Capstone Term II
04/09/2023

Hindi and Tamil Question Answering

Group #15
Bindia Biju (*100886575*)
Spandana Rajamahanthi (*100866969*)
Abishek Kailasam (*100856936*)
Sanath Davis (*100884693*)

Abstract

The final project progress report conveys the results and conclusions of the project. It explains the motivation behind the project and delves into the details of the dataset. The existing projects based on the same dataset are also explored and the differences and advantages of the project are outlined when compared to these current projects. Details and results of the conducted experiments are also laid out. Future steps are discussed.

Keywords: tamil, hindi, chaii, question answering, natural language understanding

1. Introduction

The field of natural language understanding (NLU) or interpretation (NLI) is a branch of natural-language processing in AI that focuses on machine comprehension of written language. This is a challenging problem in AI and is of significant commercial interest due to its various applications such as machine translation, question-answering, text categorization, and more. [1]

Figure 1. Natural Language Understanding - a representative image [2]

* bindia.biju@dcmail.ca, spandana.rajamahanthi@dcmail.ca, abishek.kailasam@dcmail.ca, sanath.davis@dcmail.ca

However, popular NLU models perform poorly when it comes to Indian languages like Hindi and Tamil, which are underrepresented on the web. This leads to subpar experiences for Indian users when using web applications.

This study aims to improve the performance of NLU models in Hindi and Tamil. We will use a new dataset, "chaii-1," that contains question-answer pairs in these languages, collected without the use of translation. The goal is to predict answers to real questions about Wikipedia articles.

The ultimate objective is to enhance the NLU performance in Hindi and Tamil and provide a better web experience for the 1.4 billion people in India. This research also contributes to the development of multilingual natural language processing, which could have wider applications beyond these two languages. The goal is to eliminate language barriers and help Indian users fully utilize the web. [3]

We would like to try to improve upon the current scores of accuracy. Much of the current work focuses on combining a similar dataset called TyDi with the chaii dataset. Also, much of the effort has been targeted at adding new data to the dataset. But we want to try it using the chaii dataset alone. Some of the existing work has not attempted to do a lot of hyperparameter fine-tuning. They just use the default parameters. We would like to try and improve performance by spending significant time on hyperparameter tuning. We also want to focus on the problem of tokenization which arises due to some punctuation marks. We also want to look at ways to re-rank the generated answers using a meta-model. [4]

2. Related Work

- Multilingual Contrastive Training for Question-Answering in Low-resource Languages [6]

  This is a well-researched paper on low-resource language NLU. This focuses on multiple Dravidian languages than just Hindi and Tamil. Also, according to them, they were not able to conduct enough hyperparameter tuning.
- chaii - EDA & Baseline [7]

  This provides a good EDA and Baseline analysis. We can improve upon this by doing even more EDA and pre-processing. Also, we will be trying to improve upon the score.
- chaii QA - 5 Fold XLMRoberta Torch | FIT [8]

  This is more in line with the aim of our project. This kernel implements the 5-Fold XLMRoberta QA Model. We would like to check other options and make the process a bit simpler. We will also provide a good comparison between various models.
- chaii-qa: Character/Token/Languages EDA [9]

  This is a good exploration of the data and focuses on tokenization which we would also like to explore. We would like to improve upon how they handled the unknown tokens and provide more insights into what some of these tokens can mean

3. Experimental Design

3.1. Main purpose

The high-level main purpose of the Chaii Hindi and Tamil Question Answering project is to encourage the development of accurate question-answering systems in low-resource languages like Hindi and Tamil. We aim to advance the state-of-the-art in natural language processing and promote research in the field. Ultimately, we seek to improve the quality of language processing tools for these languages and enable their widespread use in various applications.

3.2. Flowchart

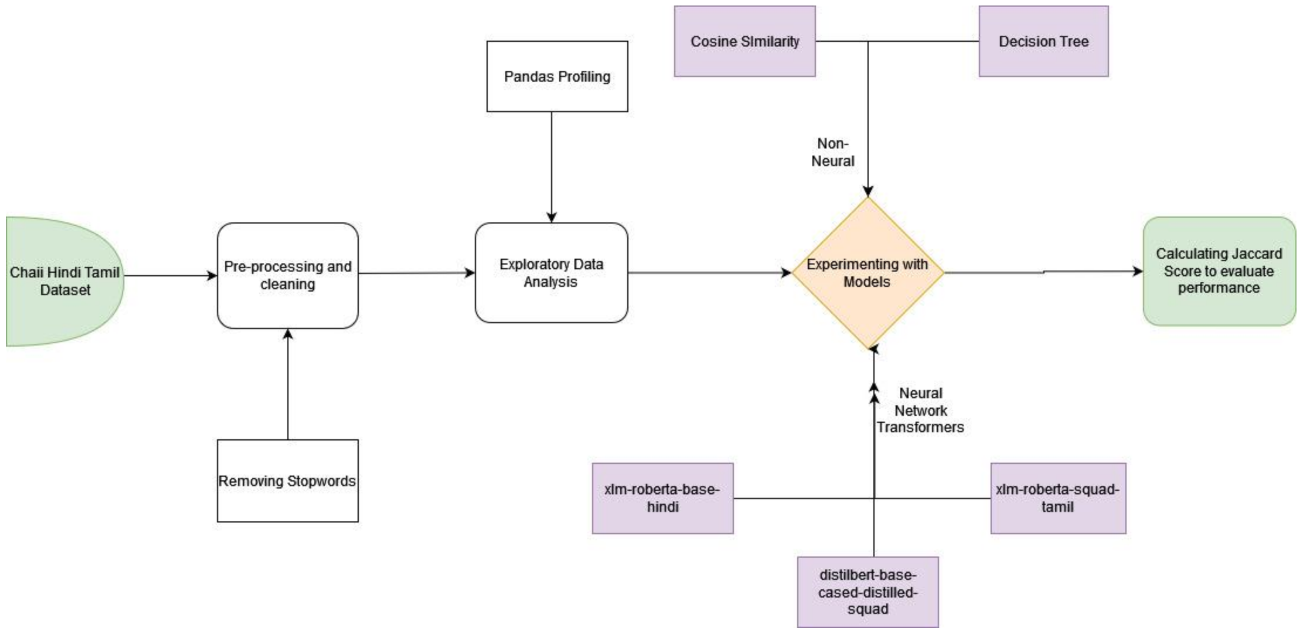

Chaii Hindi Tamil Dataset
Pre-processing and cleaning
Removing Stopwords
Pandas Profiling
Exploratory Data Analysis
Cosine Similarity
Decision Tree
Non-Neural
Experimenting with Models
Neural Network Transformers
xlm-roberta-base-hindi
xlm-roberta-squad-tamil
distilbert-base-cased-distilled-squad
Calculating Jaccard Score to evaluate performance

Figure 2.Updated flowchart of the machine learning system

3.3. Dataset Details

We will be using the chaii 2021 dataset for the project. It has 6 columns. A separate small Test Dataset is also provided.

The six columns are:

- *id* - a unique identifier
- *context* - the text of the Hindi/Tamil sample from which answers should be derived
- *question* - the question, in Hindi/Tamil
- *answer_text* (train only) - the answer to the question (manual annotation)
- *answer_start* (train only) - the starting character in context for the answer
- *language* - whether the text in question is in Tamil or Hindi

The chaii 2021 dataset was created using a two-step process. During the first step, annotators were presented with Wikipedia text snippets and asked to generate questions that they wanted to have answered, making sure that the questions could not be answered from the provided text. The questions were meant to have clear and precise answers.

In the second step, for each question generated in step one, the top result from a Google search was selected and annotators were instructed to identify the answer within that page.

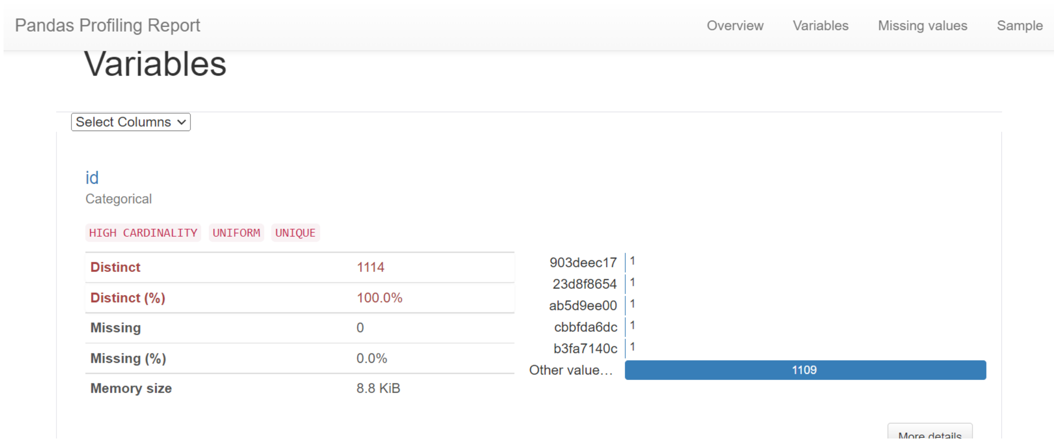
The selected answer was to be the first valid one found in the document. For Hindi questions, only Hindi Wikipedia pages were used, and the same was done for Tamil.

