AI SINGAPORE

AIAP Batch 12 Mini Project

18 Apr 2023

Hugging Face Model Recommender System (HFMRS)

By Bryan, Jia Hao, Kok Wai, Wan Ying

Table of Contents

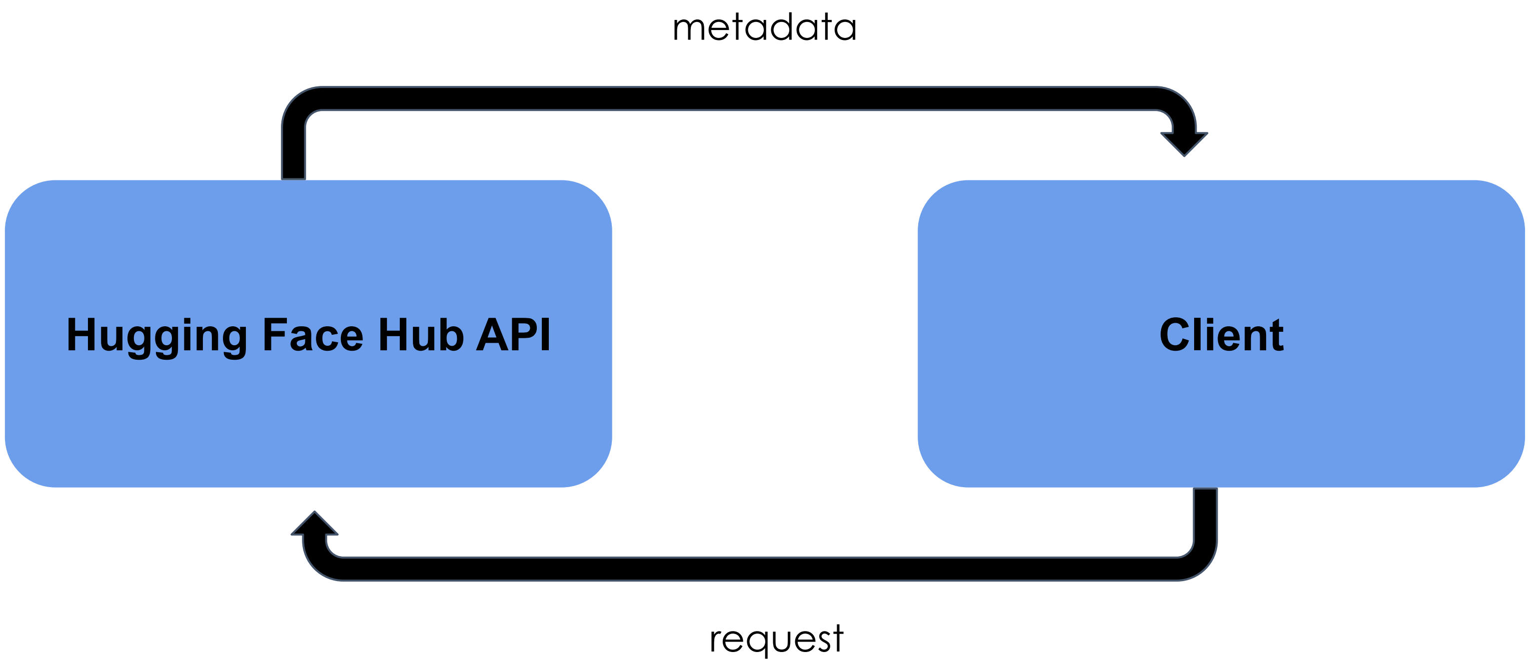
Problem Statement

- Challenge to identify the appropriate models from vast number of AI models available
- Evolution of AI technology and the emergence of new models adds to the complexity
- Limitation with existing model search on Hugging Face

Introduction

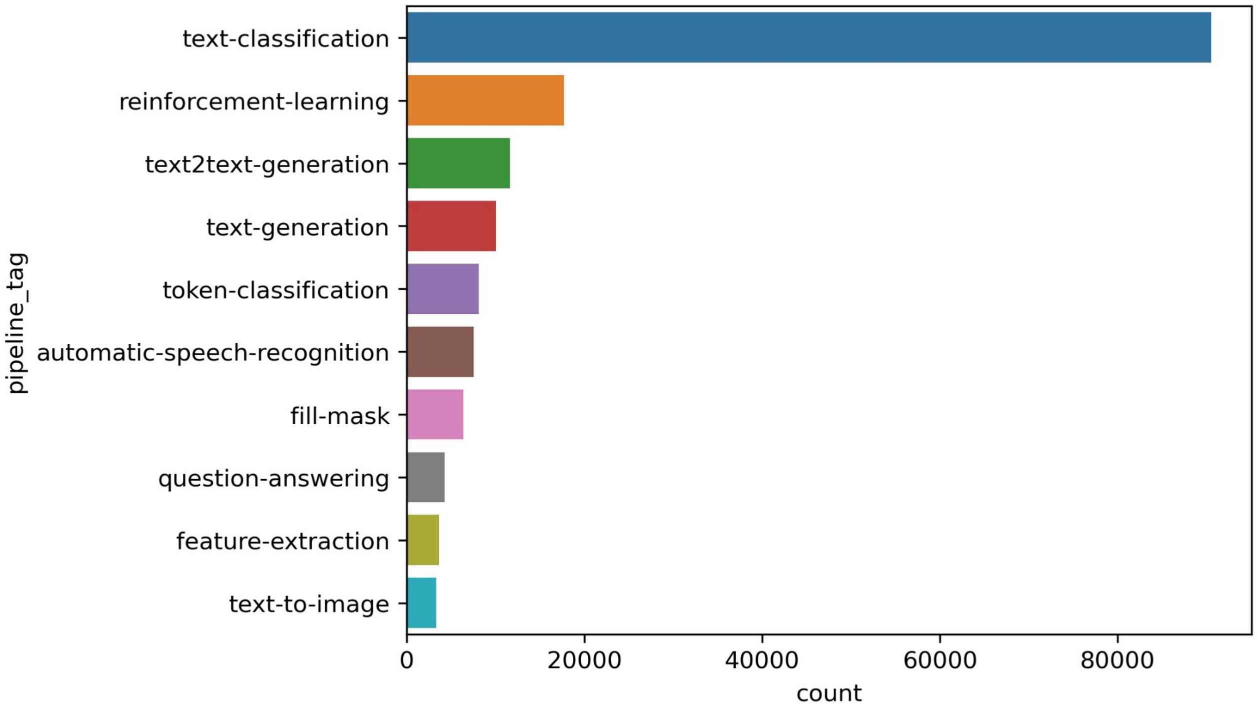
```python
from huggingface_hub import HfApi

hf_api = HfApi()
models = hf_api.list_models()
```

