Research Article

Reinforced Model for Abstractive Summarization

Vijeyata Ramkrushna Tupurwadi¹, Pratik Tukaram Shinde², Nida M. Shaikh³, Priyanka Goragh Nell⁴, S. S. Deshmukh⁵

BE Student¹, 2, 3, 4, Assistant Professor⁵
Department of Computer Engineering
AISSMS COE, Pune, India

Abstract:
In the modern technology environment, the need for faster results have become the major requirement from any system. We opt for the process which is intelligent enough to provide us with accurate and faster results. In our work, we propose to build a Framework which helps to get a Summarized format for any given dataset. There are many popular approaches such as Feature Selection(FS) and Feature Extraction(FE). Earlier these approaches where Studied independently and the final result would either contain the Selected Feature or just the extracted in other words transformed Feature. The work we are aiming for is to combine both the Features to get a Reduced and Minimum data. It is focused on the Theoretical Paper which can be conducted like the online papers i.e. MCQ type questions and the way the results are immediately displayed our Framework does the same for Theory papers and generates a Faster result. It also Maintains the orthogonality property among the selected original and linear combinations of features of a dataset. A unified iterative algorithm, for both supervised and unsupervised cases, is also developed under this framework. With the help of this we are keeping a great hand on encouraging the use of Green Computing.

Index Terms: Artificial Intelligence, Green Computing, Machine, Learning, Summarization.

I. INTRODUCTION:
In the gift there's solely the net paper i.e. MCQ question that have the power to get a fast and immediate Results, however once it involves Theory papers we tend to area unit lacking behind in providing the outputs quicker. That the Framework is made in such the simplest way that makes a outline of any given datasets. The big-data analysis is formed here and solely the vital options area unit extracted and reworked. The techniques of Machine learning and computing play a serious role and additionally offer with the right ways. Including inexperienced Computing technique is additionally AN another goal. the speculation papers that area unit written on papers that results in a loss of surroundings health is slow down because the writing of answers are shifted on Computers and with this Framework we tend to generate a dataset that encompass vital keywords and solely the Stem words area unit hold on within the dataset. The interface selects solely those words and generate a summarized result. Therefore, by doing this we tend to get a right away result for the speculation paper that saves not solely the paper however additionally the time. By mistreatment many alternative ways, the Framework is build that aims in providing with quick Results and Reduced information.

II. LITERATURE SURVEY:
Sreevani and C. A. Murthy. [1] It is crucial to cut back the spatiality of the info to enhance each the potency and effectiveness of most of the info mining algorithms, additionally it’s vital for higher image, information compression, noise removal, improved understanding ability, and generalization of the training algorithms. ancient and progressive spatiality reduction strategies be two categories: feature choice and have extraction. These approaches are with success applied in several real applications, like Image process, text categorization, bioinformatics. Feature choice aims at finding a set of most helpful options from the initial set of options, whereas feature extraction strategies offer combos (linear or nonlinear) of the initial options. Stefan Berchtold, Daniel A. Keim, Hans-Peter Kriegel. [2]The problem of finding all part similar components could be a troublesome task. The progressive approach preponderantly employed in business could be a search supported feature vectors. Feature vectors comprises a hard and fast range of attributes describing necessary properties of the polygons. expertise shows, however, that an exploration supported feature vectors isn't comfortable, since solely apriori outlined options will be employed in looking for similar components. what's required to realize higher results is to permit a similarity search supported the precise pure mathematics of the polygons given by the CAD system.

III. BRIEF DESCRIPTION
Green Computing is the Digitization process of converting information into digital format. In this format information is organized into discrete units of data that can be separately addresses. To reduce use of paper for enhancement of Green Computing System which will conduct popular approaches defined for Feature Selection(FS) and Feature Extraction (FE). It gives summarization of given dataset using Distributed database. In Education System the result is a dynamic process which needs to b continuously monitored. The aim of System is proposed for giving fast result. The prime goal was to develop a reliable System for Green Computing and better transformation in Education System

Aims and Objective:
- Better enhancement of results.
- Reduce the use of paper through online test.
- Fast out-turns
- Use of machine learning for summarization.

Problem Statement:
To process the exam assessment of comprehensive question using Artificial Intelligence model of NLP for effective text segmentation using Machine Learning approach to facilitate the need of faster result and centralized approach.

http://ijesc.org/
System Architecture:
System Description
User Input:
Answer of given question.
System Input:
Data set of specimen question and answer.
Functions (F):
Set Theory Analysis:
a. Let ‘S’ be the Error detection in big data as the final set S = {…………}
b. Identify the inputs as D
S = {D, …}
D = {D1, D2, D3, D4} ‘D’ gives data files

c. Identify the outputs as O
S = {D, L, A…
D = {D1, D2, D3, D4} ‘D’ gives data files
L = {L1, L2 …} ‘L’ gives the log files for Text Classification and KNN
A = {A1, A2, A3, …} ‘A’ gives Result
D. Identify the functions as ‘F’
S = {D, L, A, F…
F = {F1(), F2(), F3(), F4(), F5(), F6()}
F1(V): Data Preparation
F2 (V): Text Classification
F3 (V): User Input Validation
F4 (T): KNN Classification and Data Analysis
F4 (D): Storing in Data store and Result
F6 (V): User Alert
Hence the functionality can be shown as,

Success Condition:

Output: Percentage of similarity between the dataset and input given by user.

