WIKIPEDIA & FANDOM ARTICLE RECOMMENDER SYSTEM

A content-based recommender system that suggests similar Wikipedia or Fandom articles based on previously visited pages

Monika Dominiak 160307
Antonina Grdzielewska 160286

1. Data Collection: Crawling and Scraping

Article data (2000 articles) was collected directly from Wikipedia and Fandom Wiki using a custom spider written in Scrapy (my_spider.py).
This spider automatically follows internal article links, downloads content, and saves each page's URL, title, and cleaned text to a CSV file (data/wiki.csv or data/fandom.csv).

Key features

Article Validation	Ensures only valid article URLs are scraped
Text Extraction	Extracts <p> tag text, removes punctuation, normalizes whitespace
Minimum Word Count	Filters out short or non-content pages
Random Fallback	When stuck, fetches a random article (Special:Random) to continue exploration
CSV Export	Saves url, title, and text to csv file

Parameters

max_links	Maximum number of pages to collect
start_url	Starting article URL
min_word_count	Minimum number of words to save a page
allow_random	Enables fetching random pages when stuck
user_source_type	Manually sets source to "wiki" or "fandom" for fetching random pages

2. Text Preprocessing

TF-IDF similarity

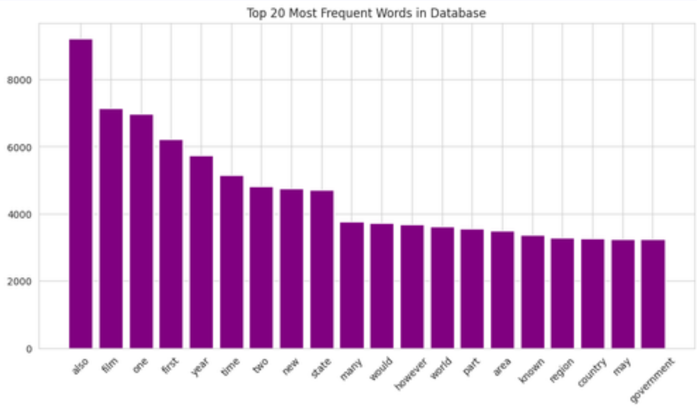
- Text is preprocessed before vectorization:

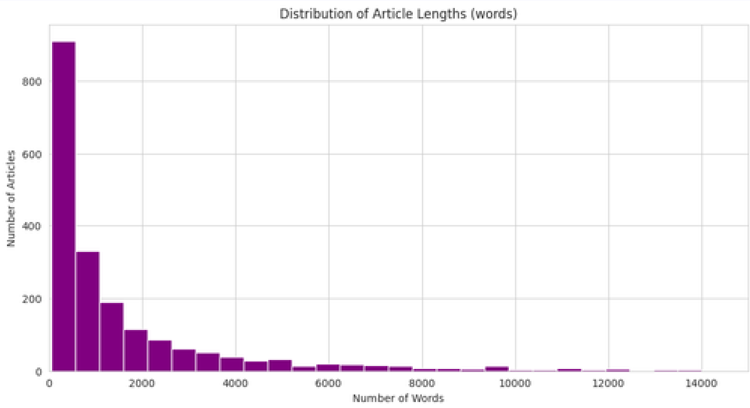
Steps applied using NLTK and regex:
1. Lowercasing
2. Stopword removal (stopwords.words('english'))
3. Filtering non-alphabetic tokens
4. Lemmatization and stemming comparison

Embedding similarity

- Raw text is generally used without preprocessing.

Database statistics





3. Processing User Input and Selecting Diverse Articles

When a user provides a list of previously visited articles, the system needs to process them in a way that maximizes recommendation quality while keeping computation efficient. This step involves scraping, preprocessing, and selecting the most diverse subset of articles.

- The system first scrapes the user-provided URLs using a simplified scraping function (run_scraper_simple).
- For each URL:
- The system checks whether it is a valid article (is_article).
- It extracts the text and title.
- Articles below a minimum word count (e.g., 10 words) are discarded.
- The result is a dataframe of valid user articles, ready for preprocessing.

