

Descriptive Answers Evaluation Using Semantic Similarity

Donis Abraham
Department of Computer Application
Amal Jyothi College of Engineering
Kanjirappally, India
donisabraham2022a@mca.ajce.in

Sruthimol Kurian
Department of Computer Application
Amal Jyothi College of Engineering
Kanjirappally, India
sruthimolkurian@amaljyothi.ac.in

Abstract — To ensure the confidentiality, correctness, and impartiality of the assessment, an automatic descriptive replies evaluation method will be employed. In this research, we offer a system for evaluating descriptive replies called *Descriptive answers evaluation*, which uses several semantic similarity algorithm models. It aids in resolving the manual evaluation's dispute. For preprocessing, a branching and forming morphological variants of base words method was presented, which helps to reduce the content of descriptive type answers. This system compares sentences using bert-base-nli-mean-tokens and stsb-roberta-large models, and measures similarity using the cosine similarity algorithm.

Keywords—Semantic similarity, Machine learning, Cosine similarity, NLP.

I. INTRODUCTION

Everything is done online in today's environment. Today, numerous organisations and institutions provide aptitude tests in an online format. Everyone is mostly focused on conducting objective style exams in an online environment. Some institutions and colleges give tests online due to unforeseen circumstances, but the valuation is done offline. This exam is set out in a descriptive manner. Descriptive responses are lengthy written responses to queries. As a result, valuers faced numerous challenges while evaluating such online answer sheets in a descriptive model. This study shows how to analyse such descriptive answers using a computer intelligence-based strategy.

This system is based on Natural Language Processing (NLP), which is a subdomain of artificial intelligence (Artificial Intelligence). BERT (Bidirectional Encoder Representations utilising Transformers) and the STSB Roberta big model are used in the proposed system. STSB Roberta Large is a meaning pooling method that aids in the detection of semantic similarity. The major goal of this study is to analyse descriptive answers using the notion of phrase analysis.

II. LITERATURE SURVEY

Semantics analysis is a type of NLP that compares two pieces of textual data or scripts to see how similar they are[1]. There are a variety of algorithms that can be employed to do this[1]. The Siamese Manhattan LSTM technique is used in the model reported in this study (MaLSTM)[1]. The first phase is preprocessing, which comprises tokenization, stop word removal, and word vector initialization [2]. The second step is to compare the two statements to see how similar they are[2]. The similarity calculation depends on two measures[2]. The first is

semantic similarity, which is based on a pre-trained vector representation and a knowledge base called WordNet. [2]. The second is calculating word order similarity, which considers the difference in order of words in the two sentences[2]. This study proposes three methods for computing similarities between brief text data based on methods for computing feature vectors and similarity measures[3]. Using a vectorized tf-idf model, the pre-processed documents were turned into tf-idf vectors. A sparse matrix comprising tf-idf weights for each word of each document with a dimension of [number of documents * number of features(unique words)] was obtained[3]. The matrix's tf-idf weights are now used as a feature for each document, and document similarity is computed using cosine similarity[3]. The cosine similarity module in sklearn was used to calculate the similarity. [3]. The vector values or word embeddings of each word in all the preprocessed documents were computed using this word2vec model[3]. The word2vec vectors for terms that did not exist in the pre-trained model's vocabulary were set to zero[3]. A text's average word vector was generated, and the resulting vector was used as the document's feature word2vec vector. [3]. As a result, all of the documents were converted to word2vec vectors from n-gram vectors[3]. These word2vec vectors were used as feature vectors to compute the cosine similarity within the documents. [3].

The pre-processed documents were used to create a dictionary or a bag of words model, and the resulting dictionary was used to create a corpus (a sparse vector that holds unique word ids and the number of times they appeared)[3]. Because the soft cosine measure uses a similarity matrix between pairs of features, the dictionary (feature of the dataset) and term similarity index (features of the word2vec model) were used to construct a term similarity matrix. [3]. Another way for representing words semantically is deep learning[4]. A large corpus is utilised for training to find word representation in a semantic space[4]. The co-occurrence of terms in the corpus has an impact on the word representation that results. [4]. The goal of deep learning is to train the model to guess a word based on the context[4].

A vector representation for words can be learned using this technique[4]. Deep learning algorithms have demonstrated to be effective in semantically representing words[4]. The cosine similarity between the vectors of the words is used to measure word similarity. [4].

DOI: 10.5281/zenodo.6880934

ISBN: 978-93-5607-317-3@2022 MCA, Amal Jyothi College of Engineering Kanjirappally, Kottayam

Pre-processing techniques such as pruning and stemming are employed to reduce the number of student answers[5]. Take a look at answers A1, A2, A3, and so forth. Student advertisement for a specific question with answer key Ak[5]. Pruning and stemming techniques are used to each response. [5].