Dataset:
https://www.kaggle.com/competitions/chaii-hindi-and-tamil-question-answering/data [5]

Exploratory graphs:

Pandas Profiling Report

Pandas-profiling generates profile reports from a pandas DataFrame which is handy for exploratory data analysis.



id
Categorical

HIGH CARDINALITY UNIFORM UNIQUE

Distinct	1114
Distinct (%)	100.0%
Missing	0
Missing (%)	0.0%
Memory size	8.8 KiB

903deec17	1
23d8f8654	1
ab5d9ee00	1
cbbfda6dc	1
b3fa7140c	1
Other value…	1109

Figure 3. Pandas Profile Report

```
Number of training samples:  1114

Training data language count:
 hindi    746
tamil    368
Name: language, dtype: int64
```

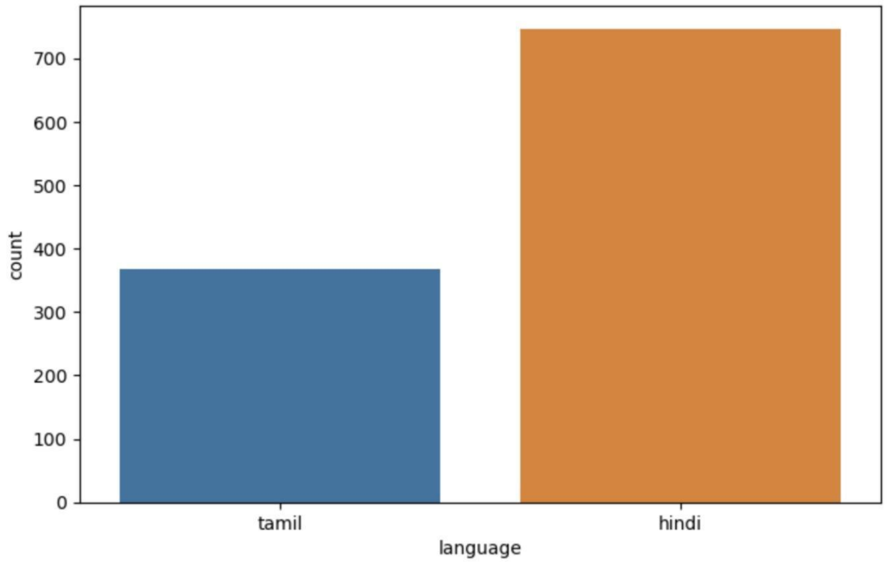


Figure 4. Count of the languages in the dataset

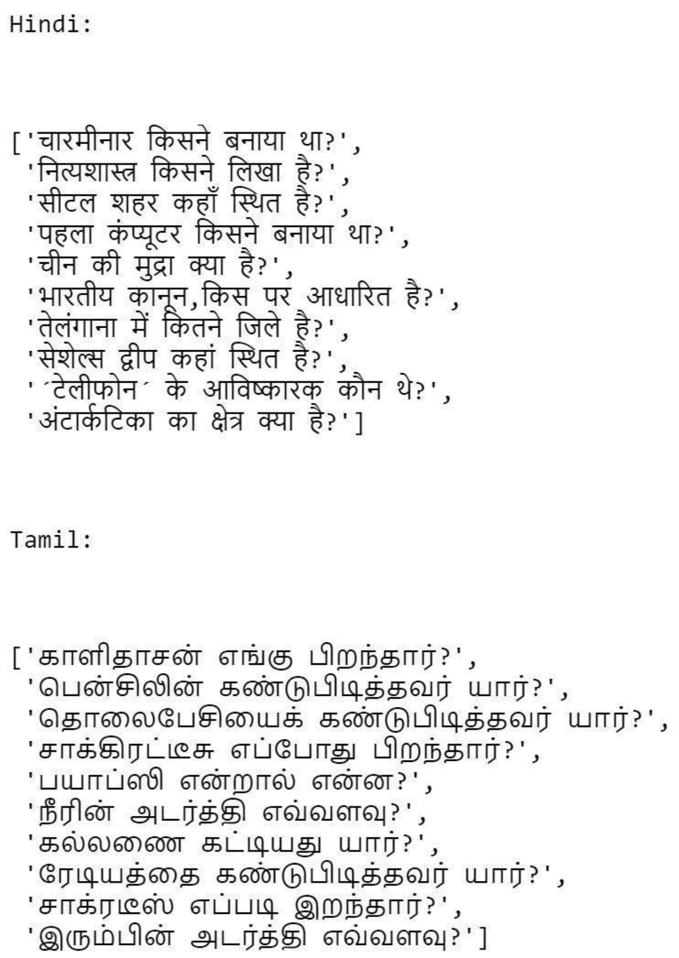
```
Hindi:


['श्यामजी कृष्ण वर्मा ने इंडियन होम रूल सोसाइटी को किस साल में स्थापित किया था?',
 'भारत सरकार ने धनतेरस को किस दिवस के रूप में मनाने का निर्णय लिया है?',
 'चेन्नई में २६ दिसंबर २००४ को आई सुनामी के दौरान कितने लोगों की मौत हुई थी?',
 'भारत में कृषि में किस दशक के मध्य तक पारंपरिक बीजों का प्रयोग किया जाता था?',
 'हम साथ साथ हैं 1999 की फील में सलमान खान के किरदार का नाम क्या था?',
 'उत्तरी हिंद महासागर में बना हुदहुद तूफ़ान किस वर्ष का अब तक का सबसे ताकतवर तूफान है?',
 'किसने अल्जीरिया में काम करते हुए पहली बार लाल रक्त कोशिका के अन्दर परजीवी को देखा था?',
 'किस वैज्ञानिक ने यह विचार रखा कि कोशिकाएँ सदा कोशिकाओं के विभाजन से ही पैदा होती हैं?',
 'सन १८८६ में किसने बताया कि तम्बाकू में मोजेक रोग एक विशेष प्रकार के वाइरस के द्वारा होता है?',
 'हॉलिवुड प्रेसबिटेरियन मेडिकल सेंटर को कम्प्यूटरों का डेटा फिर से पाने के लिए हैकर्स को कितने डॉलर की फिरौती देनी पड़ी थी?']


Tamil:


['காச நோய்க்கு எந்த ஆண்டு முதல் முதலில் மருந்து கண்டுபிடிக்கப்பட்டது?',
 'பெரியம்மை நோய் எந்த ஆண்டு இந்தியாவில் முதல் முதலில் தோன்றியது?',
 'முதல் குளிர்கால ஒலிம்பிக் விளையாட்டு போட்டி யாரால் தொடக்கி வைக்கப்பட்டது?',
 'தென்கிழக்கு ஆசிய ஒப்பந்த அமைப்பில் எத்தனை நாடுகள் உறுப்பினர்களாக உள்ளன?',
 'ஜப்பானில் முதல் முதலில் வீசப்பட்ட அணு குண்டின் பெயர் என்ன?',
 'நாடக ஆசிரியர் வில்லியம் சேக்சுபியர் எப்போது திருமணம் செய்து கொண்டார்?',
 'திமுக அரசியல் கட்சித் தலைவர் மு. கருணாநிதி எப்போது இறந்தார்?',
 'பூட்டு செய்யும் தொழிலாளியான பீட்டர் ஹென்கின் எந்த ஆண்டு கடிகாரம் உருவாக்கினார்?',
 '2009ஆம் ஆண்டில் 2 ஆஸ்கார் விருதை வென்ற இந்திய இசையமைப்பாளர் யார்?',
 'இந்திய அறிவியல் அறிஞர் சர் சந்திரசேகர வெங்கட ராமன் எப்போது பிறந்தார்?']
```

Figure 5. Largest Questions in each language



```
Hindi:


['चारमीनार किसने बनाया था?',
 'नित्यशास्त्र किसने लिखा है?',
 'सीटल शहर कहाँ स्थित है?',
 'पहला कंप्यूटर किसने बनाया था?',
 'चीन की मुद्रा क्या है?',
 'भारतीय कानून,किस पर आधारित है?',
 'तेलंगाना में कितने जिले है?',
 'सेशेल्स द्वीप कहां स्थित है?',
 '´टेलीफोन´ के आविष्कारक कौन थे?',
 'अंटार्कटिका का क्षेत्र क्या है?']


Tamil:


['காளிதாசன் எங்கு பிறந்தார்?',
 'பென்சிலின் கண்டுபிடித்தவர் யார்?',
 'தொலைபேசியைக் கண்டுபிடித்தவர் யார்?',
 'சாக்கிரட்டீசு எப்போது பிறந்தார்?',
 'பயாப்ஸி என்றால் என்ன?',
 'நீரின் அடர்த்தி எவ்வளவு?',
 'கல்லணை கட்டியது யார்?',
 'ரேடியத்தை கண்டுபிடித்தவர் யார்?',
 'சாக்ரடீஸ் எப்படி இறந்தார்?',
 'இரும்பின் அடர்த்தி எவ்வளவு?']
```

