Sanskrit Words Frequency Analysis

Praveen Patel, 241110404, praveenp24@iitk.ac.in
Puspesh Kumar Srivastava, 241110405, puspeshk24@iitk.ac.in
Uddeshya Raj, 24110406, uddeshya24@iitk.ac.in

CS685: Data Mining

Abstract

We intend to create a usage timeline for words present in Sanskrit texts using documents from sanskritdocunents.org and the digital corpus of Sanskrit. The finished project will be able to display a graph showing word usage trends in Sanskrit texts over different periods, given Sanskrit words as input. It would also show how those words are used in various Sanskrit texts and similar words in their vocabulary. The fundamental techniques used would be Text mining and preprocessing (tokenisation, lemmatisation), Word frequency analysis across different time windows, Time series analysis, or trend analysis using visualisation tools (e.g., Matplotlib, Seaborn).

1 Motivation of the Problem

Sanskrit, one of the world's most ancient languages, holds a vast and rich literary tradition spanning millennia. However, the evolution of word usage in Sanskrit texts over time remains a comparatively underexplored area of linguistic research. By creating a usage timeline for Sanskrit words, we wanted to design a platform to help linguistic researchers understand the rise and fall of using certain Sanskrit words or terms over centuries. Such an analysis would provide insights into how specific words and their meanings evolved and illuminate trends in their prominence across different genres and periods. Furthermore, visualising these trends and identifying similar words within the Sanskrit vocabulary would foster a deeper understanding of the language's structure and usage. This project uses modern computational linguistics, enabling researchers, students, and enthusiasts to explore Sanskrit texts dynamically and interactively.

2 Data Used

We took the Sanskrit text corpus from Digital Corpus of Sanskrit (DCS) and the dictionary they used for word analysis and found similar vocabulary words. The DCS corpus is a comprehensive collection of Sanskrit texts that includes various works across different genres and periods. Since the time of creation for each document was not available in a single location, this information was gathered from multiple sources across the internet. In total, 282 Sanskrit texts, including 281 profiled books and one dictionary, were scraped and processed for this project. The original corpus contained multiple unprofiled sub-corpora, also included for thorough analysis. This dataset provides a rich foundation for studying the usage patterns of Sanskrit words and their evolution across time, using techniques such as word frequency analysis and time-series analysis.

3 Methodology

The methodology of the project includes:

3.1 Scrapping

We decided to scrape the text from the DCS because its API is not publicly available. From each of the Sanskrit texts on the DCS, we extracted the name of the text and each sentence, along with the root words of the words used in those sentences. The extracted information for each book was saved in individual JSON files. Additionally, the dictionary provided by the DCS was scraped and stored as a Python dictionary, which was then serialised into a JSON file for further use.

3.2 Frequency calculation

We read a dictionary of Sanskrit words and process multiple JSON files containing structured Sanskrit text data to determine how frequently each word appears. Using the JSON and Os libraries, we analysed the content of each text file, counted the occurrences of each word in specific fields, and captured up to three example sentences for context. The outcome is a JSON file summarising the frequency of each word in each document, along with relevant example sentences, providing a detailed overview of word usage across the texts.

3.3 Time series analysis

We implemented a script to visualise the frequency of specified Sanskrit words over historical periods. First, we loaded data from JSON files containing a Sanskrit dictionary, word frequency information, and a time mapping for various texts. We prepared a list of words to analyse, and for each word, we aggregated its occurrences over time by mapping the word's frequency data to corresponding time ranges in the text data.

For visualisation, we divided the period from -1500 to 1200 into 100-year intervals and grouped word occurrences within these intervals. We plotted the results on a bar chart, each representing the frequency of a word's occurrence in that period. To ensure clarity when displaying multiple words, we used distinct colours for each word and adjusted bar positions slightly for easy comparison.