Similar to the main article database, each user-provided article undergoes:
- Lowercasing
- Stopword removal
- Lemmatization using WordNetLemmatizer

This ensures that user articles are in the same clean, normalized format as the database articles.

Problem:
Users may provide too many articles. Including all articles could:
- Slow down similarity computations
- Over-represent some topics if multiple similar articles are provided

Solution:
Select a subset of the most diverse articles based on their semantic embeddings.
1. Embed articles: Convert all user articles into dense sentence embeddings using SentenceTransformer.
2. Cluster embeddings: Apply KMeans clustering to group articles into top_n clusters (20).
3. Select representatives: For each cluster, pick the article closest to the cluster centroid.
4. Use these diverse articles as input for the recommendation engine.

4. Feature Representation

In this project, we use two main types of features: TF-IDF vectors (lexical features) and sentence embeddings (semantic features), combined into a hybrid representation.

TF-IDF Representation

- Implements TfidfVectorizer from scikit-learn.
- Captures word-level importance relative to the corpus.
- All articles from a user are combined into one long, normalized, and lemmatized text
- Highlights important words in each article.
- Reduces the weight of common words.
- Works well for matching articles that share similar vocabulary.

Sentence Embeddings

- Uses pretrained SentenceTransformer (all-MiniLM-L6-v2).
- Encodes entire documents into dense vectors representing semantic meaning.
- User input (multiple articles) is combined as a weighted average (later articles are weighted slightly higher for recency).
- Captures semantic similarity beyond exact word matches.
- Handles synonyms, paraphrasing, and conceptual similarity.
- Essential for recommending articles from different topics that are conceptually related.

Hybrid Representation

Both methods are combined to balance textual and conceptual similarity:
Final Score = 0.4 * tfidf_sim + 0.6 * embedding_sim

5. Recommendation algorithm

To make the recommendation process more personalized, the system can take into account the user's reading history — i.e., the articles that the user has already viewed or interacted with. The most recent views receive higher weights to better capture current interests. This is a function parameter, so user can choose which way he wants to do this. We were checking everything taking weighted history into consideration.
This history is used as a context vector that influences which new articles are recommended. The most recently seen articles are the most important.

- We take into consideration only articles that user did not see before
- Preprocess text using the same pipeline.
- Compute TF-IDF and embedding representations.
- Calculate similarity with all articles in the dataset.
- Rank by hybrid score
- Return articles above min_similarity threshold (0.9) or if there are no such articles, return top 10 results.

Example 1

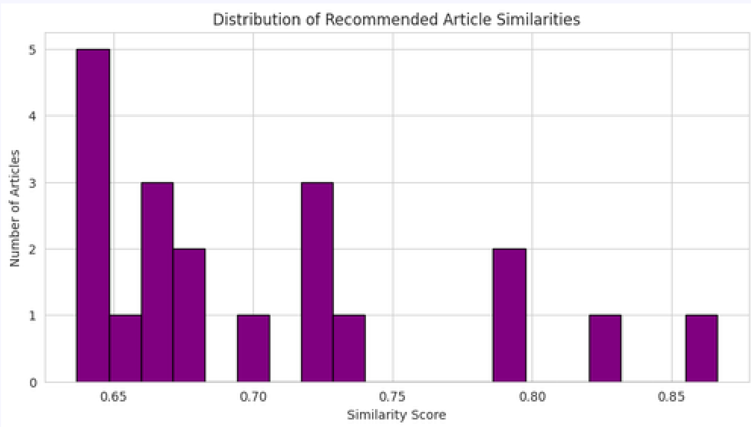
User urls:
https://en.wikipedia.org/wiki/Monkey
https://en.wikipedia.org/wiki/Tiger