Figure 6. Shortest Questions in each language

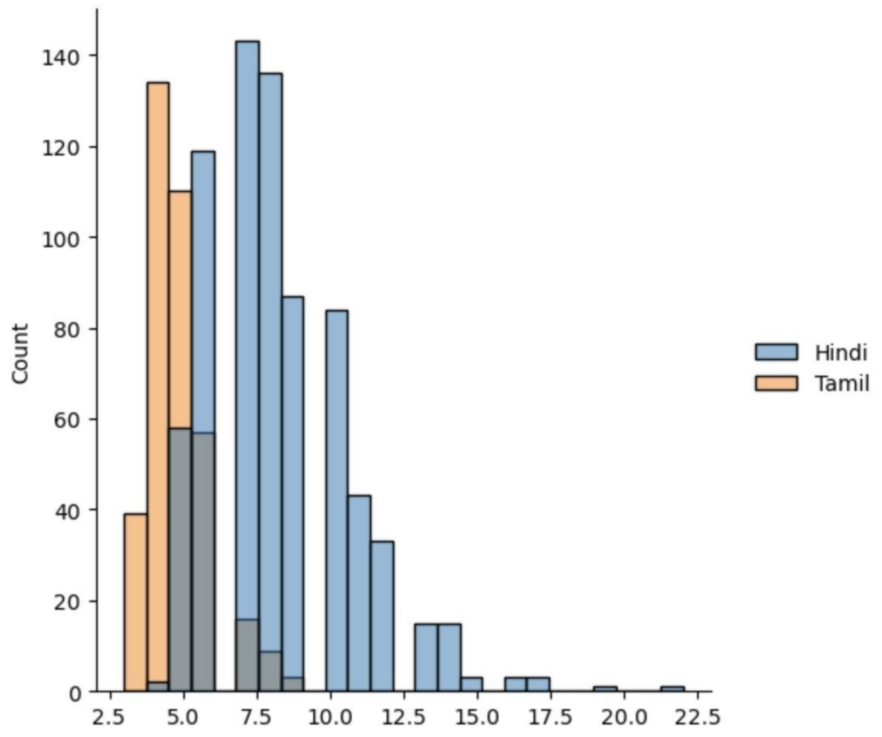


Figure 7. Distribution of questions length

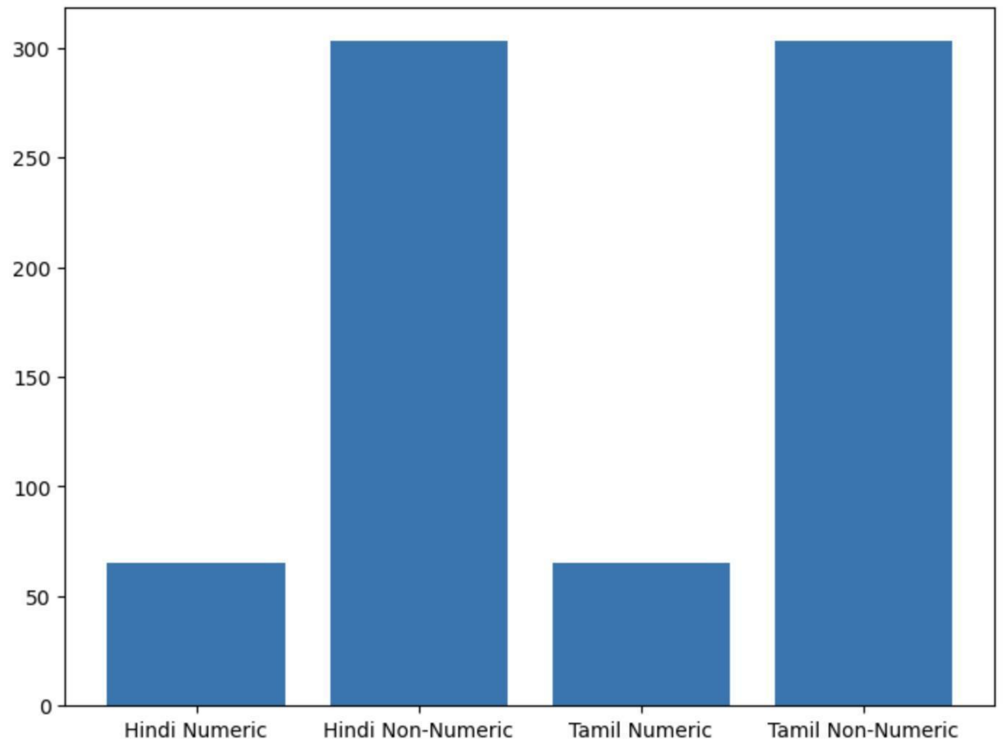


Figure 8. The number of numeric answers in each language

3.4. Data Preparation

3.4.1. Removing unwanted columns

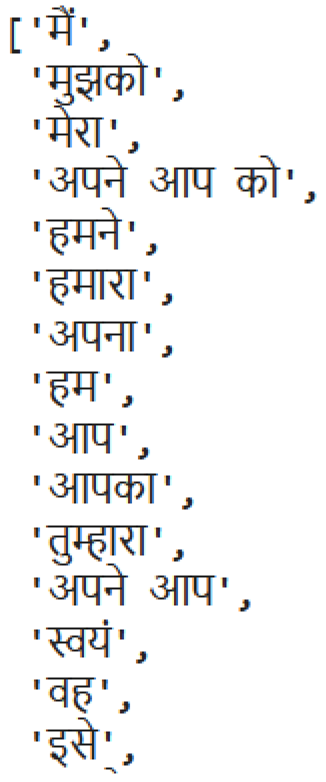
The column answer_start was not required and was removed.

3.4.2. Removing stopwords

Stop words are commonly used words in a language such as "the," "a," "an," "in," "is," "and," etc., that do not carry any significant meaning on their own and are generally removed from text during the data cleaning process in Natural Language Processing (NLP) applications.

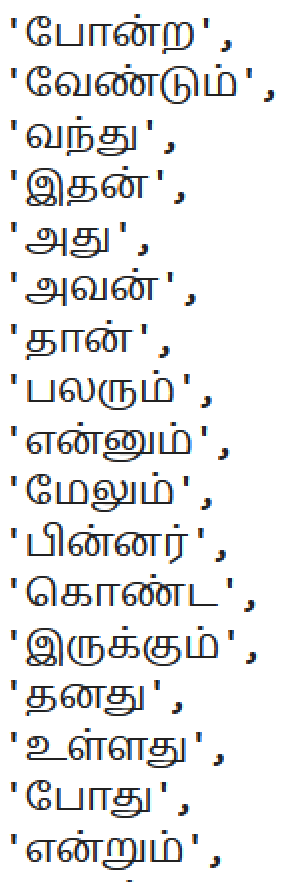
Removing stop words can be useful for several reasons:

- Efficiency: Stop words often appear frequently in text, which can slow down the processing of large amounts of text data. By removing stop words, the processing time can be reduced and the efficiency of NLP algorithms can be improved.
- Accuracy: Stop words can also add noise to the text data, as they do not add much value to the meaning of the text. By removing them, the accuracy of NLP algorithms can be improved as the focus is shifted to the more important words in the text.
- Consistency: Removing stop words can help in creating consistent representations of text data, which can help in tasks such as document classification, sentiment analysis, and topic modeling.



```
['मैं',
 'मुझको',
 'मेरा',
 'अपने आप को',
 'हमने',
 'हमारा',
 'अपना',
 'हम',
 'आप',
 'आपका',
 'तुम्हारा',
 'अपने आप',
 'स्वयं',
 'वह',
 'इसे',
```

Figure 9. Some Stopwords in Hindi [10]



```
'போன்ற',
'வேண்டும்',
'வந்து',
'இதன்',
'அது',
'அவன்',
'தான்',
'பலரும்',
'என்னும்',
'மேலும்',
'பின்னர்',
'கொண்ட',
'இருக்கும்',
'தனது',
'உள்ளது',
'போது',
'என்றும்',
```

Figure 10. Some Stopwords in Tamil [11]

3.5. Model Training and Evaluation

We conducted three types of experiments:

3.5.1. Cosine Similarity

We use the TF-IDF vectorizer and cosine similarity to find the sentence in the context that is most similar to the question. Finally, it prints the answer from the most similar sentence. [12]

3.5.2. Decision Tree Classifier

The code fits the Decision Tree algorithm on the features and labels and predicts the sentence in the context that contains the answer. Finally, it extracts the answer from the predicted sentence and prints it. [13]

3.5.3. Transformer 1 - distilbert-base-cased-distilled-squad

We use the pipeline function from the Hugging Face Transformers library to load a pre-trained question-answering model. The default model is the distilbert-base-cased-distilled-squad modal. It then defines the context and the question and uses the model to generate an answer. Finally, it prints the answer. [14]

3.5.4. Transformer 2 - Hindi Specific Model

To cater to the needs of the Hindi questions alone, we use the AswiN037/xlm-roberta-squad-tamil modal. AswiN037/xlm-roberta-squad-tamil is a Tamil language variant of the XLM-RoBERTa model fine-tuned on the Squad dataset, which is a question-answering benchmark dataset. XLM-RoBERTa is a multilingual version of the RoBERTa model, which is a masked language model trained on a large corpus of text data. Fine-tuning on the Squad dataset specifically helps the model to learn how to answer questions by extracting relevant information from a given context.

AswiN037/xlm-roberta-squad-tamil is trained to understand and generate responses in Tamil language, making it suitable for tasks that require question-answering capabilities in Tamil. It can be used for various applications such as information retrieval, conversational agents, and language understanding tasks that involve answering questions in Tamil. [17]

3.5.5. Transformer 3 - Tamil Specific Model

The Bhavikardeshna/xlm-roberta-base-hindi is a variant of the XLM-RoBERTa model that has been fine-tuned on Hindi language data. XLM-RoBERTa is a multilingual version of the RoBERTa model, which is a state-of-the-art masked language model trained on a large corpus of text data. Fine-tuning on Hindi language data specifically helps the model to understand and generate responses in Hindi language with improved accuracy.

bhavikardeshna/xlm-roberta-base-hindi can be used for a wide range of natural language processing (NLP) tasks that require understanding or generation of text in Hindi language. This includes tasks such as sentiment analysis, text classification, named entity recognition, text generation, and other language understanding or generation tasks. [18]

3.6. Baseline

We chose the cosine similarity and its evaluation scores as the baseline. Cosine Similarity is a simple approach and comparing it with it is a good way of measuring improvements.

