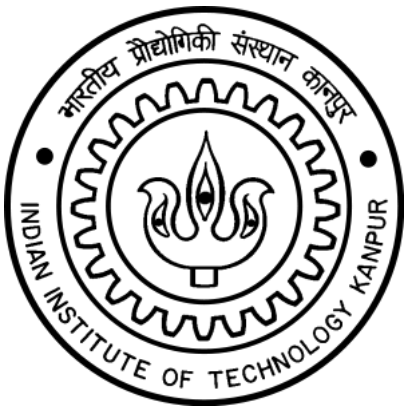
Information Retrieval System On Semantic Meaning Of Bhagavad Gita



Indian Institute of Technology Kanpur

Department Of Computer Science and Engineering

Project Report - Group 3
CS657A: Information Retrieval System
Under the guidance of
Prof. Arnab Bhattacharya
Academic Year 2022-2023

ABHISHEK SAHU (21111002)
abhisheks21@iitk.ac.in

ALOK KUMAR TRIVEDI (21111008)
alokt21@iitk.ac.in

ASHITOSH MORE (21111017)
ashitoshvm21@iitk.ac.in

SHIVAM KHARWAR (21111058)
skharwar21@iitk.ac.in

TANIKELLA SAI KIRAN (21111061)
tskiran21@iitk.ac.in

Acknowledgement

We would like to extend our sincere gratitude to our Project Guide, Prof. Arnab Bhattacharya for his advice, guidance, patience and timely help during the project. The freedom he gave us, to work with anything at any time, encouraged us to try out new things and helped to work more efficiently. It is a great honor for us to have been a part of his course. The successful completion of the work has been only possible due to his excellent guidance, meticulous observation and critical analysis.

ABHISHEK SAHU (21111002)
ALOK KUMAR TRIVEDI (21111008)
ASHITOSH MORE (21111017)
SHIVAM KHARWAR (21111058)
TANIKELLA SAI KIRAN (21111061)

Contents

Abstract

In one of the world's oldest religions, Hinduism and other similar faiths across South Asian and South-East Asian countries, Bhagavad Gita is considered the quintessence of the philosophy of life. This invaluable treasure of knowledge is limited to those who can understand its complex language and deep essence. Many learned people and scholars explained the intrinsic meaning of the Bhagavad Gita in their own words for the understanding of ordinary people. In this project, we intended to provide instant solutions from the Bhagavad Gita for the problems stated by the user by retrieving the relevant commentaries. We designed a Multi-lingual Information Retrieval System for the same. Various contextual models like Bert, Word2vec, Glove, LSA, Word2Vec and non-embedding models like TF-IDF and BM25 language models extracted the most relevant and appropriate Gita Slokas corresponding commentary for a given query in four different languages. In the later part, Cross-Lingual Information Retrieval for Hindi and English languages are implemented.The results reflected that the BM25 and LSI models work well for all four languages. They are lightweight and fast. In comparison, BERT models performed well for semantic and cross-lingual searching tasks for Hindi and English with Mean Pooling and Cosine Similarity metrics. For all other models, Clustering with Max pooling performs well. Also, Mean Pooling with cosine similarity shows high values in the evaluation.

Chapter 1

INTRODUCTION

1.1 Motivation

Bhagavat Gita (The Song by God) is considered one of the most important and widely discussed religious scripture of Hinduism. It is 700 verse scripture which contains dialogues between Prince and disciple Arjuna and his mentor and guide Krishna, known as a text that summarizes and comprises the core of Hindu philosophy. In the Mahabharata war, Arjuna, the most skilled and prominent archer and Krishna as his charioteer and other Pandavas were at war with their cousins Kauravas. Arjuna faced a moral dilemma where he did not want to go to war with his relatives and wanted to renounce everything to become a Yogi. As his mentor and guide, Lord Krishna reminded Arjuna of his Kshatriya duty and taught him the concepts of a karma yogi.

Although the Bhagavad Gita might be thousands of years old, the philosophy, logic, and knowledge are timeless. Understanding the appropriate meaning and interpretation of slokas in one's native language can helps in gaining a deeper understanding of the hows and whys of everyday life.

During the tough phases of life, most of us wish to refer to Bhagavad Gita for the solutions. However, understanding the divine concepts and language is not very easy and requires broad knowledge. So this motivated us to work on the semantic meaning of the Bhagavad Gita which will then provide the solution for any problem stated by the user using teachings of the Lord Krishna.