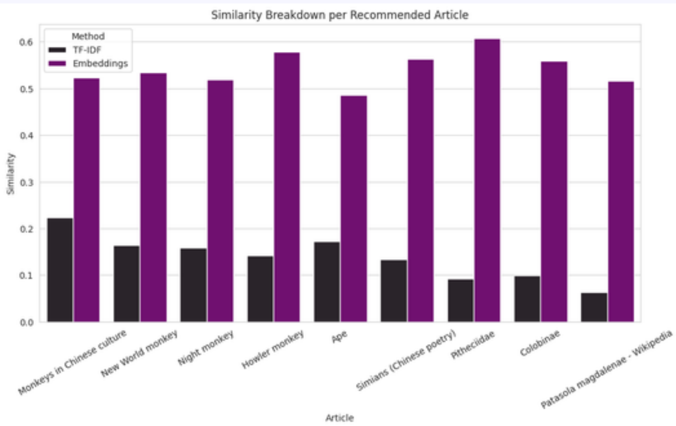
Recommendations:

1	https://en.wikipedia.org/wiki/Monkeys_in_Chinese_culture	0.866419
2	https://en.wikipedia.org/wiki/Platyrrhini	0.822383
3	https://en.wikipedia.org/wiki/Aotidae	0.797528
4	https://en.wikipedia.org/wiki/Howler_monkey	0.79152
5	https://en.wikipedia.org/wiki/Apes	0.730016
6	https://en.wikipedia.org/wiki/Simians_(Chinese_poetry)	0.722337
7	https://en.wikipedia.org/w/index.php?title=Simians_(Chinese_poetry)&oldid=1310610400	0.722337
8	https://en.wikipedia.org/wiki/Pitheciidae	0.718461
9	https://en.wikipedia.org/wiki/Colobinae	0.694462
10	https://en.wikipedia.org/wiki/Patasola_magdalenae	0.680305

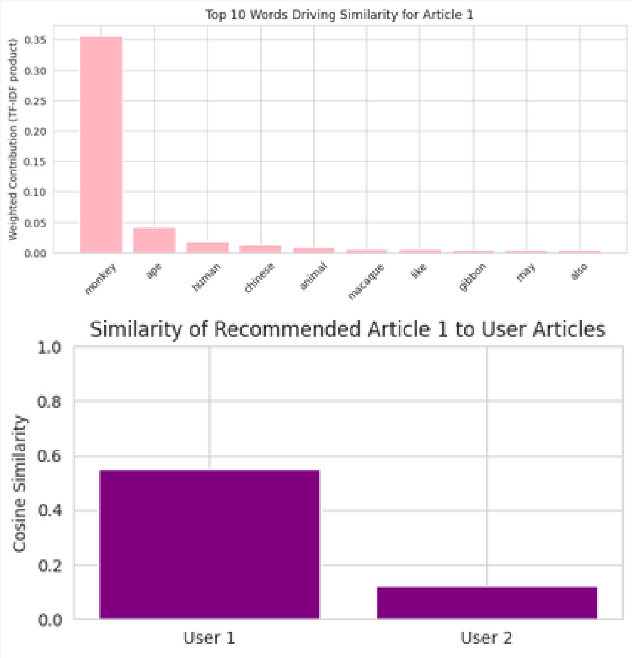
For comparison, order_importance = False

No.	title	URL	Similarity
1	Kangchenjunga	https://en.wikipedia.org/wiki/Kangchenjunga	0.793849
2	Ronald Tompkins	https://disney.fandom.com/wiki/Ronald_Tompkins	0.710561
3	Nilgiri Mountains	https://en.wikipedia.org/wiki/Nilgiri_Mountains	0.681216
4	Cricket	https://en.wikipedia.org/wiki/Cricket	0.668438
5	Idiyappam	https://en.wikipedia.org/wiki/String_hoppers	0.662858
6	Rasgulla	https://en.wikipedia.org/wiki/Rasgulla	0.640722
7	Kabaddi	https://en.wikipedia.org/wiki/Kabaddi	0.631143
8	Sulaiman Mountains	https://en.wikipedia.org/wiki/Sulaiman_Mountains	0.619976
9	Kashmir Valley	https://en.wikipedia.org/wiki/Kashmir_Valley	0.614973
10	Geography of West Bengal	https://en.wikipedia.org/wiki/Geography_of_West_Bengal	0.613574

