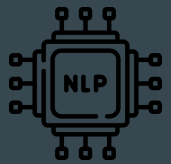
Trail Sense: Optimising Trek Selection through Semantic Analysis

RAHUL PATIL
Student ID : 1109004
Thesis Supervisor : Ritupurna Dutta
Masters of Science in Computer Science
Liverpool John Moores University, UK

INTRODUCTION & PROBLEM STATEMENT

Adventure tourism, particularly trekking, is a rapidly growing sector. This growth has led to an explosion of online information regarding treks, including reviews, descriptions, difficulty levels, altitudes, seasons, durations, weather conditions, and regional highlights. The sheer volume and variety of this information make it challenging to sift through and select the right trek. This thesis aims to address the problem by leveraging Natural Language Processing (NLP) and state-of-the-art models to simplify the trek selection process, extracting relevant and important information from extensive data



Literature Review

The niche topic of NLP in the context of trekking has limited literature, but we can draw parallels from the broader tourism industry where NLP has been impactful. Here are some relevant studies and their contributions over time:

2022

Tourist Sentiment Mining Based on Deep Learning - (Li et al., 2022)

Sentiment Analysis

2023

Natural language processing: state of the art, current trends and challenges. - (Khurana et al., 2023)

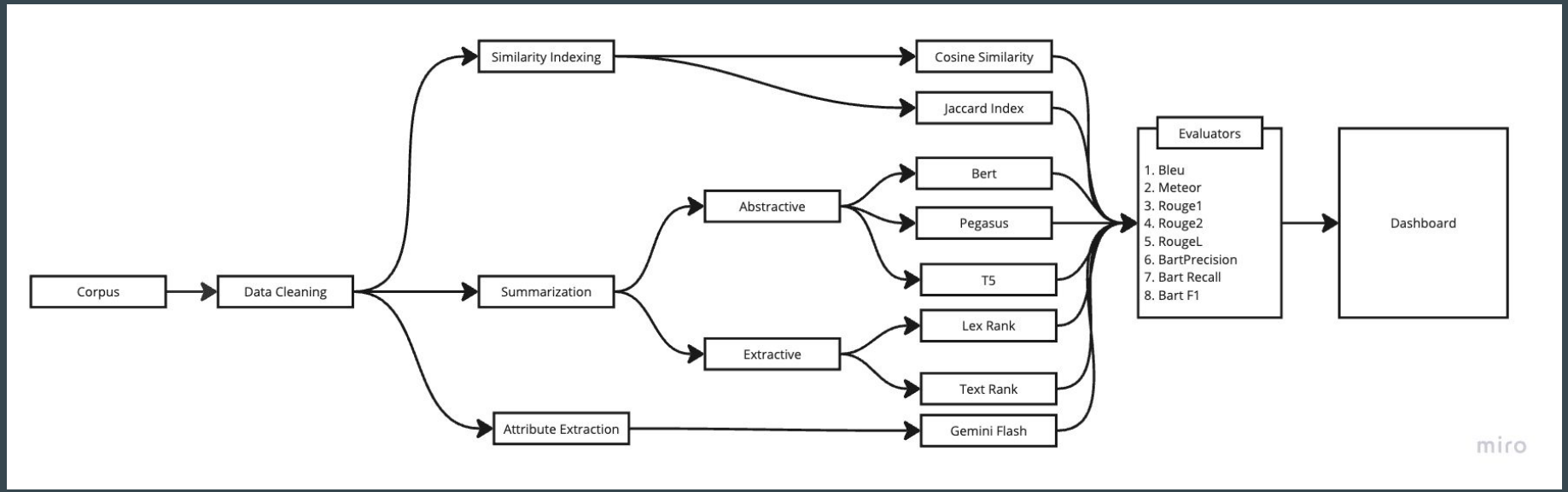
ChatBot

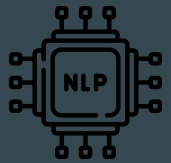
2015

Development of Social Media Strategies in Tourism Destination (Kiralova, Alzbeta & Pavlicek, Antonin. (2015))

Marketing

Methodology





Data Acquisition & Cleaning

Data Acquisition

Web scraping data from trekking website to get trek description for 225+ t¯reks all over India using Python.

HTML Data Cleaning

Extracting the body of the trek description by removing unwanted tags from the html data using beautifulsoup library in Python.

Removing stop words

Removing english stop words such as “the”, “and” from the sentences as they do not add value to the content.

