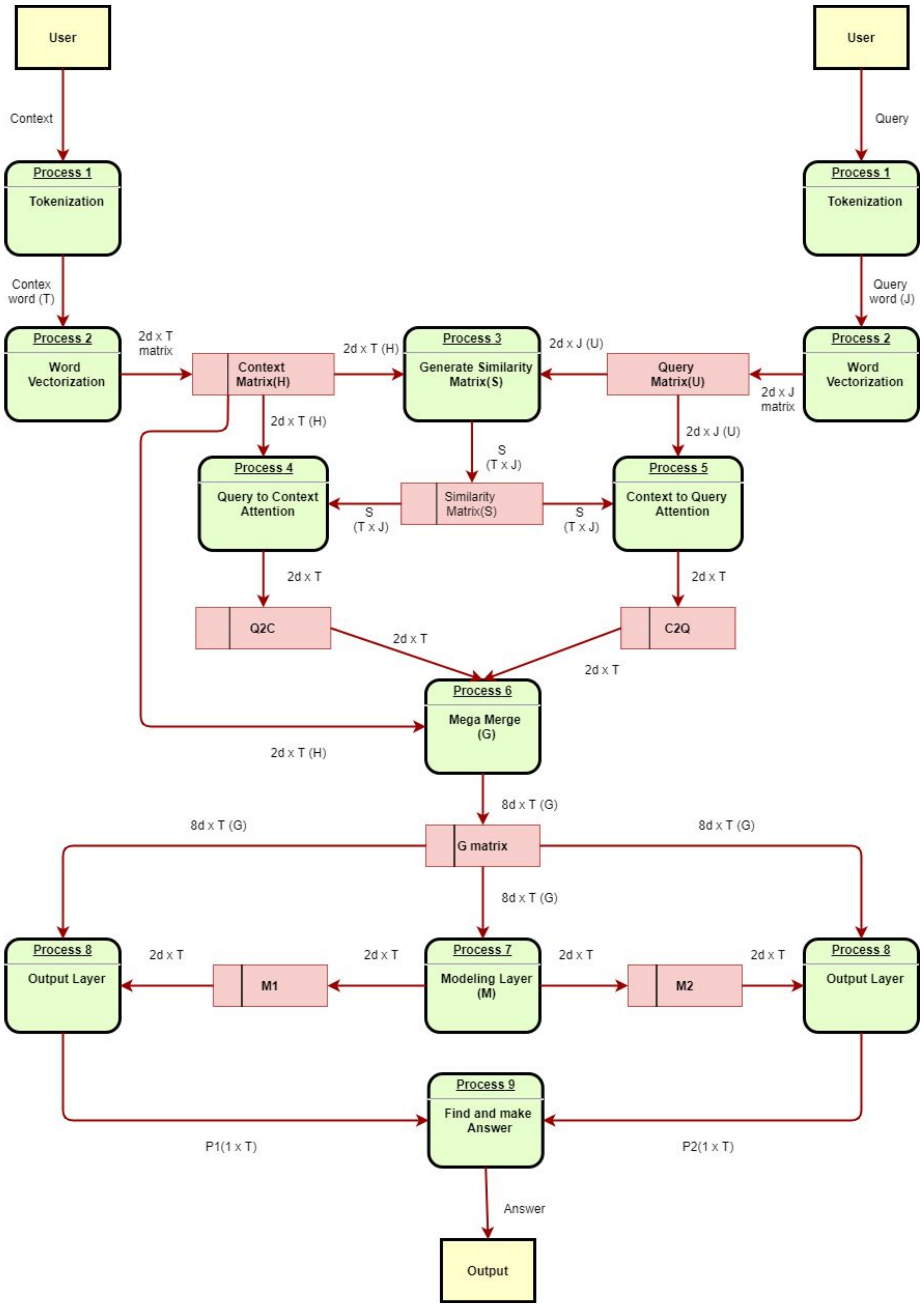


Description:

Our Q&A Model is a closed-domain, extractive Q&A model that can only answer factoid questions. Here Our model takes an English short passage as a context and takes factoid Question as Query. Then our model tries to predict the actual substring of our passage that contains the fact of our Query. In the end, it shows the most probable answer string to the user. In the process , it uses pretrained vectors and matrices and generates many new matrices and saves them.