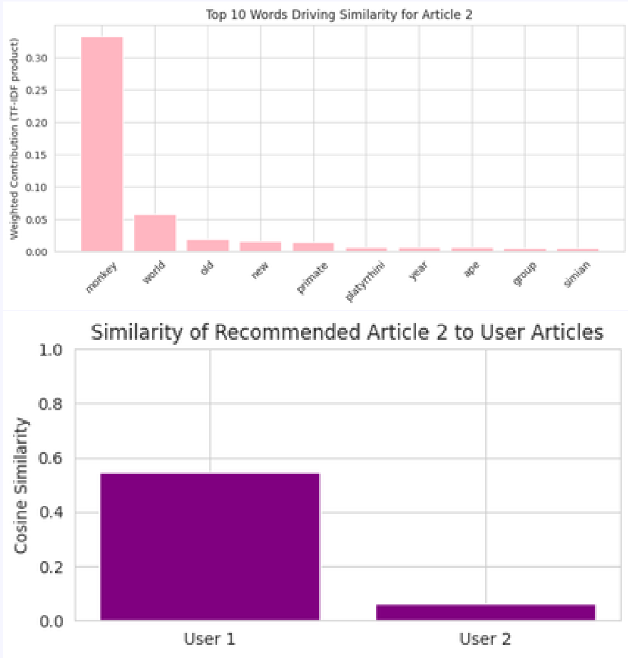




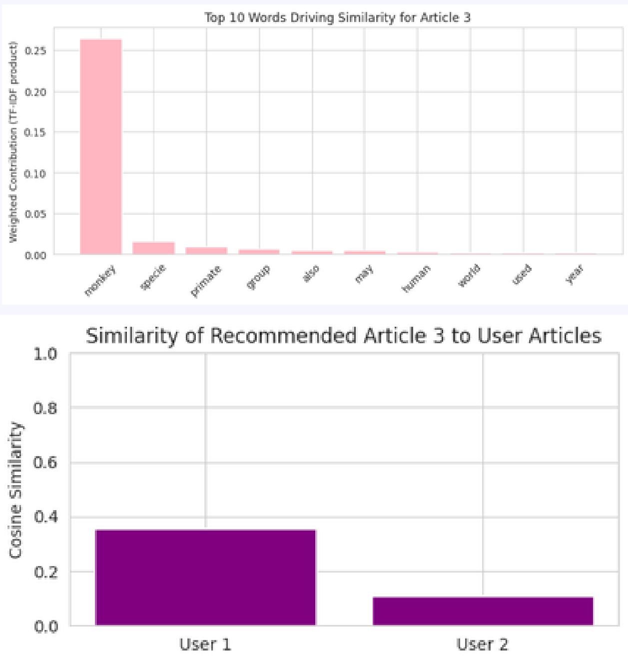
Article 1 breakdown



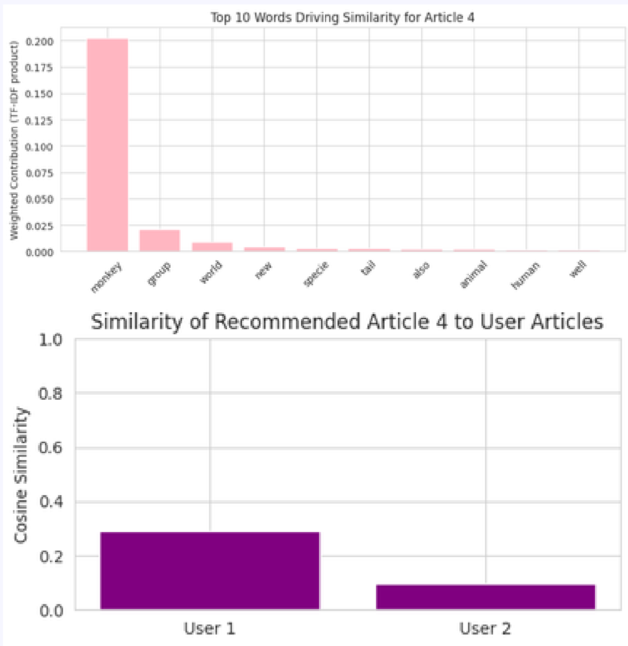
Article 2 breakdown



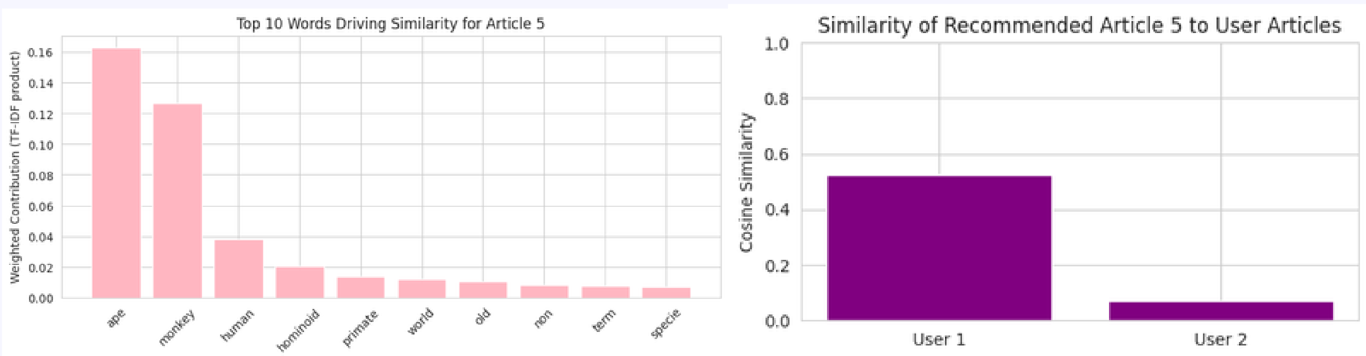
Article 3 breakdown

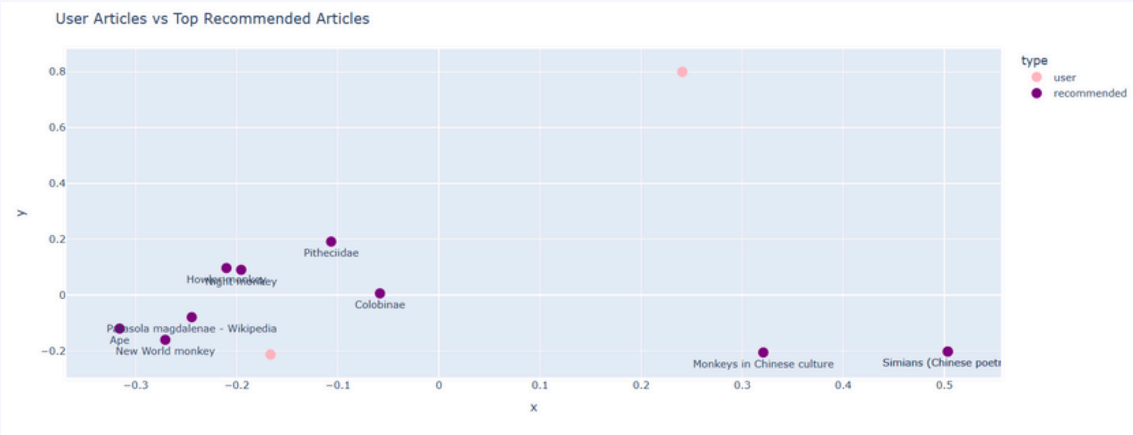


Article 4 breakdown



Article 5 breakdown





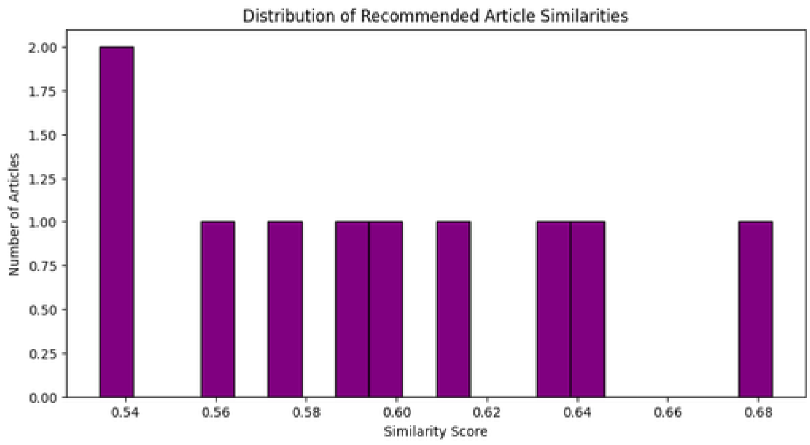
We can see that most of recommended articles are near the 'Monkey' article, that means they are related