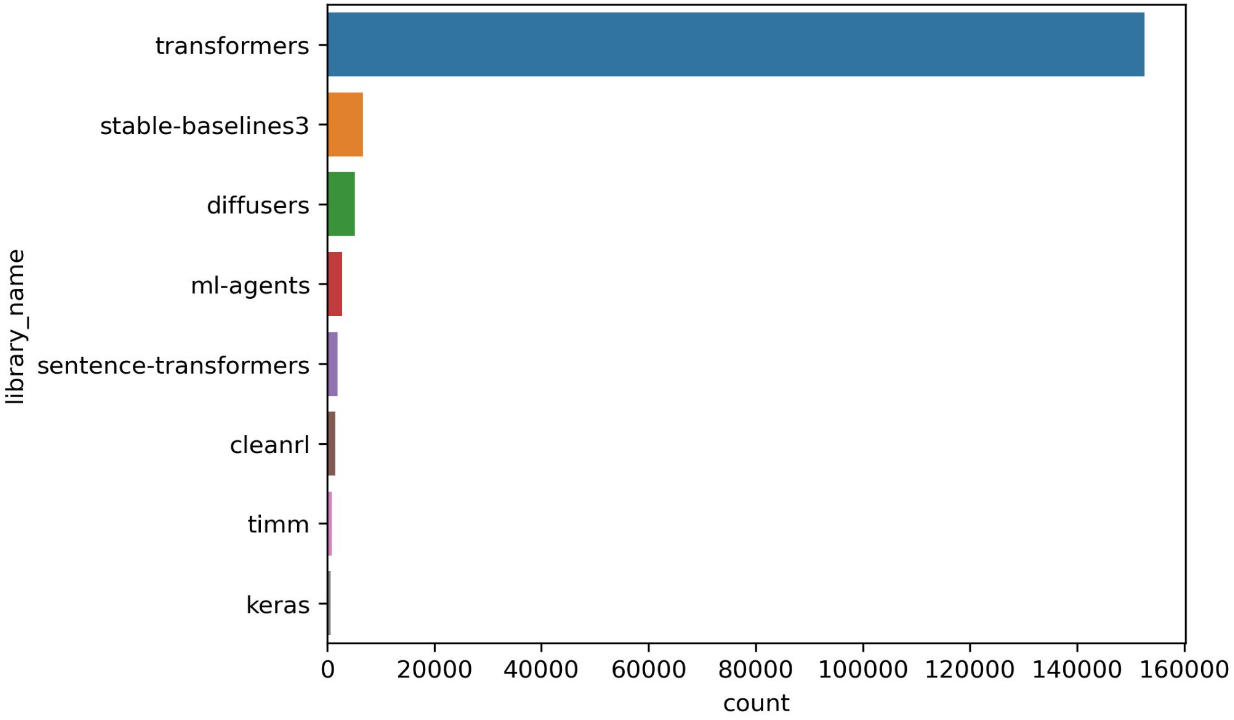
huggingface_hub



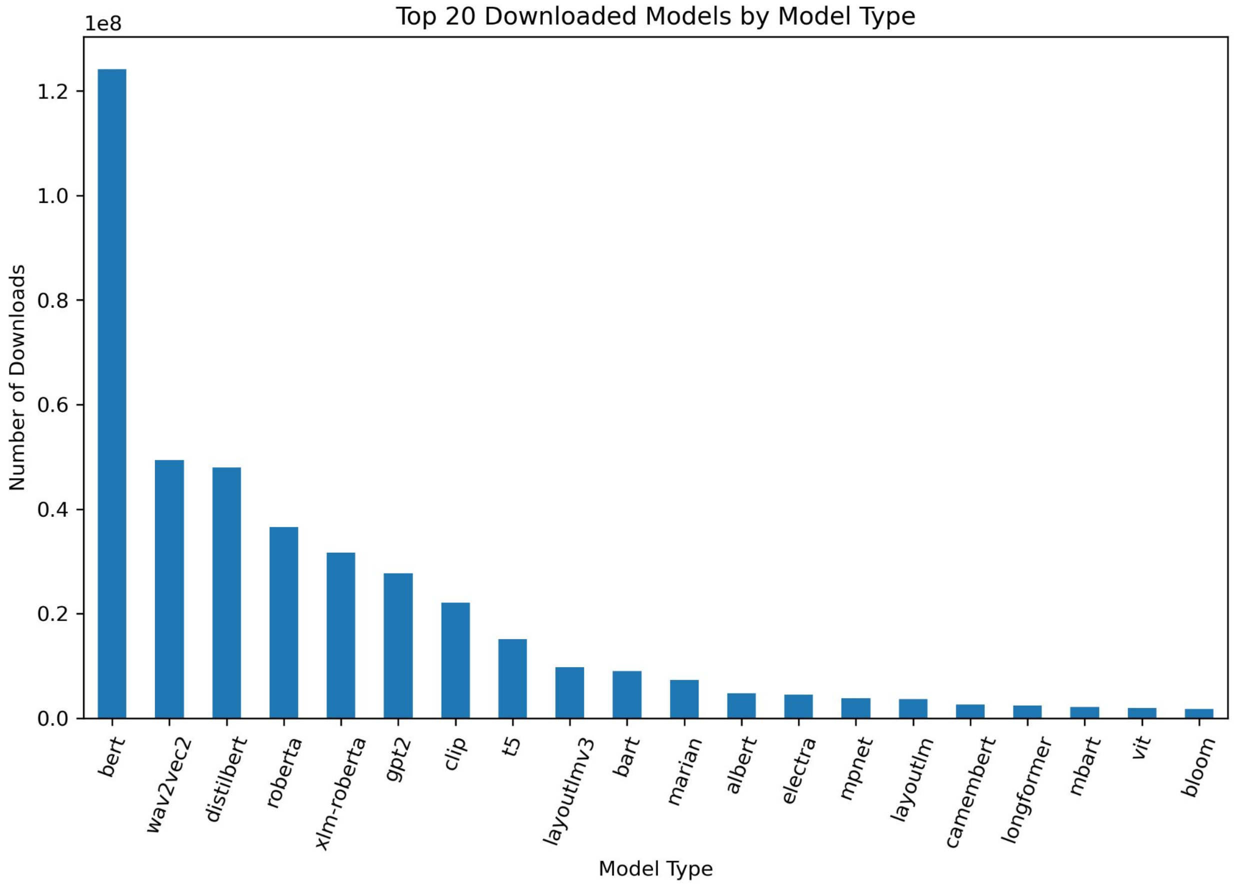
Field	Description	Example
modelId	Unique identifier for the model.	albert-large-v1
tags	Additional information about the model, such as programming languages, tasks, language, datasets, and license.	['pytorch', 'tf', 'albert', 'fill-mask', 'en', 'dataset:bookcorpus', 'dataset:wikipedia', 'arxiv:1909.11942', 'transformers', 'license:apache-2.0', 'autotrain_compatible', 'has_space']
pipeline_tag	Type of task the model was specifically trained for.	fill-mask
config	Technical information about the model's architecture and type.	{'architectures': ['AlbertForMaskedLM'], 'model_type': 'albert'}
pdownloads	The number of downloads the model has received.	357
library_name	The name of the library that hosts the model.	transformers

