Question Recommendation System

Jayant Kumar Mittal
Department of Applied Mathematics
Delhi Technological University
New Delhi, India
jayant.mittal13@gmail.com

Kartikay Kashyap
Department of Applied Mathematics
Delhi Technological University,
New Delhi, India
kartikaykashyap12@gmail.com

Mahesh
Department of Applied Mathematics
Delhi Technological University
New Delhi, India
jangramahesh98@gmail.com

Abstract— In the Modern World automation has become more of a requirement than a desire. And with the massive usage of Online Education Platforms, the idea of automation can be an efficient tool. However, preparing questionnaires and quizzes related to a course is still a hassle. This paper is revolving around creating a recommendation system for suggesting questions related to a video. In order to test our system, we take a data set of multiple videos and process them to get their transcript (textual data) and then use a transformer to check for sentence similarity to recommend relevant questions according to the video source based on their similarity scores.

Keywords—Natural Language Processing, Semantic Text Similarity, Sentence-Similarity, Recommendation System, Education.

I. INTRODUCTION

In this digital era, the use of the web has increased due to the advancement of the internet, hence sorting and filtering data has become a vital task, where we can do wonders using sentence similarity. Due to the availability of an extensive and exhausting amount of textual data, measuring similarity between texts is one of the most important tasks which enables the necessity to use the Natural Language Processing (NLP) techniques.

In other words, sentence similarity is used in various natural language applications such as semantic search, summarization, question answering, document categorization, sentiment analysis, as well as plagiarism detection. Sentences might be lexically or semantically similar (String-based similarity is referred to as lexical similarity.)

A learning model for learning word representation forms a large corpus in Natural Language Processing. Word semantic properties are captured in the resulting vectors. In the semantic space, similar words should have closed vectors [1].

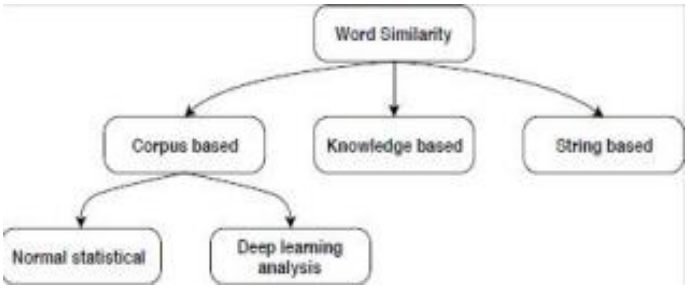


Fig 1. Different approaches for calculating

word-to-word similarity.

According to [1], the survey classifies, the approach of sentence similarities into three broad categories: -

1. *Word-to-word based* - In this category, measuring the similarities between two sentences is best done by using a word-to-word (Semantic Text Similarity) STS. Sometimes, we can combine one or even more methods of word similarity.

2. *Structure-based* – Structure-based STS uses structural information of sentences to improve the accuracy of measuring sentences' semantic similarity.

3. *Vector-based* - In vector-based STS sentence similarity is assessed in two steps: - First, a vector representation for the sentence is created and then the similarity is measured based on that generated vector. Vector-based method sentences are represented in two forms: statistical-based and learning-based.

Semantic textual similarity includes paraphrasing or duplicate detection, query, and matching applications. The standard method which is used for determining sentence similarity is obtaining the embedding of the sentences or it can be done by taking the cosine similarity between them.

In general, sentence similarity plays an important role in any search algorithm that analyses context. Currently, there are few recommendation models which operate on sentence similarities having low accuracy and precision measure.

So, in this paper, our goal is to create a recommendation system that uses Natural Language Processing (NLP) algorithms to determine sentence similarity for suggesting video content-related questions which can then be embedded into python videos which will be very helpful if anyone wants to hands-on apply and test the concepts.

IT professionals and students or Educational Institutions are the main focus groups for testing and adoption of this recommendation system. We have taken a group of 40 people for the testing phase in which 35 are students and 5 are python experts for in-depth reviewing and checking the usefulness and efficiency of our recommendation system.

The rest of the paper has been divided into various sections with the first one explaining the related works in the field of Semantic Text Similarity. In the second section, the experimental design of the proposed work/system is explained in detail. Then the final section articulates the observations obtained after performing the experimentation.

