Maximizing E-Commerce Efficiency: Advanced Search Engine, Precise Product Categorization and Smart Recommendation System

THESIS SUBMITTED TO

Symbiosis Institute of Geoinformatics

FOR PARTIAL FULFILLMENT OF THE M. Sc. DEGREE

By

Kinjal Bandopadhyay

(Batch 2022-2024 / PRN 22070243028)

Symbiosis Institute of Geoinformatics

Symbiosis International (Deemed University) 5th Floor, Atur Centre, Gokhale Cross Road, Model Colony, Pune - 411016

CERTIFICATE

Certified that this thesis titled "Maximizing E-Commerce Efficiency: Advanced Search Engine, Precise Product Categorization and Smart Recommendation System" is a bonafide work done by Mr. Kinjal Bandopadhyay(22070243028) at Symbiosis Institute of Geoinformatics, under our supervision.

Supervisor, Internal

Mr. Sahil Shah

Symbiosis Institute of Geoinformatics

Index

ACKNOWLEGEMENT

I extend my heartfelt gratitude to my internal supervisor, Mr. Sahil Shah, for his unwavering support, invaluable guidance, and insightful inputs throughout this project. His expertise and encouragement have been instrumental in shaping the direction and success of this research endeavour.

I am deeply indebted to Symbiosis Institute of Geoinformatics for providing me with the opportunity to pursue my M. Sc. degree and for furnishing the necessary resources and conducive environment for conducting this study. The academic rigor and research facilities offered by the institution have been integral to the completion of this project.

I would like to express my sincere appreciation to all the faculty members of Symbiosis Institute of Geoinformatics whose expertise, mentorship, and dedication have enriched my academic journey and broadened my understanding of the subject matter.

Lastly, but certainly not least, I extend my heartfelt gratitude to my family for their unwavering support, encouragement, and understanding throughout this endeavour. Their love, patience, and belief in me have been a constant source of inspiration and motivation.

Kinjal Bandopadhyay

M. Sc. DSSA (Batch 2022-2024)

PRN: 22070243028

LIST OF FIGURES

PREFACE

This project report outlines the development and implementation of advanced systems for an e-commerce platform, focusing on personalized recommendations, sophisticated search capabilities, and efficient product categorization using deep learning techniques.

E-commerce platforms have transformed the way people shop, offering convenience, variety, and accessibility. However, the rapid increase in online shopping activity has also presented new challenges, particularly in providing a personalized and seamless user experience. Traditional methods for recommendation, search, and categorization often fall short in addressing the complex needs of modern e-commerce environments. This project aims to bridge this gap by leveraging the power of deep learning to create more intelligent and responsive systems.

The motivation behind this project stems from the desire to enhance user satisfaction and engagement by providing personalized product suggestions, improving search accuracy, and automating product categorization. These enhancements not only improve the user experience but also drive sales and operational efficiency for the e-commerce platform.Throughout this project, various deep learning models and techniques have been explored and implemented. From using TF-IDF vectors for content-based recommendations to employing word embeddings for understanding search queries, each component of the project is designed to leverage cutting-edge technology. Additionally, the integration of advanced information retrieval models and the support for multilingual queries demonstrate the project's commitment to inclusivity and precision.

The development process involved extensive research, experimentation, and validation to ensure that the systems not only meet but exceed the performance requirements. The results highlight significant improvements in recommendation accuracy, search relevance, and categorization precision, underscoring the effectiveness of deep learning in transforming e-commerce experiences.This report is a culmination of the collaborative efforts of a dedicated team committed to pushing the boundaries of what is possible in e-commerce technology. It provides detailed insights into the methodologies, challenges, and successes encountered throughout the project. We hope that this work will serve as a valuable resource for further advancements in the field and inspire future innovations in e-commerce.

ABSTRACT

This project aims to enhance an e-commerce platform by designing and implementing a state-of-the-art recommendation system, an advanced search engine, and an efficient product categorization mechanism using deep learning techniques. Key objectives include creating a content-based recommendation system utilizing TF-IDF vectors to deliver personalized product suggestions, developing an advanced search engine employing word embeddings for contextual relevance, and integrating multilingual capabilities through Deep Text Search.

The recommendation system analyzes user preferences and product attributes, resulting in tailored suggestions that improve user engagement and satisfaction. The search engine, leveraging models like PyTerrier and Sentence-BERT, ensures contextually accurate search results, enhancing the user experience. Multilingual support broadens accessibility, catering to a diverse user base.

Additionally, we developed predictive models for product categorization, significantly improving efficiency and accuracy. The Fine-Tuned BERT Base model achieved a high accuracy of 98.62%, outperforming other models like RNN-LSTM and CNN-LSTM. These advancements have created a robust, user-friendly platform capable of handling large data volumes and peak usage times without compromising performance.

Overall, the implementation of these advanced techniques has resulted in a comprehensive e-commerce system that enhances user experience, drives engagement, and boosts sales, effectively catering to a global audience.

INTRODUCTION

In the modern digital age, the shopping experience has been revolutionized by e-commerce, allowing consumers to make purchases anytime and anywhere. This transformation has led to a rapid expansion of e-commerce businesses, each striving to offer the best user experience to attract and retain customers. Central to this enhanced user experience are recommendation systems and advanced search engines.

Recommendation systems are widely used across various sectors, including entertainment and online retail, to provide personalized suggestions based on users' previous interactions. This personalization not only improves customer satisfaction but also boosts sales by recommending products that users are more likely to purchase. For instance, Sharma and Yadav (2020) developed an item-based collaborative filtering method for movie recommendations, and Balush et al. (2021) investigated the use of intelligent search, NLP, and machine learning in recommendation systems. On the other hand, advanced search engines play a crucial role in delivering accurate and relevant search results, which helps retain users by preventing poor search experiences. Jiang et al. (2020) enhanced BERT for e-commerce search ranking, and Zhou et al. (2023) utilized large language models to improve product descriptions in e-commerce.

This project aims to develop an integrated solution combining a recommendation system and an advanced search engine specifically for an e-commerce platform. By employing deep learning techniques rather than traditional word comparison methods, we seek to achieve superior performance in both recommendation and search functionalities. This involves leveraging state-of-the-art models like PyTerrier and Sentence-BERT for information retrieval, ensuring the system effectively handles user queries. Fan et al. (2024) explored the impact of large language models on recommender systems, emphasizing the potential of these advanced techniques.

Our recommendation system is based on a content-based approach using Term Frequency-Inverse Document Frequency (TF-IDF) vectors. This method transforms product descriptions into numerical data, enabling the system to compute similarity scores and recommend products with similar attributes. For the search engine component, word embeddings are utilized to understand the context and significance of search terms, moving beyond simple string matching to deliver more relevant results. The Word2Vec model, a popular method for generating word embeddings, will be used to capture the semantic relationships between words, thereby improving search result accuracy. Kumar and Sarkar (2022) introduced ListBERT for ranking e-commerce products, demonstrating the effectiveness of advanced models in enhancing search outcomes.