To create aesthetically distinct colours, we generated a set of unique colour codes to avoid overlap and enhance visualisation. The output is a graph showing the frequency distribution of the chosen words over time, allowing us to compare the usage trends of different words throughout history. This helps understand the historical context and prominence of specific words in ancient texts.

3.4 Finding similar words

We have used the Sentence Transformers library and the paraphrase-MiniLM-L6-v2 model to generate sentence embeddings(size 384). This model efficiently captures semantic meaning and encodes sentences in batches of 1024 for faster processing. The embeddings are high-dimensional vectors, and batching ensures optimised speed and memory usage. The process includes a progress bar and can be run on a GPU for further speed improvements.

Top-N Words Calculation: Cosine similarity is used to identify the most similar words by normalising the embeddings with Scikit-learn's normalise function. The FAISS library is employed for efficient similarity search, using the IndexFlatIP index for fast nearest-neighbour searches. The top-n similar words are retrieved based on the highest cosine similarity scores, and the results are stored in a dictionary.

Optimisations: Batch processing during embedding generation speeds up the process, while FAISS enhances the similarity search. Normalisation ensures accurate cosine similarity, and for larger datasets, leveraging parallelisation, GPU support, and adjusting batch sizes can improve efficiency. Pickle is used to serialise the dictionary of similar words, allowing for easy storage and retrieval of the results without recomputing them each time the program is run. This improves efficiency, especially when dealing with large datasets, and ensures that the results are readily available for future use. Compared to using the BERT model and calculating the cosine similarity matrix for similarity calculation, with optimisations, the embeddings of 289846*289846 phrases were reduced from 300GBs to 31MBs and the calculation time from many hours to 40 seconds.

3.5 Interface

We used Streamlit to create an interactive interface for this project, enabling an accessible user experience. The app allows users to choose between Comparative Study and Word Analysis modes. In Comparative Study mode, users can input multiple words to compare their historical frequencies over a time range selected. In contrast, the Word Analysis mode offers detailed insights into a single word, including its meaning, similar words, and occurrences in historical texts. Sidebar options provide word selection, number of comparisons, and adjustable time ranges using dropdowns and sliders. The output includes dynamic visualisations of word frequencies over time and detailed word data, making exploration and analysis straightforward and insightful.

4 Results

In the first part interface of our project shows the complete word analysis of a choosen word representing the meaning of the word and words similar to that word.Below that it shows a graph representing the frequency of the word used in a range of time.Below that it represents the name of all the books and texts in which the word is present. In the second part it displays a comparative analysis graph of more than one word.