3.10.3 1-level DFD:

In this level we highlight the main functions of the system and break down the high level process of 0-level DFD into subprocesses.


User
Context
Process 1
Tokenization
Contex word (T)
Process 2
Word Vectorization
2d x T matrix
Context Matrix(H)
2d x T (H)
Process 3
Generate Similarity Matrix(S)
2d x J (U)
Query Matrix(U)
2d x J matrix
User
Query
Process 1
Tokenization
Query word (J)
Process 2
Word Vectorization
2d x T (H)
S (T x J)
2d x J (U)
Process 4
Query to Context Attention
S (T x J)
Similarity Matrix(S)
S (T x J)
Process 5
Context to Query Attention
2d x T
2d x T
Q2C
2d x T
C2Q
2d x T
Process 6
Mega Merge (G)
2d x T (H)
8d x T (G)
8d x T (G)
G matrix
8d x T (G)
8d x T (G)
Process 8
Output Layer
2d x T
M1
2d x T
Process 7
Modeling Layer (M)
2d x T
M2
2d x T
Process 8
Output Layer
Process 9
Find and make Answer
P1(1 x T)
P2(1 x T)
Answer
Output

Description(1-level DFD):

__Process 1:__ User inputs: Context and query are passed through the process “Tokenization”. The incoming Query and its Context are first tokenized, i.e. these two long strings are broken down into their constituent words.Here, the symbols T and J are used to denote the number of words in Context and Query, respectively.

__Process 2:__ After Tokenization process, The resulting words are then subjected to the "Vectorization" process, where they are converted into vectors of numbers. These vectors capture the grammatical function (syntax) and the meaning (semantics) of the words, enabling us to perform various mathematical operations on them. "Vectorization" process takes T context word and generates (2d x T) context matrix(H), J query word and generates (2d x J) query matrix(U). The vectorization processing outputs H and U carry within them the syntactic, semantic as well as contextual information from all words in the Query and the Context.

__Process 3:__ Then H and U are passed through the "*Generate Similarity matrix*" process. This process takes H and U and generates (T x J) similarity matrix (S).First concatenate the three vectors : H, U and H*U (* = element wise multiplication). Then the dot product of the resulting vector and a trainable weight form the S matrix. The value in row t and column j of the matrix S represents the similarity of t-th Context word and j-th Query word. This matrix helps us to make attention mechanisms.

__Process 5:__ **S** and **U** are also passed through another process "*Context to Query Attention*". This process generates a (2d x T) matrix **C2Q**. First, we use the scalar values in **S** to calculate the attention distribution. This is done by taking the row-wise softmax of **S.** Let's call it matrix **A**. The matrix **A**, whose dimension is the same as **S**, indicates which Query words are the most relevant to each Context word. We then take every row of **A** to get the attention distribution **At:** which has a dimension of 1-by-J. **At:** *reflects the* *relative importance of each Query word for the t-th Context word.* We then calculate the weighted sum of the query matrix **U** with respect to each element in the attention distribution **At.** The result of this step is the attention output matrix called **C2Q**. **C2Q** encapsulates the information about the relevance of each query word to each Context word.

<u>Process 4:</u> Then **S** and **H** are passed through the "*Query to Context Attention*" process. This process generates a (2d x T) matrix **Q2C**. We first take the maximum across the row of the similarity matrix **S** to get a column vector(**z**). We apply softmax on **z** to get an attention distribution called **b**. We then use **b** to take a weighted sum of the Context matrix **H.** The resulting attention output is a **2d**-by-T column vector called **Q2C . Q2C** is a representation of the Context that encapsulates the information about the most important words in the Context with respect to the Query.