Lemmatizing words

Converting words to their base format. For example the word “running” becomes “run”

Standardizing Data

Remove special character and unwanted line endings and convert everything to lowercase.

Similarity Index Calculation

Cosine Similarity

Cosine Similarity as Angular Distance:

1. Each trek description is a vector in a high-dimensional space.
2. Each dimension represents a word or term (e.g., "summit," "ridge," "waterfall").
3. The angle between two trek vectors is a measure of their similarity.
4. Smaller angle = more similar.

Key Points:

1. Accounts for the frequency of terms in each trek.
2. Works well when the length of descriptions varies.
3. Range: 0 (no similarity) to 1 (identical)

Jaccard Index

Jaccard Index as Set Intersection:

1. Each trek description is a set of unique words.
2. Jaccard similarity is the ratio of the intersection (shared words) to the union (all words) of the two sets.
3. Higher Jaccard similarity = more words in common.

Key Points:

1. Ignores word frequency, focusing only on the presence or absence of terms.
2. Useful when the vocabulary used to describe treks is consistent.
3. Range: 0 (no shared words) to 1 (all words are shared)

Extractive Summarization

Text Rank

1. What it is:
    a. Graph-based ranking model
    b. Inspired by Google's PageRank algorithm
    c. Ranks sentences based on importance
2. Advantages:
    a. Unsupervised method
    b. No need for labeled training data
    c. Effective for identifying key sentences
3. Limitations:
    a. Ignores semantic meaning
    b. Relies on sentence connectivity
    c. May not perform well on very short texts
4. Use Cases:
    a. Summarizing news articles
    b. Extracting key points from academic papers
    c. Generating abstracts for long documents

Lex Rank

1. What it is:
    a. Graph-based algorithm
    b. Uses cosine similarity for sentence connectivity
    c. Ranks sentences by eigenvector centrality
2. Advantages:
    a. Captures global structure of the text
    b. Effective for multi-document summarization
    c. Unsupervised method
3. Limitations:
    a. Computationally intensive for large texts
    b. May struggle with context understanding
    c. Depends on quality of sentence similarity measures
4. Use Cases:
    a. Multi-document summarization
    b. Summarizing legal documents
    c. Extracting key sentences from large corpora

Abstractive Summarization

Pegasus

1. What it is:
   a. Transformer-based model for abstractive summarization
   b. Focuses on generating summaries by masking entire sentences
2. Size of Model: Large pre-trained model (up to 568 million parameters)
3. Advantages:
   a. High-quality, coherent summaries
   b. Specialized for summarization tasks
   c. Effective at handling long documents
4. Limitations:
   a. Requires significant computational resources
   b. Training data dependency for domain-specific tasks

Text to Text (T5)

1. What it is:
   a. Text-to-Text Transfer Transformer
   b. Treats all NLP tasks as text generation tasks
2. Advantages:
   a. Unified framework for various NLP tasks
   b. Highly flexible and versatile
   c. Strong performance across multiple benchmarks
3. Limitations:
   a. Larger models require extensive computational resources
   b. May overfit without careful fine-tuning
   c. Complexity in training large model

Bert

1. What it is:
   a. Bidirectional Encoder Representations from Transformers
   b. Originally designed for various NLP tasks, adapted for summarization
2. Advantages:
   a. Captures context from both directions
   b. Highly versatile and pre-trained on diverse corpora
   c. Effective at understanding sentence context
3. Limitations:
   a. Not specifically optimized for summarization
   b. Fine-tuning required for specific tasks
   c. Computationally intensive

Gemini Flash Model

What it is? : Gemini Flash 1.5 is built upon a transformer architecture, which is a powerful neural network designed for processing sequential data like text. It excels at understanding language nuances, context, and long-range dependencies. Its pre-training on a massive dataset of text and code gives it a broad understanding of the world and enables it to perform a wide range of tasks.

How we used it : We created a custom prompt where we passed trek description to the model and asked it to extract important attributes from the trek, using these attributes we were able to group and find similar treks to each other based on different attributes.