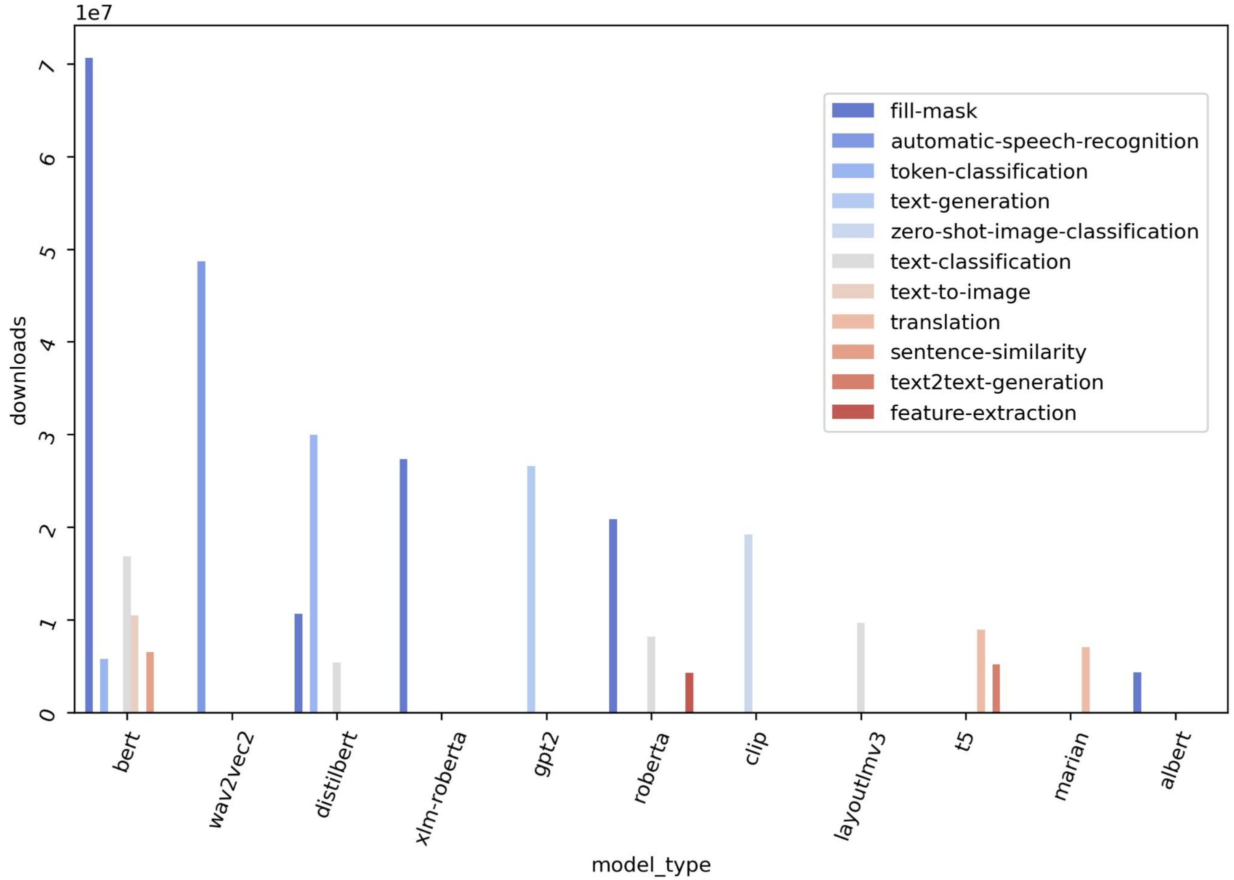
Exploratory Data Analysis





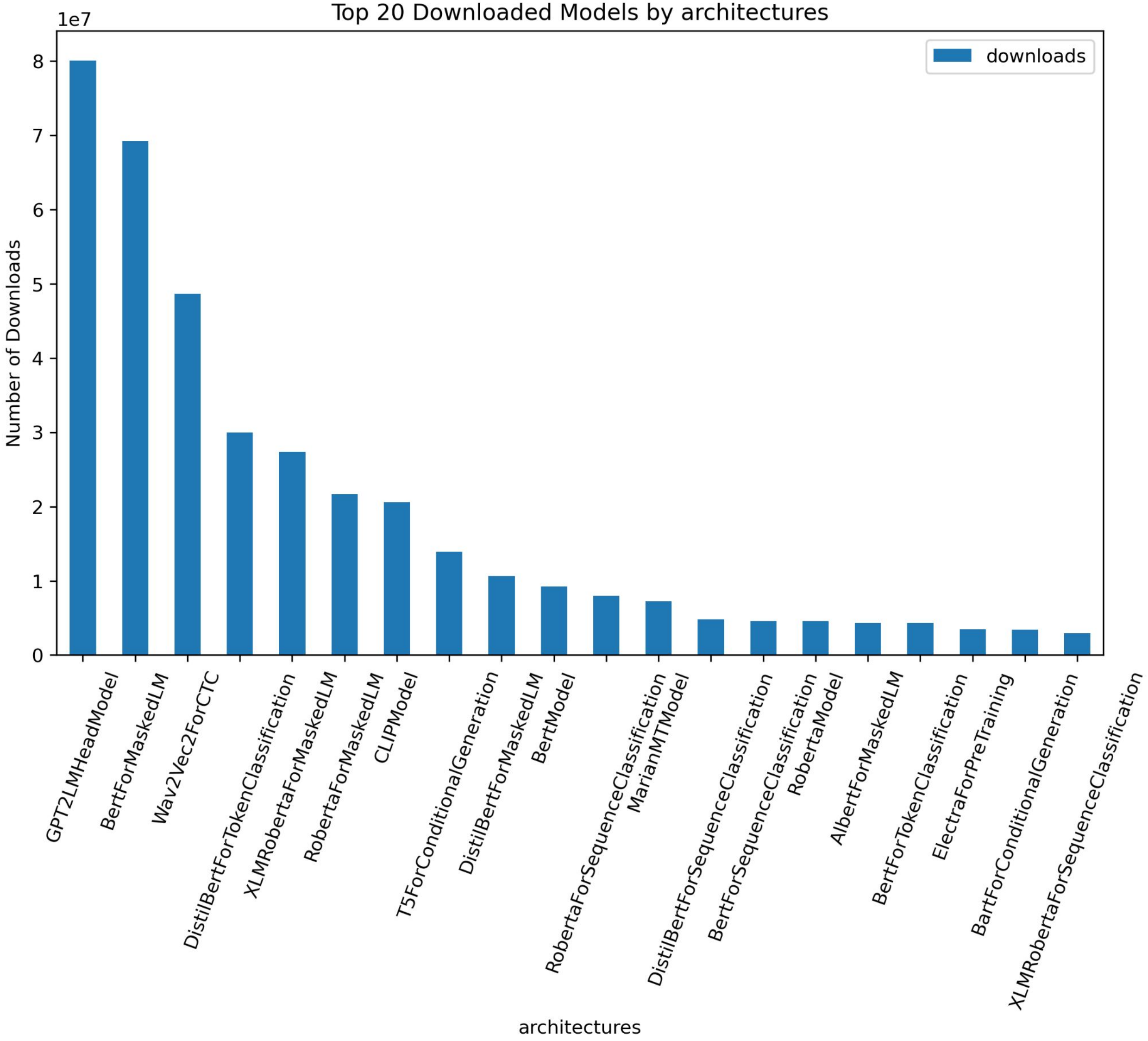
Exploratory Data Analysis

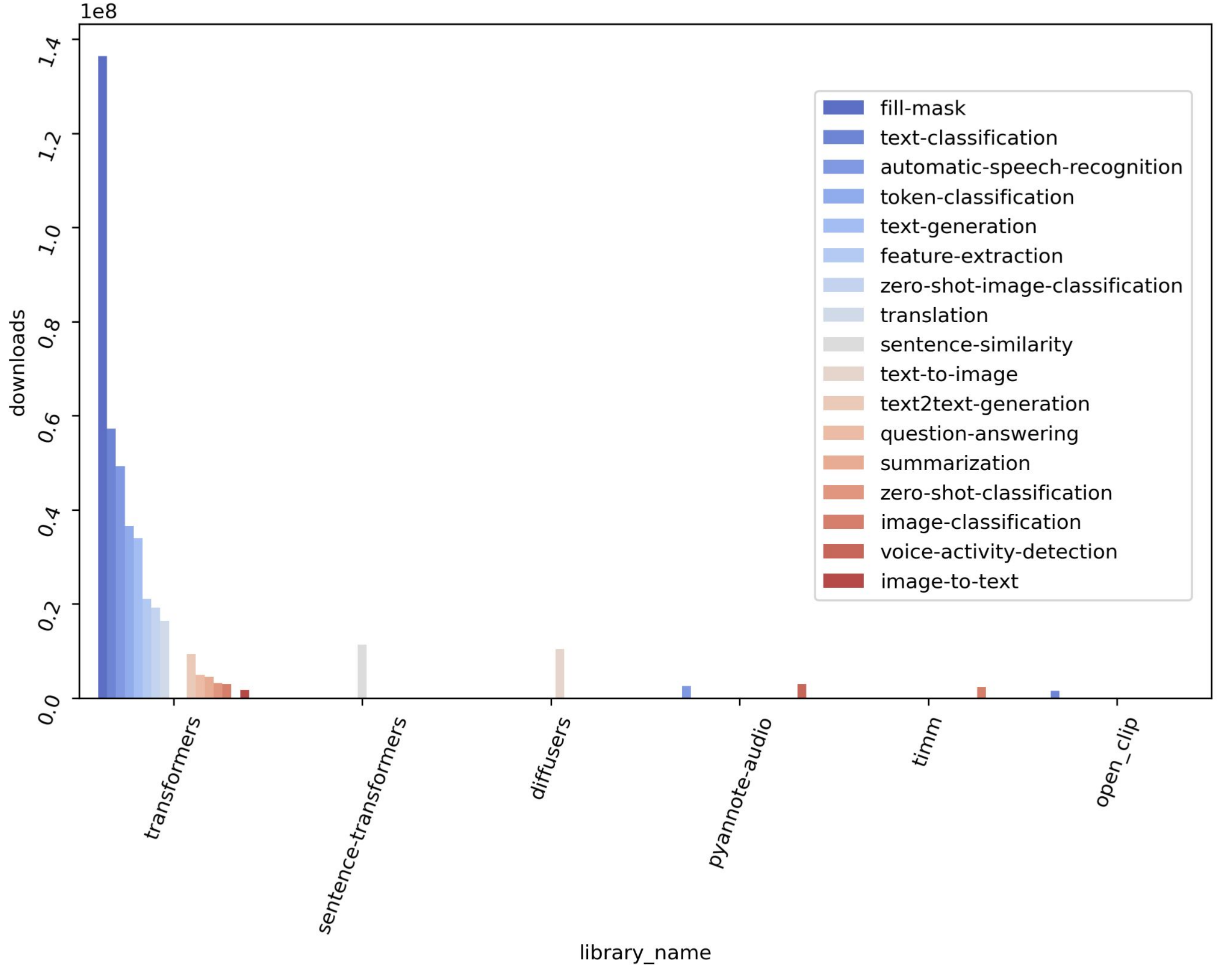




AI SINGAPORE

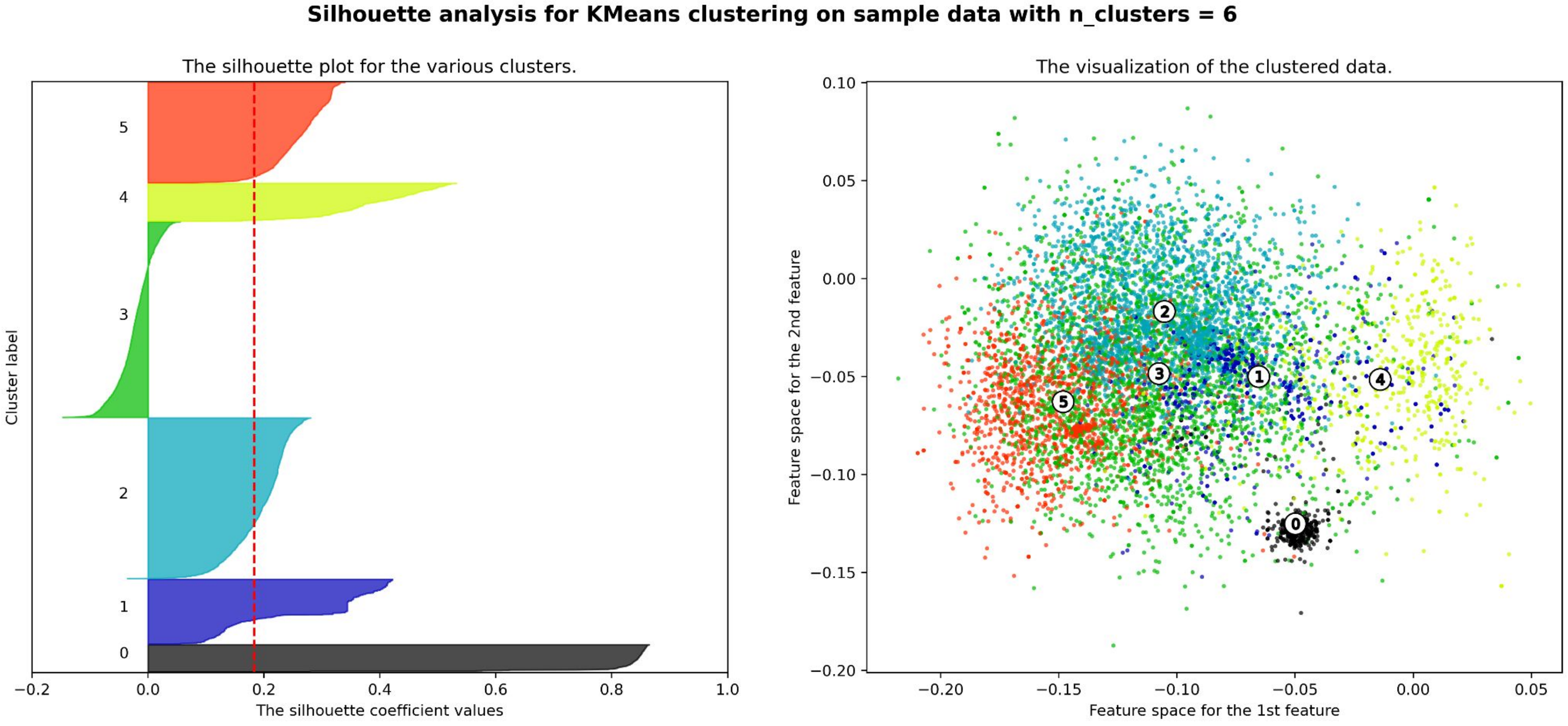
Exploratory Data Analysis



