# Develop a question answering system that can answer When/Where/Who type questions from a given set of documents

This is how the code works:

1. Convert paragraph into sentences metrics.
2. Cleaning the data.
3. Conversion of sentences to corresponding word embedding
4. Conversion of question into the word embedding
5. Apply Euclidean distance
6. Put the results in heap with index.
7. Pop and Print the result.

Loading Libraries:

1. NLTK is a platform for building Python programs to work with human language data. It provides easy-to-use interfaces, text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning
2. re: A regular expression or RegEx is a sequence of characters which enable you to find string or set of string using a specialized pattern.
3. numpy: NumPy is a Python library used for working with arrays.
4. Gensim is a free open-source Python library for representing documents as semantic vectors efficiently. It is designed to process raw, unstructured digital texts (“plain text”) using unsupervised machine learning algorithms.

```
import numpy as np
import nltk
import re
import gensim
from gensim.parsing.preprocessing import remove_stopwords
from gensim import corpora
from sklearn.feature_extraction.text import TfidfVectorizer
import heapq
```

Loading and reading file (6.txt) Note: this file is to be uploaded, and can be found in unlabelled datasets provided in the drive link as per lms

```
filename "6.txt"
```

```
filename='6.txt'
f = open(filename, "r")#creating a file object
txt=f.read() #Read the contents of the file into text
f.close()
```

class for preprocessing and creating word embedding

here we convert the text in lower case and remove the stop words.

Stop words can be imported using nltk packages. A stop word is a commonly used word (such as “the”, “a”, “an”, “in”) that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query.

We also perform Data Cleaning by removing the extra spaces.

These "cleaned sentences" are stored together.

TF-IDF stands for term frequency-inverse document frequency and it is a measure, used in the fields of information retrieval (IR) and machine learning, that can quantify the importance or relevance of string representations (words, phrases, lemmas, etc) in a document amongst a collection of documents

it is like calculating score for each sentence as per its statistical information.

```
class Preprocessing:
    #constructor
    def __init__(self,txt):
        # Tokenization
        nltk.download('punkt')  #punkt is nltk tokenizer
        # breaking text to sentences
        tokens = nltk.sent_tokenize(txt)
        self.tokens = tokens
        self.tfidfvectoriser=TfidfVectorizer()

    def clean_sentence(self, sentence, stopwords=False):
        sentence = sentence.lower().strip()
        sentence = re.sub(r'[^a-z0-9\s]', '', sentence)
        if stopwords:
          sentence = remove_stopwords(sentence)
        return sentence

    # store cleaned sentences to cleaned_sentences
    def get_cleaned_sentences(self,tokens, stopwords=False):
        cleaned_sentences = []
        for line in tokens:
          cleaned = self.clean_sentence(line, stopwords)
          cleaned_sentences.append(cleaned)
        return cleaned_sentences

    #do all the cleaning
    def cleanall(self):
        cleaned_sentences = self.get_cleaned_sentences(self.tokens, stopwords=True)
        cleaned_sentences_with_stopwords = self.get_cleaned_sentences(self.tokens,
        # print(cleaned_sentences)
        # print(cleaned_sentences_with_stopwords)
        return [cleaned_sentences,cleaned_sentences_with_stopwords]

    # TF-IDF Vectorizer
    def TFIDF(self,cleaned_sentences):
        self.tfidfvectoriser.fit(cleaned_sentences)
        tfidf_vectors=self.tfidfvectoriser.transform(cleaned_sentences)
        return tfidf_vectors

    #tfidf for question
    def TFIDF_Q(self,question_to_be_cleaned):
        tfidf_vectors=self.tfidfvectoriser.transform([question_to_be_cleaned])
        return tfidf_vectors

    # main call function
    def doall(self):
        cleaned_sentences, cleaned_sentences_with_stopwords = self.cleanall()
        tfidf = self.TFIDF(cleaned_sentences)
        return [cleaned_sentences,cleaned_sentences_with_stopwords,tfidf]
```

This is a class for answering the question that we will be putting.

The Parameter that we are using is the EUclidean distance.

```
class AnswerMe:
    #Euclidean distance
    def Euclidean(self, question_vector, sentence_vector):
        vec1 = question_vector.copy()
        vec2 = sentence_vector.copy()
        if len(vec1)<len(vec2): vec1,vec2 = vec2,vec1
        vec2 = np.resize(vec2,(vec1.shape[0],vec1.shape[1]))
        return np.linalg.norm(vec1-vec2)

    # main call function
    def answer(self, question_vector, sentence_vector, method):
        return self.Euclidean(question_vector,sentence_vector)
```

Function to retrieve the answer

```
def RetrieveAnswer(question_embedding, tfidf_vectors,method=1):
  similarity_heap = []
  max_similarity = float('inf')

  index_similarity = -1

  for index, embedding in enumerate(tfidf_vectors):
    find_similarity = AnswerMe()
    similarity = find_similarity.answer((question_embedding).toarray(),(embedding).
    if method==1:
      heapq.heappush(similarity_heap,(similarity,index))
    else:
      heapq.heappush(similarity_heap,(-similarity,index))
  return similarity_heap
```

The question can be put here.

Some other sample questions are:

1. What number has the global number of confirmed cases of COVID-19 has surpassed?
2. What does the World Health Organization (WHO) remind all countries and communities
3. How can the spread of the virus be slowed and the impact reduced?
4. What does WHO call on all countries to do?
5. Who is demonstrating that spread of the virus can be slowed?

```
user_question = "What does WHO call on all countries to do?"

preprocess = Preprocessing(txt)
cleaned_sentences,cleaned_sentences_with_stopwords,tfidf_vectors = preprocess.doall

question = preprocess.clean_sentence(user_question, stopwords=True)
question_embedding = preprocess.TFIDF_Q(question)

similarity_heap = RetrieveAnswer(question_embedding , tfidf_vectors ,method)
print("Question: ", user_question)
print()

#we are printing just one sentence
number_of_sentences_to_print = 1
while number_of_sentences_to_print>0 and len(similarity_heap)>0:
  x = similarity_heap.pop(0)
  print(cleaned_sentences_with_stopwords[x[1]])
  number_of_sentences_to_print-=1
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
Question:  What does WHO call on all countries to do?

who calls on all countries to continue efforts that have been effective in lim
```