Additionally, the project addresses the challenge of multilingual search, which is essential for serving the diverse linguistic user base of e-commerce platforms. By incorporating Deep Text Search, the search engine will support multiple languages, thus enhancing accessibility and user satisfaction across different regions. Yao et al. (2023) improved domain-specific recommendations with knowledge plugins, highlighting the importance of specialized approaches in multilingual contexts.

To sum up, this project integrates advanced deep learning models to develop a sophisticated recommendation system and a robust search engine for e-commerce platforms. By focusing on personalization and relevance, we aim to significantly enhance the user experience, thereby increasing user engagement and driving sales.

OBJECTIVES

The main objective of this project is to design and implement a state-of-the-art recommendation system, an advanced search engine, and an efficient product categorization mechanism for an e-commerce platform, leveraging deep learning techniques to significantly enhance user experience and boost sales.

- Firstly, the project aims to create a content-based recommendation system. This system will utilize Term Frequency-Inverse Document Frequency (TF-IDF) vectors to analyse product descriptions. By calculating similarity scores based on these vectors, the system will recommend products with attributes similar to those the user has shown interest in, thereby delivering personalized product suggestions. This method ensures that users receive recommendations closely matching their preferences, thereby improving their overall shopping experience.
- Secondly, the project focuses on developing an advanced product search engine. Unlike traditional search engines that rely on simple string matching, this search engine will use word embeddings to understand the context and semantics of search queries. By employing models such as Word2Vec, the system will capture the meaning and context of search terms, ensuring that the search results are contextually relevant and accurate. This will help in providing users with the most pertinent products based on their search queries, thus enhancing their satisfaction and reducing the likelihood of churn.
- Another critical objective is to integrate deep learning models to improve the performance of both the recommendation system and the search engine. To improve the precision and relevance of search and recommendation outcomes, advanced information retrieval models such as PyTerrier and Sentence-BERT will be utilized. This guarantees users a smooth and error-resistant experience.
- Supporting multilingual capabilities is also a significant goal of this project. By implementing Deep Text Search, the search engine will be able to handle queries in multiple languages. This feature is essential for catering to a diverse user base, improving accessibility, and ensuring that users from different linguistic backgrounds can effectively use the platform.
- A major challenge in e-commerce is the categorization of products, as there are billions of items, and manually tagging each one is inefficient and costly. This project aims to

solve this problem by building a predictive model to categorize products in an e-commerce dataset. Product categorization is a supervised classification problem where the categories are the target classes, and the features are the words extracted from the product descriptions or images. By using state-of-the-art machine learning and deep learning techniques, the goal is to classify product categories with high precision, thus saving time and reducing operational costs.

To sum up, optimizing system performance is a crucial objective. The recommendation, search, and categorization systems must be efficient and scalable, capable of handling large volumes of data and user queries without compromising on performance. This involves ensuring that the systems maintain high performance under varying loads, providing a smooth and responsive user experience even during peak usage times. By focusing on these objectives, the project aims to enhance the overall user experience, leading to increased engagement and higher sales.

The project scope encompasses the development of several key components aimed at enhancing the functionality and user experience of an e-commerce platform. Firstly, a content-based recommendation system will be created, utilizing TF-IDF vectors to offer personalized product suggestions based on user preferences. Secondly, an advanced product search engine will be implemented, leveraging word embeddings like Word2Vec to achieve contextual search accuracy, ensuring that search results are relevant and precise. Additionally, deep learning models such as PyTerrier and Sentence-BERT will be integrated to further enhance recommendation and search accuracy. The project will also implement Deep Text Search to support multilingual queries, improving accessibility for a diverse user base. Furthermore, automation of product categorization using predictive models will be pursued to classify products with high precision, reducing manual effort and operational costs. Finally, system performance will be optimized to ensure efficiency, scalability, and responsiveness, particularly during peak usage times, to provide a seamless and satisfying user experience.

<u>LITERATURE REVIEW</u>

In Meshal Alfarhood, Jianlin Cheng(2018), the research introduces DeepHCF, a model for rating prediction in recommender systems. By employing joint training with a multi-layer perceptron (MLP) and a convolutional neural network (CNN), the model aims to minimize prediction error and enhance system performance. Evaluated on datasets such as MovieLens-1M, Amazon Instant Video, Amazon Android Apps, and Amazon Digital Music, DeepHCF achieves notable performance improvements, with MAE/RMSE scores of 0.674/0.867, 0.688/0.994, 0.896/1.180, and 0.520/0.830 respectively.

In Poonam Sharma, Lokesh Yadav(2020), the paper presents a system that categorizes users with similar interests using AI-based algorithms for personalized content suggestions. Utilizing item-based collaborative filtering, the system enhances recommendation accuracy and relevance by considering user profiles, search history, and demographic traits. Applied to the Netflix movies and TV shows dataset, the system integrates clustering, similarity, and classification techniques, improving recommendation precision and reducing Mean Absolute Error.

In Yunjiang Jiang et al. (2020)., the study explores the integration of BERT into E-commerce non-default search ranking through a two-stage ranking approach. The approach includes implementing parallel prediction on multiple GPU hosts and introducing a C++-based BERT tokenizer. Using a dataset with 899k documents and 150 queries, producing 150 million query/item pairs, the study achieves a 0.37% improvement with WholeWord Masking BERT-Large and enhanced F1 scores (0.8293), securing 1st place in the supervised phase and 2nd in the final phase.

In Yue Shang et al. (2020), the paper introduces a learning framework employing transfer learning and knowledge distillation to tackle the challenge of limited human-labeled data in industrial applications. Using four in-house datasets, the study achieves 97% test accuracy with significantly lower serving costs. A/B tests demonstrate improvements in CVR and GMV metrics, and the framework effectively imitates teacher models with simple feed-forward student networks.

In Lakshya Kumar, Sagnik Sarkar(2022)., the paper introduces an innovative approach for end-to-end training of a transformer-based model using a listwise surrogate loss function. The study compares the efficacy of listwise and pairwise loss functions based on the NDCG metric and emphasizes model application in low-latency e-commerce search with knowledge distillation. Utilizing a fashion dataset for pre-training the RoBERTa model, the study achieves a 13.9%

NDCG improvement, with the student model achieving an NDCG of 0.73 and significantly lower ranking latency (~15 ms). ListBERT outperforms other loss functions, achieving an NDCG of 0.754.

In the study by Wenqi Fan et al. (2024), the literature delivers an exhaustive analysis of Large Language Models (LLMs) within recommender systems.It focuses on paradigms of pre-training, fine-tuning, and prompting, and discusses potential future directions and challenges in LLM-empowered recommender systems. The paper covers methods for implementing LLMs, including pre-training tasks like Masked Language Modeling and fine-tuning strategies. It concludes with future research directions, addressing issues such as hallucination mitigation and advanced techniques.