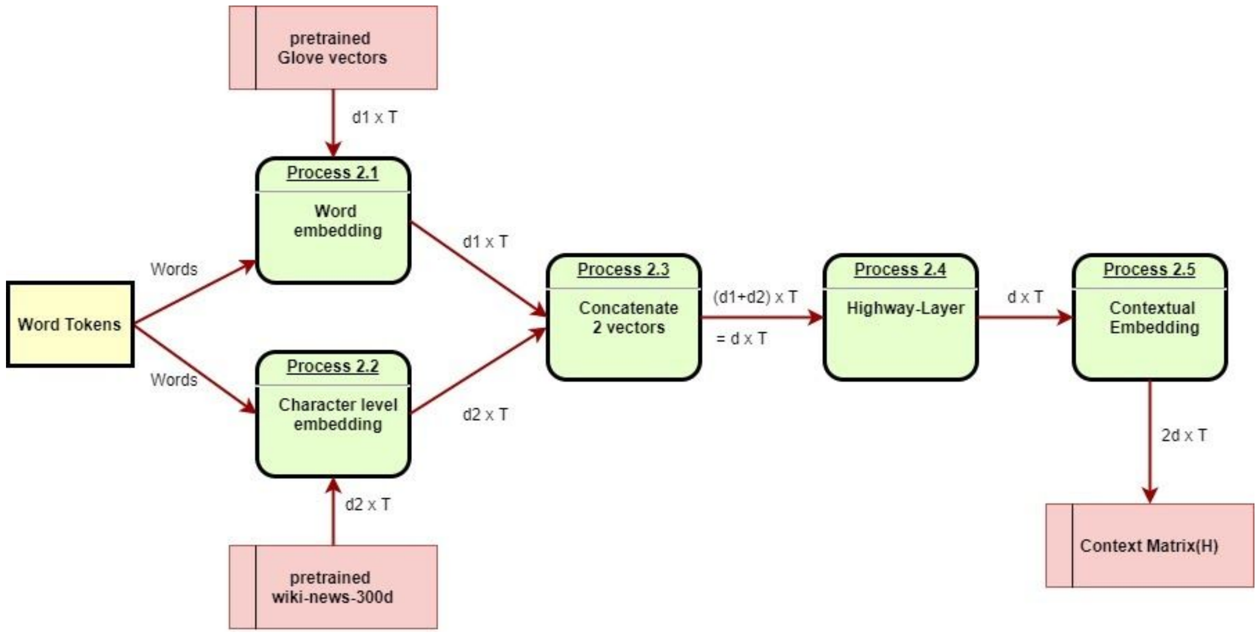
<u>Process 6:</u> Then the matrices: **H**,**C2Q** and **Q2C** are passed through the "*Mega Merge*" process. It generates a (8d x T) matrix (**G**). In this process we concatenate these 4 vectors across rows: **H**, **Q2C**, (**Q2C * C2Q**) and (**H * C2Q**). Here * denotes the element wise multiplication. The giant matrix **G** contains all information in H, C2Q and Q2C.

<u>Process 7:</u> The G matrix is passed through the "*Modeling Layer*". The Modeling Layer takes G and outputs 2 matrices M1 (2d x T) and M2 (2d x T). M1 and M2 are yet another matrix representation of Context words. The difference between M1 and M2 and the previous representations of Context words are that M1 and M2 have embedded in them information about the entire Context paragraph as well as the Query.
<u>Process 8:</u> Then G,M1 and M2 are passed through "Output Layer". Here G and M1 generate P1 (probability distribution of first word of the answer) and M2 and G generate the P2(probability distribution of last word of the answer).

<u>Process 9:</u> P1 and P2 are then passed through the "Find and make answer" process which makes the final answer string and shows the output.

3.10.4 2-level DFD:

It goes one step deeper into parts of 1-level DFD.



