MCQ Generator - Algorithm Documentation

# **Step 1 – Import the text file/article:**

Firstly, we will import the text file/article that has to be used for the MCQ generation with the help if open and read functions.

# **Step 2 – Extraction of important word (keywords):**

In this step, we try to extract the most important words, also known as keywords, in our text article that will result in quality questions. These keywords will act as the answer to questions and will be a part of the 4 options, the other 3 being the wrong options.

To do this, we use Python Keyphrase Extraction (PKE). The PKE that we use in this project is from an open source github repository that allows us to find keywords. This uses a multipartite unsupervised model to extract keywords. Multipartition ensure the efficiency of the extraction. So, in this process we get the best 25 keyphrases/keywords from our text article.

# **Step 3 – Split the article into sentences:**

In order to ease our access to the article, we split the whole large article into an array/list of sentences. This is called sentence tokenization. This will also help us fetch the sentences related to the keyword easily.

During this step, we also stem/trim the sentences that have length smaller than 15 words. This ensures that the questions which will be made with these sentences later will have enough context for understanding it.

# **Step 4 – Map the sentences to the keywords:**

Once we are done with the extraction of the important keywords and splitting the sentences, we move forward to mapping the sentences which contain keywords, to those keywords. This is done, so that when we need to generate questions from the keywords, these sentences will serve as the question with a blank for the answer.

We also sort the sentences in descending order if their lengths so that the best sentence will be chosen for the question generation.

# **Step 5 – Generate sense of the keywords:**

Let’s first talk about the term sense. A single English word might have multiple meanings and are used for different senses. For example, “fly”. The word fly is related to aviation as well as the house insect fly. The sentences, “A plane is about to fly” and “I killed a fly”, both have very different meanings of the word fly and the way in which the word fly is interpreted by us is known as the sense of the word. So in order to tackle this problem, Google provides the sense support which we used in this project.

# **Step 6 – Generate distractors from Wordnet:**

What are distractors? In short, distractors are the words that we will use as the remaining 3 incorrect options of our MCQ questions that will distract the quiz taker from the actual answer. The key point to keep in mind about finding these distractors is that they should have the same sense as the keyword/correct answer, if not, it will make the quiz fairly easier for the taker.

How do we do this? First we extract the sense of the word from the previous step. Next we obtain the hypernym, also known as the superclass/super-category of the current word. Once we have the hypernym, we go on to find the hyponyms, as known as the children/sub-category/members of that hypernym. These hyponyms will be of the same category of the keyword and of the same sense as well (ensured by the previous step). This will lead us to the distractors.

# **Step 7 – Generate distractors from Conceptnet:**

The question arises why use another method when we already have Wordnet? The answer to this is sometimes we encounter situations where we are unable to find any sense for the word, in that case Wordnet fails to generate the distractors and to level up our game, we use Conceptnet, an online API that helps us in finding the distractors, also the disadvantage for this is that it might not be in the best sense, hence Wordnet is always our first choice, if it fails, we go for Conceptnet.

How it works? It is an online API and is loaded to the program with URL and its object is created. It helps us in finding the best distractors for our word.

# **Step 8 – Map the distractors to the keywords:**

Once we are done creating the distractors, our next task is to map the distractors to the keywords. This step also check if the sense if available and also decides as to go for the Wordnet approach or the Conceptnet approach.

# **Step 9 – Present our MCQ:**

Now that we are done with all the major steps, our last but not the least step is to print all the questions as well as their options in a nice and readable manner.

To do this, firstly we convert the sentence into a regular expression and replace the keyword with a blank and print it with the question serial number and the four options.

With these 9 steps we are done with our project for now and we can generate a fairly nice MCQ Quizlet. This was my brute force approach for the assignment that I had to do within the given deadline but there are a lot more improvements that I would like to add to my current code in order to ensure even better results with the MCQs. I have added those ideas in my ‘Room for improvement’ file.

Also, I have added few articles as well as the generated MCQ quizzes in the repository which will help one see the working of the project. Looking forward to further improving the project.