II. RELATED WORK

Across all the sources we have searched, we came across recommendation systems that are used for various fields in education like E-learning [2], Curriculum Recommendation and Scholarship Recommendation Systems, etc. However, we could not find any research paper that proposes a similar work that is done in this paper.

A. Papers using NLP for Sentence Similarity

1. Sentence Similarity Based on Semantic Nets and Corpus Statistics

In [3] the author proposed a method to measure sentence similarity between sentences of very short length by taking an algorithm that considers the implied semantic and word order information in the sentences.

A structured lexical database and corpus statistics are used to calculate the semantic similarity of two sentences. This method can mimic human common sense due to the usage of a lexical database and the incorporation of corpus statistics which makes it adaptable to other fields.

The similarity measure used by this method yielded a reasonable result with a Pearson correlation coefficient of 0.816. For this algorithm due to a lack of other published results on sentence similarities and a number of issues in re-implementing these algorithms for this domain, comparison with some of the other algorithms presented is now impossible. These include a large number of parameters that must be manually set as well as feature definition [3].

B. Deep Learning technique

1. A novel sentence similarity version with phrase embedding primarily based totally on convolution neural community.

2. Enhanced-RCNN: A green approach for studying sentence similarity.

In [4] the authors endorse a neural community-primarily based totally version, namely, “the phrase set vectors -the shallow convolutional neural community-the sentence vector” (WSV-SCNN-SV).

The fundamental frame of the version is the shallow convolutional neural community, which takes the organization of “phrase set vectors” because they enter the characteristic map and output the sentence representation—the sentence vector.

The “phrase set vector” is a brand new concept thinking about because of the development of the phrase embedding. Compared with phrase embedding, it carries greater semantic facts of sentences. Then they computed the similarity primarily based totally on the sentence vectors that were found via way of means of the convolutional neural community in place of using the convolutional neural community.

Meanwhile, without pre-education, the end result of this version does now no longer carry out well. As an end result, pre-education is an important element in enhancing this version's performance. They trust that their version will carry out higher performance if it's miles pre-skilled on a massive quantity of applicable education data.

In [5] proposed Enhanced-RCNN model is composed of three components: input encoding, interactive sentence representation, and similarity modeling.

Input encoding consists of three parts: bidirectional GRU (BiGRU) [5], CNN, and Pooling. Each part extracts different features from a multi-dimensional perspective. The module of interactive sentence representation aims to obtain an appropriate representation for each sentence with consideration of the interactive effects of the other sentence.

While the system performs well, the Enhanced-RCNN model without a CNN encoder fails to determine the similarity of two sentences.

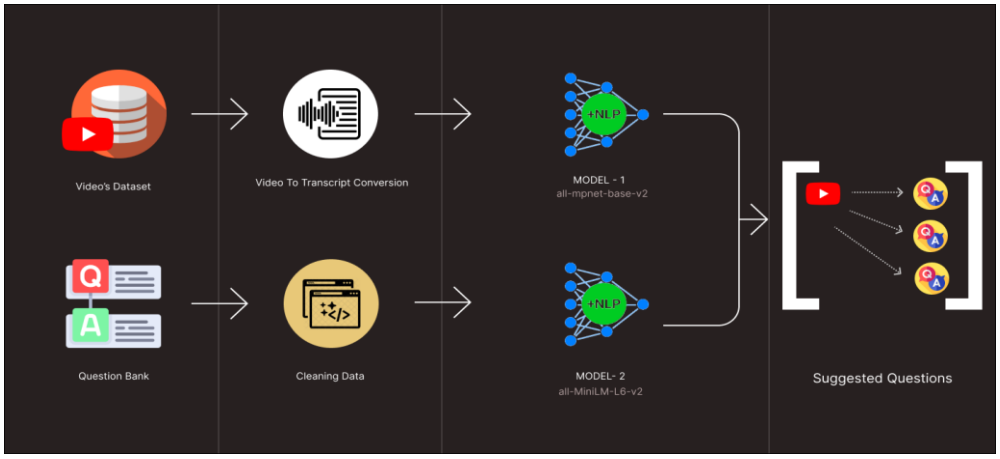


Fig 2. Flowchart depicting the methodology of the recommended technique.

III. EXPERIMENTAL DESIGN

Phase 1. Data Collection

For the purpose of this study, two data sets were compiled.

In the first one, a dataset consisting of around 100 python videos was compiled from different media sources. In the second data set, a question bank of around 1000 questions related to python was extracted from various online sources using Web Scraping.