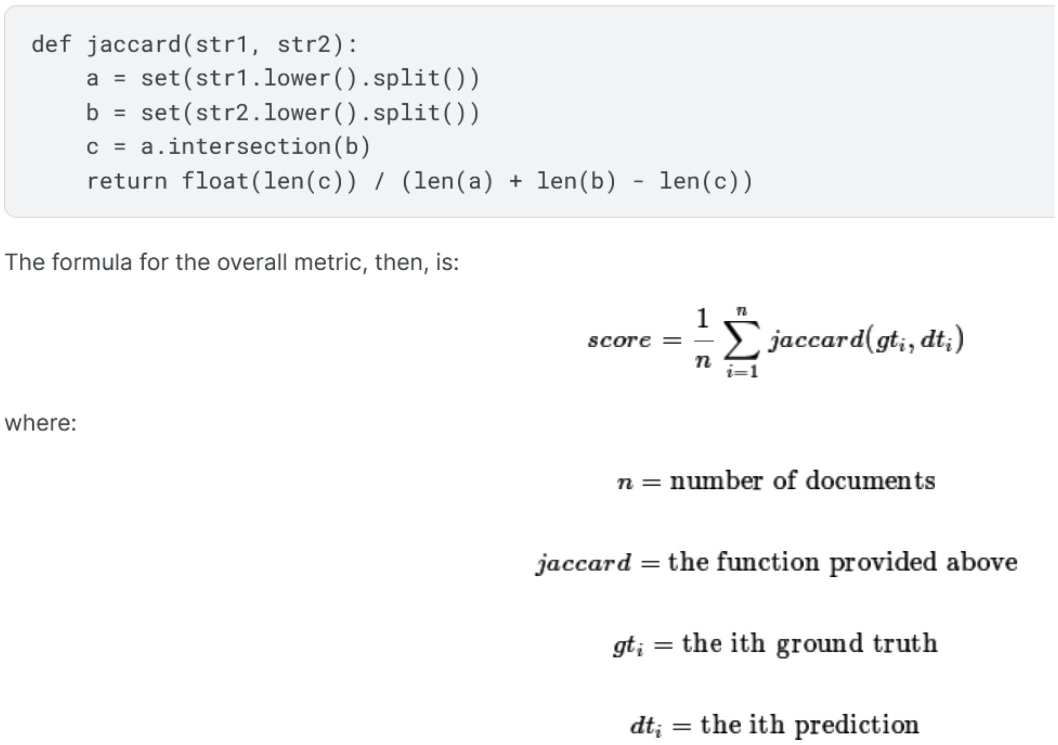
3.7. Performance Metrics - The Jaccard Score

In this project, we will rely on the word-level Jaccard Score for evaluating and comparing performances.

The word-level Jaccard score is a measure of similarity between two sets of words that compares the intersection and union of the words in the two sets. It is calculated as the size of the intersection of the two sets divided by the size of their union.

In Natural Language Processing (NLP), the word-level Jaccard score is often used in tasks such as text classification, information retrieval, and text similarity analysis. It can help identify the similarity between two texts based on the overlap of words in the texts.

For example, it can be used to compare the similarity of two documents, to identify the similarity of search queries, or to evaluate the performance of text classification models. One of the advantages of using the word-level Jaccard score is that it is simple to calculate and can be computed efficiently for large amounts of text data. [15]



```
def jaccard(str1, str2):
    a = set(str1.lower().split())
    b = set(str2.lower().split())
    c = a.intersection(b)
    return float(len(c)) / (len(a) + len(b) - len(c))
```

The formula for the overall metric, then, is:

$$score = \frac{1}{n}\sum_{i=1}^{n} jaccard(gt_i, dt_i)$$

where:

$$n = \text{number of documents}$$

$$jaccard = \text{the function provided above}$$

$$gt_i = \text{the ith ground truth}$$

$$dt_i = \text{the ith prediction}$$

Figure 11. The code and formula for calculating the Jaccard Score [16]

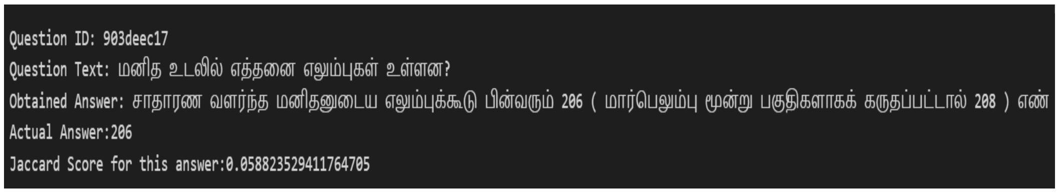
4. Experimental Results

4.1. Main Findings

- Through the various experiments we found out how to properly clean the dataset, specifically removing the stopwords from the dataset using the Hindi and Tamil Stopword list and tokenizing.
- Through the various explorative analyses and pandas profiling, we got many inferences about the data.
- We used three types of models and found out the Jaccard scores of each.
- We then proceeded to use three specific neural network transformers with very good success rates
- We found that the bhavikardeshna/xlm-roberta-base-hindi model provided the best scores.

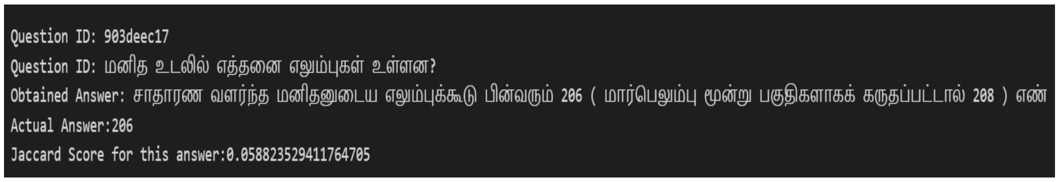
4.2. Figures

We can see the answer to the Tamil Question "How many bones are there in a human body" here in each of the three models. The Context given is a Wikipedia article about the human body. The correct answer is '206'.



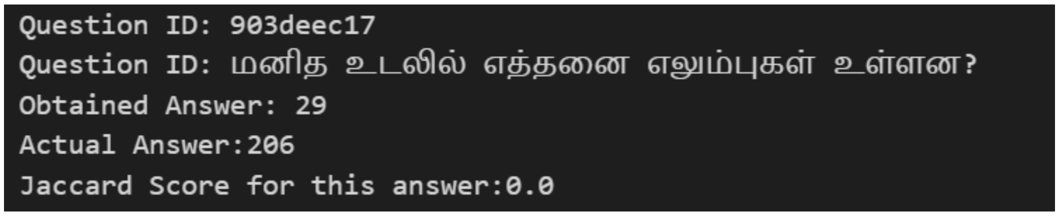
```
Question ID: 903deec17
Question Text: மனித உடலில் எத்தனை எலும்புகள் உள்ளன?
Obtained Answer: சாதாரண வளர்ந்த மனிதனுடைய எலும்புக்கூடு பின்வரும் 206 ( மார்பெலும்பு மூன்று பகுதிகளாகக் கருதப்பட்டால் 208 ) எண்
Actual Answer:206
Jaccard Score for this answer:0.058823529411764705
```

Figure 12. Answer for the first question using Cosine Similarity



```
Question ID: 903deec17
Question ID: மனித உடலில் எத்தனை எலும்புகள் உள்ளன?
Obtained Answer: சாதாரண வளர்ந்த மனிதனுடைய எலும்புக்கூடு பின்வரும் 206 ( மார்பெலும்பு மூன்று பகுதிகளாகக் கருதப்பட்டால் 208 ) எண்
Actual Answer:206
Jaccard Score for this answer:0.058823529411764705
```

Figure 13. Answer for the first question using Decision Tree



```
Question ID: 903deec17
Question ID: மனித உடலில் எத்தனை எலும்புகள் உள்ளன?
Obtained Answer: 29
Actual Answer:206
Jaccard Score for this answer:0.0
```

Figure 14. Answer for the first question using the default Transformer approach

Here we can see the overall Jaccard score of the three models:

Model	**Overall Jaccard Score**
Cosine Similarity	0.0219
Decision Tree	0.0154
Default Neural Network Transformer	0.0273

Figure 15. Overall Jaccard Score Comparison

Here we can see the overall Jaccard score of the three Neural Network models:

Model	**Overall Jaccard Score**
distilbert-base-cased-distilled-squad (Default Question Answering Model)	0.0273
xlm-roberta-base-hindi	0.6522
xlm-roberta-squad-tamil	0.5183

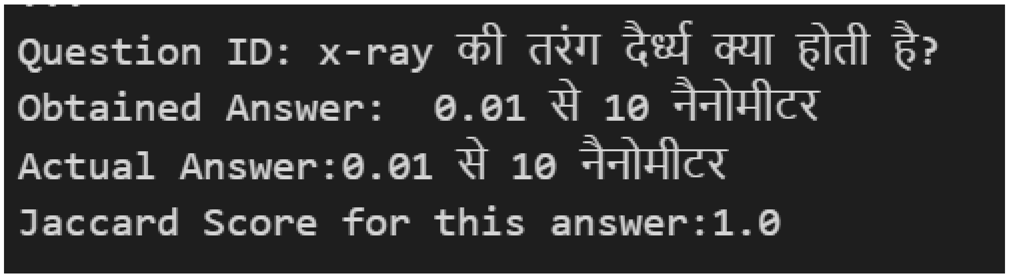
Figure 15. Overall Jaccard Score Comparison of Transformers only