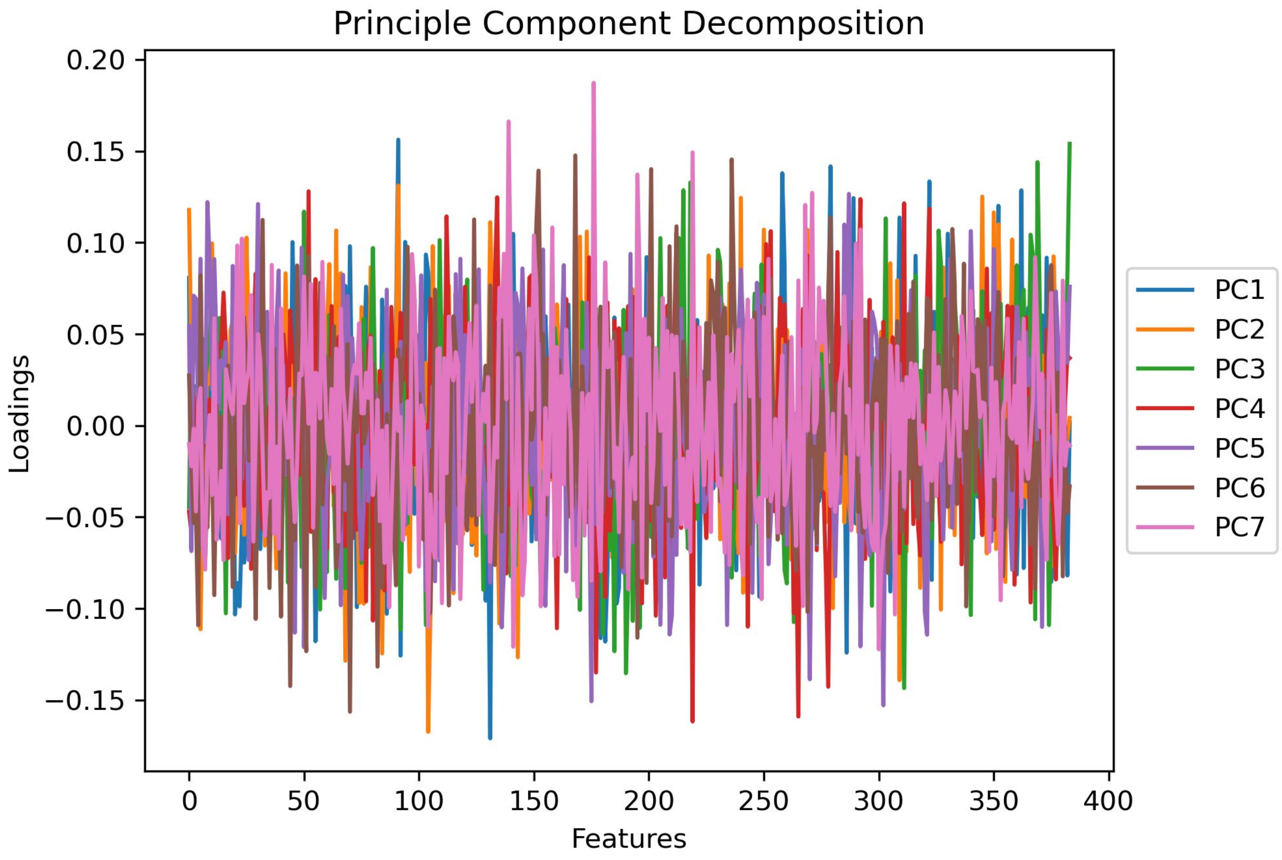


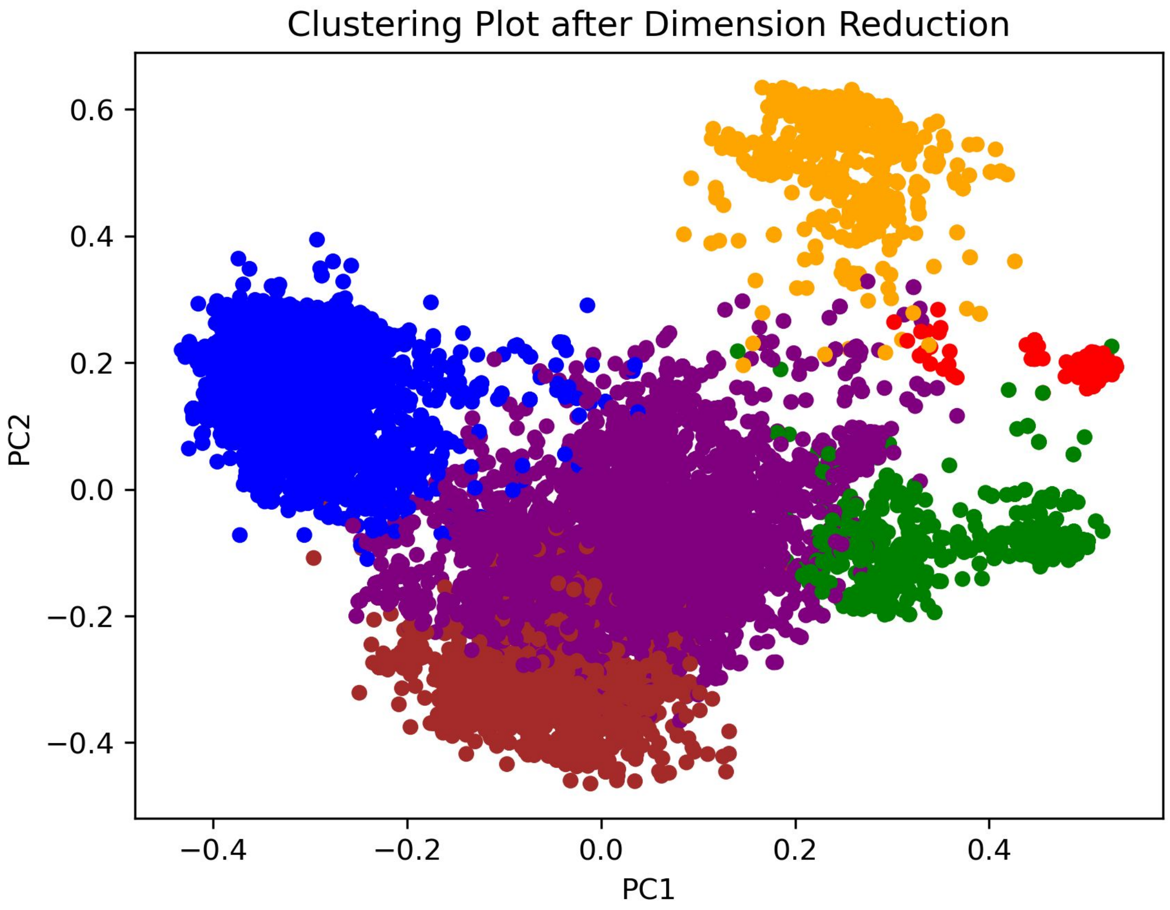
Data Visualization with K-means Clustering

Silhouette Score: $$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, \text{ if } |C_I| > 1$$



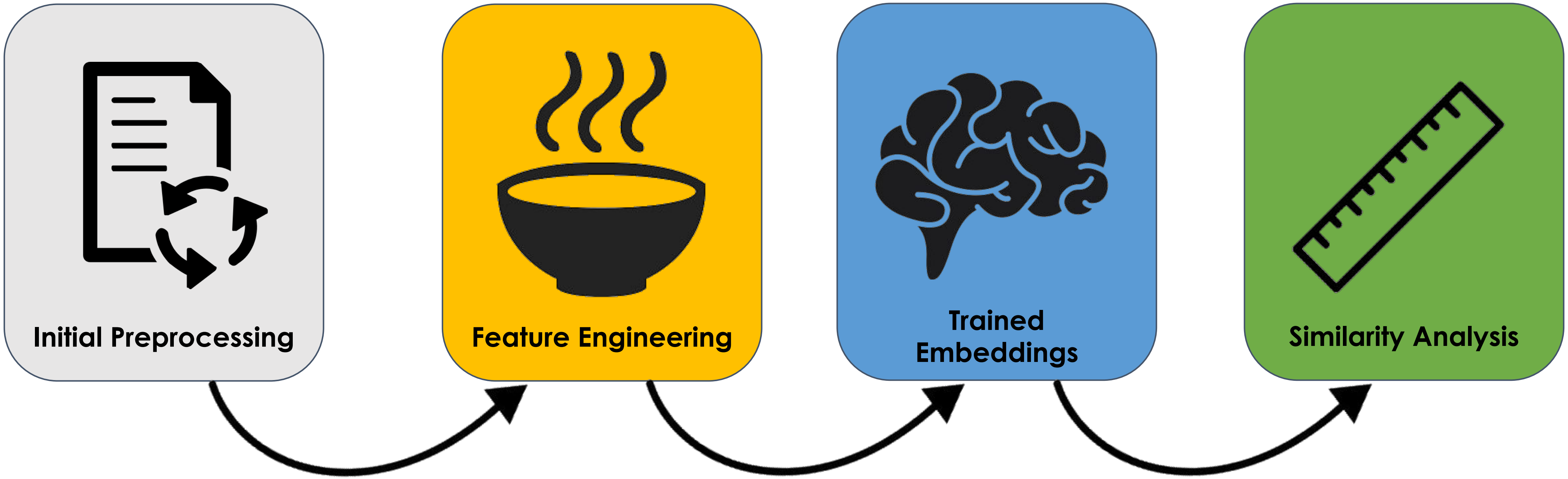
Data Visualization with K-means Clustering





AI SINGAPORE

Preprocessing



Useful information from EDA

- Features contain lists.
- Repeated information in *tags* & *datasets.*
- NaN represented by [ ].
- 140k/170k rows less than 10 downloads.