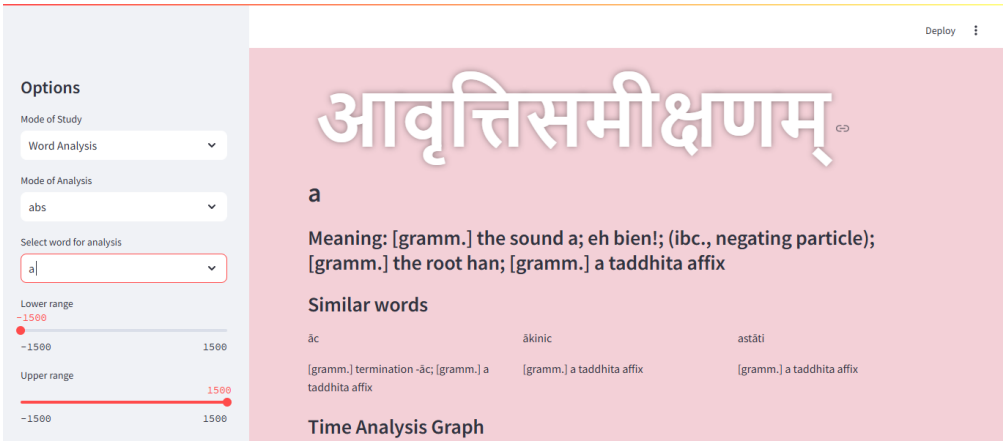

Figure 1: Word analysis showing similar words and meaning

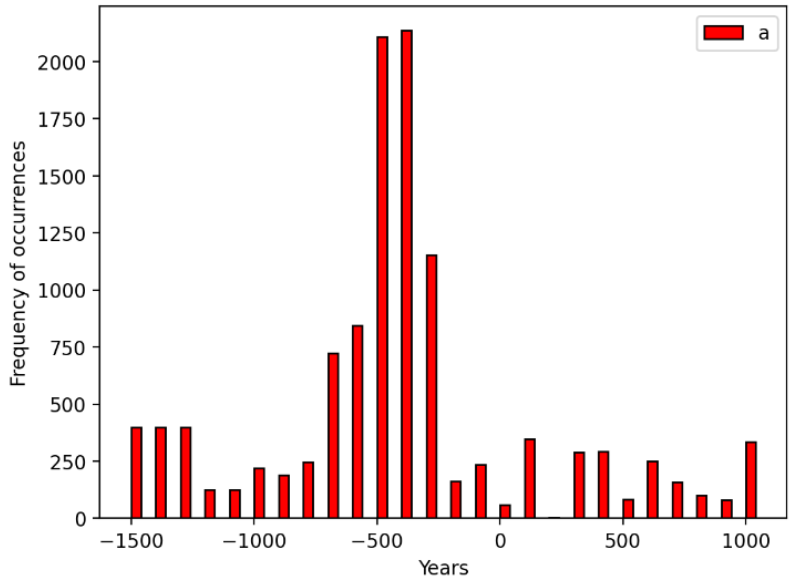

Figure 2: Time Graph

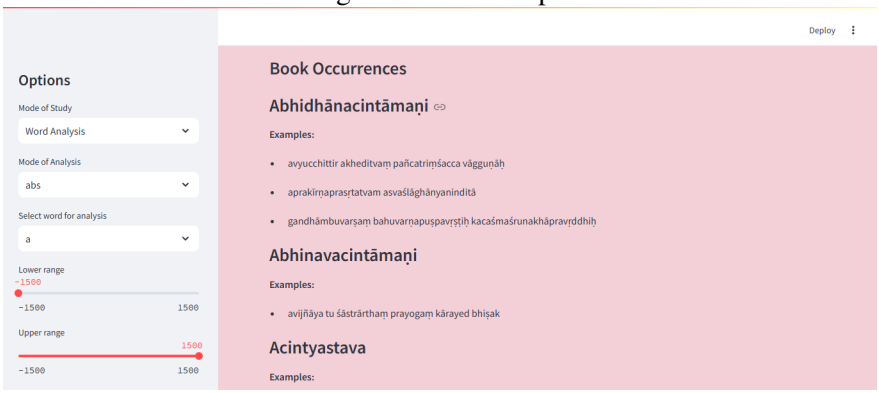

Figure 3: Books occurrence and Examples

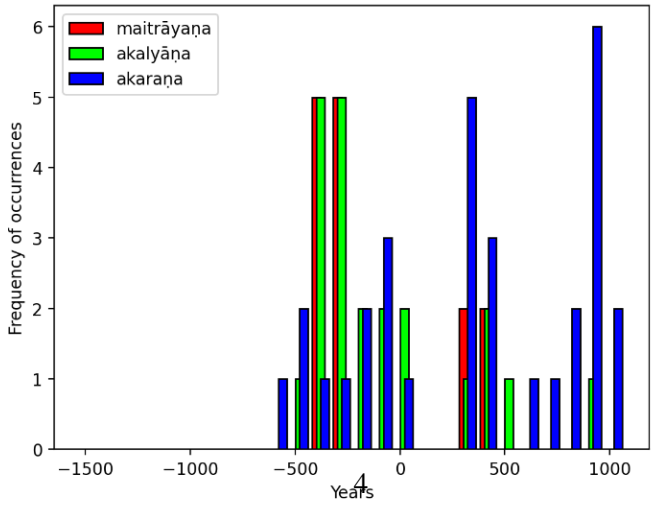

Figure 4: Comparative graph showing comparison of three words

5 Conclusions and Future Directions

This project aims to provide valuable insights into the evolution of word usage in Sanskrit texts across different periods. By leveraging the Sanskrit DCS corpus and applying modern text-mining techniques, we have generated a usage timeline for Sanskrit words, visualised their trends, and identified similar words within the vocabulary. Combining word frequency analysis, time-series analysis, and advanced computational tools enables a deeper understanding of how Sanskrit words have evolved in meaning and prominence over the centuries. This approach enriches our knowledge of Sanskrit literature and opens new avenues for digital humanities research in classical languages.

There are several exciting possibilities for further development. Future iterations of this project could integrate more extensive and diverse Sanskrit corpora, expanding the scope of word usage trends. Incorporating additional computational techniques, such as machine learning models for semantic analysis and context-based word relationships, could further enhance the accuracy and depth of the word usage timeline. Expanding the project to include real-time updates from newly digitised texts would ensure the analysis remains current and dynamic. The corpus could also be used to build a comprehensive thesaurus, contributing to a richer understanding of the relationships between words. Instead of focusing solely on root words, we could analyse compound words, which can be broken down into root words for more granular frequency and other linguistic analyses. Ultimately, this work could be a foundation for building interactive tools that allow researchers, students, and enthusiasts to explore Sanskrit's historical, cultural, and linguistic significance engagingly and comprehensively.

6 Team Contributions

