CH5019 PROJECT

THE CORRECTION MAESTRO

Group 67

Tamminaina Sai Gowtham -ED19B063

Aaditya Edupuganti - AE19B104

Royyuru Sai Prasanna Gangadhar - ME19B190

Course Professor : Dr.Raghunathan Rengaswamy

INDEX

June 3, 2022

Contents

List of Figures

1 Problem Statement of the project

Machine Learning and Artificial Intelligence have been making the lives of people very easier, we are looking at one such case where the professor's or at least his TA's life has been made easier.

We observe that professors are very busy these days as they are being engaged in their respective research works and hence this algorithm is useful for them in balancing their work load.

In our given project, we were asked to develop an algorithm for finding out whether the descriptive answers given by students in a written exam actually corresponds to the main answer

We can do this either by checking if the answer is right or wrong, or by estimating how much percentage of the given answer is matched with the original template.

2 Approach used in our code

A **Condition** for test cases is that the students are required to answer according to what they have been taught by the prof or the text book, this is put so that the keywords get matched and to avoid synonyms.

For our project, we wanted to keep things easier and would only classify people's marks based on some categories listed below.

1. 0 - 20 % Keywords matched- 0 Marks awarded
2. 20 - 40 % Keywords matched- 0.5 Marks awarded
3. 40 - 70 % Keywords matched- 0.8 Marks awarded
4. 70 - 100 % Keywords matched- 1 Marks awarded

We have kept these conditions by manually going through the test cases with some intuitions and also to account for synonym usage in the answer.

There are many approaches to this algorithm and we have taken TF - IDF to be our method here.

TF-IDF is a method which gives us a numerical weightage of words which reflects how important the particular word is to a document in a corpus. A corpus is a collection of documents. Tf is Term frequency, and IDF is Inverse document frequency. This method is often used for information retrieval and text mining. For detailed explanation on this method, please refer to this research paper.
In simple words, what this method does is that it assigns the values to keywords and these values will imply the importance of the word in that paragraph. If a student uses these high rated words in his/her answers, then he/she has a higher chance of getting their answer correct.

3 Our code

We first remove the Stop Words (words such as "a","the","of" etc) from both the professor's as well as the student's answer.