In Xu Huang et al. (2024) the paper introduces InteRecAgent, a framework that integrates traditional recommender models with Large Language Models (LLMs) to create versatile and interactive recommender systems. The paper evaluates the effectiveness of InteRecAgent using three datasets: Steam, MovieLens, and Amazon Beauty, each comprising user-item interaction history data and item metadata. InteRecAgent employs components such as a memory bus, dynamic demonstration-augmented plan-first execution, and reflection, showcasing its effectiveness in diverse domains supported by an ablation study and case studies in chit-chat, gaming, and e-commerce.

In the study conducted by Jianghong Zhou et al. (2023), the aim is to automate the creation of product descriptions in eCommerce by employing the LLAMA 2.0 7B language model. The research utilized a meticulously curated dataset of genuine product descriptions sourced from Walmart. The methodology encompassed training the LLAMA 2.0 7B model on the Walmart dataset, refining it for domain-specific characteristics, and concentrating on language appeal, factual details, dimensions, attributes, and assurances within the product descriptions. Evaluation metrics confirmed scalability, reduced human workload, and highlighted the potential for automated optimization in eCommerce, enhancing search functionality and boosting sales.

In Dario Di Palma(2017). the paper explores language models in recommendation systems, focusing on tasks like utilizing large language models for reasoning over knowledge graphs, addressing continual fine-tuning challenges, and assessing their effectiveness and limitations as recommender systems. The research employed three datasets: ML-1M, Beauty, and Online Retail, demonstrating the effectiveness of enhancing LLMs for practical tasks by incorporating domain-specific knowledge without fine-tuning costs. This method proved critical for improving LLM performance, showcasing robustness to prompt modification and outperforming other approaches in achieving significant performance improvements.

In Xuejiao Wang and Chao Liu(2023), the research paper investigates Retrieval-augmented Recommendation Systems (RaRS) with Large Language Models (LLMs) to enhance recommender system performance. The study evaluates LLM advantages and challenges, conducts experiments on benchmarks, and compares with existing methods, aiming to integrate conversational AI with recommendation systems for effective LLM utilization. Experimental analysis using GPT-3.5 on MovieLens100K and Facebook Books datasets demonstrated ChatGPT's efficacy in Zero-Shot scenarios, generating top 50 recommendations comparable to state-of-the-art models. However, the black-box nature of LLMs poses interpretability challenges.

In the paper Xuejiao Wang and Chao Liu(2023), the authors aim to combat information overload in the news domain with a personalized recommendation system. Using 5000 users and 120,000 reading records, the system enhances user collaborative filtering to improve recall, accuracy, and F1 score. The algorithm integrates user reading records and news title similarity, achieving superior precision and recall over other models.

In the paper Zeyu Chena, Kailun Jianb(2023) the authors analyze user behavior to develop a personalized recommendation system. Using data like User ID, Video ID, and viewing duration, the collaborative filtering algorithm achieves a recall rate and precision surpassing AI and GA algorithms. After 300 iterations, the model demonstrates a recommended accuracy of 93.3%, outperforming other methods.

In the paper Illia Balush, Victoria Vysotska and Solomiia Albota(2021), the authors outline objectives for an intelligent search system in e-commerce. Using e-commerce purchase data, the system employs Levenshtein's fuzzy search algorithm to enhance search quality, relevance, and price sorting. The results show improved speed and accuracy, emphasizing the algorithm's effectiveness over existing systems.

In the paper Ms. Shakila Shaikh Dr. Sheetal Rathi,Asst Prof. Prachi Janrao(2017), the authors aim to enhance recommendation methods by addressing limitations and proposing semantic integration. Using user data from multiple e-commerce platforms, the graph-based approach emphasizes semantic integration and overlap techniques. Comparative analysis reveals a need for improved semantic recommendation properties in popular websites.

In the paper Soodabeh Sarafrazi, Darwin Wheeler , David Garcia, Shane Henrikson, Naveed Sharif, Hui Wu(2023), the authors aim to improve SEO in healthcare by utilizing NLP and graph techniques. Using keyword data, search volume, and page rankings, the study employs the BERT model and node2vec for content optimization and keyword cannibalization. The approach enhances SEO performance, improving health-related content visibility.

Table for analysis of existing techniques:

Reference Number	**Key Words**	**Objective(Abstract)**	**Dataset Used**	**Methods Used (Name of any process or model)**
[1]	Deep Learning, Collaborative Filtering, Neural Networks, Recommendation Systems	The research paper introduces DeepHCF, a model for rating prediction in recommender systems. Employing joint training with a multi-layer perceptron (MLP) and a convolutional neural network (CNN), it aims to minimize prediction error and enhance system performance. Comparative analysis with baseline approaches assesses the effectiveness of DeepHCF.	The research paper used four different real-world datasets for evaluation. These datasets include MovieLens-1M, Amazon Instant Video, Amazon Android Apps, and Amazon Digital Music. Each dataset was used to assess the performance of the DeepHCF model in the context of recommendation system accuracy and effectiveness.	The research paper introduces DeepHCF, a hybrid recommender system addressing collaborative filtering sparsity using both ratings matrix and item reviews. The model achieves excellent performance with MAE/RMSE scores of 0.674/0.867, 0.688/0.994, 0.896/1.180, and 0.520/0.830 on MovieLens-1M, Amazon Instant Video, Amazon Android Apps, and Amazon Digital Music datasets.
[2]	Movies, Recommendation system, CBF- Content-based filtering, CF- Collaborative filtering	The Movie Recommendation System aims to categorize users with similar interests, employing AI-based algorithms for personalized content suggestions. Utilizing item-based collaborative filtering, the system	In the Movie Recommendation system Netflix movies and TV shows dataset is used and also the system considers user profiles, search history, and demographic	The research employed content-based filtering, emphasizing user comfort in film recommendations. By integrating clustering, similarity, and classification

		considers user profiles, search history, and demographic traits to enhance recommendation accuracy and relevance.	traits to enhance recommendation accuracy and relevance.	techniques, the system improved accuracy by considering user preferences, viewing history, and reviews. This strategy enhanced recommendation precision and reduced Mean Absolute Error.
[3]	E-Commerce Search, Information Retrieval, Deep Learning	The study explores the integration of BERT into E-commerce non-default search ranking, presenting a two-stage ranking approach for high-accuracy recall. Objectives include implementing parallel prediction on multiple GPU hosts, introducing a C++-based BERT tokenizer via Tensorflow custom op, and experimenting with various BERT fine-tuning schemes and features.	The system presented on this research paper uses corpus offered by data challenge, there are 899k documents and 150 queries, producing 150 millions of query/item pairs.	The study introduces a two-stage ranking approach for E-commerce Non-Default Search Ranking, combining refined query matching and BERT-Large fine-tuning. Experimental results revealed a 0.37% improvement with WholeWord Masking BERT-Large and enhanced F1 scores (0.8293) with a multilayer perceptron. Ensemble models secured 1st place in the supervised phase and 2nd place in the final phase.
[4]	transfer learning; knowledge distillation; ensemble learning	This paper introduces a learning framework employing transfer learning and knowledge distillation to address the challenge of limited human-labeled data in industrial applications. Experimental results and case studies	The study employs four in-house datasets, each divided into 90% training and 10% test sets, preventing cross-leakage. These datasets include human-labeled	The study presents a data-driven framework for e-commerce search relevance, distilling BERT knowledge into a feed-forward network. BERT2DNN