Web scraping, web harvesting, or web data extraction is data scraping used for extracting data from websites. The web scraping software may directly access the World Wide Web using the Hypertext Transfer Protocol or a web browser [6].

Phase 2. Data Pre-Processing and Translation

In this, the video dataset consisted of two types of videos, some had their transcript embedded within them while the others didn't.

As a result, for the later ones the transcript was prepared using the "Speech-Recognition pydub" wherein the process of converting audio to text is divided into two parts.

First, breaking the audio into chunks, and second, extracting the content using Speech Recognition [7]. Then the video transcripts were converted into the JSON format for further computation.

The extracted data for the python related questions was cleaned of its HTML tags or any ambiguity and then stored in the required JSON format.

The questions and the transcript of the videos were then passed on to the next step for further calculations.

Phase 3. Model

In Sentence embedding, the entire sentence or text along with its semantics information is mapped into vectors of real numbers [8].

Thus, after processing the data, sentence embedding of video transcripts and the question bank were found.

The models used in to find the sentence embedding's are: -

1. *Sentence-transformers/ all-mpnet-base-v2*

This is a sentence-transformers model that maps sentences & paragraphs to a 768-dimensional dense vector space and can be used for tasks like clustering or semantic search.

The model is intended to be used as a sentence and short paragraph encoder. Given an input text, it outputs a vector that captures the semantic information. The sentence vector may be used for information retrieval, clustering, or sentence similarity tasks. By default, input text longer than 384-word pieces is truncated [9].

2. Sentence-transformers/all-MiniLM-L6-v2

This is a sentence-transformers model that maps sentences & paragraphs to a 384-dimensional dense vector space and can be used for tasks like clustering or semantic search.

The model is intended to be used as a sentence and short paragraph encoder. Given an input text, it outputs a vector that captures the semantic information. The sentence vector may be used for information retrieval, clustering, or sentence similarity tasks. By default, input text longer than 256-word pieces is truncated [10].

Similarity scoring is equivalent to seeking out points in the multidimensional space that are near enough to each other to potentially represent the same true location.

However, the curse of dimensionality refers to the increase in the "volume" of the search space as one adds additional dimensions [8].

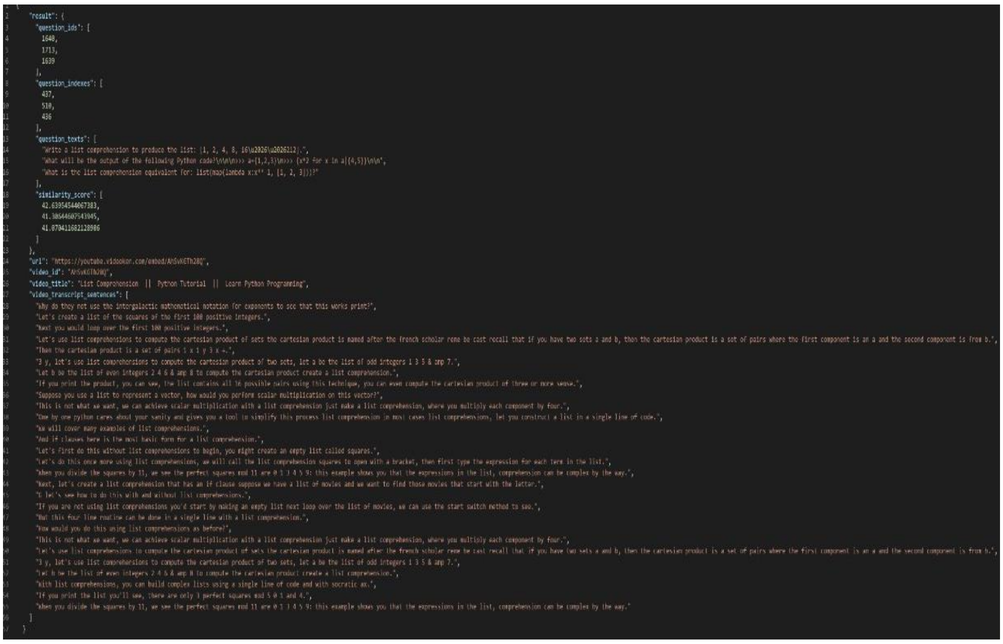
The similarity score of all questions with all the video transcripts was found using the vectors received in the previous step and the question indices, question_ids, question texts, and similarity score of the questions with maximum similarity score were returned for a particular video.