1.2 Broad Aim Of The Project

This project aims to develop a Multi-lingual Information Retrieval System on Bhagavad Gita. We have incorporated various contextual like **Bert, FastText, Glove, LSA, Word2Vec** and Basic models like **TF-IDF** and **BM25** language models to extract the most relevant and appropriate Gita Slokas and corresponding commentary for a given query in four different languages.
Also, we tried to add cross-lingual Information Retrieval in the Hindi and English languages.

Chapter 2

METHODOLOGY

2.1 Data Collection And Preprocessing

The data is scraped from various popular Gita web pages by web scrapping, and then the data is formatted into a shape and structure that will be easy to use. The websites used for web scrapping for the Gita Database are given below:
English : English data.
Hindi : Hindi data
Odia : Odia data
Telugu : Telugu data
Then the data is stored in a python dictionary which is converted into CSV files. The Key of the dictionary is the **shloka** and value is its corresponding **commentary**
After Data Collection and storing, we got four CSV files as output for each language: Hindi, English, Telugu, and Odia. These CSV files still have some inconsistencies. We worked on some manual inconsistencies in the data we got from web scrapping, like missing commentaries for some shloka for which we had to remove them from the database, and there are some commentaries which considered 2-3 shlokas simultaneously while explaining them, so we kept those commentaries in our dataset.

2.2 Basic Model

Following model based on Basic have been implemented for our project on the scrapped Gita dataset:

1. Boolean Retrieval System
2. Tf-Idf
3. BM-25

Before implementing all the models, we need to do some standard preprocessing for all three models.

2.2.1 Preprocessing

Preprocessing is performed on all the mentioned languages. The details of preprocessing for each language is mentioned below.

1. **English:**
   English Commentary corresponding to each shloka is passed through a data cleaning module which removed the unwanted chatacters like punctuations, symbols, numbers etc and also the non-English characters present in the text. The cleaned data is then passed to the tokenizer to generate the tokens list. These tokens are used for stopwords removal and were converted to lower case. Finally, using Porter Stemming algorithm, all the tokens were stemmed.

2. **Hindi:**
   Hindi Commentary is processed just as the English commentary file. It went through the above mentioned steps like data cleaning where non-Hindi words, exceptional punctuations, unnecessary characters were removed. Then with the help of the tokenizer, the text was converted into the tokens and stopwords were removed from it.

3. **Odia:**
   Odia Commentary file is cleaned with the help of the cleaning module and then tokenized, and stopwords are removed.

4. **Telugu:**
   Telugu Commentary file is same way processed like Odia first cleaned then tokenized and then stopwords are removed
   After preprocessing is done, the next step is to generate the posting list or inverted indexing.

2.2.2 Posting List Generation

After getting a list of useful tokens from every language dataset, the whole list of tokens is traversed again to generate the posting lists, term frequency, and unique word list

1. Posting List: Dictionary of inverted index list of words in tokens as key and its list of commentaries in which it appears as value.

2. Term frequency: It is a nested Dictionary of words as a key of the first dictionary and value as a dictionary which has commentary index as key and no. of occurrence in that commentary as value

3. Unique Word: List of unique words in all the commentaries.

After computing these lists, they are stored as pickle files so as to use them in the future to avoid repeated computations. Now we can implement each model as we have all the necessary preprocessed files required for that particular model. The implementation of each and every model is explained below.

2.2.3 Implementation

1. **Boolean Retrieval System:**
   First, the query input module takes the input query from the user. It cleans the user query by tokenizing and removing the stop words and converts the query into Boolean Expression, then creates the Boolean Incidence Matrix of terms in the query against Commentaries and puts 0 if the commentary does not contain a term and one otherwise and then evaluates the expression based on 'and', 'or' and 'not' to check the commentaries that are relevant or not.

2. **TF-IDF System:**
   Similar to the Boolean Retrieval System. Initially, the query is taken from the user preprocessed by cleaning the query by tokenizing and removing the stop words, and it calculates the

score based on the product of term frequency and inverse document frequency and retrieves the relevant scores based on this product as this product is used as metric of relevancy.

$$\text{tf}(t,d) = \frac{f_d(t)}{max f_d(w)}$$

$$\text{idf}(t,D) = ln(\frac{|D|}{|d \epsilon D : t \epsilon d|})$$

$$\text{tfidf}(t,d,D) = \text{tf}(t,d) * \text{idf}(t,D)$$

3. **BM25:**
   Again, the query is taken from the user and preprocessed to clean the query. And then BM25 score is calculated between query and commentaries as a measure of relevancy to retrieve relevant models.

$$\sum_{i}^{n} \text{IDF}(q_i) \frac{f(q_i,D) * (k1+1)}{f_{t,D} + k_1.(1 - b + b.\frac{|D|}{avgd})}$$

$$\text{IDF}(q_i) = log \frac{N - \eta_t + 0.5}{\eta_t + 0.5}$$

2.3 Latent Semantic Analysis

To implement the LSA model, we have used the gensim library. We have trained the LSI model with 30 topics. Below are the details

2.3.1 Preprocessing

The following steps are used for the preprocessing of the LSI model.

