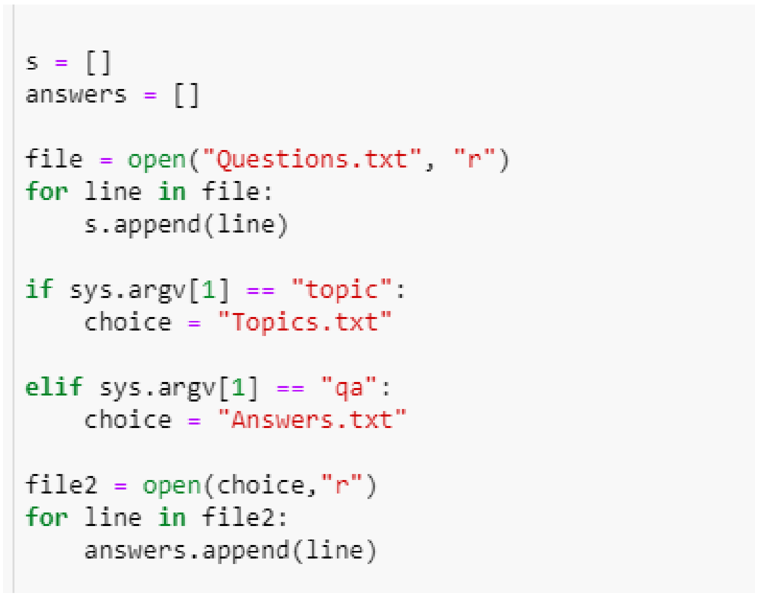
Claire Chemutai
Zaneta Asare
Bruno Edoh
Ethel Adongo

Implementation of the cosine similarity for both the Topic modelling and Q&A

Cosine similarity was implemented to model both topic and Question-Answer problem. Implementation steps taken to implement this are as below.

Handle data

The first step is to load the data files needed. This is the file containing the questions and answers and the other contains the topics. We open the files using open functions and read data lines. We then append the read lines into an array



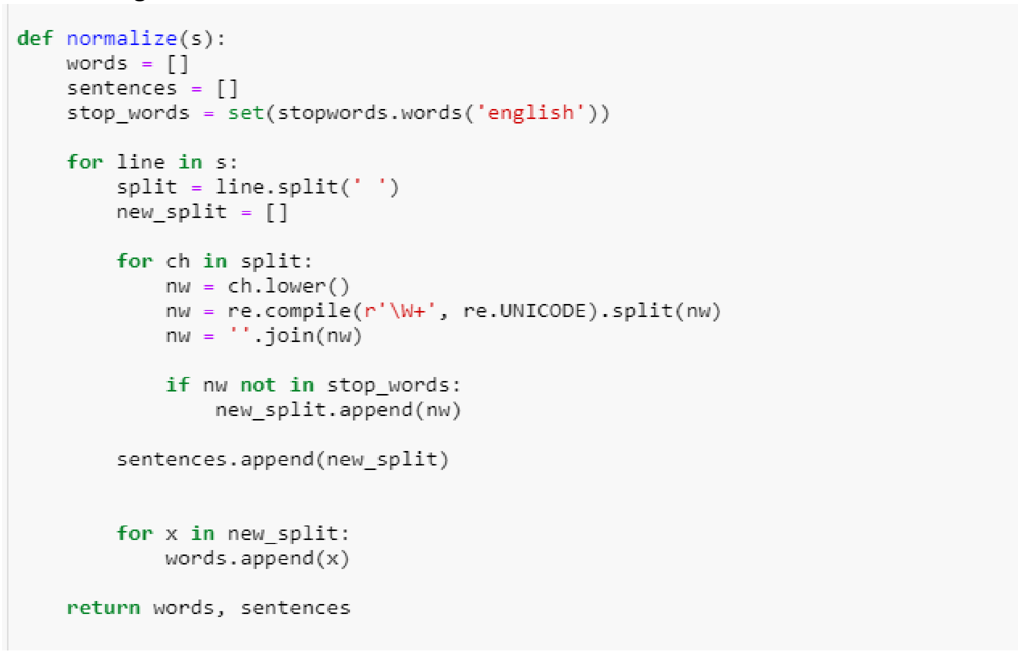
```
s = []
answers = []

file = open("Questions.txt", "r")
for line in file:
    s.append(line)

if sys.argv[1] == "topic":
    choice = "Topics.txt"

elif sys.argv[1] == "qa":
    choice = "Answers.txt"

file2 = open(choice,"r")
for line in file2:
    answers.append(line)
```

Normalization

A normalize function is then created in order to normalize the loaded data and put it in a more convenient form.

In normalization, we set stopwords, which consist of only English words. The data is also changed all to lower case. We then return two arrays, one consisting of the words, and the other consisting of the sentences.

