Interim Report

Team 3:

P Shiridi Kumar	2021121005
Yash Bhatia	2020101007
Abhishek Sharma	2020101050

Problem statement: Question and answer generation [Brief Description: Given a context document D and a sequence of user questions, the goal is to generate answers for these Questions using the given document D as a reference]

Tasks Completed

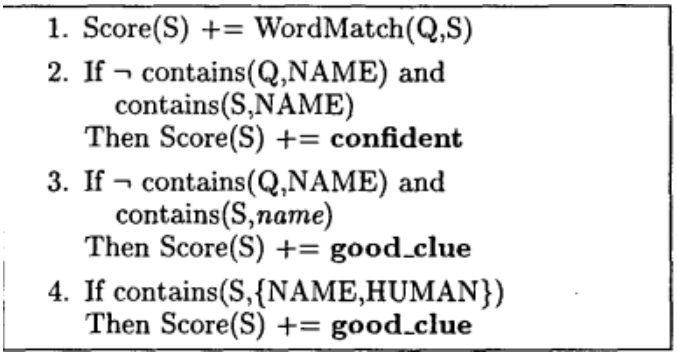
- **Review/research of methodologies :**
  - We began by evaluating different approaches, which encompassed fine-tuning pre-trained models and developing new implementations from the ground up. Ultimately, we have settled on the following strategies:
    - Fine-tuning BERT

    - Fine-tuning T5
    - Employing a neural ranking model
    - Implementing rule-based systems
    - Implementing an Encoder-decoder model

    As a final step, we will try to generalize the training through domain adversarial training across the models

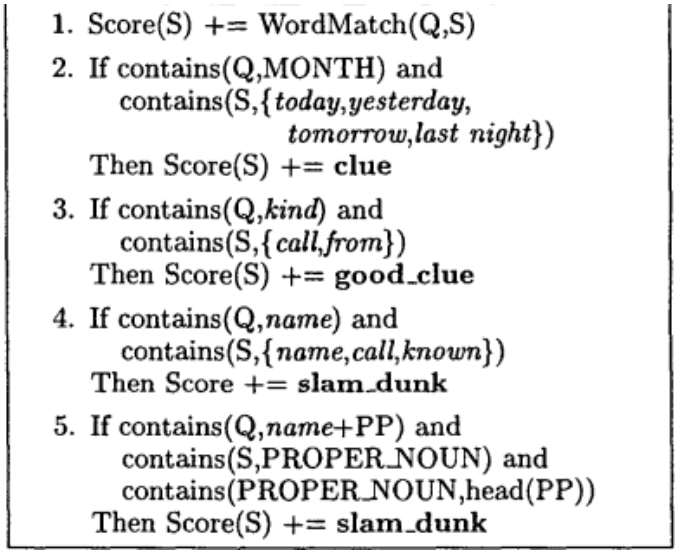
- **Data collection / Preprocessing :**
    - We have used the Squad dataset for the Neural ranking model (will be using the same for other fine-tuned models as well)
    - For neural ranking, we have experimented with Glove embeddings for vector representation of each word which resulted in the input size being huge. So we used sent_transformer from hugging faces for dense vector representation of sentences
    - Other preprocessing steps include the tokenization of sentences from a paragraph
- **Rule-based system :**
    - It uses heuristic rules that look for lexical and semantic clues in the question and the paragraph.
    - Each type of WH and How question look for different type of answers, so for every type of questions there are separate set of rules and scoring system.
    - The rules are applied to each sentence in the paragraph, as well as the title of the story, with the exception that the title is not considered for WHY question.The dateline may be a possible answer for WHEN and WHERE questions.
    - All questions share a common `WordMatch` function which counts the number of words that appear in both the question and the sentence being considered. This function uses stemmed word, and score is increased if two words have same morphological roots. The function considers that verbs are more important than non verb matches, so verb matches have more score.

- Who Rules: If the Question does not contain any proper noun, and the sentence contains a proper noun. If a question does not contain any proper noun, and the answer contains the word “noun”. If the sentence contains any occupation, organization, etc.