1. Cleaning Of Commentaries
2. Finding Unique Words in the Corpus and assigning index
3. Constructing Document Bag Of Word
4. Training LSI using SVD
5. Saving Model to avoid future computations

2.3.2 Implementation

1. We have to find query embedding using the LSI index given any query Q.
2. We need to encode the Query to vector using preprocessing files generated in the preprocessing step.
3. Then we have to compare the query vector with the document vectors using similarity metrics like cosine similarity.
4. Ranking based on the score in non-increasing order.

2.4 Word2Vec

We have implemented the Word2Vec (Cbow and SkipGram) model for our Bhagavad Gita dataset. Initially, we trained the model on the dataset and pickled the Word2Vec model for further use. To train the Word2Vec model, we have used Google Colab with GPU enabled using Keras library.

2.4.1 Preprocessing

Before the preprocessing part, let us examine the different strategies used for encoding any given passage. This will help us build the background for understanding the working of those models discussed in later stages.

Passage Embedding is the process of representing the whole passage into a vector that closely resembles its semantic meaning. So that we can use it for our computations
There are the following strategies that we have used to generate the passage embedding:

1. Max Pooling
2. Mean Pooling
3. Line by Line
4. Whole

Max Pooling

First, we need to calculate the sentence vector for a sentence. For the calculation of the sentence vector, we tokenize the sentence into words and then for every word, we find the embedding. After getting the embedding, we find the mean of all the embeddings of the word present in the sentence. In this way, we compute the sentence vector.

The **Max Pooling** takes the maximum sentence vector out of all the sentences present in a passage.
Steps involved in Max Pooling:

1. Tokenize Passage into Sentences
2. Tokenize Sentence into Words
3. Find Word Embeddings
4. Mean all word embeddings in a sentence
5. Taking the maximum sentence vector out of all sentences in a passage

Mean Pooling

The **Mean Pooling** takes the mean sentence vector out of all the sentences present in a passage.
Steps involve in Mean Pooling:

1. Tokenize Passage into Sentences
2. Tokenize Sentence into Words
3. Find Word Embeddings

4. Mean all word embeddings in a sentence
5. Taking the mean sentence vector of all sentences in a passage

Line By Line

The **Line-By-Line** strategy keeps each sentence vector present in a passage.
Steps involved in this:

1. Tokenize Passage into Sentences
2. Tokenize Sentence into Words
3. Find Word Embeddings
4. Mean all word embeddings in a sentence
5. Take all the sentence vector present in a passage

Whole

The **Whole** strategy takes the passage as independent sentence and computes sentence embedding for it.
Steps involved in this:

1. Tokenize Passage into Words
2. Find Word Embeddings
3. Mean all word embeddings in a passage

Now let us jump into **preprocessing** part. The different steps involved in the preprocessing step are mentioned below:

1. Cleaning all Passages
2. Load Pre-trained Model
3. Calculating Passage embedding by Max Pooling
4. Calculating Passage embedding by Mean Pooling
5. Calculating Passage embedding by Line by Line
6. Calculating Passage embedding by Whole
7. Pickle all files so that they can be used in the future

2.4.2 Implementation

First, We have loaded the pre-trained values of both CBow and Skipgram models that we have trained on the Bhagavad Gita dataset. Then given any query Q, we need to calculate the query embedding. The query is a sentence, so we can use the sentence embedding technique discussed in the preprocessing part. After that, we need to follow these steps

1. **Max Pooling:** We will load the Max Pooling Matrix that we have generated earlier. Then we need to take each passage embedding and compare it to query embedding using metrics like Cosine Similarity and Euclidean Distance. And then output of those passages which have better metrics.

2. **Mean Pooling:** We will load the Mean Pooling Matrix that we have generated earlier. Then we need to take each passage embedding and compare it to query embedding using metrics like Cosine Similarity and Euclidean Distance. And then output of those passages which have better metrics.

3. **Line By Line:** We will load the Line by Line 3D Matrix that we have generated earlier. Then we need to take each sentence embedding and compare it to query embedding using metrics like Cosine Similarity and Euclidean Distance and then take the better metrics out of all sentence scores for a passage. And then output of those passages which have better metrics.

4. **Whole:** We will load the Whole Pooling Matrix that we have generated earlier. Then we need to take each passage embedding and compare it to query embedding using metrics like Cosine Similarity and Euclidean Distance. And then output of those passages which have better metrics.