Description: Vectorization

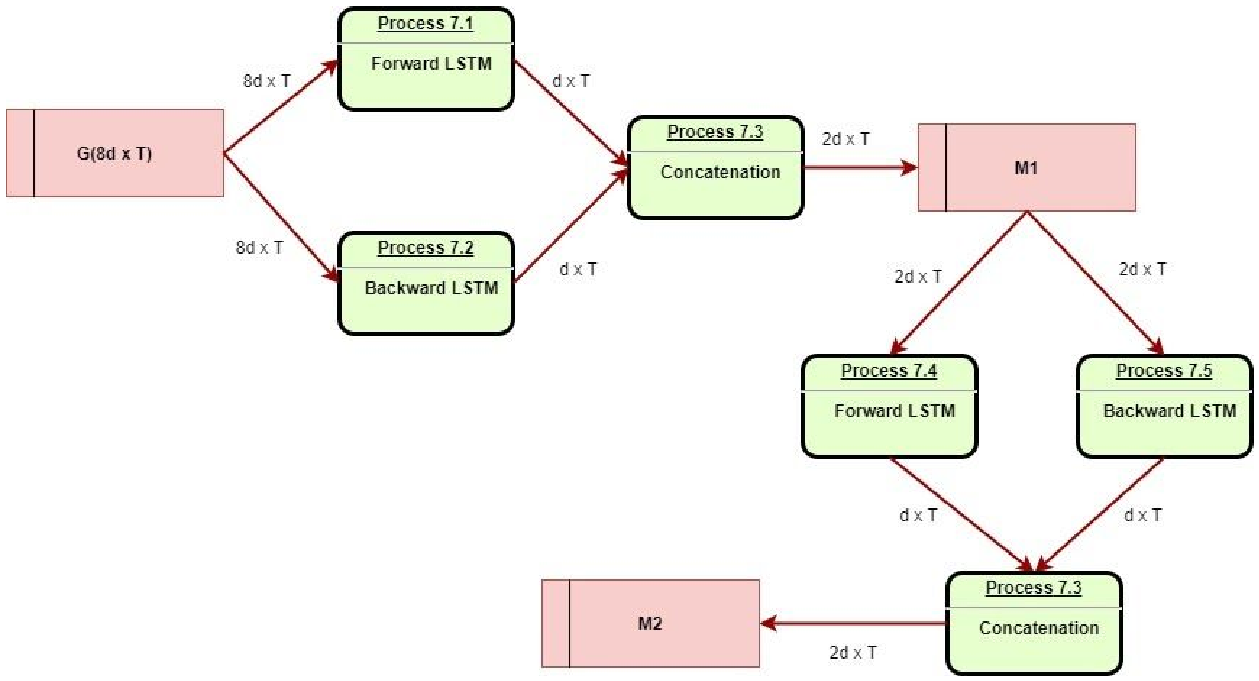
Process 2.1: For word embedding our model uses pre-trained GloVe embeddings to get the vector representation of words in the Query and the Context. The output of the word embedding step is two matrices — one for the Context and one for the Query. The lengths of these matrices equal the number of words in the Context and the Query (**T**). Their height , **d1**, is a preset value that is equal to the vector dimension from GloVe; this can either be 50, 100, 200 or 300.

<u>Process 2.2:</u> As glove handle "out of vocabulary words" (OOV) by assigning random values, it may confuse our model. For this reason, we need character level embedding to handle OOV words. In Character level embedding we use "pretrained wiki news 300d" vectors. The output of the Character level embedding step is two matrices — one for the Context and one for the Query. The lengths of these matrices equal the number of words in the Context and the Query (**T**). Their height , **d2**, is a preset value that is equal to the vector dimension. In our case **d2 = 300.**

<u>Process 2.3:</u> Concatenate two vectors step takes 2 matrices from process 2.1 and process 2.2 and concatenate them. The output of this step is a d x T shaped matrix. Here, d = d1+d2.