1. Score(S) += WordMatch(Q,S)
2. If ¬ contains(Q,NAME) and
   contains(S,NAME)
   Then Score(S) += **confident**
3. If ¬ contains(Q,NAME) and
   contains(S,*name*)
   Then Score(S) += **good_clue**
4. If contains(S,{NAME,HUMAN})
   Then Score(S) += **good_clue**

- What Rules: First reward sentences that contain a date expression if the question contains a month of the year (What occurred on a specific date?). Questions that include “What kind” are also included. Similarly other rules are implemented as mentioned in the paper.



1. Score(S) += WordMatch(Q,S)
2. If contains(Q,MONTH) and
   contains(S,{*today*,*yesterday*,
   tomorrow,*last night*})
   Then Score(S) += **clue**
3. If contains(Q,*kind*) and
   contains(S,{*call*,*from*})
   Then Score(S) += **good_clue**
4. If contains(Q,*name*) and
   contains(S,{*name*,*call*,*known*})
   Then Score += **slam_dunk**
5. If contains(Q,*name*+PP) and
   contains(S,PROPER_NOUN) and
   contains(PROPER_NOUN,head(PP))
   Then Score(S) += **slam_dunk**

- WHEN , WHERE and WHY Rules :

1. If contains(S,TIME)
   Then Score(S) += **good_clue**
   Score(S) += WordMatch(Q,S)
2. If contains(Q,*the last*) and
   contains(S,{*first*,*last*,*since*,*ago*})
   Then Score(S) += **slam_dunk**
3. If contains(Q,{*start*,*begin*}) and
   contains(S,{*start*,*begin*,*since*,*year*})
   Then Score(S) += **slam_dunk**

Figure 4: WHEN Rules

1. Score(S) += WordMatch(Q,S)
2. If contains(S,LocationPrep)
   Then Score(S) += **good_clue**
3. If contains(S, LOCATION)
   Then Score(S) += **confident**

Figure 5: WHERE Rules

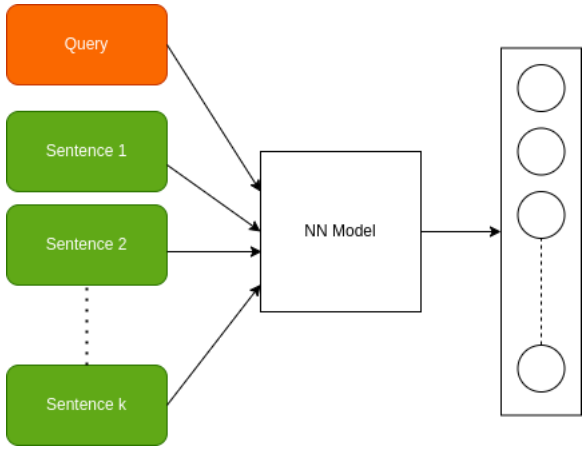
1. If S ϵ BEST
   Then Score(S) += **clue**
2. If S immed. precedes member of BEST
   Then Score(S) += **clue**
3. If S immed. follows member of BEST
   Then Score(S) += **good_clue**
4. If contains(S,*want*)
   Then Score(S) += **good_clue**
5. If contains(S,{*so*,*because*})
   Then Score(S) += **good_clue**

Figure 6: WHY Rules

- The dataset used is group of different paragraphs, where questions and answer are both given for each paragraph. The accuracy is calculated in a way that if a 50% words in predicted words are available in given answer of the question, then it considered as a good answer. This approach leads to have **30-35%** accuracy which is descent for a very naive model for Question Answering.