5. **Clustering**: It is not always true that we will get all our desired output. Sometimes, it is better to get a cluster of outputs. For this reason, we have used Clustering. Furthermore, the Clustering used is KMeans Cluster with 60 clusters. We have used different pooling techniques in Clustering that is mentioned below:

   (a) **Clustering By Max Pooling** We are going to use the Max Pooling Matrix that we have generated during preprocessing for our KMeans Algorithm

   (b) **Clustering By Mean Pooling** We are going to use the Mean Pooling Matrix that we have generated during preprocessing for our KMeans Algorithm

   (c) **Clustering By WholePooling** We are going to use the Whole Pooling Matrix that we have generated during preprocessing for our KMeans Algorithm

2.5 FastText

2.5.1 Preprocessing

The different steps involved in the preprocessing are mentioned below:

1. Cleaning all Passages

2. Load Pre-trained Model

3. Calculating Passage embedding by Max Pooling

4. Calculating Passage embedding by Mean Pooling
5. Calculating Passage embedding by Line by Line
6. Calculating Passage embedding by Whole
7. Pickle all files so that they can be used in the future

2.5.2 Implementation

We have loaded the standard pre-trained values of FastText models. Then given any query Q, we need to calculate the query embedding. The query is a sentence, so we can use the sentence embedding. After that, we need to follow these steps

1. **Max Pooling:** We will load the Max Pooling Matrix that we have generated earlier. Then we need to take each passage embedding and compare it to query embedding using metrics like Cosine Similarity and Euclidean Distance. And then output of those passages which have better metrics.
2. **Mean Pooling:** We will load the Mean Pooling Matrix that we have generated earlier. Then we need to take each passage embedding and compare it to query embedding using metrics like Cosine Similarity and Euclidean Distance. And then output of those passages which have better metrics.
3. **Line By Line:** We will load the Line by Line 3D Matrix that we have generated earlier. Then we need to take each sentence embedding and compare it to query embedding using metrics like Cosine Similarity and Euclidean Distance and then take the better metrics out of all sentence scores for a passage. And then output of those passages which have better metrics.
4. **Whole:** We will load the Whole Pooling Matrix that we have generated earlier. Then we need to take each passage embedding and compare it to query embedding using metrics like Cosine Similarity and Euclidean Distance. And then output of those passages which have better metrics.
5. **Clustering**: It is not always true that we will get all our desired output. Sometimes, it is better to get a cluster of outputs. For this reason, we have used Clustering. Furthermore, the Clustering used is KMeans Cluster with 60 clusters. We have used different pooling techniques in Clustering that is mentioned below:
   (a) **Clustering By Max Pooling** We are going to use the Max Pooling Matrix that we have generated during preprocessing for our KMeans Algorithm
   (b) **Clustering By Mean Pooling** We are going to use the Mean Pooling Matrix that we have generated during preprocessing for our KMeans Algorithm
   (c) **Clustering By WholePooling** We are going to use the Whole Pooling Matrix that we have generated during preprocessing for our KMeans Algorithm

2.6 Glove

2.6.1 Preprocessing

The different steps involved in the preprocessing step are mentioned below:

1. Cleaning all Passages
2. Load Pre-trained Model
3. Calculating Passage embedding by Max Pooling
4. Calculating Passage embedding by Mean Pooling
5. Calculating Passage embedding by Line by Line
6. Calculating Passage embedding by Whole
7. Pickle all files so that they can be used in the future

2.6.2 Implementation

Again, We have loaded the standard pre-trained values of Glove models. Then given any query Q, we need to calculate the query embedding. The query is a sentence, so we can use the sentence embedding technique discussed in the preprocessing part. After that, we need to follow these steps

1. **Max Pooling:** We will load the Max Pooling Matrix that we have generated earlier. Then we need to take each passage embedding and compare it to query embedding using metrics like Cosine Similarity and Euclidean Distance. And then output of those passages which have better metrics.
2. **Mean Pooling:** We will load the Mean Pooling Matrix that we have generated earlier. Then we need to take each passage embedding and compare it to query embedding using metrics like Cosine Similarity and Euclidean Distance. And then output of those passages which have better metrics.
3. **Line By Line:** We will load the Line by Line 3D Matrix that we have generated earlier. Then we need to take each sentence embedding and compare it to query embedding using metrics like Cosine Similarity and Euclidean Distance and then take the better metrics out of all sentence scores for a passage. And then output of those passages which have better metrics.
4. **Whole:** We will load the Whole Pooling Matrix that we have generated earlier. Then we need to take each passage embedding and compare it to query embedding using metrics like Cosine Similarity and Euclidean Distance. And then output of those passages which have better metrics.
5. **Clustering**: It is not always true that we will get all our desired output. Sometimes, it is better to get a cluster of outputs. For this reason, we have used Clustering. Furthermore, the Clustering used is KMeans Cluster with 60 clusters. We have used different pooling techniques in Clustering that is mentioned below:
   (a) **Clustering By Max Pooling** We are going to use the Max Pooling Matrix that we have generated during preprocessing for our KMeans Algorithm
   (b) **Clustering By Mean Pooling** We are going to use the Mean Pooling Matrix that we have generated during preprocessing for our KMeans Algorithm
   (c) **Clustering By WholePooling** We are going to use the Whole Pooling Matrix that we have generated during preprocessing for our KMeans Algorithm