		showcase the effective imitation of teacher models by simple feed-forward student networks, accompanied by significant enhancements in online serving speed.	(query, item) pairs, two months of filtered search logs, and ten months of unfiltered search logs.	achieves 97% test accuracy with significantly lower serving costs. A/B tests demonstrate improvements in CVR and GMV metrics, and case studies reveal semantic relations in learned embeddings.
[5]	Transformer, RoBERTa, BERT, Ranking, Retrieval, E-commerce Products, Knowledge-Distillation, NDCG, Listwise Loss, Pairwise Loss etc.	The objective of this paper is to present a novel method for training a transformer-based model end-to-end, utilizing a listwise surrogate loss function. It aims to assess the effectiveness of listwise versus pairwise loss functions, measured by the NDCG metric, and underscore the model's applicability in low-latency e-commerce search through knowledge distillation.	The study utilizes a fashion dataset for pre-training the RoBERTa model from scratch, incorporating a BPE tokenizer for effective tokenization. The pre-training dataset comprises product descriptions and user queries, totaling over 5 million instances.	The study introduces ListBERT, A novel method that combines a RoBERTa model with listwise loss functions for ranking e-commerce products. Pre-training on a fashion corpus and fine-tuning with approxNDCG yields a 13.9% NDCG improvement. Knowledge distillation enables a student model with a 0.73 NDCG and significantly lower ranking latency (~ 15 ms). ListBERT outperforms other loss functions, achieving an NDCG of 0.754.
[6]	Recommender Systems, Large Language Models (LLMs), Pre-training and Fine-tuning, In-context	The literature seeks to offer an extensive examination of Large Language Models (LLMs) within recommender systems, concentrating on the concepts of pre-training, fine-tuning,	Within this paper, Large Language Models (LLMs) undergo training on an extensive corpus containing varied	The research paper explores methods for implementing Language Models (LLMs) in recommender systems, covering

	Learning, Prompting.	and prompting. Additionally, it explores the potential avenues for future developments and the challenges faced in the realm of LLM-driven recommender systems.	and unlabeled data.	pre-training tasks like Masked Language Modeling and fine-tuning strategies. It discusses LLM application in recommendation tasks, challenges like hallucination mitigation, and advanced techniques, concluding with future research directions.
[7]	InteRecAgent, Memory bus, Dynamic demonstration-augmented task planning, Reflection, Natural language interface, ID-based matrix factorization models, Conversational recommender system,	The objective of this paper is to present InteRecAgent, a framework that combines conventional recommender models with Large Language Models (LLMs) to develop adaptable and interactive recommendation systems. It aims to tackle the hurdles related to the integration of LLMs into recommender systems and underscore the potential of InteRecAgent in delivering personalized and interactive recommendations across diverse domains.	The effectiveness of the methods proposed in the document is evaluated using three datasets: Steam, MovieLens, and Amazon Beauty. Each dataset comprises user-item interaction history data and item metadata.	The paper introduces InteRecAgent, employing Large Language Models (LLMs) as the brain and recommendation models as tools. Key components include a memory bus, dynamic demonstration-augmented plan-first execution, and reflection. Experimental results showcase InteRecAgent's effectiveness in diverse domains, supported by an ablation study and case studies in chit-chat, gaming, and e-commerce.
[8]	Search visibility, Market trends, Click-through rates, Automation,	The objective of the research is to automate the generation of product descriptions in eCommerce using the LLAMA 2.0 7B language	The study employed a meticulously curated dataset comprising authentic	The method involved training LLAMA 2.0 7B on the Walmart dataset, fine-tuning for

	LLAMA 2.0 7B language model,, NDCG, Human workload reduction, Large language models, Optimizing,	model. The aim is to enhance search visibility, customer engagement, and ultimately increase click-through rates by leveraging advanced language models.	product descriptions sourced from Walmart, a prominent player in the eCommerce sector on a global scale.	domain-specific features, and emphasizing linguistic appeal, factual content, dimensions, attributes, and warranties within product descriptions. Evaluation metrics confirmed scalability, reduced human workload, and highlighted the potential for automated optimization in eCommerce, enhancing search functionality and boosting sales.
[9]	large language model, knowledge plugins, recommender system	The paper aims to explore language models in recommendation systems, focusing on tasks like utilizing large language models for reasoning over knowledge graphs, addressing continual fine-tuning challenges, and assessing their effectiveness and limitations as recommender systems.	The research paper employed three datasets: ML-1M with 6,034 users, 3,533 items, and 575,272 interactions (95.35% density); Beauty with 22,363 users, 12,101 items, and 198,502 interactions (99.93% density); and Online Retail with 16,517 users, 3,466 items, and 514,694 interactions (99.10% density).	The paper explores enhancing Large Language Models (LLMs) for practical tasks by incorporating domain-specific knowledge without fine-tuning costs. The method proves critical for improving LLM performance, showcasing robustness to prompt modification and outperforming other approaches in achieving significant performance improvements.
[10]	Large Language Models,	The research paper investigates Retrieval-	The research paper employed experimental	The research paper employed experimental

	Recommender Systems, Prompt Engineering, New Item Recommendation, Hallucination Problem, Retrievalaugmented Recommender System	augmented Recommendation Systems (RaRS) with Large Language Models (LLMs) to enhance recommender system performance. It evaluates LLM advantages and challenges, conducts experiments on benchmarks, compares with existing methods, and aims to integrate conversational AI with recommendation systems for effective LLM utilization.	analysis to assess ChatGPT's capabilities in recommendation scenarios, using GPT-3.5 on MovieLens100K and Facebook Books datasets.	analysis to assess ChatGPT's capabilities in recommendation scenarios, using GPT-3.5 on MovieLens100K and Facebook Books datasets. Results showed ChatGPT's efficacy in Zero-Shot scenarios, generating top 50 recommendations comparable to state-of-the-art models, but its black-box nature poses interpretability challenges.
[11]	News Recommendation Algorithm User Behavior Analysis Hybrid Recommendation Strategies Precision and Recall Evaluation System Architecture and Components	The paper aims to combat information overload in the news domain by implementing a personalized news recommendation system. Utilizing an enhanced user collaborative filtering algorithm, it seeks to improve performance and achieve higher recall, accuracy, and F1 score ratio compared to other algorithms, ultimately providing superior news recommendations.	The dataset used in the paper includes 5000 users and nearly 120,000 reading records for the experimentation and evaluation of the news recommendation algorithm.	The paper introduces a news recommendation algorithm integrating user reading records and news title similarity. Through fusion similarity calculation and parameter optimization, with α set to 0.4, the algorithm achieves superior precision and recall, outperforming PNR, NR_LDA, and FALS by 2.7%, 3.9%, and 12.6% in accuracy,
[12]	User behavior analysis, Collaborative filtering algorithm, Digital media,	The research paper aims to analyze user behavior, extract log data to construct a user model, and develop a personalized	In this paper the presented model is trained using User ID ,VIDeo id ,Watch start time	The research paper employed a collaborative filtering algorithm for recommendation