Preprocessing

Initial Preprocessing

Steps:

- col.apply(str)
- lowercase & remove punctuation
- Handle exceptions [ ]
- Limit samples

Preprocessing

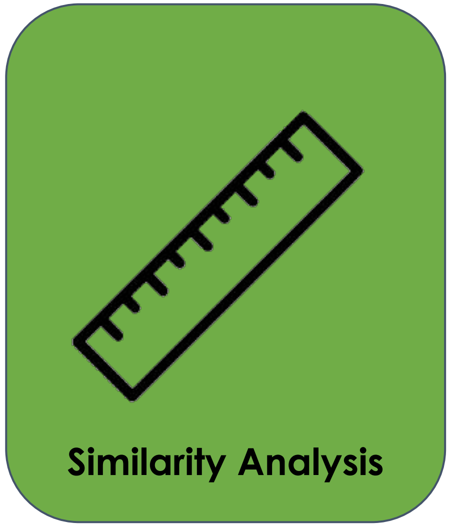
- Concatenate features.
- Create feature "*soup*".
- Essentially a corpus of information for each row

Preprocessing

import SentenceTransformer

- Pre-trained Word Embeddings
- Encode into vectors which capture the semantic meaning of corpus

Preprocessing



Baseline Modelling

- **Cosine Similarity**

$$\text{cosine similarity} = S_C(A,B) := \cos(\theta) = \frac{\mathbf{A}\cdot\mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum\limits_{i=1}^{n} A_i B_i}{\sqrt{\sum\limits_{i=1}^{n} A_i^2}\sqrt{\sum\limits_{i=1}^{n} B_i^2}},$$

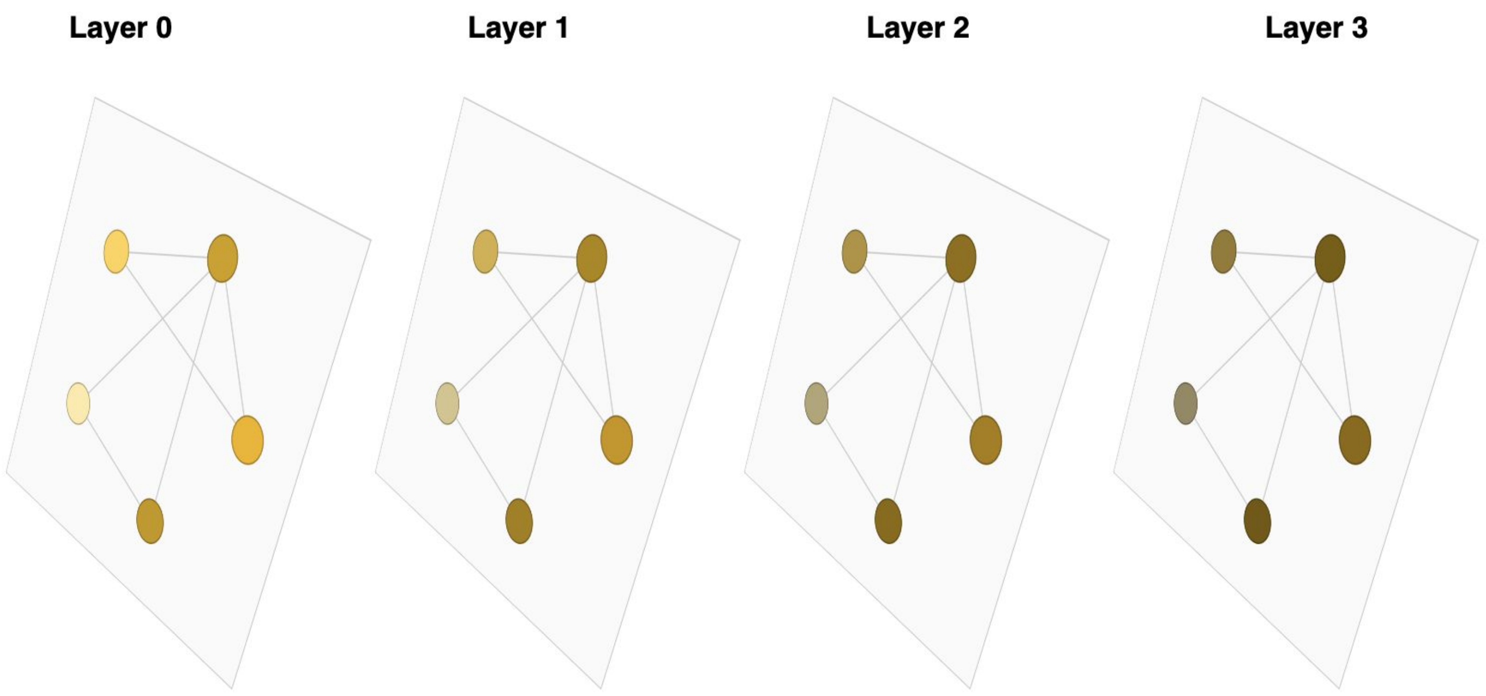
- **Jaccard Similarity**

$$J(A,B) = \frac{|A\cap B|}{|A\cup B|} = \frac{|A\cap B|}{|A| + |B| - |A\cap B|}.$$

Graph Neural Network (GNN)

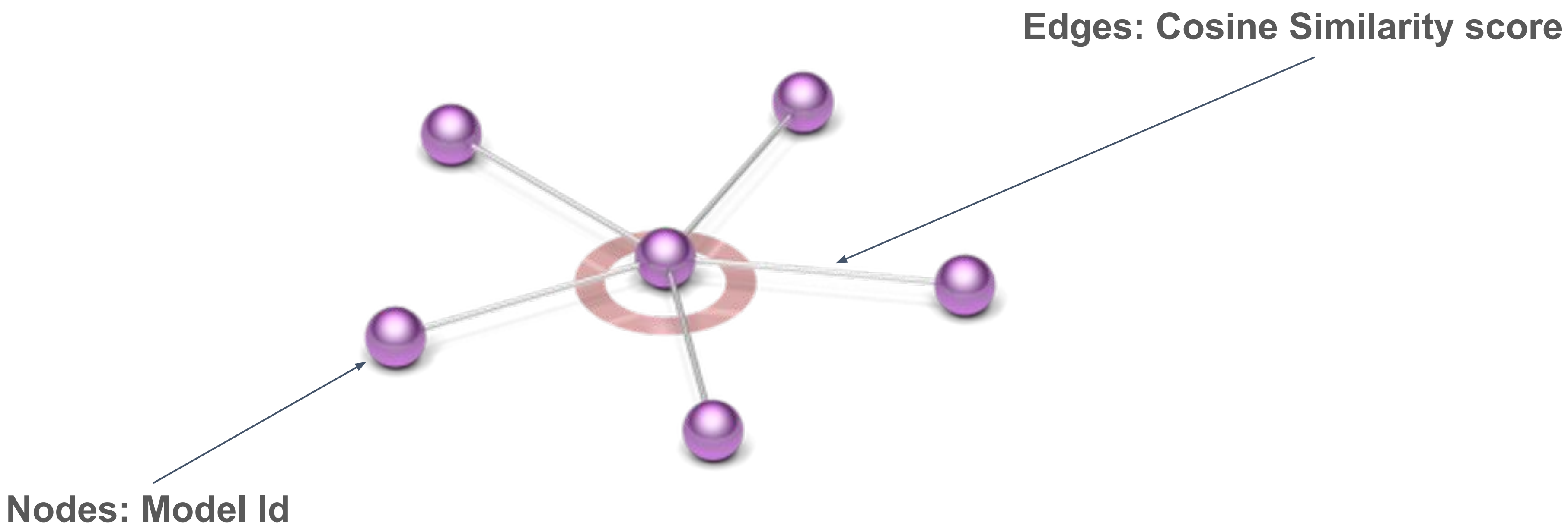
1. Overview of Graph Neural Network
2. Context of HFMRS
3. Implementation