2.7 Sentence BERT

Here we have used the Sentence Bert transformer, trained in 100 different languages, including Hindi and English. Unfortunately, we couldn't use SentBert for Odia and Telugu due to resource constraints. We have downloaded the pre-trained Sentence Bert model for our task.

2.7.1 Preprocessing

The different steps involved in the preprocessing step for BERT model are mentioned below:

1. Cleaning all Passages
2. Load Pre-trained Model
3. Calculating Passage embedding by Max Pooling
4. Calculating Passage embedding by Mean Pooling
5. Calculating Passage embedding by Line by Line
6. Calculating Passage embedding by Whole
7. Pickle all files so that they can be used in the future

2.7.2 Implementation

Given any query Q either in Hindi or English Language, we need to calculate the query embedding. The query is a sentence, so we can use the sentence BERT to encode it. After that, we need to follow these steps

1. **Max Pooling:** We will load the Max Pooling Matrix that we have generated earlier. Then we need to take each passage embedding and compare it to query embedding using metrics like Cosine Similarity and Euclidean Distance. And then output of those passages which have better metrics.
2. **Mean Pooling:** We will load the Mean Pooling Matrix that we have generated earlier. Then we need to take each passage embedding and compare it to query embedding using metrics like Cosine Similarity and Euclidean Distance. And then output of those passages which have better metrics.
3. **Line By Line:** We will load the Line by Line 3D Matrix that we have generated earlier. Then we need to take each sentence embedding and compare it to query embedding using metrics like Cosine Similarity and Euclidean Distance and then take the better metrics out of all sentence scores for a passage. And then output of those passages which have better metrics.
4. **Whole:** We will load the Whole Pooling Matrix that we have generated earlier. Then we need to take each passage embedding and compare it to query embedding using metrics like Cosine Similarity and Euclidean Distance. And then output of those passages which have better metrics.
5. **Clustering**: It is not always true that we will get all our desired output. Sometimes, it is better to get a cluster of outputs. For this reason, we have used Clustering. Furthermore, the Clustering used is KMeans Cluster with 60 clusters. We have used different pooling techniques in Clustering that is mentioned below:

(a) **Clustering By Max Pooling** We are going to use the Max Pooling Matrix that we have generated during preprocessing for our KMeans Algorithm

(b) **Clustering By Mean Pooling** We are going to use the Mean Pooling Matrix that we have generated during preprocessing for our KMeans Algorithm

(c) **Clustering By WholePooling** We are going to use the Whole Pooling Matrix that we have generated during preprocessing for our KMeans Algorithm

Chapter 3

OUTPUTS

ఆత్మహత్యా ఆలోచనలు

ధైర్యం వస్తుంది.ఆధ్యాత్మిక పురోగతి పథంలో ప్రతికూలతలు ఎదురైనప్పుడు, ఇతరులు మనకు తీవ్ర అన్యాయం చేసారు అని, వారే మనకు శత్రువులని ఫిర్యాదు చేస్తాము. కానీ, మన మనస్సే మన ప్రధాన శత్రువు. పరిపూర్ణ సిద్ధి కోసం మనం చేసే ప్రయత్నాలకు అవరోధం కల్పించే వినాశకారి అదే. ఒక పక్క, జీవాత్మ యొక్క గొప్ప శ్రేయిభిలాషి లాగా, మనకు అత్యంత శ్రేయస్సుని కలుగ చేసే శక్తి, మనస్సుకి, ఉంది; మరో పక్క, మన ప్రగాఢ శత్రువుగా, మన కు తీవ్ర హాని చేసే శక్తి కూడా దానికి ఉంది. నియంత్రించబడిన మనస్సు ఏంతో మేలు కలుగ చేయవచ్చు, అ దేవిధంగా, నిగ్రహింపబడని మనస్సు తుచ్ఛమైన తలంపులతో జీవాత్మను పతనానికి గురి చేస్తుంది.ఒక మిత్రునిగా ఉ పయోగించుకోవటానికి, మనస్సు యొక్క పనితీరుని అర్థం చేసుకోవటం ముఖ్యం. మన మనస్సు నాలుగు స్థాయిల్లో ప ని చేస్తుంది.మనస్సు: ఆలోచనలు సృష్టించినప్పుడు, దాని \'మనస్సు\' అంటాము.బుద్ధి : ఆలోచించి నిర్ణయాలు