	Recommendation system	recommendation system. It focuses on refining and merging user behavior connectives, categorizing behavior characteristics, and accurately identifying nearest neighbor users for precise personalized recommendations.	,Viewing duration.	systems. Simulation experiment analysis revealed that the algorithm achieved the highest recall rate and precision, surpassing AI and GA algorithms. Specifically, after 300 iteration steps, the paper's algorithm demonstrated a recommended accuracy of 93.3%, outperforming others
[13]	Search engine, Elasticsearch, referral systems, collaborative filtering, web application, search, fuzzy search, inverted index, search query, intelligent search system, entry point, recommendation system,	The research paper outlines objectives for an intelligent search system, emphasizing fast, efficient search, user satisfaction, and the incorporation of recommendation systems. It aims to be relevant for e-commerce, focusing on information storage, convenience, algorithm application, and the development of a robust search engine.	The system presented in this paper uses the user purchase data of an e-commerce website for the evaluation.	The paper explores Levenshtein's fuzzy search algorithm to enhance search quality on trading platforms. Results demonstrate the algorithm's effectiveness in finding goods with vague queries, sorting results by relevance and price. Comparisons show improved speed and accuracy over existing systems, emphasizing the algorithm's significance.
[14]	Content-based; Collaborative filtering; hybrid recommendati	The paper aims to enhance recommendation methods by addressing limitations, proposing semantic integration to	The dataset used in the paper includes user data of multiple ecommerce platform.	The paper presents a graph-based approach for enhancing e-commerce recommendation

	on; item recommendation, Overlap, Graph algorithm.	improve user and item modeling. It seeks to evaluate and demonstrate the potential of semantics in recommendation systems, especially in addressing the absence of semantic recommendations on platforms like flickr.com.		systems, emphasizing semantic integration using an overlap technique. Comparative analysis of popular websites reveals a need for improved semantic recommendation properties
[15]	Search Engine Optimization (SEO), Natural Language Processing (NLP), Large Language Model (LLM), Graph Analysis	The paper aims to improve search engine optimization in healthcare and clinical domains by utilizing natural language processing and graph techniques. It targets issues like insufficient content and keyword cannibalization, employing machine learning, specifically the BERT model, and graph analysis to offer practical solutions for SEO challenges.	The dataset used in this paper comprises the keyword itself, search volume, difficulty, URLs of pages within the KP.org domain, alongside their corresponding rankings, as well as URLs and rankings for competitor pages.	The paper employs a combination of natural language processing (NLP) techniques, including the BERT model, and graph analysis methods such as node2vec to enhance search engine optimization in healthcare. Focusing on content optimization and keyword cannibalization, the study utilizes diverse machine learning techniques to empower SEO teams and improve the visibility of health-related content.

METHODOLOGY

Data Description:

In this project, we employed various datasets to develop a recommendation system and a multilingual search engine tailored for an e-commerce platform. These datasets span multiple languages and domains, ensuring a comprehensive and versatile system.

The primary dataset utilized in this project is a comprehensive e-commerce dataset sourced from Flipkart, available on DataWorld.com. This dataset includes 15 columns and 20,000 observations, providing a rich source of information for building our recommendation system. The columns in this dataset are as follows:

- uniq_id: A unique identifier for each product.
- crawl_timestamp: The timestamp when the data was crawled.
- product_url: The URL of the product on Flipkart.
- product_name: The name of the product.
- product_category_tree: A hierarchical categorization of the product.
- pid: Product ID.
- retail_price: The retail price of the product.
- discounted_price: The discounted price of the product.
- image: A list of URLs to images of the product.
- is_FK_Advantage_product: A boolean indicating whether the product is a Flipkart Advantage product.
- description: A description of the product.
- product_rating: The product's rating.
- overall_rating: The overall rating of the product.
- brand: The brand of the product.
- product_specifications: Specifications of the product in JSON format.

In this dataset, the column is_FK_Advantage_product is of boolean type, while retail_price and discounted_price are numerical. The remaining columns are categorical, providing diverse information to support both recommendation and search functionalities.

For the multilingual search engine, the necessary columns were translated into Hindi, Marathi, and Bengali using the Googletrans library. This allows the search engine to effectively handle queries in multiple languages, enhancing accessibility and user experience across different linguistic groups.

For product categorization, we considered the top 15 categories, leveraging the product_category_tree column to classify products accurately. This approach ensures that new items added to the platform are categorized quickly and correctly, which is crucial for maintaining an organized inventory and enhancing the user shopping experience.

By integrating these diverse datasets, we have developed a comprehensive system that not only recommends products based on user preferences but also supports robust multilingual search functionality. This approach enhances the overall utility and reach of the e-commerce platform, catering to a diverse user base with varied linguistic backgrounds.

Natural Language Processing (NLP)(James F. Allen(2003))

Natural Language Processing (NLP) is a specialized field within computer science dedicated to the interaction between human language and computational systems. It encompasses a range of tasks aimed at enabling computers to understand, interpret, and generate human language in a manner akin to human communication.

NLP involves processing and analyzing large volumes of natural language data, which is often unstructured and diverse. It encompasses tasks such as text parsing, sentiment analysis, language translation, speech recognition, and information extraction. These tasks enable computers to comprehend and extract meaning from text, enabling a wide array of applications across various domains.

One fundamental aspect of NLP is the development of algorithms and models that can understand and generate human language. This involves techniques such as machine learning, deep learning, and natural language understanding, which enable computers to learn patterns and relationships within textual data.

Another key area of NLP is language generation, where computers generate human-like text based on input data or predefined rules. This includes tasks such as language translation, text summarization, and dialogue generation, which aim to produce coherent and contextually relevant output.

NLP finds applications across numerous industries and domains, including healthcare, finance, customer service, education, and entertainment. In healthcare, NLP is used for tasks such as medical record analysis, clinical documentation, and patient monitoring. In finance, it is used for sentiment analysis, fraud detection, and automated trading. In customer service, NLP powers chatbots and virtual assistants, enabling automated interaction with users. In education, NLP is used for language learning, automated grading, and content recommendation. In entertainment, it is used for content analysis, recommendation systems, and natural language interfaces.