Phase 4. Validation

Human-Validation of our system was done in order to determine its usefulness and efficiency. 50 People comprising of 40 people of various python skill levels (novice, beginner, intermediate, advanced) and 10 Python Experts were made to watch 10 and 2 videos from our dataset respectively, and based on the watched videos, 5 questions each were suggested using both the models. All user's experiences and suggestions were recorded in the survey.


```
{
    "id": 1640,
    "question_text": "
Write a list comprehension to produce the list: [1, 2, 4, 8, 16……212].
",
    "options": {
      "options": [
        " [(2**x) for x in range(0, 13)]",
        " [(x**2) for x in range(1, 13)]",
        " [(2**x) for x in range(1, 13)]",
        " [(x**2) for x in range(0, 13)]"
      ]
    },
    "answer": {
      "answer": [
        0
      ],
      "answer_text": [
        " [(2**x) for x in range(0, 13)]"
      ],
      "explanation": "The required list comprehension will print the numbers from 1 to 12, each raised to 2. The required answer is thus, [(2**x) for x in range(0, 13)]."
    },
    "difficulty": "Moderate",
    "tags": [
      "List Comprehension"
    ],
    "categories": [
      "Python"
    ]
}
```


Fig 3. Sample of the question bank created in JSON




```
{
  "result": {
    "question_ids": [
      1640,
      1713,
      1639
    ],
    "question_indexes": [
      437,
      510,
      436
    ],
    "question_texts": [
      "Write a list comprehension to produce the list: [1, 2, 4, 8, 16\u2026\u2026212].",
      "What will be the output of the following Python code?\n\n\n>>> a=[1,2,3]\n>>> [x*2 for x in a[(4,5)]\n\n",
      "What is the list comprehension equivalent for: list(map(lambda x:x** 1, [1, 2, 3]))?"
    ],
    "similarity_score": [
      42.63954544067383,
      41.38644607543945,
      41.078411682128986
    ]
  },
  "url": "https://youtube.vidookon.com/embed/AhSvKGTh28Q",
  "video_id": "AhSvKGTh28Q",
  "video_title": "List Comprehension || Python Tutorial || Learn Python Programming",
  "video_transcript_sentences": [
    "Why do they not use the intergalactic mathematical notation for exponents to see that this works print?",
    "Let's create a list of the squares of the first 100 positive integers.",
    "Next you would loop over the first 100 positive integers.",
    "Let's use list comprehensions to compute the cartesian product of sets the cartesian product is named after the french scholar rene be cast recall that if you have two sets a and b, then the cartesian product is a set of pairs where the first component is an a and the second component is from b.",
    "Then the cartesian product is a set of pairs 1 x 1 y 3 x +.",
    "3 y, let's use list comprehensions to compute the cartesian product of two sets, let a be the list of odd integers 1 3 5 & amp 7.",
    "Let b be the list of even integers 2 4 6 & amp 8 to compute the cartesian product create a list comprehension.",
    "If you print the product, you can see, the list contains all 16 possible pairs using this technique, you can even compute the cartesian product of three or more sense.",
    "Suppose you use a list to represent a vector, how would you perform scalar multiplication on this vector?",
    "This is not what we want, we can achieve scalar multiplication with a list comprehension just make a list comprehension, where you multiply each component by four.",
    "One by one python cares about your sanity and gives you a tool to simplify this process list comprehension in most cases list comprehensions, let you construct a list in a single line of code.",
    "We will cover many examples of list comprehensions.",
    "And if clauses here is the most basic form for a list comprehension.",
    "Let's first do this without list comprehensions to begin, you might create an empty list called squares.",
    "Let's do this once more using list comprehensions, we will call the list comprehension squares to open with a bracket, then first type the expression for each term in the list.",
    "When you divide the squares by 11, we see the perfect squares mod 11 are 0 1 3 4 5 9: this example shows you that the expressions in the list, comprehension can be complex by the way.",
    "Next, let's create a list comprehension that has an if clause suppose we have a list of movies and we want to find those movies that start with the letter.",
    "G let's see how to do this with and without list comprehensions.",
    "If you are not using list comprehensions you'd start by making an empty list next loop over the list of movies, we can use the start switch method to see.",
    "But this four line routine can be done in a single line with a list comprehension.",
    "How would you do this using list comprehensions as before?",
    "This is not what we want, we can achieve scalar multiplication with a list comprehension just make a list comprehension, where you multiply each component by four.",
    "Let's use list comprehensions to compute the cartesian product of sets the cartesian product is named after the french scholar rene be cast recall that if you have two sets a and b, then the cartesian product is a set of pairs where the first component is an a and the second component is from b.",
    "3 y, let's use list comprehensions to compute the cartesian product of two sets, let a be the list of odd integers 1 3 5 & amp 7.",
    "Let b be the list of even integers 2 4 6 & amp 8 to compute the cartesian product create a list comprehension.",
    "With list comprehensions, you can build complex lists using a single line of code and with socratic ax.",
    "If you print the list you'll see, there are only 3 perfect squares mod 5 0 1 and 4.",
    "When you divide the squares by 11, we see the perfect squares mod 11 are 0 1 3 4 5 9: this example shows you that the expressions in the list, comprehension can be complex by the way."
  ]
}
```