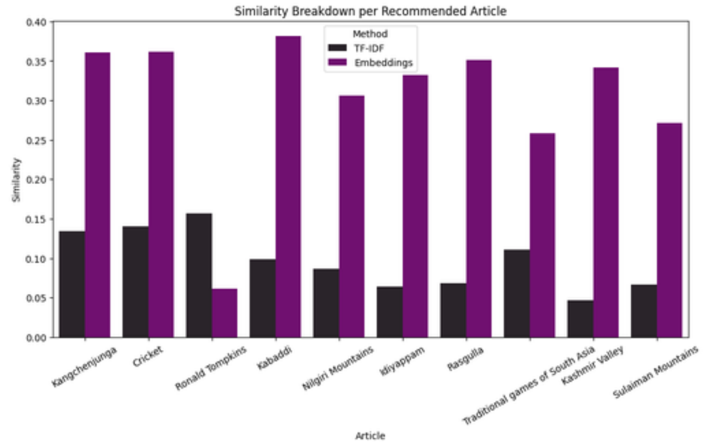
Example 2 - more diverse urls

User urls:
https://en.wikipedia.org/wiki/Pozna%C5%84_University_of_Technology
https://harrypotter.fandom.com/wiki/Hermione_Granger
https://en.wikipedia.org/wiki/Mount_Everest
https://en.wikipedia.org/wiki/Doughnut
https://en.wikipedia.org/wiki/Volleyball

Recommendations:

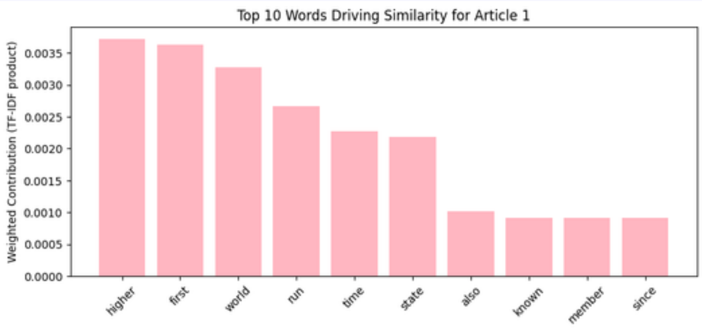
1	https://en.wikipedia.org/wiki/Kangchenjunga	0.683098
2	https://en.wikipedia.org/wiki/Cricket	0.642806
3	https://disney.fandom.com/wiki/Ronald_Tompkins	0.638422
4	https://en.wikipedia.org/wiki/Kabaddi	0.610859
5	https://en.wikipedia.org/wiki/Nilgiri_Mountains	0.59479
6	https://en.wikipedia.org/wiki/String_hoppers	0.593738
7	https://en.wikipedia.org/wiki/Rasgulla	0.574868
8	https://en.wikipedia.org/wiki/Traditional_games_of_South_Asia	0.562681
9	https://en.wikipedia.org/wiki/Kashmir_Valley	0.536844
10	https://en.wikipedia.org/wiki/Sulaiman_Mountains	0.534404