Overall, NLP plays a crucial role in bridging the gap between human language and computers, enabling the development of intelligent systems that can understand, interpret, and generate natural language text. Its applications are diverse and far-reaching, with the potential to revolutionize how we interact with technology and process textual information.

Recommendation System(Hyeyoung Ko et al (2021)):

A recommendation system is an advanced technology deployed by e-commerce platforms to offer personalized product suggestions to users. By analysing various aspects of user behaviour, preferences, and past interactions, these systems predict and suggest products that users are likely to find interesting. This tailored approach aims to enhance the overall shopping experience by streamlining the process of discovering new products that align with user interests.

The benefits of recommendation systems in e-commerce are multifaceted. Firstly, they significantly increase sales and conversion rates. By presenting users with products that match their interests and needs, the likelihood of purchase is greatly enhanced. Personalized recommendations lead to more informed and satisfying purchase decisions, translating into higher sales for the platform.

Secondly, recommendation systems greatly enhance the user experience. They simplify the shopping journey by helping users discover products they might not have found on their own. This personalized touch makes the shopping experience more enjoyable and efficient, encouraging users to spend more time on the platform. By providing relevant suggestions, these systems help users navigate through extensive product catalogues with ease, improving overall satisfaction.

Another significant benefit is improved customer retention and loyalty. When users feel understood and valued through personalized recommendations, they are more likely to return to the platform for future purchases. This fosters a sense of loyalty and trust between the user and the e-commerce platform. Satisfied customers are not only likely to become repeat buyers but also to recommend the platform to others, contributing to organic growth.

Recommendation systems also facilitate efficient product discovery. With the vast number of products available on e-commerce platforms, users can easily become overwhelmed. Recommendation systems act as a guide, directing users to products that are most relevant to their interests. This not only saves time but also enhances the user's ability to find desirable products, making the shopping experience more productive.

Furthermore, implementing a sophisticated recommendation system provides a competitive advantage. In a crowded e-commerce market, the ability to offer superior personalization can set a platform apart from its competitors. Users are more likely to choose and remain loyal to platforms that offer a personalized shopping experience. This differentiation can be a critical factor in attracting and retaining customers in a competitive landscape.

In this project, we implemented a a content-based recommendation system employing Term Frequency-Inverse Document Frequency (TF-IDF) vectors. The primary dataset utilized was a comprehensive e-commerce dataset from Flipkart, which included detailed product descriptions, prices, ratings, and other relevant information. This rich dataset provided the foundation for building an effective recommendation system.

The implementation process began with data collection and preprocessing. TF-IDF was then used to convert product descriptions into numerical vectors, allowing the system to quantify the importance of various words within the descriptions. This step was crucial for identifying the key attributes of each product.

Next, the system computed similarity scores between products based on their TF-IDF vectors. By comparing these scores, the system could identify products with similar attributes to those a user had previously shown interest in. These products were then recommended to the user, providing a personalized shopping experience.

The recommendation system continually updates and refines its suggestions by analysing ongoing user interactions. This guarantees that the recommendations stay relevant and in tune with the user's changing preferences. By delivering personalized product suggestions, the system enhances the user experience and drives sales for the e-commerce platform.

Overall, the implementation of this content-based recommendation system demonstrates how advanced data analysis and machine learning techniques can be leveraged to create a more engaging and effective e-commerce environment. This approach not only improves user satisfaction but also contributes to the platform's success by increasing sales and fostering customer loyalty.

Below are different categories of recommendation engines employed in diverse contexts:

- Market Basket Analysis (Association Rule Mining): This approach delves into analyzing patterns of items frequently purchased together. By identifying associations between items, it can suggest complementary or related products to customers based on their current selections.
- Content-Based: Content-based recommendation systems rely on analyzing the attributes or features of items and comparing them to the user's preferences. Recommendations are made by suggesting items with similar attributes to those the user has previously shown interest in.
- Collaborative Filtering: This technique generates recommendations for individuals by utilizing the behaviour and preferences of a larger user group. It identifies users with similar tastes and preferences and recommends items that these similar users have liked or purchased.
- Hybrid Systems: These systems amalgamate various recommendation techniques to deliver recommendations that are both precise and varied. By incorporating different approaches such as collaborative filtering and content-based filtering, hybrid systems aim to overcome the limitations of individual methods.
- ML Clustering-Based: Machine learning clustering-based recommendation systems use clustering algorithms to group items or users based on similarities in their attributes or behavior. Recommendations are then made by suggesting items that are popular or preferred within the same cluster.
- ML Classification-Based: Machine learning classification-based recommendation systems classify users or items into predefined categories or classes. Recommendations are generated by predicting the class to which a user or item belongs and suggesting items that are popular or relevant within that class.
- Deep Learning and NLP Based: Deep learning and natural language processing (NLP) based recommendation systems utilize advanced neural network architectures and NLP

techniques to extract and analyze textual data such as product descriptions or user reviews. Through grasping the meaning and context embedded within textual data, these systems have the capacity to produce recommendations that are more precise and tailored to individual preferences.

Content Based Filtering:

For this project, we've applied content-based filtering, a technique commonly used in e-commerce and various other fields, for recommendation purposes. Content-based filtering tailors suggestions to users by evaluating item characteristics. In contrast to collaborative filtering, which relies on user actions and preferences, content-based filtering centres on examining the intrinsic attributes or qualities of items.

In content-based filtering, items are described using a set of features or attributes, such as product descriptions, tags, or metadata. These features are then used to create profiles or representations of items. When a user interacts with the system, their preferences are inferred based on their past interactions or explicit feedback.

Suggestions are produced by comparing the user's profile to the profiles of items within the system. Items that closely match the user's preferences, based on their features or attributes, are recommended to the user. For example, in e-commerce, if a user has previously shown interest in purchasing sports shoes, the system may recommend other sports-related products such as activewear or fitness equipment.

Content-based filtering is particularly useful in scenarios where there is rich item metadata available, such as textual descriptions or tags. It can also be effective for recommending niche or less popular items based on their content attributes. However, one limitation of content-based filtering is that it may struggle to recommend items outside of a user's known preferences or interests, as it relies solely on item features rather than user interactions or community feedback. Overall, content-based filtering provides a valuable approach for generating personalized recommendations tailored to individual user preferences based on the characteristics of items.