Here we can also see the time taken for running each model:

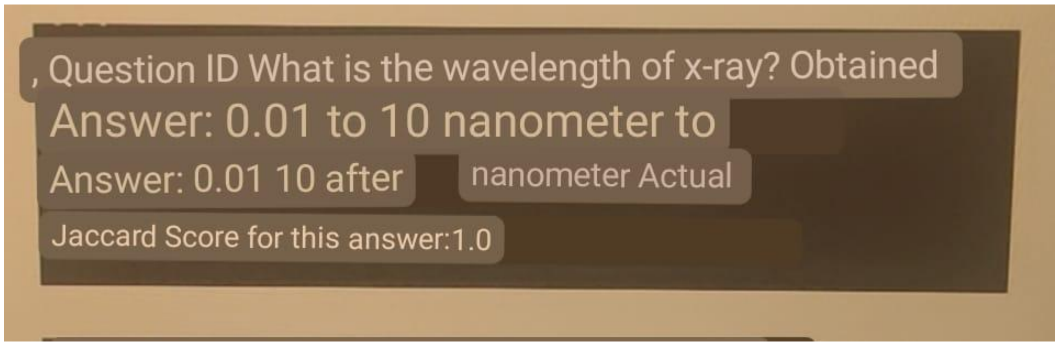
Model	Time Taken
Cosine Similarity	23.1 Seconds
Decision Tree	23.6 Seconds
Default Neural Network Transformer	128 Minutes
Hindi Specific Transformer	218 Minutes
Tamil Specific Transformer	406 Minutes

Figure 16. Time Taken Comparison

Now we can see some of the attempts at answering questions by the Hindi Specific Transformer. We will add translations using Google Lens for easy understanding:

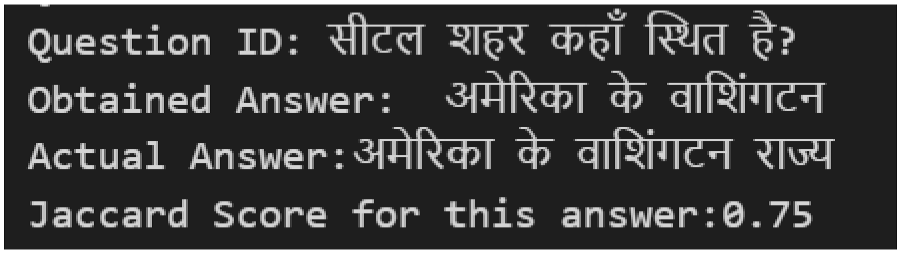


```
Question ID: x-ray की तरंग दैर्ध्य क्या होती है?
Obtained Answer:  0.01 से 10 नैनोमीटर
Actual Answer:0.01 से 10 नैनोमीटर
Jaccard Score for this answer:1.0
```

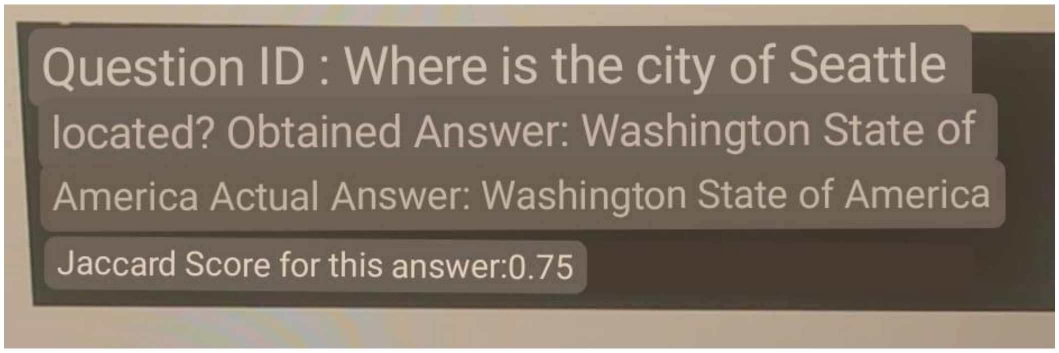


, Question ID What is the wavelength of x-ray? Obtained
Answer: 0.01 to 10 nanometer to
Answer: 0.01 10 after nanometer Actual
Jaccard Score for this answer:1.0

Figure 17. Hindi Specific Sample 1 with Translation

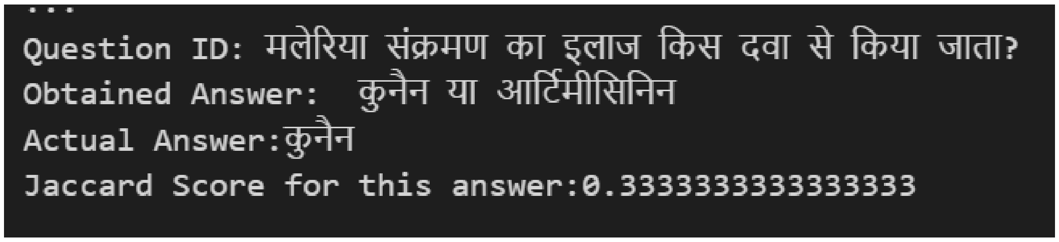


```
Question ID: सीटल शहर कहाँ स्थित है?
Obtained Answer:  अमेरिका के वाशिंगटन
Actual Answer:अमेरिका के वाशिंगटन राज्य
Jaccard Score for this answer:0.75
```

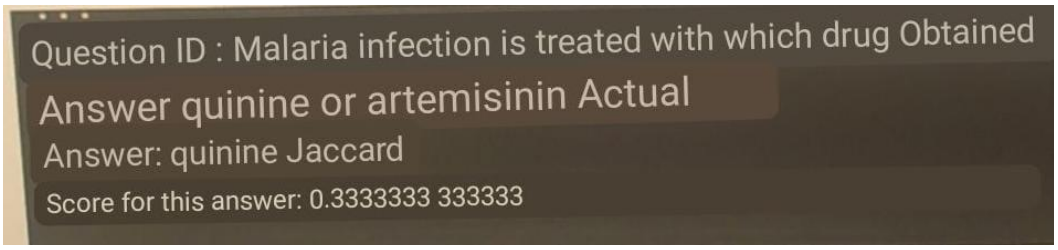


```
Question ID : Where is the city of Seattle
located? Obtained Answer: Washington State of
America Actual Answer: Washington State of America
Jaccard Score for this answer:0.75
```

Figure 17. Hindi Specific Sample 2 with Translation



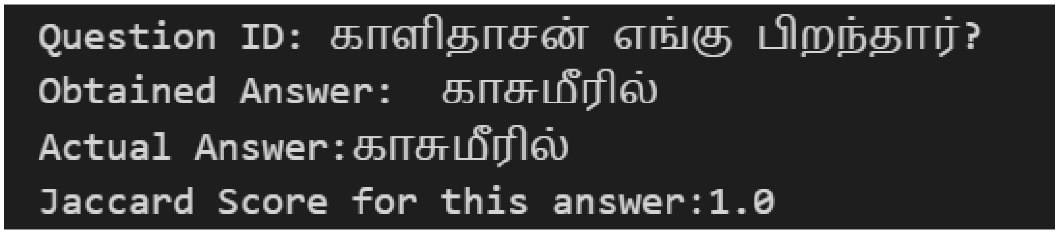
```
...
Question ID: मलेरिया संक्रमण का इलाज किस दवा से किया जाता?
Obtained Answer:  कुनैन या आर्टिमीसिनिन
Actual Answer:कुनैन
Jaccard Score for this answer:0.3333333333333333
```



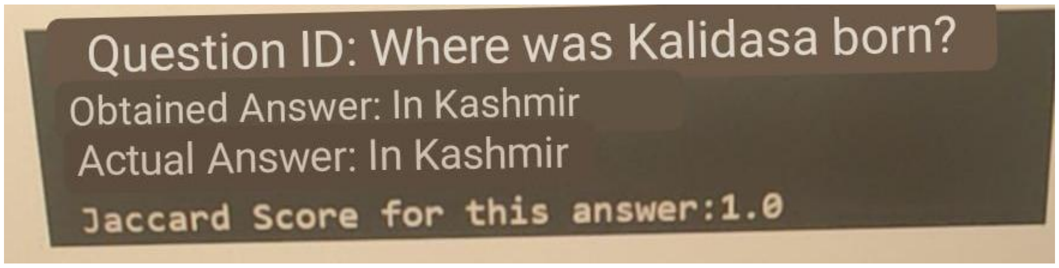
```
...
Question ID : Malaria infection is treated with which drug Obtained
Answer quinine or artemisinin Actual
Answer: quinine Jaccard
Score for this answer: 0.3333333 333333
```

Figure 18. Hindi Specific Sample 3 with Translation

Now we can see some of the attempts at answering questions by the Tamil Specific Transformer:

