Methods implemented for entity linking:

Use Tc_generate_entitylinking_results.py to run the entity linking implementation. When used the option use_entity_linking = "enhanced". The code will run for combined entity linking methods used. The implementation uses Tagme to find annotations and spots in the queries as well as in the paragraphs. From results, I have found out using spots on the text gives better results than using annotations or both.

Implemented different methods to enhance the processed query and paragraph text using tagme's different methods. Text processing has been done on both the methods to remove the stop words, repeated words, unwanted symbols etc. Stemming is also done here.

Tried giving different minimum scores for the methods to generate the best results. For example, using an annotation score 0f 0.2 gives more meaningful words than with a score of 0.1. The mentions help in automatically removing the unwanted words from the list.

Found the similarity between words in queries and words in paragraphs and compared the results using different retrieval methods. Generated the results using test200 data with 4000 passages. After I have the ranking of paragraphs for all 7 million passages, I can use the top retrieved passages to apply the methods specially on that data.

Also implemented the interpretation algorithm also known as Greedy interpretation algorithm discussed in "Entity linking in queries: Tasks and evaluation" paper by Faegheh Hasibi, Krisztian Balog and Svein Erik Bratsberg. The method helps in finding interpretations of the query by machine-understanding them. The result is the set of entities including the interpretations and has no over-lapping words. Each mention corresponds to single entity.

For comparison, the results generated using entity linking with improved TFIDF_improved retrieval method and using simple baseline approach are shown below:

Entity linking performance:
mrr = 0.341,
p@5=0.126,
r-prec=0.191,
map=0.230

Baseline approach
Eval(mrr=0.30530690657174037,
p@5=0.11312775330396337,
r-prec=0.18206986490518473,
map=0.22395140874924307)

How to run the code:

1. Results file – you can directly use my results file in your evaluation framework with the qrels file and results file.

2. You can use the bash scripts baseline.run.sh for baseline approach and entitylinking.run.sh for entity linking implementation.

   baseline.run.sh

   python tc_generate_entitylinking_results.py $1 $2 $3 notenhanced

   $1 = outlines file
   $2 = paragraphs file
   $3 = results file

   Example
   Python tc_generate_entitylinking_results.py all.test200.cbor.outlines
   all.test200.cbor.paragraphs output_baseline.run notenhanced

   python eval_framework.py $4 $5

   $4 = qrels file
   $5 = results file

   Example
   python eval_framework.py all.test200.cbor.hierarchical.qrels output_baseline.run

   entitylinking.run.sh

   python tc_generate_entitylinking_results.py $1 $2 $3 enhanced

   $1 = outlines file
   $2 = paragraphs file
   $3 = results file

   Example
   Python tc_generate_entitylinking_results.py all.test200.cbor.outlines
   all.test200.cbor.paragraphs output_baseline.run enhanced

   python eval_framework.py $1 $2

   $4 = qrels file
   $5 = results file

   Example
   python eval_framework.py all.test200.cbor.hierarchical.qrels output_entitylinking.run

3. Main code **- tc_generate_entitylinking_results.py** for generating results file required in evaluation and **eval_framework.py** for evaluation.