III. METHODOLOGY

The method of determining how two text documents are contextually similar is known as semantic similarity. The proposed system uses BERT and STSB Models to find the embedding of semantic Sentence similarity between more than one sentences. Then, between the embeddings, compute the cosine similarity score.

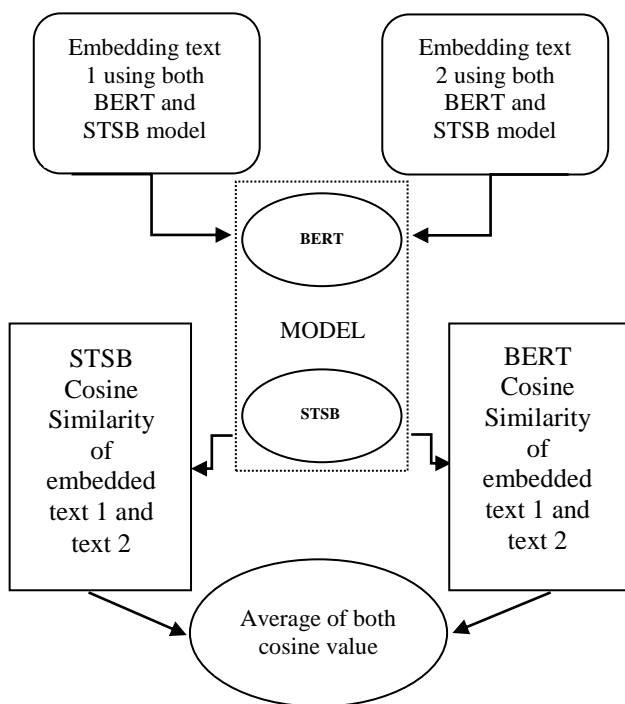


Fig 1: Proposed System

A. Semantic Similarity

It's easy to describe semantic similarity, but it's far more complicated to compute. People think intuitively in terms of symbols and concepts, so when you ask someone how two real-life objects relate, they will respond with a semantic similarity.

The context of semantic similarity is important. When determining whether two items are comparable, you must first define the elements of the objects you wish to examine. For instance, two goods may be comparable in price yet have vastly different feature sets. Neighbors may be similar in terms of location, but they may be completely different in terms of cultural heritage.

B. Word Embedding

Vectors (arrays of numbers) are used as input in machine learning models. When working with text, the first step is to devise a strategy for converting strings to numbers (or "vectorizing") before feeding it to the model. The technique

of transforming text data to numerical vectors is known as vectorization or word embedding. The numerical vectors are then utilized to create a variety of machine learning models.

C. Sentence-Transformers

Sentence-Transformers is a Python framework that allows you to create cutting-edge sentence, text, and image embeddings. Embeddings may be computed for more than 100 languages, and they're simple to utilize for applications like semantic text similarity, semantic search, and para-text mining. The framework is built on PyTorch and Transformers, and it includes a huge number of pre-trained models that may be used for a variety of tasks.

D. Models

1. BERT

Google created an approach for natural language processing pre-training called Bidirectional Encoder Representations from Transformers. It creates state-of-the-art models for a wide range of jobs using two steps: pre-training and fine-tuning. Its distinguishing characteristic is its uniform design across many downstream duties, which we will explain shortly. That means that the same pre-trained model can be fine-tuned for a range of final tasks that aren't necessarily similar to the task on which it was trained, and still produce results that are close to state-of-the-art.

2. STSB

The STSB is a type of a sentence-transformers model: It converts sentences and paragraphs into a dense vector space of 1024 dimensions, which can be utilized for applications like clustering and semantic search. The stsb bert-large model sentence transformers is a Natural Language Processing (NLP) model implemented in the Transformer library, which is written in Python.

E. Cosine Similarity

The similarity of two vectors in an inner product space is measured by cosine similarity. It determines whether two vectors are pointing in the same general direction by measuring the cosine of the angle between them. In text analysis, it's frequently used to determine document similarity. It is given by:

$$\text{similarity}(x,y) = \cos(\theta) = \frac{x \cdot y}{|x||y|}$$

IV. IMPLEMENTATION

The sentence-transformers package, which wraps most of this process into a few lines of code, is the simplest way for us to accomplish everything we just reviewed. First of all, we install required packages using pip

```

pip install sentence-transformers
pip install flask
  
```

```
from flask import Flask, render_template, request
from sentence_transformers import SentenceTransformer, util
import numpy as np
from transformers import AutoTokenizer, AutoModel
import torch
from sklearn.metrics.pairwise import cosine_similarity
app = Flask(__name__)
```