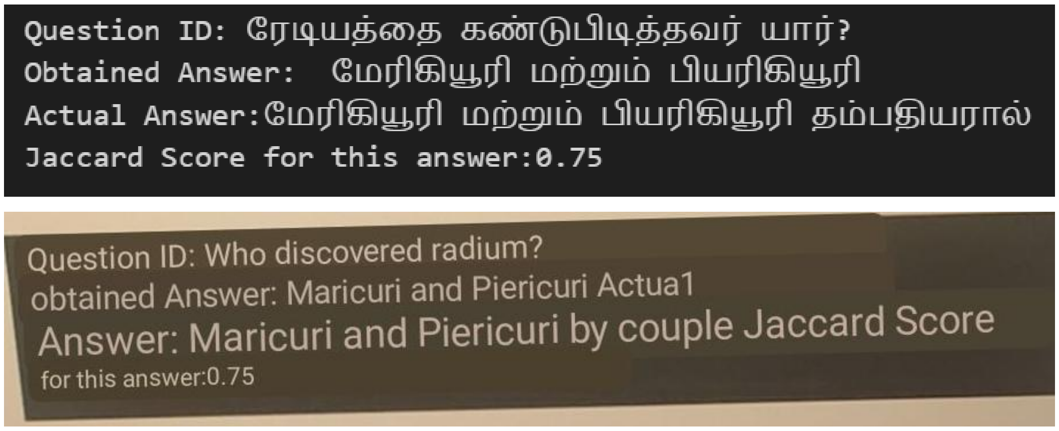


```
Question ID: காளிதாசன் எங்கு பிறந்தார்?
Obtained Answer:  காசுமீரில்
Actual Answer:காசுமீரில்
Jaccard Score for this answer:1.0
```



```
Question ID: Where was Kalidasa born?
Obtained Answer: In Kashmir
Actual Answer: In Kashmir
Jaccard Score for this answer:1.0
```

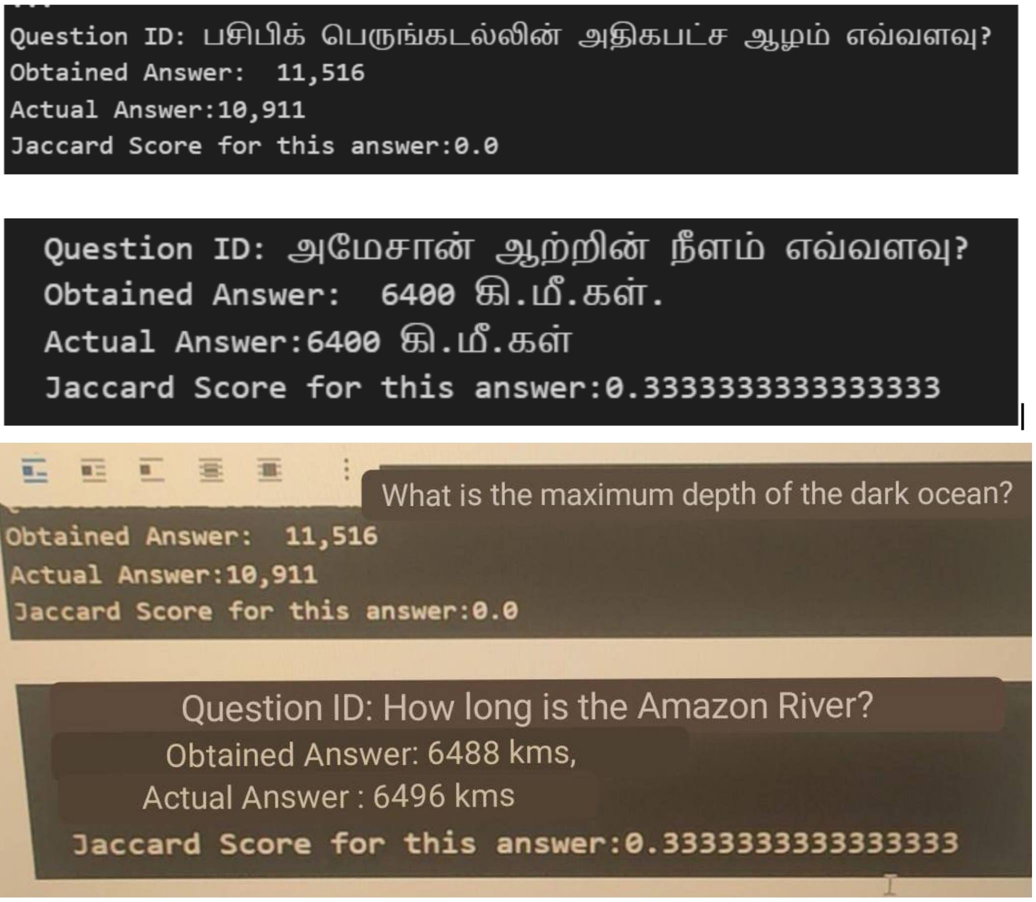
Figure 19. Tamil Specific Sample 1 with Translation



```
Question ID: ரேடியத்தை கண்டுபிடித்தவர் யார்?
Obtained Answer:  மேரிகியூரி மற்றும் பியரிகியூரி
Actual Answer:மேரிகியூரி மற்றும் பியரிகியூரி தம்பதியரால்
Jaccard Score for this answer:0.75
```

```
Question ID: Who discovered radium?
obtained Answer: Maricuri and Piericuri Actua1
Answer: Maricuri and Piericuri by couple Jaccard Score
for this answer:0.75
```

Figure 20. Tamil Specific Sample 2 with Translation



```
Question ID: பசிபிக் பெருங்கடல்லின் அதிகபட்ச ஆழம் எவ்வளவு?
Obtained Answer:  11,516
Actual Answer:10,911
Jaccard Score for this answer:0.0
```

```
Question ID: அமேசான் ஆற்றின் நீளம் எவ்வளவு?
Obtained Answer:  6400 கி.மீ.கள்.
Actual Answer:6400 கி.மீ.கள்
Jaccard Score for this answer:0.3333333333333333
```

```
What is the maximum depth of the dark ocean?
Obtained Answer:  11,516
Actual Answer:10,911
Jaccard Score for this answer:0.0
```

```
Question ID: How long is the Amazon River?
Obtained Answer: 6488 kms,
Actual Answer : 6496 kms
Jaccard Score for this answer:0.3333333333333333
```

Figure 21. Tamil Specific Sample 3 with Translation

4.3. Results and Insights

- We see that the Decision Tree approach is not a good approach for our problem. This is not a typical tree classification problem
- We see that in the models we tried, the Transformer approach gave the best results

- We see that the Transformer approach is very intensive, utilizing CPU, GPU, and a lot of time, considering the smaller scale dataset
- When we started to use very specialized transformers we received huge score increases
- The Hindi specific model worked very well with dealing with Hindi Questions and the Tamil one with Tamil Questions
- But in both the models, we found that they are not performing particularly well with numeric questions and answers. We feel this might be a deficiency of these language-specific models and numbers are not technically in Tamil or Hindi.

4.4. Future Steps

- Multilingual Approach: Expand your model to handle multiple languages beyond just Hindi and Tamil
- Interactive and Conversational QA: Extend your model to support interactive and conversational question answering.
- We can proceed to deploy and set up an API that provides Tamil and Hindi Question and Answering, with the added benefit being that the answers will also be in the language specified
- We can create a full-scale web application. Focus on creating a user-friendly and intuitive interface for your Question Answering AI.
- We can train the models used specifically for our dataset
- Performance Optimization: Continuously optimize the performance of your model by experimenting with hyperparameters, model architecture, and other techniques. You can also explore techniques such as model compression or quantization to reduce the model's size and inference time, making it more efficient for real-time applications.
- Improve the answering of numeric questions

5. Conclusion

We worked on preparing our dataset by removing unnecessary stopwords. We then proceeded to try different types of AI models to answer questions. As per our intuition, we found out that Neural Networks perform this task better. Then, we sought out and tested with three different Neural Network Transformers. We conclude that these specific models of each language work very well and provide good scores and results. The best performing model was the xlm-roberta-base-hindi model.

<u>Ethical Considerations:</u>

- Fairness: Ensure that your model does not exhibit bias towards any particular language, dialect, region, or cultural group. Bias in your AI system can lead to unfair treatment and perpetuate discrimination, so it's crucial to carefully evaluate and mitigate any biases in your model to ensure fairness and equitable access for users of both Hindi and Tamil languages.
- Privacy: Consider the privacy implications of collecting and storing user data, including the questions asked and answers provided. Ensure that you comply with applicable data privacy laws and regulations and take appropriate measures to protect user data, including data encryption, secure storage, and obtaining informed consent from users. Be transparent about how user data is used and provide options for users to control their data.
- Transparency and Explainability: Transformers are known to be complex and opaque models, which can make it challenging to explain how they arrive at their

answers. Consider incorporating methods for model interpretability and explainability, particularly for Hindi and Tamil languages, so that users can understand how your model arrives at its answers. This promotes transparency and accountability and helps users trust the AI system.

- Social Impact: Consider the potential social impact of your Question Answering AI for Hindi and Tamil languages. Evaluate how your system may affect language preservation, cultural representation, and accessibility for users in diverse regions and communities. Strive to build an AI system that positively contributes to the linguistic and cultural diversity of the Hindi and Tamil-speaking populations and minimizes any negative consequences.
- Accountability: Take responsibility for the performance and behavior of your Hindi and Tamil Question Answering AI. Regularly evaluate and monitor your model's performance, including language accuracy and cultural sensitivity, and address any biases, errors, or limitations. Be transparent about the capabilities and limitations of your AI system, particularly in the context of Hindi and Tamil languages, to set realistic expectations among users. Establish channels for user feedback and actively address concerns or issues raised by users.

6. Acknowledgments

The Capstone Project is done under the guidance of Ruba Al Omari, Faculty of Science, Engineering & Information Technology at Durham College, Oshawa, Ontario. It is also a part of an intriguing Kaggle Competition.