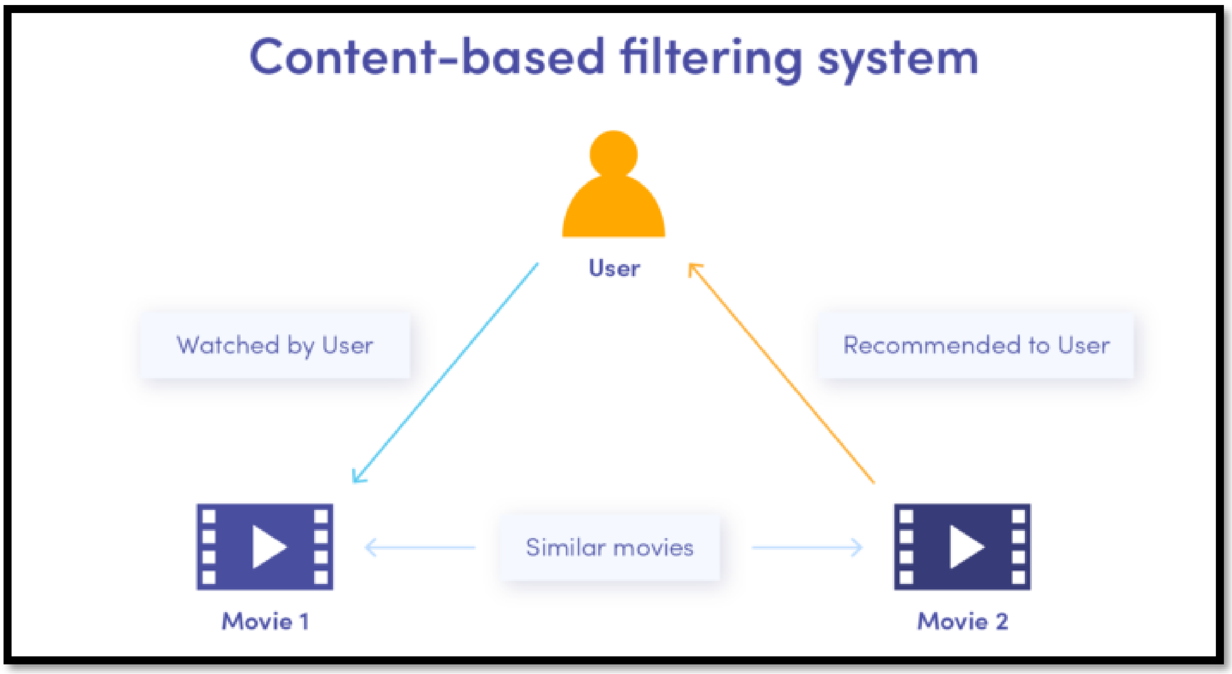


Fig 1: Content Based Filtering

(Source: https://www.miquido.com/blog/perks-of-recommendation-systems-in-business)

Term Frequency-Inverse Document Frequency (Akiko Aizawa,2003):

Natural Language Processing (NLP) stands as a pivotal domain within computer science, focusing on the intricate interplay between human language and computational systems. An integral task within NLP involves extracting pertinent information from vast amounts of unstructured data. In this discourse, we delve into a widely employed technique in NLP known as TF-IDF.

TF-IDF, which stands for Term Frequency-Inverse Document Frequency, acts as a numerical measure indicating the importance of a word within a document. It is commonly employed in NLP to assess how relevant a term is to either a specific document or a collection of documents. TF-IDF computation integrates two principal factors: the frequency of a word within a specific document (TF) and its frequency across all documents in the corpus (IDF).

The term frequency (TF) quantifies how often a term appears within a document, calculated by dividing the count of a term by the total word count within the document, yielding a value between 0 and 1. Conversely, the inverse document frequency (IDF) measures the importance of a term across the entire corpus. IDF is computed as the logarithm of the total number of documents divided by the number of documents containing the term, resulting in a value greater than or equal to 0.

The TF-IDF score, computed by multiplying TF and IDF, signifies the importance of a term within a document. Higher TF-IDF scores denote greater significance of the term within the document.

TF(i,j) = (Term i frequency in document j)/(Total words in document j) ------(1)

IDF (i) = log2((Total documents)/(documents with term i)) --------------------(2)

TFIDF (score for term i in document j) = TF(i,j) * IDF (i) ---------------------(3)

Here IDF = Inverse Document Frequency & TF = Term Frequency and t = Term & j = Document

TF-IDF operates by assigning weights to words based on their frequency and rarity, thereby ascertaining the importance of terms within documents or corpora. For instance, in a hypothetical scenario with a corpus containing five documents, TF-IDF scores for a specific term, such as "fox," would be computed by assessing its frequency within each document, its occurrence across the corpus, and subsequent calculation of TF and IDF values.

Advantages of TF-IDF encompass its ability to measure term relevance, handle large text datasets, mitigate the impact of common words (stop words), and facilitate various NLP applications. Despite its utility, TF-IDF has limitations, including its disregard for contextual nuances, assumption of term independence, and sensitivity to vocabulary size and stop words. Nonetheless, TF-IDF remains indispensable for diverse NLP tasks, including search engines, text classification, information extraction, keyword extraction, recommender systems, and sentiment analysis.

Preprocessing:

The dataset comprises 20,000 observations with 15 columns. Among these, 'is_FK_Advantage_product' is boolean, while 'retail_price' and 'discounted_price' are numerical, and the rest are categorical. To enhance the dataset, a new 'length' column is added to gauge the total character count in the 'description' variable. Additionally, another column, 'no_of_words', is introduced to tally the number of words in each description before preprocessing. The word count distribution illustrates that a significant portion of descriptions contains fewer than 100 words, with around 20% ranging from 100 to 200 words.

Preparing data is crucial for transforming raw information into a format that is appropriate for analysis. Missing value analysis reveals approximately 30% of brand variables have missing values, while other variables have negligible omissions. Text preprocessing involves several tasks, including converting text to lowercase, removing/replacing punctuations and numbers, eliminating extra whitespaces, and filtering out stop words. These tasks ensure the data is cleaned of extraneous information. Additionally, stemming and lemmatization techniques are employed to normalize words.

Further refinement involves removing domain-specific stop words related to online shopping, ensuring the descriptions contain only relevant terms. After preprocessing, the most frequent words in the dataset are identified, including 'women', 'price', and 'shirt', indicating a prevalence of fashion-related items, particularly those targeted towards women. This comprehensive preprocessing approach streamlines the dataset, facilitating more accurate analysis and interpretation.