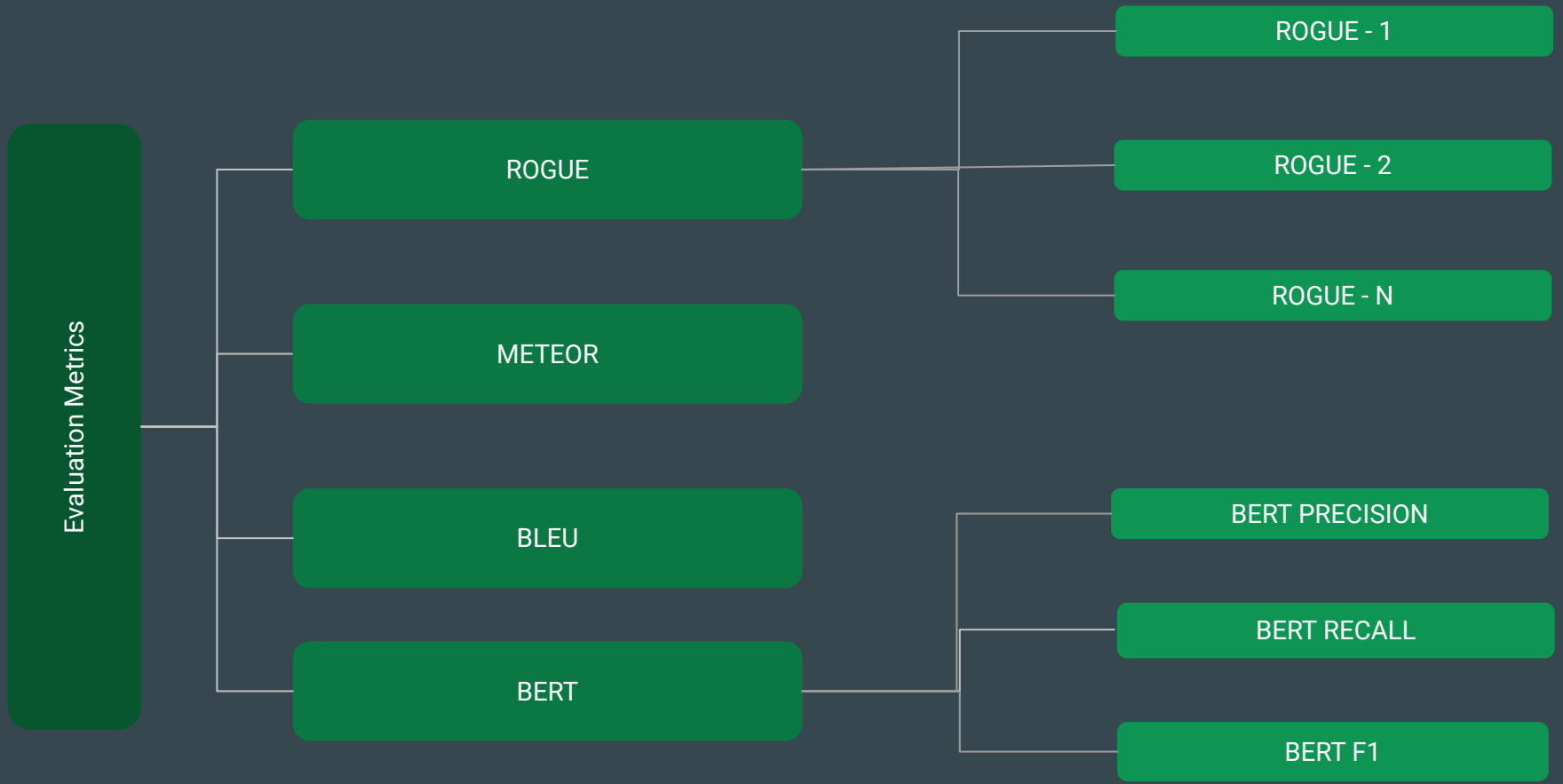
Custom Prompt :

```
Extract the following keys from the trek description by interpreting it and getting the key-value pairs: region, difficulty, season, days, altitude. The values of these keys can have multiple values as lists. If the values are not explicitly mentioned in the text, infer them.Return a JSON dictionary with the key-value pairs, without adding any additional descriptions or comments. Trek Description : ```{text}```
```

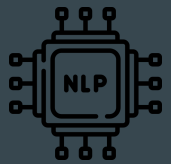
Sample output:

```
{"Ajan Top": {"region": ["Uttarakhand"], "difficulty": ["Easy"], "season": ["Summer", "Autumn"], "days": [1], "altitude": ["<10k"], "month": ["April", "May", "June", "September", "October", "November"], "temperature": ["(10 Deg C, 20 Deg C)"]}
```