7. Bibliography

[1] Wikimedia Foundation. (2022, December 25). *Natural-language understanding*. Wikipedia. Retrieved February 5, 2023, from https://en.wikipedia.org/wiki/Natural-language_understanding

[2] Develop. (2020, November 27). *How can natural language processing help your business?* Humable. Retrieved February 5, 2023, from https://humable.io/natural-language-processing-and-artificial-intelligence-the-interaction-of-the-future/?lang=en

[3] *Chaii - Discussions*. Kaggle. (n.d.). Retrieved February 5, 2023, from https://www.kaggle.com/competitions/chaii-hindi-and-tamil-question-answering/discussion/264578

[4] *Chaii -Discussions 2*. Kaggle. (n.d.). Retrieved February 5, 2023, from https://www.kaggle.com/competitions/chaii-hindi-and-tamil-question-answering/discussion/288049

[5] *Chaii 2021 -Dataset*. Kaggle. (n.d.). Retrieved February 5, 2023, from https://www.kaggle.com/competitions/chaii-hindi-and-tamil-question-answering/data

[6] Kumar, G. K., Gehlot, A., Mullappilly, S. S., & Nandakumar, K. (2022). Mucot: Multilingual contrastive training for question-answering in low-resource languages. *Proceedings of the Second Workshop on Speech and Language Technologies for Dravidian Languages*. https://doi.org/10.18653/v1/2022.dravidianlangtech-1.3

[7] Thedrcat. (2021, August 13). *Chaii - Eda & Baseline*. Kaggle. Retrieved February 5, 2023, from https://www.kaggle.com/code/thedrcat/chaii-eda-baseline

[8] Rhtsingh. (2021, August 17). *Chaii qa - 5 fold XLMROBERTA torch: FIT*. Kaggle. Retrieved February 5, 2023, from https://www.kaggle.com/code/rhtsingh/chaii-qa-5-fold-xlmroberta-torch-fit

[9] Nbroad. (2021, September 20). *Chaii-qa: Character/token/languages Eda* . Kaggle. Retrieved February 5, 2023, from https://www.kaggle.com/code/nbroad/chaii-qa-character-token-languages-eda

[10] Bhatia, R. (2020, August 11). *Hindi stop words and sentiment lexicons*. Kaggle. Retrieved March 5, 2023, from https://www.kaggle.com/datasets/ruchi798/hindi-stopwords?select=stopwords.txt

[11] GitHub. (n.d.). Retrieved March 5, 2023, from https://raw.githubusercontent.com/AshokR/TamilNLP/master/tamilnlp/Resources/TamilStopWords.txt

[12] Tidoo, Y. M. (2019, January 7). *Cosine similarity in question-answering apps*. Sweetcode.io. Retrieved March 5, 2023, from https://sweetcode.io/cosine-similarity-question-answering-apps/

[13] *Sklearn.tree.decisiontreeclassifier*. scikit. (n.d.). Retrieved March 5, 2023, from https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

[14] Question answering. (n.d.). Retrieved March 5, 2023, from https://huggingface.co/docs/transformers/tasks/question_answering

[15] Gupta, S. (2021, August 20). *Overview of text similarity metrics in Python*. Medium. Retrieved March 5, 2023, from https://towardsdatascience.com/overview-of-text-similarity-metrics-3397c4601f50

[16] *EVALUATION - Chaii - Hindi and Tamil question answering*. Kaggle. (n.d.). Retrieved March 5, 2023, from https://www.kaggle.com/competitions/chaii-hindi-and-tamil-question-answering/overview/evaluation

[17] *ASWIN037/xlm-roberta-squad-tamil · hugging face*. AswiN037/xlm-roberta-squad-tamil · Hugging Face. (n.d.). Retrieved April 9, 2023, from https://huggingface.co/AswiN037/xlm-roberta-squad-tamil

[18] *Bhavikardeshna/xlm-roberta-base-hindi · hugging face*. bhavikardeshna/xlm-roberta-base-hindi · Hugging Face. (n.d.). Retrieved April 9, 2023, from https://huggingface.co/bhavikardeshna/xlm-roberta-base-hindi

8. Google Colab URL

https://colab.research.google.com/drive/12vNRE4ypPMIZNc2JuNHnrz7-L8ITKZYN?usp=sharing

9. Appendix - Code

```
## Introduction

    ### This is a python notebook which details the various steps and processes involved in exploring,
    analysing, training and testing the Chaii 2021 dataset. The goal is to study question and answering in
    the indian languages of Tamil and Hindi.
    ### A project report is attached along with this submission.
```

## Minimum Requirements

### You will definitely need to have the following installed for running this notebook:

a) Pandas
b) Sklearn
c) Numpy
d) Seaborn
e) Matplotlib
f) Pandas Profiling
g) nltk
h) transformers
i) tensorflow

## Load dataset into pandas

The dataset is called 'Chaii' and can be downloaded from https://www.kaggle.com/competitions/chaii-hindi-and-tamil-question-answering/data

Hosted in git at
https://gitlab.com/sanathdavis/capstone-2/-/raw/main/train.csv
https://gitlab.com/sanathdavis/capstone-2/-/raw/main/test.csv

```
import pandas as pd
dataset = pd.read_csv("https://gitlab.com/sanathdavis/capstone-2/-/raw/main/train.csv")
dataset
```

## Data Preperation

### Remove Answer Start column

```
dataset = dataset.drop(['answer_start'], axis=1)
```

### Removing Stop Words

Stop words are commonly used words in a language such as "the," "a," "an," "in," "is," "and," etc., that do not carry any significant meaning on their own and are generally removed from text during the data cleaning process in Natural Language Processing (NLP) applications.

Removing stop words can be useful for several reasons:

Efficiency: Stop words often appear frequently in text, which can slow down the processing of large amounts of text data. By removing stop words, the processing time can be reduced and the efficiency of NLP algorithms can be improved.

Accuracy: Stop words can also add noise to the text data, as they do not add much value to the meaning of the text. By removing them, the accuracy of NLP algorithms can be improved as the focus is shifted to the more important words in the text.

Consistency: Removing stop words can help in creating consistent representations of text data, which can help in tasks such as document classification, sentiment analysis, and topic modeling.

Hindi stopwords obtained from https://www.kaggle.com/datasets/ruchi798/hindi-stopwords?select=stopwords.txt (hosted in git)

Tamil stopwords obtained from https://github.com/AshokR/TamilNLP/blob/master/tamilnlp/Resources/TamilStopWords.txt

Read, list and combine the stopwords

```
stop_words_hindi = pd.read_table("https://gitlab.com/sanathdavis/capstone-2/-/raw/main/hindi_stopwords.txt", header=None)
```

```
stop_words_tamil =
pd.read_table("https://raw.githubusercontent.com/AshokR/TamilNLP/master/tamilnlp/
Resources/TamilStopWords.txt", header=None)

stop_words_hindi = stop_words_hindi[0].to_list()
stop_words_tamil = stop_words_tamil[0].to_list()

stop_words = stop_words_hindi + stop_words_tamil

stop_words
```

## **Explorative graphs**

Some sources of inspiration

https://www.kaggle.com/code/aakashnain/chaii-explore-the-data
https://www.kaggle.com/code/starkking07/just-exploring-eda-xlm-baseline
https://www.kaggle.com/code/ashutosh619sudo/chaii-eda-and-data-exploration

```
import matplotlib.pyplot as plt
import seaborn as sns
```

### a) Number of training sets in each language

```
print("Number of training samples: ", len(dataset))
print("\nTraining data language count:\n",dataset["language"].value_counts())
plt.figure(figsize=(8, 5))
sns.countplot(data=dataset, x="language")
plt.show()
# Analyze Hindi and Tamil text differently
train_hindi = dataset[dataset["language"] == "hindi" ]
train_tamil = dataset[dataset["language"] == "tamil" ]
```

### b) Shortest questions(based on number of words)

```
print("Hindi: \n\n")
display(sorted(train_hindi["question"].tolist() , key = lambda x : len(x.split(sep = " "))
)[:10])
print("\n\nTamil: \n\n")
display(sorted(train_tamil["question"].tolist() , key = lambda x : len(x.split(sep = " "))
)[:10])
```

### c) Longest questions(based on number of words)

```
print("Hindi: \n\n")
display(sorted(train_hindi["question"].tolist() , key = lambda x : len(x.split(sep = " "))
)[-10:])
print("\n\nTamil: \n\n")
display(sorted(train_tamil["question"].tolist() , key = lambda x : len(x.split(sep = " "))
)[-10:])
```

### d) Distribution of questions length

```
plt.figure(figsize = ( 15, 8))
sns.displot( data = pd.DataFrame({ "Hindi" : train_hindi["question"].map(lambda x :
len(x.split(sep = " "))) ,
             "Tamil" : train_tamil["question"].map(lambda x : len(x.split(sep = " ")) ) })
)
### e) Distribution of answers based on Numeric or not
numeric_df = train_hindi["answer_text"].str.isnumeric()
len_h_numeric_non = len(numeric_df[numeric_df == False])
len_h_numeric = len(numeric_df[numeric_df == True])
numeric1_df = train_tamil["answer_text"].str.isnumeric()
len_t_numeric = len(numeric1_df[numeric1_df == True])
len_t_numeric_non = len(numeric1_df[numeric1_df == False])
len_h_numeric_non = len(numeric_df[numeric_df == False])
```

```
len_t_numeric = len(numeric1_df[numeric1_df == True])
len_t_numeric_non = len(numeric1_df[numeric1_df == False])
import matplotlib.pyplot as plt
import numpy as np
langs = ['Hindi Numeric', 'Hindi Non-Numeric', 'Tamil Numeric', 'Tamil Non-Numeric']
students = [len_h_numeric,len_h_numeric_non,len_t_numeric,len_t_numeric_non]
x_pos = np.arange(len(langs))


plt.bar(x_pos, students, color=['black', 'red', 'green', 'blue'])
plt.xticks(x_pos, langs)
plt.show()
```

## **Pandas Profiling**

pandas-profiling generates profile reports from a pandas DataFrame which is handy for exploratory data analysis.

```
import numpy as np
import pandas as pd
from pandas_profiling import ProfileReport
#creating pandas profile
profile = ProfileReport(dataset, title="Pandas Profiling Report")
#downloading an HTML version of Pandas Profiling Report
#profile.to_file("Pandas_Profiling_Report_of_the_chaii_dataset.html")
#show the interactive pandas profile inside the notebook
profile.to_notebook_iframe()
```

## **Model Training and Evaluation**

#### **Evaluation Matrics - Jaccard score**

The metric used in this project is the word-level Jaccard score.