Fig 2: Importing packages

Initializing the models BERT and STSB in two variables.

```
app = Flask(__name__)
tokenizer = AutoTokenizer.from_pretrained('sentence-transformers/bert-base-nli-mean-tokens')
model = SentenceTransformer('bert-base-nli-mean-tokens')
model2 = SentenceTransformer('stsb-roberta-large')
```

Fig 3: Load the models

Build the vector from each sentence such as the input from user and the answer key.

```
answer=request.form['answer']
bert_embedding1=model.encode(answerkey)
bert_embedding2=model.encode(answer)
embedding1=model2.encode(answerkey,convert_to_tensor=True)
embedding2=model2.encode(answer,convert_to_tensor=True)
cosine_scores= util.pytorch_cos_sim(embedding1,embedding2)
bert_cosine_scores=util.pytorch_cos_sim(bert_embedding1,bert_embedding2)
```

Fig 4: Word Embedding

Now find the cosine values of each models. By implementing below code in the program.

```
value1= cosine_scores.item()
value2= bert_cosine_scores.item()
print("Cosine Score of model a:",value1)
print("Cosine Score of model b:", value2)
```

Fig 5: finding cosine values

Now take the average from each cosine value.

```
avg= (value1+value2)/2
print("total score:", avg)
```

Fig 6: Getting average of cosine values

V. RESULTS

The suggested system employs BERT (Bidirectional Encoder Representations Using Transformers) and the STSB Roberta large model. Using these models generate the vector values.

Fig 7: User interface

```
Bert token for Answer key ['Polymorphism is the ability of any data to be processed in more than one form']Bert token
Bert token for Answer key Bert token for Answer key [[-3.25154513e-02 -5.63746281e-02 -8.17802027e-02 7.91014016e-01
5.27946174e-01 -6.04915023e-01 -1.07489377e-01 5.88698447e-01
-3.34993601e-01 1.41963705e-01 -4.52703625e-01 3.07527155e-01
-5.09297885e-02 7.08218932e-01 3.03045690e-01 -1.02496989e-01
```

Fig 8: Resultant Vector value of answer key

```
The word "poly" means many and "morphs" means forms, So it means many forms.
Bert token for Answer [[-3.25154513e-02 -5.63746281e-02 -8.17802027e-02 7.91014016e-01
5.27946174e-01 -6.04915023e-01 -1.07489377e-01 5.88698447e-01
-3.34993601e-01 1.41963705e-01 -4.52703625e-01 3.07527155e-01
-5.09297885e-02 7.08218932e-01 3.03045690e-01 -1.02496989e-01
-3.73352379e-01 -4.67189908e-01 5.52471429e-02 -4.82238173e-01
```

Fig 9 : Resultant Vector value of answer

Now we find the cosine values

```
Cosine Score of model a: 0.6693179607391357
Bert_cosine0.6693179607391357
Bert_cosineCosine Score of model b: 0.6848541498184204 0.6848541498184204
0.6848541498184204
Cosine Score of model a:total score:Cosine Score of model a: 0.6693179607391357
Cosine Score of model b:0.66931796073913570.6770860552787781
Cosine Score of model b: 0.6848541498184204
0.6848541498184204
total score:total score:
0.6770860552787781
```

Fig 10: Resultant Cosine values

VI. CONCLUSION

In the online test pattern, it is vital to include automatic evaluation of the responses. Students' answer scripts are manually assessed to avoid biasing the results. The proposed model is based on a computer script that uses the word mover's distance to evaluate a descriptive exam response. Its main purpose is to compare the semantic similarity of documents' terms and vocabulary. The online input data was extracted out for the test after a thorough pre-processing step, and semantic similarity between the student response and the answer key will be calculated. The obtained findings show that the suggested model was successful in locating the semantic similarities between the input documents, and it performed brilliantly when compared to the manual. As a result, the overhead associated with that method is eliminated. One of the limitations mentioned here is the automatic evaluation of mathematical equations and visual similarity components. These limitations may be solved in the future, allowing for the development of new innovative enhanced learning models.

REFERENCES

- [1] Vedant Bahel, and Achamma Thomas, text similarity analysis for evaluation od descriptive answers, 2021.
- [2] Mohammad Farouk Sentance Semantic similarity based on word embedding and wordnet, 2018.
- [3] Pinky Sitikhu, kritish Pahi, Pujan Thapa, Subarma Shakya A comparison of Semantic Similarity Methods for Maximum Human Interpretability 2019.
- [4] Sunilkumar P, Athira P Shaji A survey on Semantic Similarity 2019.
- [5] Meena K, Lawrance R Sematic Similarity Based Assessment of DescriptiveTypeAnswers.