

**Product Development Laboratory**

**Semester – VI**

**Department of Computer Science and Engineering**

**Title: Machine Reading Comprehension**

**First Progress Report**

**Dr. Arun Kumar**

**Submitted by**

Sikakollu N V S K Prasanth (116CS0213)

Shweta Kedas (116CS0237)

Uttam Singh (116CS0238)

**INTRODUCTION**

Machine Reading Comprehension is a task in both natural language processing and artificial intelligent research. The goal is to train a machine to understand a given passage and then answer the questions related to the passage.

As comprehension models have gone mainstream, the expectations from such models have grown to expect human like abilities to answer the queries. Reading comprehension is a task that requires question – answer system to read a text, process it, comprehend and be able to extract the span of the text and answer to the query.

Machine Reading Comprehension enables computer system to read a paragraph and answer some questions against it. While this is much easier task for humans, it’s not quite straight forward for machines. Machine tries to capture nested context in a paragraph, i.e., the subject, the predicate, the concept and the conditions from the question. Our model uses all these captured aspects and puts them together to arrive at the answer.

Our project model utilizes **word embeddings**, a technique in **Natural Language Processing (NLP)**, **Bidirectional Recurrent Neural Networks (BiRNN)** and utilizes the **attention in neural networks** to highlight some part of the text under the context of the other. It uses **Pointer Networks** to find the start and end points of the desired answer from the paragraph.

1. **Word Embeddings**

Machines only understand in binary, i.e., only 0’s and 1’s. The data we feed to the machine for training and testing are pure text that can’t be understood by a computer. A simple and efficient solution to this problem is word embeddings. It is a technique in Natural Language Processing that maps the words to fixed dimensional vectors.

A traditional method of representing the words is one-hot vector method, with its target element being 1, others being 0. The main drawback of this method comes when we look for uniqueness of each word in the vocabulary. Since English dictionary consists of millions of words, each word representation would require million bits of memory, which is practically infeasible.

Word2Vec is an efficient solution to this problem which converts the words into its respective word embeddings.

1. **Bi-Directional Recurrent Neural Networks(GRU cells)**

Recurrent Neural Network is a deep learning technique that is used for sequential data such as text, voice, stock prices, etc. This is the most powerful of all kinds of Neural Networks because it has internal memory, capable of remembering the important things about the input they received and thus, giving a precise prediction. In RNN, the information cycles through a loop. When it makes a decision, it takes into consideration the current input and also what it has learnt from the previously received inputs. In RNNs we have the problem of vanishing gradient, i.e., it fails to back propagate the error to the initial layers of the network.

To resolve this problem, we have Gated Recurrent Units (GRUs). These are invented in 2014. These have two gates namely, update gate and reset gate. These gates are the vectors which decide what information should be passed to the output and memory. The best feature of GRUs is that they can be trained to keep the information from long ago. If carefully trained, they can perform extremely well even in complex scenarios.

The meaning of a word in a sentence depends on the words that precede and succeed it. So we need to train the model from both directions of the sentence. For this purpose we use Bi-Directional RNNs.

1. **Attention**

When a neural network is given a long sentence, due to its short memory handling capacity, some of the relevant information may be lost. To work on this problem, the seq2seq approach generates a vector. When a new word is encountered by the encoder network, part of the network looks around for the word in the sentence which is relevant. The context vector highlights the part of the sentence and sends only the relevant information. It focuses the Network’s attention. With this, the decoder network can receive relevant information from Bi RNN. The sentence length matters less because only a few words are considered relevant.

1. **Pointer Networks**

Pointer Networks are sequence to sequence models where the output is discrete tokens corresponding to positions in an input sequence. These are suitable for problems like sorting, word ordering, or computational linguistic problems. One common characteristic of all these problems is that the size of the target dictionary varies depending on the input length.

Pointer Networks solve the problem of variable size output dictionaries using a mechanism of neural attention. It uses attention as a pointer to select a member of the input sequence (passage) as the output (answer).

**Work Flow:**

We start with analysis of dataset and then convert the passage and questions to its word level and character level embeddings using NLP. The entire training of the model takes place in three readings.

1. In the first phase, we feed these embeddings to a Bidirectional RNN.
2. In the second pass, the network trains itself with the context of the question using Question matched attention.
3. In the third pass, the network finds the answer to the question and ignores the rest of the passage.