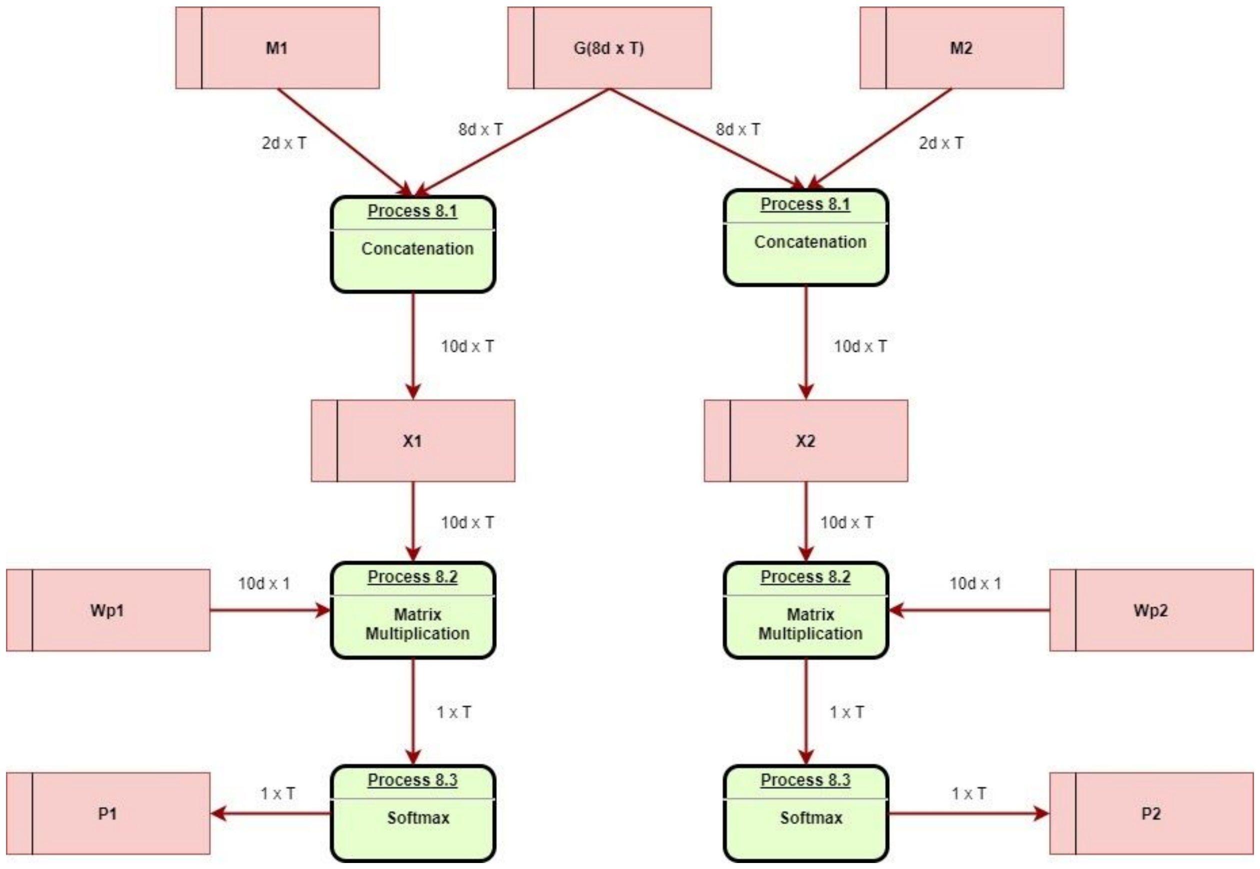
<u>Process 2.4:</u> Highway Layer takes the output of process 2.3, a d x T shaped matrix. The output of this process is also a d x T matrix. This step adjusts the relative contribution of word embedding and character level embedding steps.

<u>Process 2.5:</u> Word representation of process 2.1 and 2.2 are not taking into account the word contextual meaning. Thus, we need a contextual embedding layer. This layer consists of a bidirectional LSTM (composed of both forward and backward LSTM sequences). The combined output embeddings from the forward- and backward-LSTM simultaneously encode information from both past (backwards) and future (forward) states. The output of the contextual embedding step is two matrices — one from the Context (**H**) and the other from the Query(**U**). Both matrix sizes are 2d x T(number of words).



Description: Modeling Layer

The Modeling Layer consists of two layers of bi-LSTM. The input to the modeling layer is G. The first bi-LSTM layer converts G into a 2d-by-T matrix called M1. M1 then acts as an input to the second bi-LSTM layer, which converts it to another 2d-by-T matrix called M2.



Description: Output Layer

<u>Process 8.1:</u> **M1** and **M2** are first vertically concatenated with **G** to form **X1** = [**G**; **M1**] and **X2** = [**G**; **M2**]. Both [**G**; **M1**] and [**G**; **M2**] have a dimension of **10d** x **T**.

<u>Process 8.2:</u> In matrix multiplication steps X1 (10d x T) and a trainable weight matrix Wp1(10d x 1) are taken and a 1 x T matrix. X2 and Wp2 also produce a 1 x T matrix through this step.

<u>Process 8.3:</u> Softmax takes the output of matrix multiplication and gives a probability distribution of 1 x T matrix. Here **P1** is the probability distribution of the start index and **P2** is the probability distribution of the endindex of the answer string.

p1 and p2 are then used to find the best Answer span.