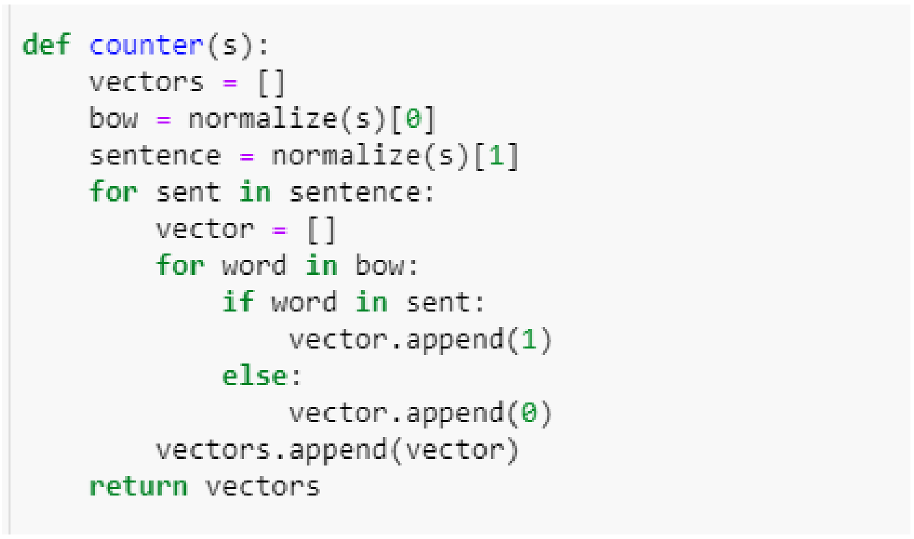
Claire Chemutai
Zaneta Asare
Bruno Edoh
Ethel Adongo



```python
def normalize(s):
    words = []
    sentences = []
    stop_words = set(stopwords.words('english'))

    for line in s:
        split = line.split(' ')
        new_split = []

        for ch in split:
            nw = ch.lower()
            nw = re.compile(r'\W+', re.UNICODE).split(nw)
            nw = ''.join(nw)

            if nw not in stop_words:
                new_split.append(nw)

        sentences.append(new_split)


        for x in new_split:
            words.append(x)

    return words, sentences
```

Computing counters

We compute the count of each word in every sentence. 1 assigned to the words that appear in a sentence and 0 assigned to the words that do not appear in a sentence

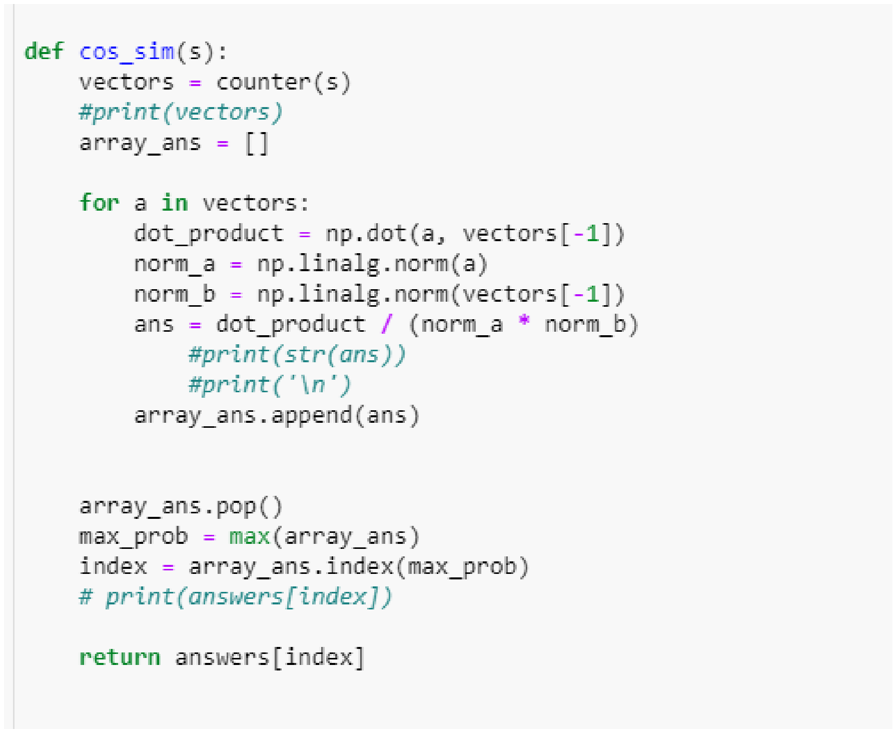


```python
def counter(s):
    vectors = []
    bow = normalize(s)[0]
    sentence = normalize(s)[1]
    for sent in sentence:
        vector = []
        for word in bow:
            if word in sent:
                vector.append(1)
            else:
                vector.append(0)
        vectors.append(vector)
    return vectors
```

Claire Chemutai
Zaneta Asare
Bruno Edoh
Ethel Adongo

Cosine similarity

We take a sentence question input by the user, normalize it and compute the counter too. We then use the dot product to compare between the sentence questions asked by the user to the sentence questions in the data file. We pick the one with the highest similarity. Using the index number of the question picked from the question file, we can pick either the corresponding answer and return it as the answer to the question the user asked, or return the corresponding topic.



```
def cos_sim(s):
    vectors = counter(s)
    #print(vectors)
    array_ans = []

    for a in vectors:
        dot_product = np.dot(a, vectors[-1])
        norm_a = np.linalg.norm(a)
        norm_b = np.linalg.norm(vectors[-1])
        ans = dot_product / (norm_a * norm_b)
            #print(str(ans))
            #print('\n')
        array_ans.append(ans)


    array_ans.pop()
    max_prob = max(array_ans)
    index = array_ans.index(max_prob)
    # print(answers[index])

    return answers[index]
```