1. Overview of Graph Neural Network

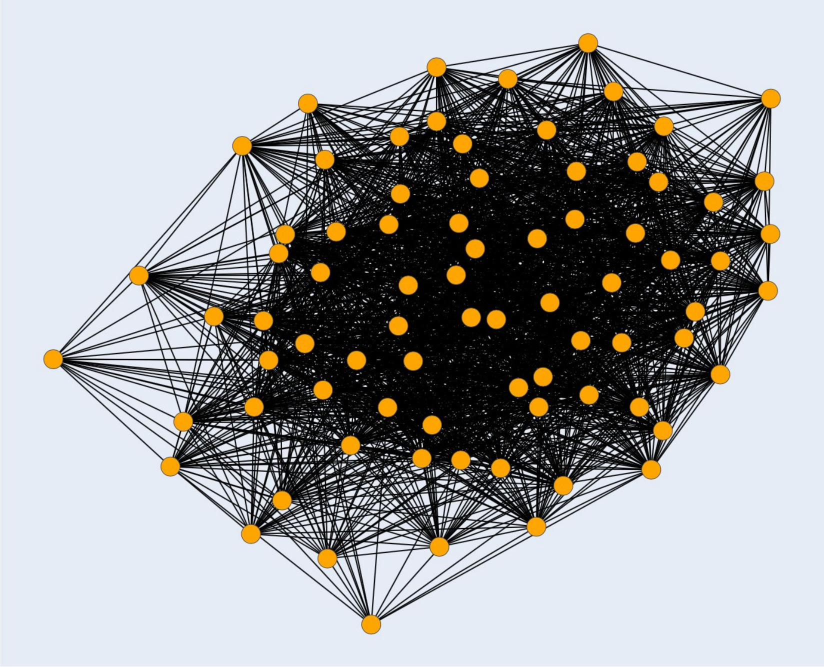


- Type of neural network which operates on graph-structured data
- Can handle complex relationship between data points represented as a graph
- Useful in domains such as Recommender System, Social Network Analysis and Bioinformatics

2. Context of HFMRS

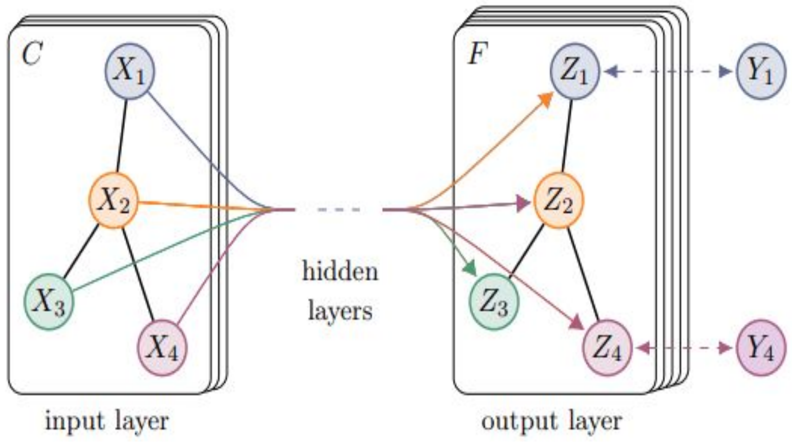


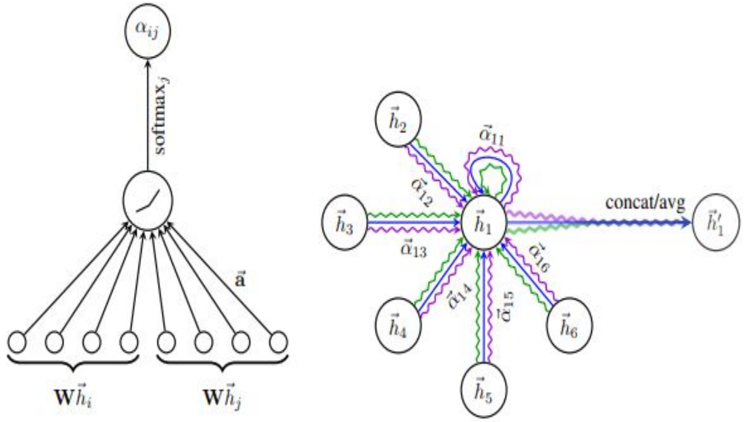
2. Context of HFMRS



- Input graph is a <u>similarity graph</u> where each node is connected to every other node in the graph through edges with corresponding edge weights (cosine similarity score)
- By leveraging information from this graph structure, our GNN model can capture interactions and relationships between each nodes to provide recommendations based on the learned representations

3. Implementation - Architecture

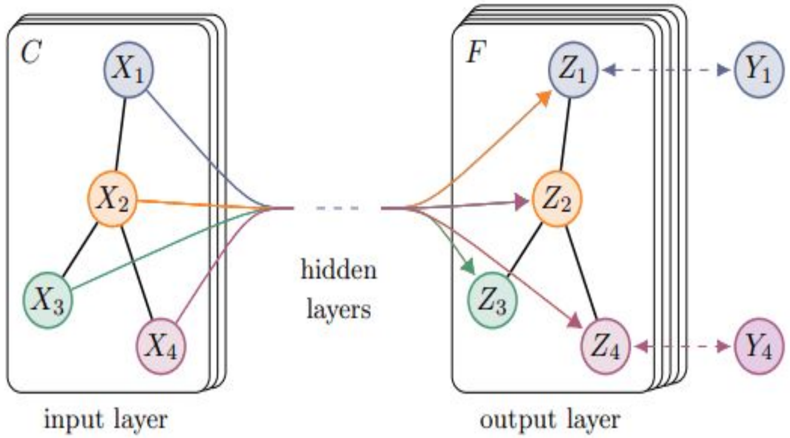




Graph Convolutional Network (GCN)

Graph Attention Network (GAT)

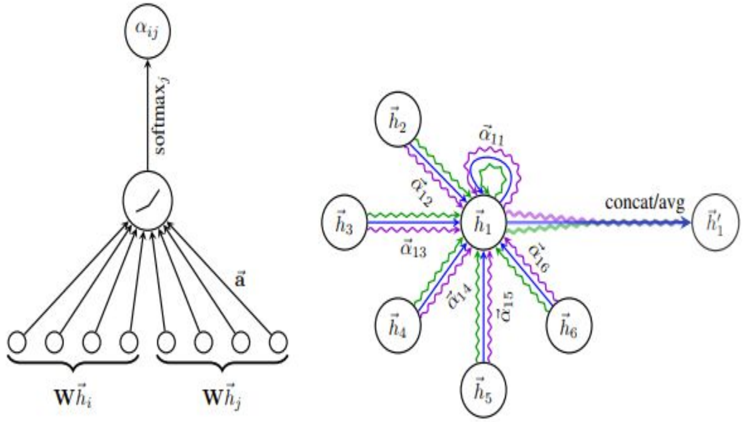
3. Implementation - Architecture



Graph Convolutional Network (GCN)