मैं मरना चाहता हूँ

'आत्मा की भगवान पर निर्भरता पर बल देते हुए श्रीकृष्ण कहते हैं-"अर्जुन! भले ही तुम मेरी आज्ञा का पालन करो या न करो, तुम्हारी स्थिति सदैव मेरे प्रभुत्व में रहेगी। जिस शरीर में तुम रहते हो वह यंत्र मेरी माया शक्ति से निर्मित है। तुम्हारे पूर्व जन्मों के अनुसार मैंने तुम्हारी पात्रता के अनुसार तुम्हें शरीर प्रदान किया है। मैं इसमें स्थित रहता हूँ और तुम्हारे विचारों, शब्दों, और कर्मों का लेखा-जोखा रखता हूँ। इस प्रकार से वर्तमान में जो कर्म तुम करते हो उसका आंकलन करते हुए मैं तुम्हारे भविष्य का निर्णय करता हूँ। यह मत सोचो कि तुम मुझसे किसी भी प्रकार से स्वतंत्र हो। इसलिए अर्जुन तुम्हारे हित में यही है कि तुम मेरी शरण ग्रहण करो।'

Query=”Importance of Bhagavad Gita”

'The Lord was acting as the spiritual master of Arjuna. Therefore it was His duty to inquire from Arjuna whether he understood the whole Bhagavad-gītā in its proper perspective. If not, the Lord w as ready to re-explain any point, or the whole Bhagavad-gītā if so required. Actually, anyone who hears Bhagavad-gītā from a bona fide spiritual master like Kṛṣṇa or His representative will find t hat all his ignorance is dispelled. Bhagavad-gītā is not an ordinary book written by a poet or fic tion writer; it is spoken by the Supreme Personality of Godhead. Any person fortunate enough to he ar these teachings from Kṛṣṇa or from His bona fide spiritual representative is sure to become a l iberated person and get out of the darkness of ignorance.'

ମୁଁ ମରିବାକୁ ଚାହେଁ

'ଆତ୍ମାର ପରମାତ୍ମାଙ୍କ ଉପରେ ନିର୍ଭରଶୀଳତା ଉପରେ ଗୁରୁତ୍ୱାରୋପଣ କରି ଶ୍ରୀକୃଷ୍ଣ କହୁଛନ୍ତି, “ଅର୍ଜୁନ, ତୁମେ ମୋର ଆଜ୍ଞା ପାଳନ କରିବାକୁ ଇଚ୍ଛା କର ବା ନ କର, ତୁମେ ସର୍ବଦା ମୋର ନିୟନ୍ତ୍ରଣାଧୀନ ରହିବ । ତୁମେ ବାସ କରୁଥିବା ଶରୀର ମାୟା ଶକ୍ତିରୁ ଗଠିତ । ପୂର୍ବ କର୍ମ ଅନୁସାରେ, ତୁମେ ଯେଉଁ ଶରୀର ପାଇବାକୁ ଯୋଗ୍ୟ, ତାହା ମୁଁ ତୁମକୁ ପ୍ରଦାନ କରିଛି । ମୁଁ ସେଠାରେ ମଧ୍ୟ ଅଛି ଏବଂ ତୁମର ସମସ୍ତ ବିଚାର, ବାଣୀ ଏବଂ କର୍ମକୁ ମୁଁ ଲିପିବଦ୍ଧ କରୁଛି । ତେଣୁ ବର୍ତ୍ତମାନ ତୁମେ ଯାହା କରୁଛ ମୁଁ ତାହା ମଧ୍ୟ ଲକ୍ଷ୍ୟ କରି ତୁମର ଭବିଷ୍ୟତ ନିର୍ଦ୍ଧାରଣ କ ରିବି । ଜୋଣସି ପରିସ୍ଥିତିରେ ମଧ୍ୟ ତୁମେ ନିଜକ ମୋ ଠାର ସ୍ୱାଧୀନ ବୋଲି ପରିକଳ୍ପନା କରନାହିଁ । ତେଣୁ ହେ ଅର୍ଜୁନ! ତୁମେ ନିଜ ସ୍ୱାର୍ଥ ପାଇଁ ସମ୍ପୂର୍ଣ୍ଣ ମୋର ଶରଣାଗତ ହେବା ଆବଶ୍ୟକ ।'