The word-level Jaccard score is a measure of similarity between two sets of words that compares the intersection and union of the words in the two sets. It is calculated as the size of the intersection of the two sets divided by the size of their union.

In Natural Language Processing (NLP), the word-level Jaccard score is often used in tasks such as text classification, information retrieval, and text similarity analysis. It can help in identifying the degree of similarity between two texts based on the overlap of words in the texts. For example, it can be used to compare the similarity of two documents, to identify the similarity of search queries, or to evaluate the performance of text classification models.

One of the advantages of using the word-level Jaccard score is that it is simple to calculate and can be computed efficiently for large amounts of text data.

Sourced at https://towardsdatascience.com/overview-of-text-similarity-metrics-3397c4601f50
https://www.kaggle.com/competitions/chaii-hindi-and-tamil-question-answering/overview/evaluation

```
#calculate jaccard score between two strings

        def jaccard(str1, str2):
            a = set(str1.lower().split())
            b = set(str2.lower().split())
            c = a.intersection(b)
            return float(len(c)) / (len(a) + len(b) - len(c))
#calculate overrall Jaccard score

        def overall_jaccard_score(jaccard_score_list):
            n = len(jaccard_score_list)
            total_jaccard_score_list = sum(jaccard_score_list)
            return round(total_jaccard_score_list/n, 4)
```

### a) Cosine Similarity
(simple approach)

Sources:
<br>
<br>
Tokenizer : https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html
<br>
Blogs: https://sweetcode.io/cosine-similarity-question-answering-apps/ , https://github.com/screddy1313/Question-Answering-system-using-Cosine-similarity

Steps:
<p>a) Remove stop words from the context and the question<p>
<p>b) Find the sentence in the context that is most similar to the question<p>
<p>c) Find the answer in the most similar sentence<p>

```
#imports

    import nltk
    nltk.download('punkt')
    from tqdm.auto import tqdm
    from nltk.tokenize import word_tokenize, sent_tokenize
    from nltk.corpus import stopwords
    from sklearn.feature_extraction.text import TfidfVectorizer
    from sklearn.metrics.pairwise import cosine_similarity
    #loop each data, tokenise, and calculate the individual jaccard scores

    predicted_answers = []
    jaccard_scores = []
    for index,row in tqdm(dataset.iterrows()) :
          context = row['context']
question = row['question']
context_tokens = word_tokenize(context)
question_tokens = word_tokenize(question)

# Remove stop words from the context and the question
context_tokens = [token for token in context_tokens if not token.lower() in stop_words]
question_tokens = [token for token in question_tokens if not token.lower() in stop_words]

# Find the sentence in the context that is most similar to the question
sentences = sent_tokenize(context)
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(sentences)
question_vec = vectorizer.transform([" ".join(question_tokens)])
similarity_scores = cosine_similarity(X, question_vec)
most_similar_sentence_index = similarity_scores.argmax()
most_similar_sentence = sentences[most_similar_sentence_index]

# Find the answer in the most similar sentence
answer_tokens = word_tokenize(most_similar_sentence)
answer = " ".join([token for token in answer_tokens if not token.lower() in stop_words])
jaccard_score = jaccard(row['answer_text'], answer)

# Print the answer
# print(answer)

#append score and answers
predicted_answers.append(answer)
jaccard_scores.append(jaccard_score)

final_jaccard_score = overall_jaccard_score(jaccard_scores)
print('Overall Jaccard score of Cosine Similarity is ' + str(final_jaccard_score))
```

### b) Decision Tree Classifier

Steps
<br>
<br>
Tree Based:
For each question, the code fits the Decision Tree algorithm on the features and labels and predicts the sentence in the context that contains the answer.
<br>
Finally, it extracts the answer from the predicted sentence and prints it

https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

```
#loop each data, tokenise, and calculate the individual jaccard scores

from sklearn.tree import DecisionTreeClassifier

predicted_answers = []
jaccard_scores = []
for index,row in tqdm(dataset.iterrows()) :
    #print(row["question"])
    context = row['context']
    question = row['question']
    context_tokens = word_tokenize(context)
    question_tokens = word_tokenize(question)

    # Remove stop words from the context and the question
    context_tokens = [token for token in context_tokens if not token.lower() in
stop_words]
    question_tokens = [token for token in question_tokens if not token.lower() in
stop_words]

    # Find the sentence in the context that is most similar to the question
    sentences = sent_tokenize(context)
    vectorizer = TfidfVectorizer()
    X = vectorizer.fit_transform(sentences).toarray()
    y = [0] * len(sentences)

    question_vec = vectorizer.transform([" ".join(question)]).toarray()
    clf = DecisionTreeClassifier()
    clf.fit(X, y)
    answer_sentence_index = clf.predict(question_vec)[0]
    answer_sentence = sentences[answer_sentence_index]
    answer_tokens = word_tokenize(answer_sentence)
    answer = " ".join([token for token in answer_tokens if not token.lower() in
stop_words])

    jaccard_score = jaccard(row['answer_text'], answer)

    # Print the answer
    # print(answer)
    predicted_answers.append(answer)
    jaccard_scores.append(jaccard_score)

final_jaccard_score = overall_jaccard_score(jaccard_scores)
print('Overall Jaccard score of Decision Tree Classifier is ' + str(final_jaccard_score))
```

### b) Neural Network - Transformers

Steps

<br>
<br>
Using Transformer:
<br>
This code uses the pipeline function from the Hugging Face Transformers library to load a pre-trained question answering model. It then defines the context and the question and uses the model to generate an answer.
<br>Finally, it prints the answer.

Sources:
<br>
https://huggingface.co/docs/transformers/tasks/question_answering

https://youtu.be/ajPx5LwJD-I

```
from transformers import pipeline

# Load the pre-trained question answering model
nlp = pipeline("question-answering")

predicted_answers = []
jaccard_scores = []
for index,row in tqdm(dataset.iterrows()) :
    #print(row["question"])
    context = row['context']
    question = row['question']
    # Use the model to generate an answer
    result = nlp(question=question, context=context)

    answer = result["answer"]
    jaccard_score = jaccard(row['answer_text'], answer)

    # Print the answer
    # print(answer)
    predicted_answers.append(answer)
    jaccard_scores.append(jaccard_score)

final_jaccard_score = overall_jaccard_score(jaccard_scores)
print('Overall Jaccard score of Transformer is ' + str(final_jaccard_score))
```

#### Transformer 3 - Hindi Specific Question Answering Model

```
from transformers import pipeline

# Load the pre-trained question answering model
nlp = pipeline("question-answering", model="AswiN037/xlm-roberta-squad-tamil")

predicted_answers = []
jaccard_scores = []
for index,row in tqdm(dataset[dataset['language'] == 'tamil'].iterrows()) :
    #print(row["question"])
    context = row['context']
    question = row['question']
    # Use the model to generate an answer
    result = nlp(question=question, context=context)

    answer = result["answer"]
    jaccard_score = jaccard(row['answer_text'], answer)

    # Print the answer
    # if index == 0:
    print('Question ID: '  + row['id'])
    print('Question ID: '  + question)
    print('Obtained Answer: ' + answer)
```

```
        print('Actual Answer:' + row['answer_text'])
        print('Jaccard Score for this answer:' + str(jaccard_score))

        predicted_answers.append(answer)
        jaccard_scores.append(jaccard_score)

final_jaccard_score = overall_jaccard_score(jaccard_scores)
print('Overall Jaccard score of Transformer is ' + str(final_jaccard_score))
```

#### **Transformer 3 - Hindi Specific Question Answering Model**

```
from transformers import pipeline

# Load the pre-trained question answering model
nlp = pipeline("question-answering", model="AswiN037/xlm-roberta-squad-tamil")

predicted_answers = []
jaccard_scores = []
for index,row in tqdm(dataset[dataset['language'] == 'tamil'].iterrows()) :
        #print(row["question"])
        context = row['context']
        question = row['question']
        # Use the model to generate an answer
        result = nlp(question=question, context=context)

        answer = result["answer"]
        jaccard_score = jaccard(row['answer_text'], answer)

        # Print the answer
        # if index == 0:
        print('Question ID: '  + row['id'])
        print('Question ID: '  + question)
        print('Obtained Answer: ' + answer)
        print('Actual Answer:' + row['answer_text'])
        print('Jaccard Score for this answer:' + str(jaccard_score))

        predicted_answers.append(answer)
        jaccard_scores.append(jaccard_score)

final_jaccard_score = overall_jaccard_score(jaccard_scores)
print('Overall Jaccard score of Transformer is ' + str(final_jaccard_score))
```