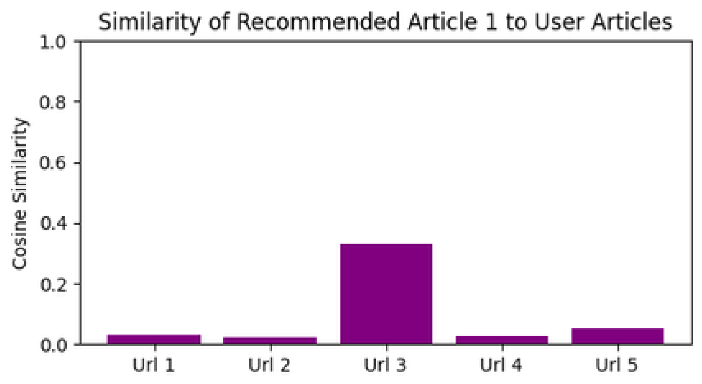

Distribution of Recommended Article Similarities
Number of Articles
Similarity Score
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
0.54
0.56
0.58
0.60
0.62
0.64
0.66
0.68


Similarity Breakdown per Recommended Article
Method
TF-IDF
Embeddings
Similarity
Article
Kangchenjunga
Cricket
Ronald Tompkins
Kabaddi
Nilgiri Mountains
Idiyappam
Rasgulla
Traditional games of South Asia
Kashmir Valley
Sulaiman Mountains

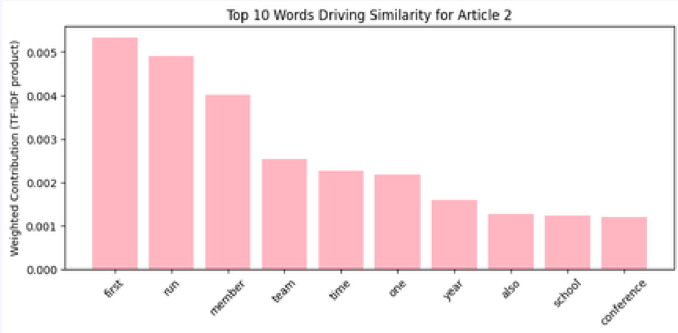

User Articles vs Top Recommended Articles
type
user
recommended
Kabaddi
Traditional games of South Asia
Cricket
Kashmir Valley
Sulaiman Mountains
Nilgiri Mountains
Kangchenjunga
Rasgulla
Idiyappam
Ronald Tompkins
x
y