Fig 4. Sample of the transcript of videos created in JSON

IV. RESULT

As per the survey conducted for the validation by python experts, the *all-mpnet-base-v2* and *all-MiniLM-L6-v2* models predicted question with 81% and 74% accuracy respectively.

According to the users with less expertise in python (novice, beginner, intermediate, advanced) for the *all-mpnet-base-v2* and *all-MiniLM-L6-v2* the accuracy ranges from 82-88% and 70-76% respectively.

Thus, according to the survey, the all-mpnet-base-v2 had an accuracy of around 81%, while the all-MiniLM-L6-v2 had an accuracy of around 75%.

	all-mpnet-base-v2	all-MiniLM-L6-v2
Python Experts	405/500	386/500
Novice	83/100	73/100
Beginner	85/100	70/100
Intermediate	80/100	76/100
Advanced	78/100	69/100
Total	731/900	674/900

Table I. Result of the survey conducted for the validation of the recommended questions.

V. CONCLUSION

The transcript of the videos collected and the question bank extracted from different online sources were processed and converted to JSON format. Later, the experiment was carried out with 2 different transformers and the result obtained was that Model-1 i.e. all-mpnet-base-v2 provides slightly better results than the later model i.e. all-MiniLM-L6-v2 which is shown in Table I.

In future research, a dynamic system can be created and Real-time-Stream of online classes be fed as input which can help teachers to test students' attention and help engage them more during the class. It is helpful for educational institutions to follow continuous evaluation.

VI. REFERENCES

1. M. Farouk, "Measuring Sentences Similarity: A Survey," Indian Journal of Science and Technology, p. 11, 2019.
2. J. G. Y. L. Huiyi Tan, "E-learning Recommendation System," in International Conference on Computer Science and Software Engineering, United States, 2008.
3. J. D. O. D. M. Z. B. Yuhua Li, "Sentence Similarity Based on Semantic Nets and Corpus Statistics," IEEE Transactions on Knowledge and Data Engineering, p. 14, 2006.
4. H. L. P. Z. Haipeng Yao, "A novel sentence similarity model with word embedding based on convolutional neural network: Sentence Similarity Model with Word Embedding based on Convolutional Neural Network," Concurrency and Computation Practice and Experience, p. 13, 2018.
5. C. H. N. X. S. L. Shuang Peng, "Enhanced-RCNN: An Efficient Method for Learning Sentence Similarity," WWW '20: The Web Conference 2020, p. 6, 2020.
6. WIKIPEDIA, "WEB SCRAPING," WIKIPEDIA, [Online]. Available: https://en.wikipedia.org/wiki/Web_scraping.
7. TUTORIALSPOINT, "PREPROCESSING," TUTORIALSPOINT, [Online]. Available: https://www.tutorialspoint.com/audio-processing-using-pydub-and-google-speech-recognition-api-in-python.

8. sciencedirect, "Similarity Score," sciencedirect, 2009. [Online]. Available: https://www.sciencedirect.com/topics/computer-science/similarity-score.
9. H. FACE, "all-mpnet-base-v2," HUGGING FACE, [Online]. Available: https://huggingface.co/sentence-transformers/all-mpnet-base-v2.
10. H. FACE, "all-MiniLM-L6-v2," HUGGING FACE, [Online]. Available: https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2.