IV. ALGORITHMS:
Comparison between things, like garments, food, merchandise and even folks, is associate degree integral a part of our way of life. It's done by assessing similarity (or differences) between 2 or additional things. Except its usual usage as associate degree aid in choosing a thing-product, the comparisons area unit helpful in looking things ‘similar’ to what you have got and in classifying things supported similarity. This post describes a selected use-case of finding similarity between 2 documents.

Measuring Similarity:
Measure of similarity can be qualitative and/or quantitative. In qualitative, the assessment is done against subjective criteria such as theme, sentiment, overall meaning, etc. In the quantitative, numerical parameters such as length of the document, number of keywords, common words, etc. are compared. The process is carried out in two steps, as mentioned below:

1. Vectorization: Transform the documents into a vector of numbers. Following are some of the popular numbers (measures): TF (Term Frequency), IDF (Inverse Document Frequency) and TF*IDF.
2. Distance Computation: Compute the cosine similarity between the document vector. As we know, the cosine (dot product) of the same vectors is 1, dissimilar/perpendicular ones are 0, so the dot product of two vector-documents is some value between 0 and 1, which is the measure of similarity amongst them. Test-case used in this post is of finding similarity between two news reports [^1, ^2] of a recent bus accident (Sources mentioned in the References). Programming language ‘Python’ and its Natural Language Toolkit library ‘nltk’[^3] are primarily used here. The similarity analysis is done in steps as mentioned below.

Documents Pre-Processing
The news reports contain many things which are not core (or irrelevant) for text analysis exercise such as finding similarity. So, they are pre-processed by converting their words into lower case and removing the ‘stopwords’, like ‘the’, ‘should’, etc.
Victimization

Characterize each text as a vector. Each text has some common and some uncommon words compared to each other. To account for all possibilities, a word set is formed which consists of words from both the documents. There are various methods by which words can be vectorised, meaning, converted to vectors (array of numbers). A few of the prominent ones are explained below.

Frequency Count Method

A simplest way to create the vectors is to count number of times each word from the common word set, occurs in individual document. Feudist counts the number of occurrence of a word in the given text. So, in the above code snippet text1_count_dict has word-count pairs of all the words from the common word set, along with their individual counts. Following table shows few words with their frequencies:

<table>
<thead>
<tr>
<th>Word</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Westbou</td>
<td>1</td>
</tr>
<tr>
<td>Whether</td>
<td>1</td>
</tr>
<tr>
<td>Windo</td>
<td>1</td>
</tr>
<tr>
<td>workers</td>
<td>1</td>
</tr>
<tr>
<td>world</td>
<td>1</td>
</tr>
<tr>
<td>Yea</td>
<td>1</td>
</tr>
</tbody>
</table>

These vectors, in a crude way, represent their respective texts and similarity can be assessed amongst them. This is the ‘Containment Ratio’ method mentioned above. TF-IDF is much better measure to represent a document.

TF-IDF Method

TF is document specific. It is a way to score the importance of words (or “terms”) in a document based on how frequently they appear. If a word appears frequently in a document, it’s important, it gets a high score. Although it is easy to compute, it is ambiguous (‘green’ the colour and ‘green’ the person’s name is not differentiated).

```
from nltk.probability import FreqDist

word_set = set(text1).union(set(text2))

freq_text1 = FreqDist(text1)
for word in text1:
    freq_text1[word] = freq_text1[word] + 0

freq_text2 = FreqDist(text2)
for word in text2:
    freq_text2[word] = freq_text2[word] + 0

freq_text1 = FreqDist(text1)
for word in text1:
    freq_text1[word] = freq_text1[word] / len(text1)

freq_text2 = FreqDist(text2)
for word in text2:
    freq_text2[word] = freq_text2[word] / len(text2)
```


IDF is for the whole collection. It is a way to score how many times a word occurs across multiple documents. If a word appears in many documents, it’s not a unique identifier, thus gets a lower score.

$$\text{TFIDF of a word} = (\text{TF of the word}) \times (\text{IDF of the word})$$

Word Embedding Method

Of-late Word embedding are being used to vectorise words, and using that the whole documents. Google’s Word2Vec and Doc2Vec available from Python’s gensim library [^6] can be used to vectorise the news reports and then find similarity between them.