Chapter 4

EVALUATION

To test the model's efficiency, we have created some ground truth value. These ground-truth values contain 30 queries and related chapter and verse number pairs around 10 for each query. For creating ground-truth value, we have checked different outputs predicted by different models and used the relevant answer for the evaluation.
We have used **Mean Average Precision(MAP)** as an evaluation metric.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{mAP} = \frac{1}{n} \sum_{k=1}^{k=n} \text{AP}_k$$

Where AP_k is average precision of class=k

4.1 Results

We have tested the ground-truth values across all models and calculated the MAP score for each model. Here we have shown some of the results.

Languages	**English**	**Hindi**	**Odia**	**Telugu**
BRS	0.067	0.059	0.025	0.046
TFIDF	0.224	0.187	0.128	0.196
BM25	0.328	0.217	0.226	0.254
LSI	0.482	0.356	0.298	0.372

Table 4.1: Results for basic models

Languages	English	Hindi	Odia	Telugu
Max Pooling Cosine	0.257	0.242	0.272	0.205
Max Pooling Euclidean	0.229	0.242	0.206	0.248
Mean Pooling Cosine	0.428	0.352	0.372	0.296
Mean Pooling Euclidean	0.321	0.204	0.198	0.267
Line Pooling Cosine	0.342	0.256	0.172	0.224
Line Pooling Euclidean	0.187	0.204	0.242	0.158
Whole Cosine	0.214	0.234	0.128	0.192
Whole Euclidean	0.167	0.128	0.192	0.182
Clustering Max Pooling	0.264	0.214	0.156	0.124
Clustering Mean Pooling	0.314	0.252	0.297	0.273
Clustering Whole	0.213	0.197	0.153	0.256

Table 4.2: Fasttext embedding models

Languages	English	Hindi	Odia	Telugu
Max Pooling Cosine	0.401	0.412	0.396	0.364
Max Pooling Euclidean	0.359	0.379	0.312	0.271
Mean Pooling Cosine	0.423	0.429	0.401	0.376
Mean Pooling Euclidean	0.216	0.271	0.296	0.284
Line Pooling Cosine	0.351	0.397	0.367	0.387
Line Pooling Euclidean	0.294	0.286	0.326	0.298
Whole Cosine	0.381	0.356	0.391	0.369
Whole Euclidean	0.302	0.291	0.284	0.278
Clustering Max Pooling	0.359	0.391	0.376	0.347
Clustering Mean Pooling	0.294	0.316	0.327	0.298
Clustering Whole	0.267	0.274	0.254	0.221

Table 4.3: Glove Embedding Models

Languages	English	Hindi	Odia	Telugu
Max Pooling Cosine	0.276	0.226	0.245	0.252
Max Pooling Euclidean	0.359	0.379	0.312	0.271
Mean Pooling Cosine	0.221	0.232	0.272	0.297
Mean Pooling Euclidean	0.321	0.282	0.345	0.272
Line Pooling Cosine	0.276	0.192	0.231	0.212
Line Pooling Euclidean	0.282	0.223	0.197	0.192
Whole Cosine	0.224	0.172	0.172	0.192
Whole Euclidean	0.272	0.252	0.112	0.121
Clustering Max Pooling	0.352	0.317	0.328	0.412
Clustering Mean Pooling	0.312	0.252	0.272	0.312
Clustering Whole	0.212	0.212	0.242	0.216

Table 4.4: SkipGram Embedding Models

Languages	English	Hindi	Odia	Telugu
Max Pooling Cosine	0.316	0.287	0.274	0.291
Max Pooling Euclidean	0.278	0.214	0.191	0.246
Mean Pooling Cosine	0.298	0.276	0.299	0.286
Mean Pooling Euclidean	0.246	0.197	0.222	0.216
Line Pooling Cosine	0.337	0.254	0.316	0.271
Line Pooling Euclidean	0.291	0.172	0.238	0.249
Whole Cosine	0.329	0.281	0.301	0.297
Whole Euclidean	0.286	0.204	0.259	0.257
Clustering Max Pooling	0.314	0.289	0.298	0.274
Clustering Mean Pooling	0.287	0.221	0.276	0.241
Clustering Whole	0.251	0.193	0.231	0.187

Table 4.5: CBow Embedding Models

Languages	English	Hindi	Cross Hindi	Cross English
Max Pooling Cosine	0.412	0.372	0.256	0.321
Max Pooling Euclidean	0.224	0.316	0.223	0.206
Mean Pooling Cosine	0.672	0.487	0.328	0.342
Mean Pooling Euclidean	0.521	0.323	0.272	0.212
Line Pooling Cosine	0.328	0.397	0.297	0.272
Line Pooling Euclidean	0.298	0.287	0.197	0.252
Whole Cosine	0.367	0.298	0.216	0.216
Whole Euclidean	0.342	0.192	0.248	0.278
Clustering Max Pooling	0.352	0.218	0.192	0.212
Clustering Mean Pooling	0.524	0.421	0.282	0.257
Clustering Whole	0.324	0.328	0.212	0.192

Table 4.6: Bert Model

4.2 Findings

We have found the following conclusions from the above tables and other experiments.