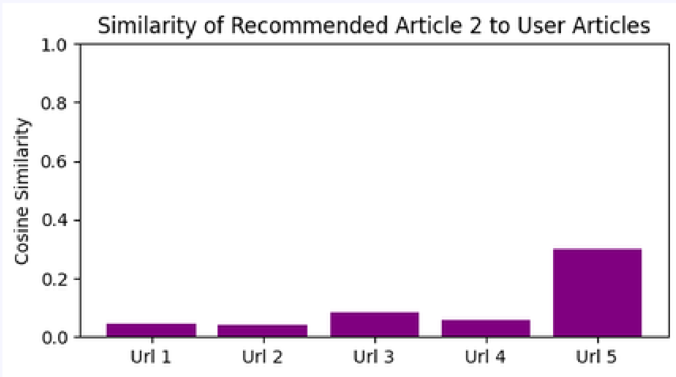
Article 1 breakdown



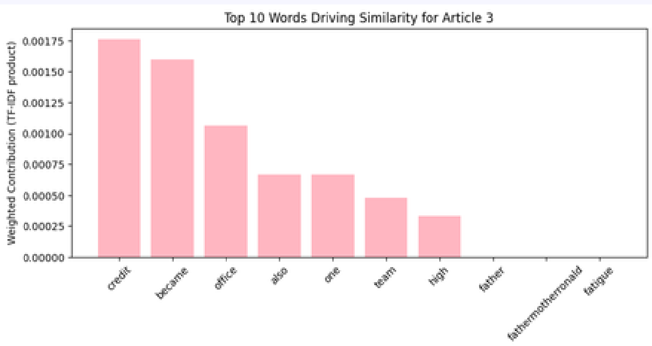


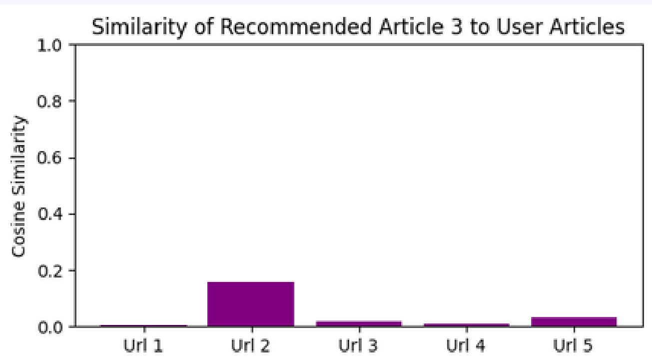
Article 2 breakdown



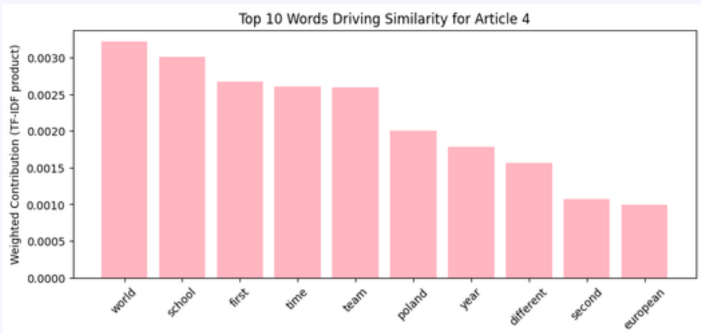


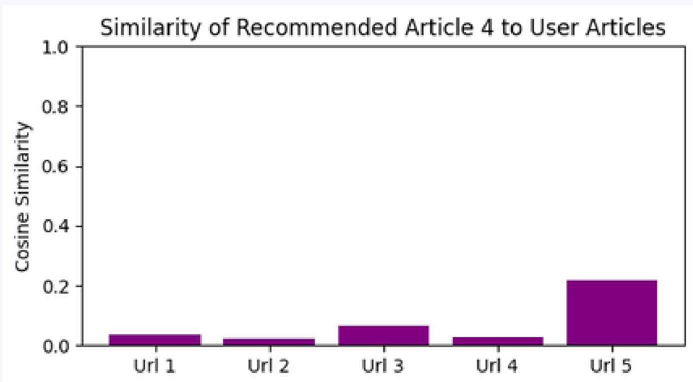
Article 3 breakdown



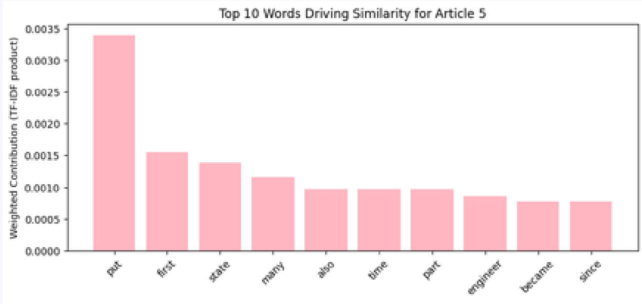


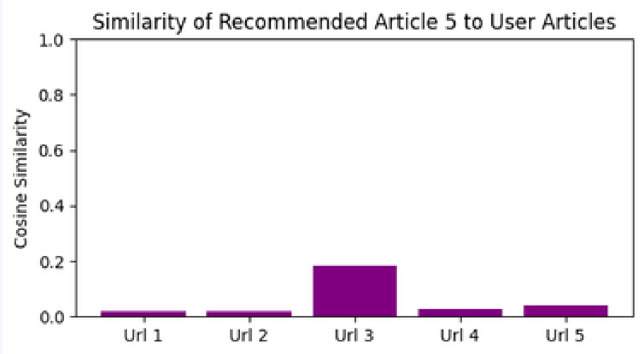
Article 4 breakdown





Article 5 breakdown





Example 3 - very domain-specific

User urls:
https://en.wikipedia.org/wiki/COVID-19_pandemic
https://en.wikipedia.org/wiki/Vaccine
https://en.wikipedia.org/wiki/Virus

Recommendations:

1	https://en.wikipedia.org/wiki/Virus_classification	1

The result for one article is above our threshold, so it is highly recommended