This project was implemented with Python as the primary language, using libraries like Tensorflow, Pytorch, SpaCy, NLTK, Numpy, Matplotlib & external tools like Google Natural Language API. Google Collab was helpful for implementing with GPU support, if you don’t have a personal computer with a GPU.

The experiment, to test embeddings with the Multi-RC Dataset, was done with USE, BERT & InferSent Embeddings. We present a general algorithm, which was followed for all of them. We have the jupyter notebooks available, along with the ‘.npy’ files for some embeddings, on our Github repository.

To implement the Graph Convolutions, we would need to generate the Knowledge Graphs for the comprehension, as told in Methodology [citation]. To use the Google Natural Language API, you would be required to set up a Google Project in Google Cloud Platform, and you can use the free trial to use the API without incurring charges.

For contributing to the community, we have made available all of the knowledge graphs for the Multi-RC Training Dataset, on our Github Repository.

We have attached some screenshots of code, to help explain the process better, in figures [citations].

**Input:**

Multi-RC Dataset, Questions & Answers, Encoder Model

Note: The Encoder Model for USE & BERT can be used on both google collab & offline. For InferSent, it is recommended to follow their Github Repository to download the requirements and the model, as instructed there.

**Steps:**

1. Initialize the Encoder Model, which will be used to generate the Embeddings
2. For each sentence in the Paragraph & Questions, on the Multi-RC Training Dataset, encode them and save them in a ‘.npy’ format. This ensures that the embeddings are stored efficiently & so they can be loaded in future.

3> Define Functions for taking dot product or cosine similarity for each Question and Answer Sentence

4> Define Functions for Metrics and Evaluation

5> For each Paragraph, for each Question in Para:

Question Embedding Loaded

For each sentence in Para:

Sentence Embedding Loaded

Take Dot Product or Cosine Similarity of Question & Sentence Embedding

6> For ‘K’ Answer Sentences, find

top ‘K’ values in sentence and

question dot product

7> Use Set Intersection to determine

matches from Got and Correct

Answers

8> Use Accuracy Metric to determine final accuracy

9> Use Relaxation Accuracy Metric to see if most answers

are gotten with 1 sentence relaxation

10> Plot statistics like answers missed to project on what the

model misses and what it gets correctly

Since the best performer were the USE Embeddings, we have also tried enriching them with Graph Convolutions, as stated in the below algorithm.

**Input:**

Multi-RC Dataset, Questions & Answers, USE Embeddings, Google Natural Language API

**Steps:**

1. Initialize the JSON key and Google Cloud Account Settings for using the Natural Language API

2> Define Functions to extract entities with NER from API & spaCy

1. Define Functions to lemmatize and filter out STOP Words to be used for entities
2. Define Functions to use WikiData to add sub-nodes for entities mapping to Wikipedia Entries
3. Define Function to take in an article and add it as a node in Knowledge Graph
4. Define Overall Function to create Knowledge Graph

7> Convert each sentence into a text file, for the Paragraph

8> Run Function to create node from sentence

9> Create 1 Knowledge Graph per Paragraph and save in folder

10> Initialize USE Embeddings for Paragraphs & Questions

11> Define Functions for Graph Convolution

12> Encode them (Embeddings) and save them, so they can be loaded in future

13> Define Functions for taking dot product for each Question and Answer Sentence

14> Before calling function for dot product, re-define matrices with Graph Convolution Function

15> Can experiment with Lambda values and test weighted averages for neighbours

16> Define Functions for Metrics and Evaluation

17> For each Paragraph, for each Question in Para:

Question Embedding Loaded

For each sentence in Para:

Sentence Embedding Loaded

Take Dot Product of Question & Sentence Embedding

18> For ‘K’ Answer Sentences, find top ‘K’ values in sentence and question dot product

19> Use Set Intersection to determine matches from Got and correct Answers

20> Use Accuracy Metric to determine final accuracy