Then we consider the remaining ones, i.e. the **Key Words** for correction.This means that the Keywords of the Professor and that of student's are only compared for evaluation.
A detailed process is involved in the code (with explanation in form of comments) the output gives marks for the students.
Given below is the code for our program

```
# -*- coding: utf-8 -*-
"""MFDS final submission.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1jHJ3TdbduawT49jPmSHZO75jpcUPSt-J
"""

import nltk
from nltk.corpus import stopwords
from nltk import tokenize
from nltk.tokenize import word_tokenize
from operator import itemgetter
import math



with open('Professor answer1.txt') as f:
    profanswer = f.read()
print(profanswer)

txt = ...
    ['s1.txt','s2.txt','s3.txt','s4.txt','s5.txt','s6.txt','s7.txt','s8.txt','s9.txt','s1
studentans = []
for i in txt:
    f= open(i,'r')
    contents = f.read()
    studentans.append(contents)
    f.close()

text = profanswer

stop_words=set(stopwords.words('english'))
total_words=profanswer.split()
total_word_length=len(total_words)
print(total_word_length)

total_sentences = tokenize.sent_tokenize(profanswer)
total_sent_len = len(total_sentences)
print(total_sent_len)

tf_score = {}
for each_word in total_words:
    each_word = each_word.replace('.','')
    if each_word not in stop_words:
        if each_word in tf_score:
            tf_score[each_word] += 1
        else:
            tf_score[each_word] = 1

# Dividing by total_word_length for each dictionary element
tf_score.update((x, y/int(total_word_length)) for x, y in tf_score.items())
```

```
print(tf_score)

def check_sent(word, sentences):
    final = [all([w in x for w in word]) for x in sentences]
    sent_len = [sentences[i] for i in range(0, len(final)) if final[i]]
    return int(len(sent_len))

idf_score = {}
for each_word in total_words:
    each_word = each_word.replace('.','')
    if each_word not in stop_words:
        if each_word in idf_score:
            idf_score[each_word] = check_sent(each_word, total_sentences)
        else:
            idf_score[each_word] = 1

# Performing a log and divide
idf_score.update((x, math.log(int(total_sent_len)/y)) for x, y in ...
   idf_score.items())

print(idf_score)

tf_idf_score = {key: tf_score[key] * idf_score.get(key, 0) for key in ...
   tf_score.keys()}
print(tf_idf_score)

def get_top_n(dict_elem, n):
    result = dict(sorted(dict_elem.items(), key = itemgetter(1), reverse = ...
       True)[:n])
    return result

text_list=list(dict(tf_idf_score).keys())
print(list(dict(tf_idf_score).keys()))
len(text_list)

def evaluate(doc,counter):
    total_words = doc.split()
    total_word_length = len(total_words)
    #print(total_word_length)
    total_sentences = tokenize.sent_tokenize(doc)
    total_sent_len = len(total_sentences)
   # print(total_sent_len)
    tf_score = {}
    for each_word in total_words:
        each_word = each_word.replace('.','')
        if each_word not in stop_words:
            if each_word in tf_score:
                tf_score[each_word] += 1
            else:
                tf_score[each_word] = 1

    # Dividing by total_word_length for each dictionary element
    tf_score.update((x, y/int(total_word_length)) for x, y in ...
       tf_score.items())
    #print(tf_score)
    def check_sent(word, sentences):
        final = [all([w in x for w in word]) for x in sentences]
        sent_len = [sentences[i] for i in range(0, len(final)) if final[i]]
        return int(len(sent_len))
    idf_score = {}
    for each_word in total_words:
```

```
            each_word = each_word.replace('.','')
            if each_word not in stop_words:
                if each_word in idf_score:
                    idf_score[each_word] = check_sent(each_word, total_sentences)
                else:
                    idf_score[each_word] = 1

        # Performing a log and divide
        idf_score.update((x, math.log(int(total_sent_len)/y)) for x, y in ...
            idf_score.items())

        #print(idf_score)
        tf_idf_score = {key: tf_score[key] * idf_score.get(key, 0) for key in ...
            tf_score.keys()}
        #print(tf_idf_score)
        def get_top_n(dict_elem, n):
            result = dict(sorted(dict_elem.items(), key = itemgetter(1), ...
                reverse = True)[:n])
            return result
        test_list=list(dict(tf_idf_score).keys())
        count=0
        for i in test_list:
          for j in text_list:
            if (i==j):
              count+=1
              #print(i)
        #print(count)
        if(count>=0 and count<=round(0.20*len(text_list))):
            print("Marks: 0.0--"+' Student'+str(counter))
        elif (count>round(0.20*len(text_list)) and ...
            count<=round(0.4*len(text_list))):
            print('Marks: 0.5--'+' Student'+str(counter))
        elif count>round(0.4*len(text_list)) and count<=round(0.7*len(text_list)):
            print('Marks: 0.8--'+' Student'+str(counter))
        else:
            print('Marks:1.0--'+' Student'+str(counter))

z = 0
for i in studentans:
    z = z+1
    evaluate(i,z)
```

In the code above, we have used the test case of a **Paragraph about tsunami** and have our 10 written student answers, i.e. The test cases.
The text files of the professor's along with the student answers are given in this drive link.
The ipynb formatted code is also presented in this drive link. After running through the code. We get the output of marks of students as shown in fig:1.

```
Marks: 0.5-- Student1
Marks: 0.5-- Student2
Marks: 0.5-- Student3
Marks: 0.5-- Student4
Marks: 0.0-- Student5
Marks: 0.5-- Student6
Marks: 0.0-- Student7
Marks: 0.8-- Student8
Marks: 0.0-- Student9
Marks: 0.0-- Student10
```

Figure 1: Marks of 10 students for the essay

4 Conclusion

From the report, we can make out the points like:

1. Many students have scored 0.5 marks, only 1 scored 0.8 marks and the rest scored 0 marks. This implies that the correction maestro is a bit strict in awarding marks but is also doing justice for relative grading on the other hand.
2. The TF-IDF scores method is a good method in identifying the importance of some specific keywords in paragraph.
3. In this report, we didn't account for the grammatical errors since the removed all the stop words from our final list.

5 References

- Reference for TF-IDF scores Method
- Reference on Stop words used in NLT

6 *Acknowledgment*

We, the group 67, have not copied from other people's report and have only used the references mentioned in section of references.
Thank you Dr.Raghunathan Rengaswamy and TA Karthick Raj for your support in this course and also for providing insights for this project
Work Shared by the team:

1. Gowtham ED19B063 - Ideation of the code and majority of it
2. Aaditya AE19B104 - Report Writing and assisting test cases
3. Gangadhar ME19B190 - Preparing test cases and Assisting in code writing.