1. Boolean Retrieval System and TF-IDF are good models for exact keyword searching for all four languages. However, they are not efficient for semantic searching.

2. BM25 and LSI model work well for all four languages; also, it is very lightweight and fast compared to other models.

3. BERT models are performing very well for semantic and cross-lingual searching tasks for Hindi and English. It works well with Mean Pooling and Cosine Similarity metrics.

4. For all other models, Clustering with Max pooling performs well. Also, Mean Pooling with cosine similarity shows high values in the evaluations.

Chapter 5

CONTRIBUTION

ABHISHEK SAHU (21111002)

- LSA - Hindi, Odia
- Glove - Hindi, Odia
- Word2Vec - English, Odia, Hindi
- SentBert - Hindi, English
- FastText - Odia
- Basic - Hindi, Odia

ALOK KUMAR TRIVEDI (21111008)

- LSA - English, Hindi
- Glove - English
- Word2Vec - Hindi
- FastText - English
- Basic - Hindi

ASHITOSH MORE (21111017)

- LSA - English, Hindi
- Glove - Hindi
- Word2Vec - English, Hindi
- FastText - Hindi
- Basic - Hindi, English

SHIVAM KHARWAR (21111058)

- LSA - English, Hindi
- Glove - English, Hindi
- Word2Vec - English, Hindi
- FastText - Hindi,English
- Basic - Hindi,English

TANIKELLA SAI KIRAN (21111061)

- LSA - English, Telugu
- Glove - Telugu, English
- Word2Vec - Telugu, English
- SentBert - English, Hindi
- Basic - Telugu

Chapter 6

FUTURE WORKS

This project has a lot of scope in the future. Our idea is just a kickstart to this concept. We can improve the project as

- Including better commentaries for different languages
- Improving word embeddings which are highly specific for Bhagavad Gita
- Construction of Knowledge graph, which helps to find a better result
- Text Summarisation features which will help to find a summary of a relevant commentary
- Extending this work for more languages
- Use of State-Of-Art model like BERT, which can capture more context.
- Many More

Chapter 7

CONCLUSION

The Multi-Lingual Information Retrieval System used for the Semantic meaning of the Bhagavad Gita provided a diverse range of results based on the Language and Model. Multiple queries are used for retrieval in different languages. While using the exact key word matching technique, BM25 gave nearly 0.4 mAP for English and Hindi Languages, whereas LSI worked better for Odia and Telugu.

In the Embedding models, Mean Pooling Cosine performed better in Fasttext and Glove embeddings for all the four languages with MAP in the range 0.45 to 0.30. But in the SkipGram and CBow embeddings, different strategies like Mean Pooling Cosine, Max Pooling Cosine, Clustering Mean Pooling etc., had performed better based on the language.

While the BERT model has higher performance for Mean Pooling in English Language with about 0.7 mAP, it has relatively lesser mAP for Cross-Lingual English. Hindi and Cross-Lingual Hindi for BERT has decent performance with MAP in the 0.35-0.45 range. There is a lot of scope for further development in the project, given the access to better commentary in various languages and the implementation of multiple models.

Bibliography

[1] Prabhupada, AC Bhaktivedanta Swami, and Bhaktivedanta Swami. Bhagavad-Gita as it is. Los Angeles: Bhaktivedanta Book Trust, 1972.

[2] Chandra, Rohitash, and Venkatesh Kulkarni. "Semantic and sentiment analysis of selected Bhagavad Gita translations using BERT-based language framework." arXiv preprint arXiv:2201.03115 (2022).

[3] Stein, Daniel. "Multi-Word Expressions in the Spanish Bhagavad Gita, Extracted with Local Grammars Based on Semantic Classes." LREC 2012 Workshop Language Resources and Evaluation for Religious Texts (LRE-Rel). 2012.

[4] The Bhagavad Gita: a new translation. WW Norton Company, 2013.

[5] Fosse, Lars Martin. The Bhagavad Gita: the original Sanskrit and an English translation. YogaVidya. com, 2007.