Finally, we use the concept of Pointer Networks to find the starting and ending point of the answer to obtain the relevant answer as output.

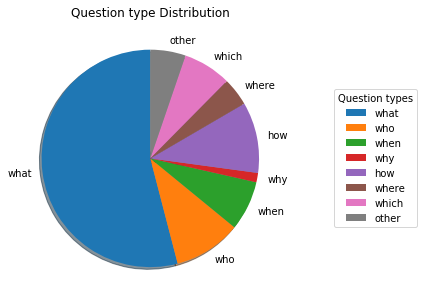
**Work Progress**

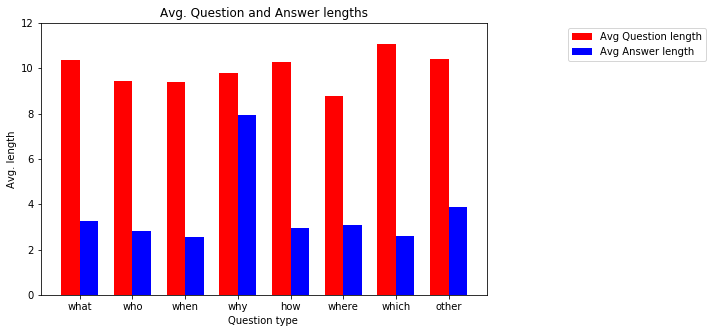
* For the data analysis, **pandas**, **numpy**, **matplotlib** libraries were used under anaconda environment.
* The training Stanford Question Answering Dataset has 442 titles with varying number of paragraphs in each title.
* **SQuAD** utilizes python data structures lists and dictionaries

**Dataset Description of Training Dataset:**

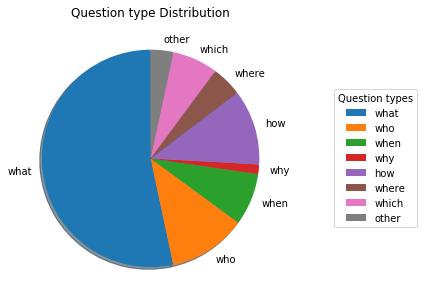
1. Each title is a key with values as paragraphs.
2. Each paragraph consists a list of dictionaries.
3. Each dictionary has keys ‘context’ and ‘qas’.
4. ‘qas' is a list of dictionaries and each dictionary has keys ‘answers’, ‘id’, ‘question’.

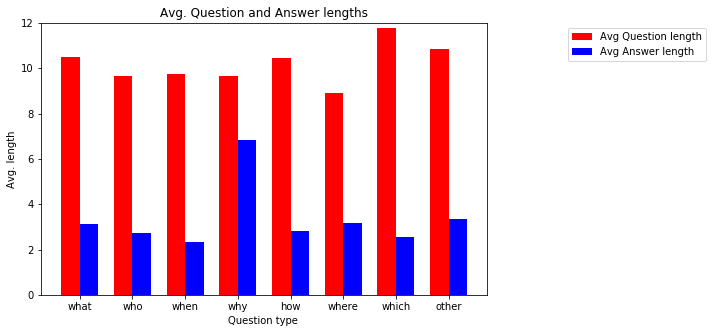
**Training Set charts**





**Test Set charts**





**Exploratory Data Analysis table**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.no** | **Statistical aspect** | **Train set** | **Test set** |
| 1 | Title count | 442 | 48 |
| 2 | Total word count | 91,87,544 | 10,93,051 |
| 3 | Total Question count | 87,599 | 10,570 |
| 4 | Context count | 18,896 | 2,067 |
| 5 | Avg Question length | 10.46 | 10.82 |
| 6 | Avg Answer length | 3.26 | 3.15 |
| 7 | Avg Context length | 117 | 123 |
| 8 | Avg Question count per context | 4.64 | 5.11 |

**Applications of this project:**

1. Text summarization
2. Chat bots in helpline desks
3. Speech-text translation

**References:**

1. Wissam Baalbaki, Dan Zylberglejd ; CS224N: Natural Language processing with Deep learning Reading Comprehension
2. Natural Language Computing group, Microsoft Research Asia; R-NET: Machine Reading Comprehension with Self Matching Networks