Once the words in the text are vectorised, the similarity score between them is nothing but the ‘distance’ between them.

Distance computation: Following are the steps to compute the similarity of two texts using TF-IDF Method. It is computed using the dot product of given vectors $v1$ and $v2$.

For the given two news items the similarity score came to about 72.62 %.

In case of Word Embedding method, the Doc2Vec model itself can compute similarity of given texts. For the given two news items the similarity score came to about 79.06 %.
Implementation:

```python
import nltk, string
from sklearn.feature_extraction.text import TfidfVectorizer

# download 'punkt' if necessary...

stemmer = nltk.stem.porter.PorterStemmer()
remove_punctuation_map = dict((ord(char), None) for char in string.punctuation)

def stem_tokens(tokens):
    return [stemmer.stem(item) for item in tokens]

"""remove punctuation, lowercase, stem""

def normalize(text):
    return stem_tokens(nltk.word_tokenize(text.lower().translate(remove_punctuation_map)))

vectorizer = TfidfVectorizer(tokenizer=normalize, stop_words='english')

def cosine_sim(text1, text2):
    tfidf = vectorizer.fit_transform([text1, text2])
    return ((tfidf * tfidf.T).A[0,1])

ans1="Paas is a cloud providing service. Types of cloud services are IaaS, Paas and SaaS."
ans2="There are different types of cloud services mainly IaaS, Paas, SaaS."

def main():
    print(str(cosine_sim(ans1, ans1)*100) + "% Similar")
    print(str(cosine_sim(ans1, ans2)*100) + "% Similar")
```

Table 1. Test Cases:

<table>
<thead>
<tr>
<th>Test Case Id</th>
<th>Test Case</th>
<th>Objectives</th>
<th>Steps</th>
<th>Expected Result</th>
<th>Actual Result</th>
<th>Test Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Data Acquisition</td>
<td>To Connect the Data Set link and Acquire the data for Model Training</td>
<td>API Class Object Object Created</td>
<td>Object Created</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>API Function Call Function Calling</td>
<td>Function Called</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>API Response Response Received</td>
<td>Response received from the object</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Data feed collection All data is collected in runtime object</td>
<td>Data collected</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Analysis</td>
<td>Check that all the functions are working properly according to plan</td>
<td>Parameters The output should be according to only selected parameters</td>
<td>The output is according to selected parameters</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Invalid Selection If invalid parameter that are not available an error should come</td>
<td>Display of error message</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Erroneous data Discard Erroneous data</td>
<td>Erroneous data Discarded</td>
<td>Fail</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Test data Collect all the Test</td>
<td>Data Collected</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>System</td>
<td>Check that the systems performance does not degrade</td>
<td>Performance System performance should not degrade</td>
<td>System performance does not degrade</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Application</td>
<td>Check that the application works properly with all system configuration and response time must not degrade</td>
<td>Generation of Modal Doctor Type Prediction should be done within minimum expected time.</td>
<td>The output is in time limit.</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Adding new Test data Test Data entered by user should not take more than expected time</td>
<td>Data is uploaded in the expected time range</td>
<td>Pass</td>
<td></td>
</tr>
</tbody>
</table>
### V. APPLICATION

It is used in Business Intelligence (BI), Survey of different Systems or Data. In Education System it will give faster result for paper checking. Used in Marketing Strategy. Input for Survey Reports for using Distributed Database.

### VI. FUTURE SCOPE:

For future work, we are planning to extend the proposed architecture to make it compatible with Big Data analysis for all types of exam.

### VII. CONCLUSION:

With the help of this framework we can check theoretical papers with faster speed and also we will be able to generate accurate result. And this leads in contribution to Green Computing.

### VIII. REFERENCES:

[1]. Sreevani, Student Member, IEEE and C. A. Murthy, “Bridging Feature Selection and Extraction: Compound Feature Generation”

[2]. Stefan Berchtold, Daniel A. Keim, Hans-Peter Kriegel, Institute for Computer Science, University of Munich, Oettingenstr. 67, D-80538 Munich, Germany; {berchtol, keim, kriegel}@dbs.informatik.uni-muenchen.de

[3]. BENJAMIN BUSTOS, DANIEL A. KEIM, DIETMAR SAUPE, TOBIAS SCHRECK, and DEJAN V. VRANIC

[4]. Neighbor Searching in Fixed Dimensions SUNIL ARYA, The Hong Kong University of Science and Technology, Kowloon, Hong Kong DAVID M. MOUNT AND ANGELA Y. WU The American University, Washington, D.C.


[8]. Li-Wei Kang, Member, IEEE, Chao-Yung Hsu, Hung-Wei Chen, Chun-Shien Lu, Member, IEEE, Chih-Yang Lin, Member, IEEE, and Soo-Chang Pei, Fellow, IEEE