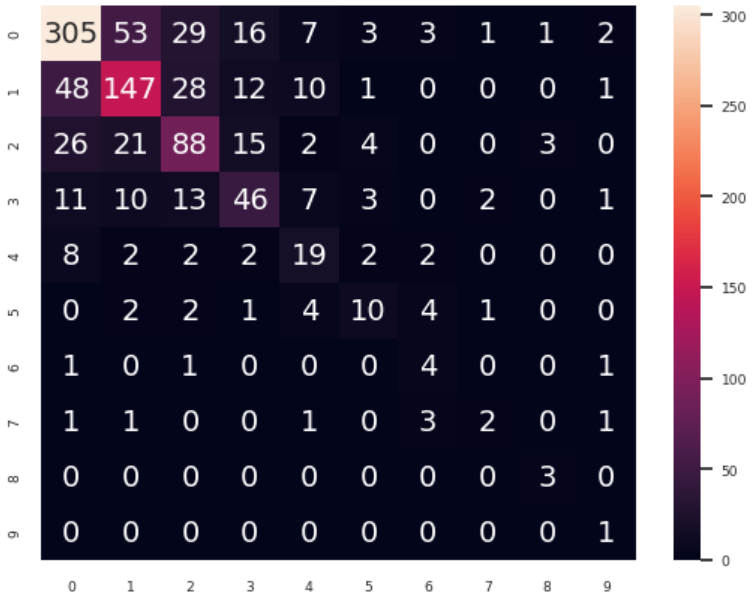
- **Neural ranking model :**
  - Neural ranking-based model: Given a Document D and a query Q, document D is divided into k individual sentences {s0,s1,s2,..sk} and is fed to the neural network model along with the query separated by a <sep> token.

- As discussed in the pre-processing step, we basically divided the paragraph into sentences using nltk.sent_tokenize and later represented these sentences as dense vectors(dimension of 384) using sent_transformers from hugging face.
- Initial architecture overview :



- But later we fed the documents individually with 0/1 as the output label rather than together because of computational complexity

- Initial results:
  - F1-score : **0.6292**
  - Precision: **0.6371**, Recall: **0.625**
  - Confusion matrix: Each entry at the ith row and jth column in the below matrix corresponds to the number of questions for which the predicted sentence index is j but the actual answer is i.
  - As we can see, the diagonal elements have a higher value indicating that there is a greater overlap between predicted and actual answers



- **Sequence to Sequence Encoder-Decoder with Attention:**
  - The input to this model is a paragraph followed by its questions, separated by <SEP> token.
  - On the decoder side, we try to generate the output of the questions.
  - While doing so, since even LSTM can forget very long dependencies, we introduce skip connections from the final encoder layer to the decoder. We call this attention.
  - While doing so, in order to increase the generalizability of the model, we use **Domain Adversarial Training** which tries to map the train domain space to test domain space.
  - We are done with the entire code apart from DAT, but it's yet to be tested.

Tasks to be Done

- **Fine-tune BERT / T5 for question-answering**
- **Implement Encoder-Decoder Model [Not yet tested and results aren't yet available]**
- **Fine- Comparison and analysis based on various metrics of all the models/implementations**

Code Structure:

Note: only important files and folders are described

- neural_ranking \ :
  - neural_ranking.ipynb [consists of the implementation of neural ranking model]
- miscellaneous \:
  - NeuralRanking_Baseline.ipynb [failed trails]
  - roughwork.ipynb [data cleaning of Squad dataset and extracting it from JSON to csv.]
- rule_based \:
  - code.ipynb [all implementation]
  - names.txt, month.txt, location.txt, occupation.txt etc [dictionaries for rules]
- encoder-decoder \:
  - seq2seq.py : [Not yet completed, preprocessing the data and other building blocks are done]

Remarks (from team perspective) :

- We are on track as of now and will try to complete the remaining tasks as mentioned in the timeline in the below section

Work Plan for remaining tasks:

- Check for further improvements for the implemented models till now : [October: 25th]
- Encoder-Decoder model : [October: 28th]
- Fine-tuning BERT/T5 : [November 7th]
- Applying domain adversarial training for the models(if there is a scope) : [November 10th]
- Analysis and comparison of all the implementation : [November 15th]

- Final check and submission : [November 17th]