- **Work by propagation information from a node's neighbours to update the node's representation**
- **In practice, each node is presented by a vector and the vector is updated by aggregating the vectors of the node's neighbours**
- **Aggregation function can be mean/sum/weighted mean/weighted sum.**
- **Each GCN can have multiple layers with each layer updating the node presentations.**

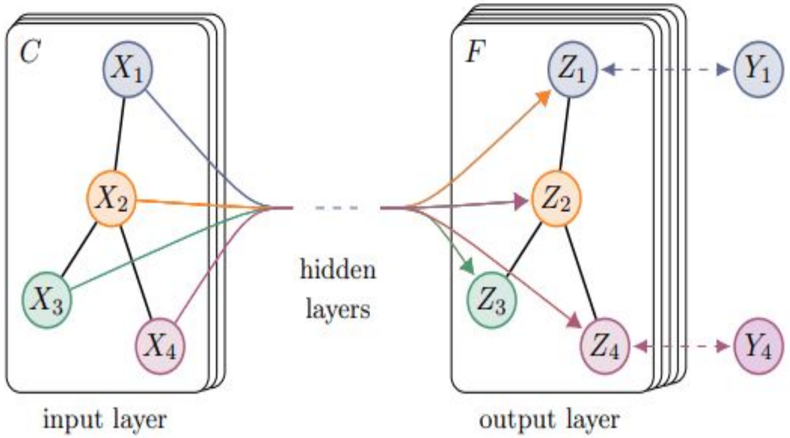
3. Implementation - Architecture

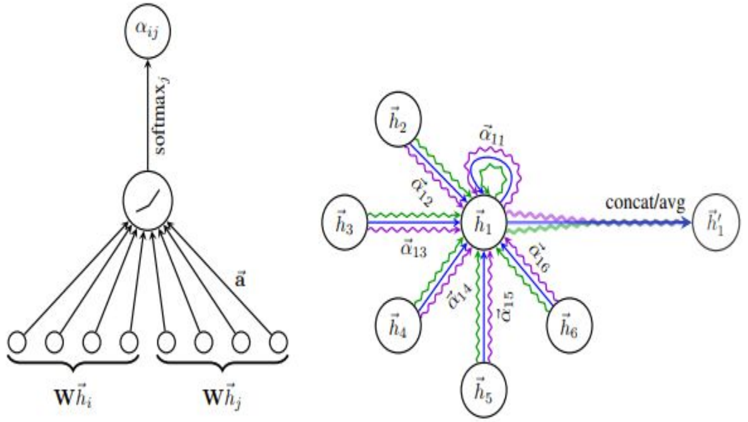


Graph Attention Network (GAT)

- **Uses attention mechanisms to weight the contribution of each neighbour node to the update of a node's representation which allows GAT to focus on the most relevant neighbour for each node**
- **In GAT, each node computes an attention coefficient for each of its neighbours based on a learned weight matrix and a non-linear activation function**
- **The attention coefficients are used to to compute a weighted sum of the neighbour vectors to update the node's representations**

3. Implementation - Architecture





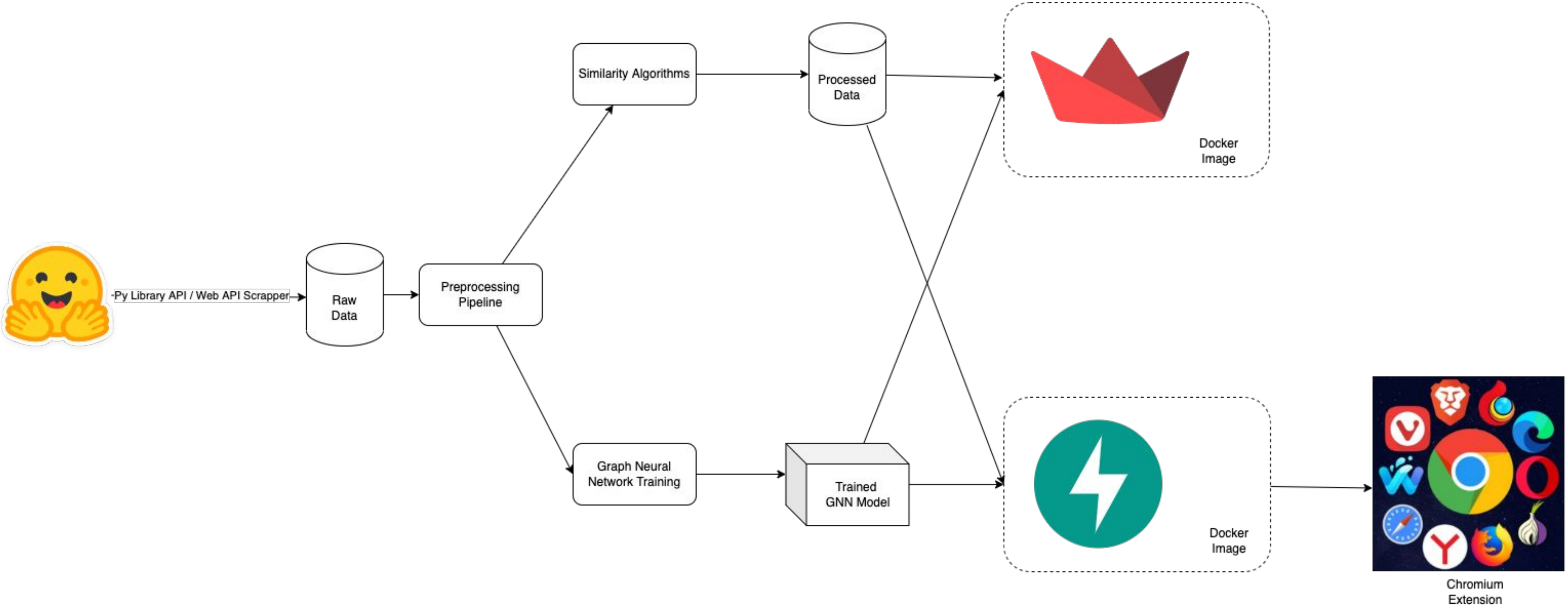
Graph Convolutional Network (GCN)

Graph Attention Network (GAT)

3. Implementation - Limitations

1. Cold start problem
    - Requires further user data such as click-through rate or interactions to train and tune the model
    - Lack of access to user data can be addressed through our web implementation
2. Limited scalability
    - Computational complexity increases as number of nodes and edges increases
    - Utilise sampling techniques to address scalability issue
    - Utilise graph analysis techniques to identify subgraphs
3. Lack of explainability
    - Learned representations and output recommendations lack transparency and explainability
    - Utilise explainability techniques such as Explainable GNNs

System Architecture



Demo

Streamlit

FastAPI

+

AI SINGAPORE

Challenges

Lack of User-Model Interactions

Lack of Universal Metrics

Lack of Gold Label Dataset

Future Work

User Management
+
Personalized
Recommendations

More
Model Tuning

Cloud
Deployment

Data
Updates

Imbalanced
Representation

AI SINGAPORE®

Conclusion

- Explored different recommender techniques for users to try
- Git clone our repository and experience it yourself!

AI SINGAPORE®

Thank you

www.aisingapore.org