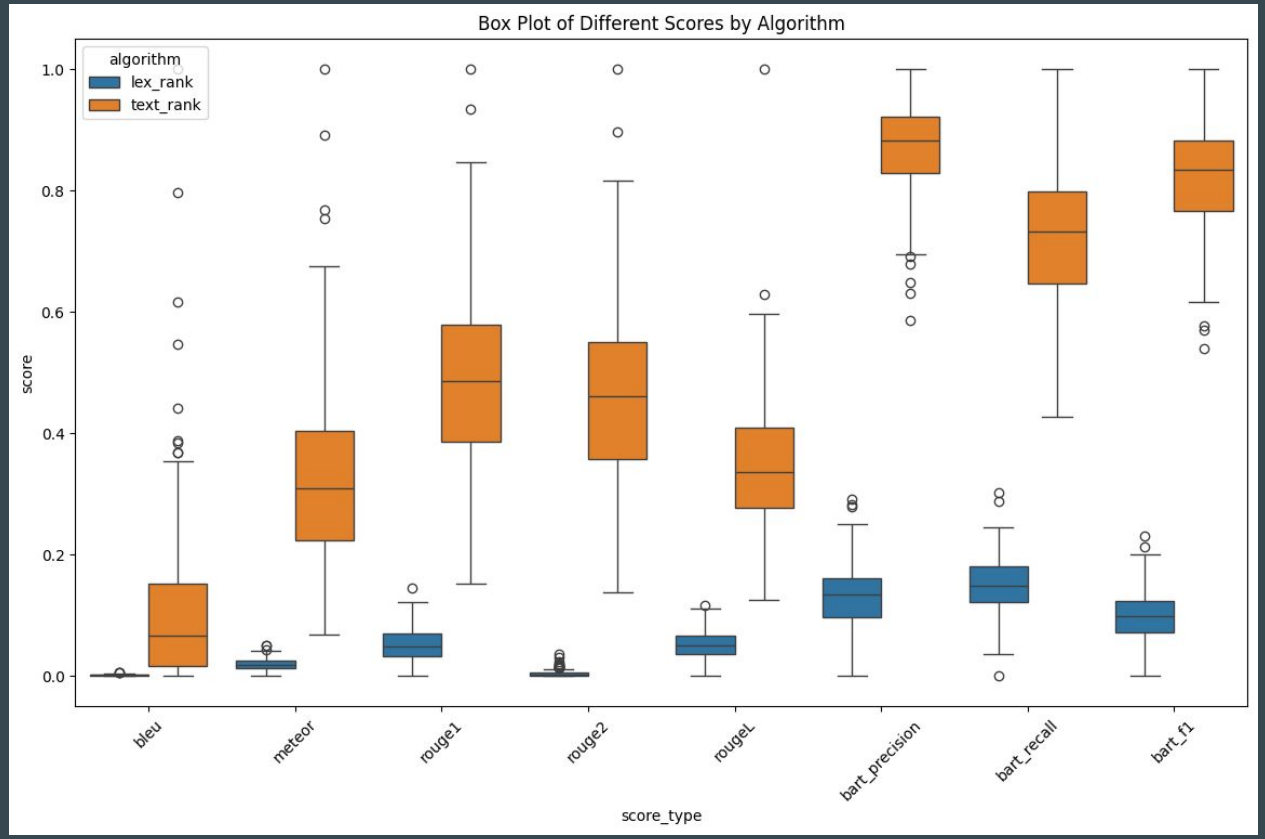
Evaluation Metrics



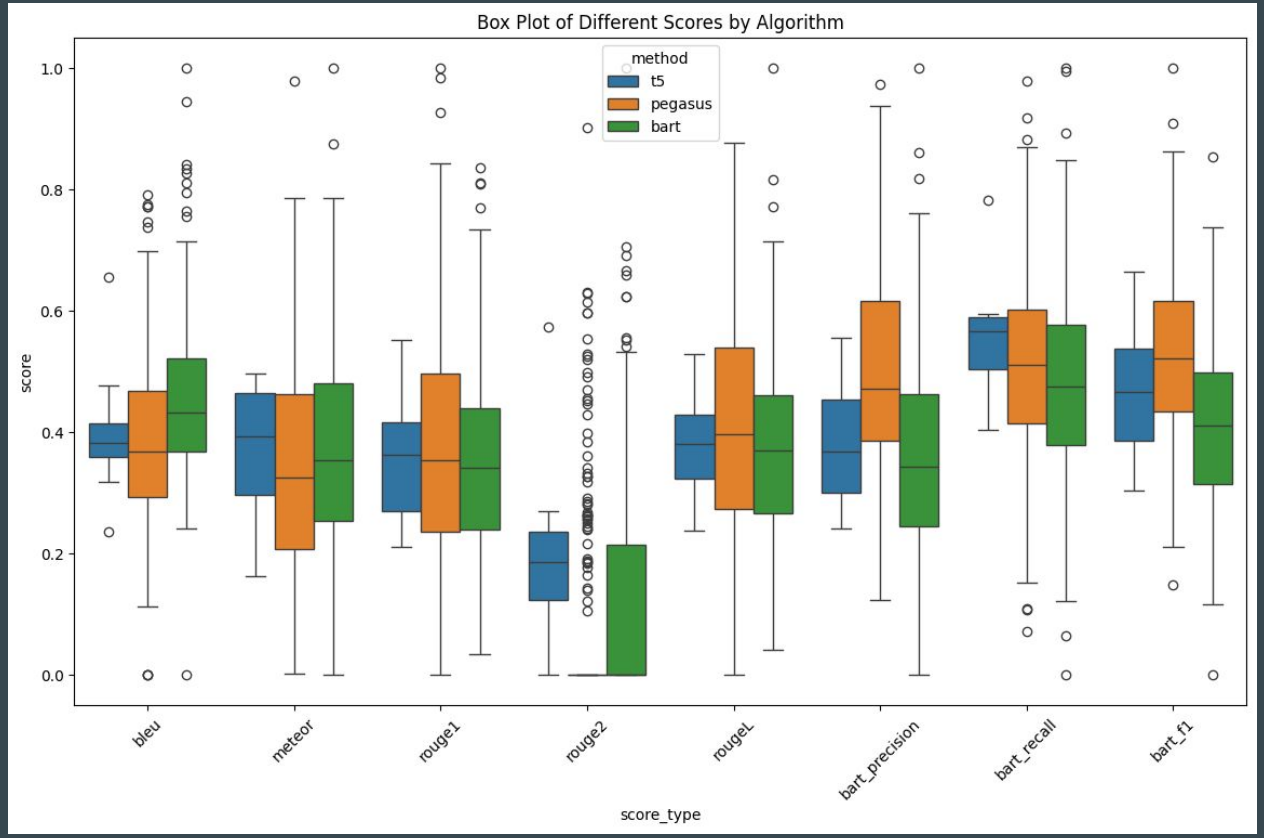
Results & Discussion



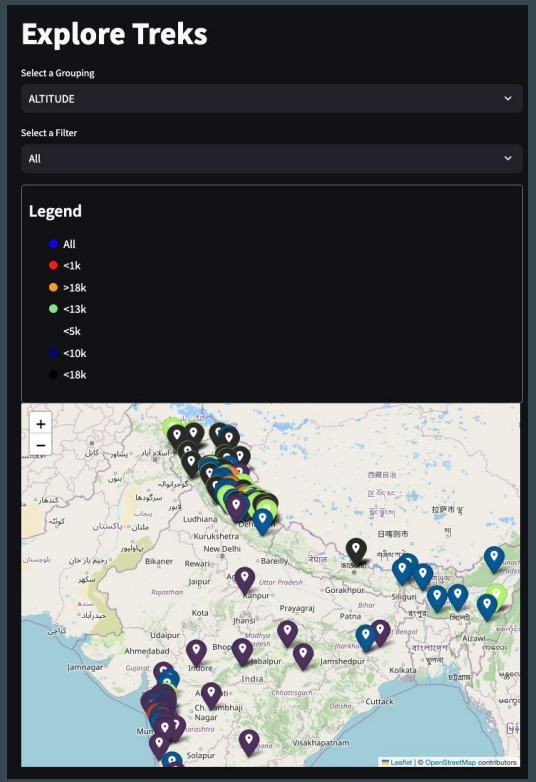
Extractive Summarization Results

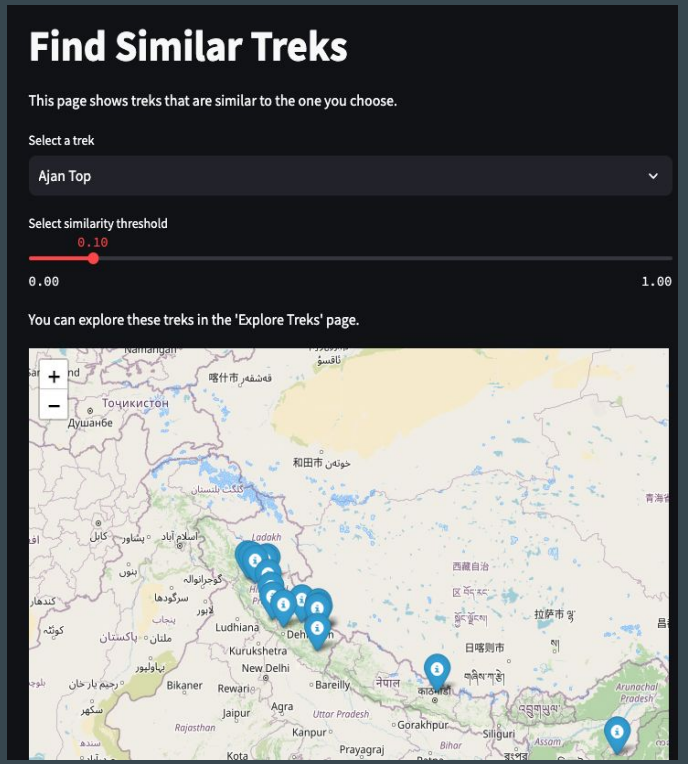


Abstractive Summarization Results



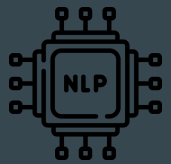
Sample Dashboard



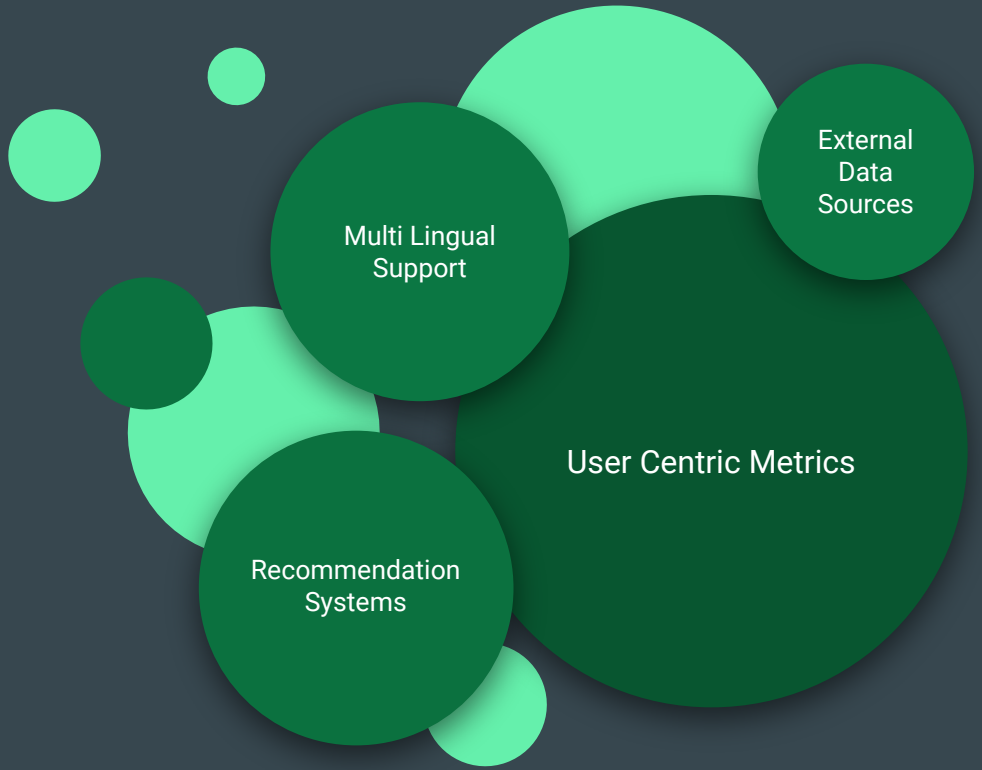


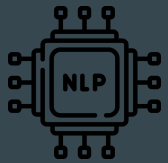
Conclusion

In summary, our research demonstrates the effectiveness of both extractive and abstractive summarization techniques in generating useful summaries of trek descriptions. The interactive dashboard provides a user-friendly way to explore and analyze these summaries. While the qualitative feedback from trekkers has been positive, future work will focus on incorporating quantitative metrics, developing a sophisticated recommendation system, integrating external data sources, enhancing summarization techniques, expanding evaluation metrics, and adding multi-lingual support. These steps will ensure that our data generation and summarization system remains a valuable tool for trekkers looking to evaluate and select trekking options.



FUTURE WORKS





Thank you!