The project was carried out by a three-member team, with each of us contributing to different aspects of the work while also collaborating on all major tasks like data scraping. Praveen Patel took the lead in developing the user interface and designing and implementing the interactive elements that allow users to input words and view results. Uddeshya Raj focused more on the time-series analysis, particularly analyzing word usage trends across different books and time periods. Puspesh Srivastava was primarily responsible for detecting similarity in words, using cosine similarity and advanced computational techniques to identify similar words in the Sanskrit vocabulary. Although each of us had a primary area of focus, we all worked together to ensure smooth integration and completion of the project.

Bibliography

1. **Sanskrit DCS (Digital Corpus of Sanskrit)**
   Website: `http://www.sanskrit-linguistics.org/dcs/`
   Provides access to a large corpus of Sanskrit texts for linguistic and historical research.

2. **Sentence-Transformers Documentation**
   Website: `https://www.sbert.net/`
   Official site for the Sentence-Transformers library, offering pre-trained models and tutorials for generating sentence embeddings.

3. **FAISS Documentation**
   Website: `https://faiss.ai/`

The official site for FAISS, a library for efficient similarity search and clustering of dense vectors, used for fast nearest-neighbor searches.

4. **Pickle Documentation**
   Website: `https://docs.python.org/3/library/pickle.html`
   Official Python documentation for Pickle, which allows for object serialization and deserialization.

5. **Streamlit Documentation**
   Website: `https://streamlit.io/`
   Official site for Streamlit, an open-source app framework to create custom data applications and visualizations with Python, ideal for building web interfaces for your project.

6. **Selenium Documentation**
   Website: `https://www.selenium.dev/documentation/en/`
   The official site for Selenium, a tool for automating web browsers, which can be used for web scraping tasks where data is dynamically loaded via JavaScript.

7. **Real Python - Web Scraping**
   Website: `https://realpython.com/beautiful-soup-web-scraper-python/`
   A tutorial on Real Python for learning how to scrape websites using Python, covering libraries like BeautifulSoup, Requests, and other scraping tools.

8. **OpenAI Tools Documentation**
   Website: `https://platform.openai.com/docs`
   Official site for OpenAI, providing documentation for accessing various AI tools like GPT, Codex, and other APIs for tasks like text generation, analysis, and more.