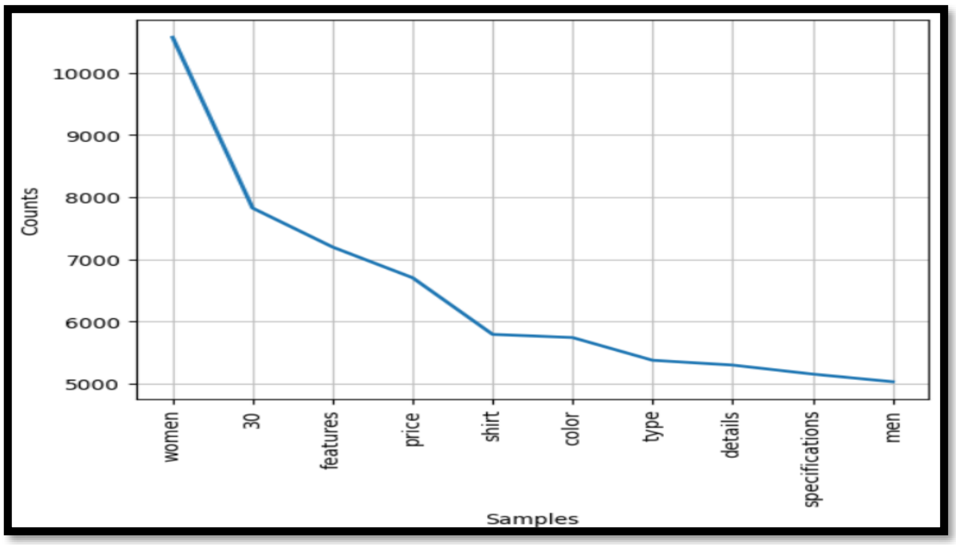


Fig 2: Word count Distribution

In essence, the data preprocessing phase plays a crucial role in enhancing the quality and usability of the dataset, ensuring that it is primed for subsequent analysis and modeling tasks. By cleaning, transforming, and refining the data, potential biases and inconsistencies are mitigated, enabling more robust and reliable insights to be derived from the dataset.

Modelling:

In the realm of model building, the focus shifts from understanding data to implementing solutions using algorithms. In this project, two pivotal models are targeted: a content-based recommendation system and a product search engine. Leveraging various Natural Language Processing (NLP) techniques, such as TF-IDF and word embeddings, both models are designed to enhance user experience and provide valuable insights into product recommendations and search results.

Beginning with the content-based recommendation system, TF-IDF is employed to analyze product descriptions and compute similarity scores. By cleansing the text data and defining a TF-IDF vectorizer, a matrix of TF-IDF vectors is generated, encapsulating the unique characteristics of each product description. With over 26,000 distinct words identified, the TF-IDF matrix lays the foundation for calculating similarity scores between products.

Utilizing cosine similarity as the metric, a function is developed to determine the N most similar products based on a given product description. This function facilitates reverse mapping of product names to their respective indices, enabling efficient retrieval of similar products. By encapsulating the entire process within a single function, testing and evaluation become more streamlined and manageable.

The function workflow involves several key steps:

- Obtaining the index corresponding to the input product.
- Calculating cosine similarity scores between the input product and all other products.
- Arranging the similarity scores in a descending sequence.
- Choosing the N most similar products from the top-ranked list.
- Outputting the names of the similar products.

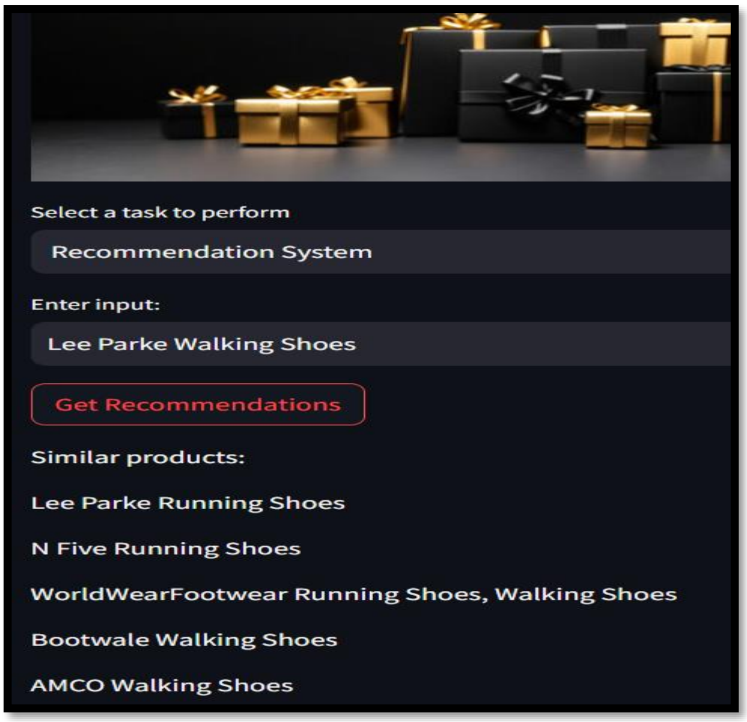
Upon execution, the function prompts the user to input a product name, after which it retrieves and displays the most similar products. This iterative process allows users to explore and discover products that closely resemble their preferences, enhancing their overall shopping experience.

The cosine similarity(Wolfram Research (2007)) of two vectors A and B is given as,

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

Where n is the dimension of the vectors and Ai and Bi are the ith element.

In brief, the content-based recommendation system employs TF-IDF and cosine similarity to deliver personalized product recommendations by analysing the textual attributes of each item. By harnessing the power of NLP techniques, this model aims to deliver tailored recommendations that resonate with users' preferences and interests, ultimately driving engagement and satisfaction.


Select a task to perform

Recommendation System

Enter input:

Lee Parke Walking Shoes

Get Recommendations

Similar products:

Lee Parke Running Shoes

N Five Running Shoes

WorldWearFootwear Running Shoes, Walking Shoes

Bootwale Walking Shoes

AMCO Walking Shoes

Fig 3: Recommendation System

Advanced Search Engine Using PyTerrier ,Sentence-BERT & DeepTextSearch

A search engine is a software program designed to retrieve information from a database or network in response to user queries. In the context of e-commerce platforms, a search engine plays a crucial role in facilitating product discovery and navigation. Users can enter keywords or phrases related to the products they are looking for, and the search engine returns relevant results based on factors such as product descriptions, titles, and attributes.

Search engines employ various techniques to rank and retrieve results, including keyword matching, relevance scoring, and machine learning algorithms. These techniques help optimize search performance and ensure that users find what they are looking for quickly and efficiently. Additionally, search engines often incorporate features like autocomplete suggestions, filters, and sorting options to further enhance the user experience.

PyTerrier :

PyTerrier is a Python library designed for information retrieval research and experimentation. It provides a wide range of tools and functionalities for building, evaluating, and deploying search systems. PyTerrier integrates with the Terrier platform, a leading open-source toolkit for building search engines.

The PyTerrier package offers modules for indexing, retrieval, evaluation, and visualization of search results. It supports various retrieval models, including BM25, TF-IDF, and Divergence from Randomness (DFR). Additionally, PyTerrier provides access to pre-trained neural retrieval models, such as BERT-based models, for advanced search applications.

With PyTerrier, researchers and developers can easily prototype and benchmark different search algorithms and techniques. Its modular architecture and intuitive API make it suitable for both academic research and industrial applications in fields such as e-commerce, healthcare, and information retrieval. Overall, PyTerrier empowers users to build powerful and scalable search systems tailored to their specific needs and requirements.

SentenceBERT:

SentenceBERT, also known as Sentence Transformer, is a state-of-the-art deep learning model used for text embedding and similarity calculation. It is built on top of the BERT (Bidirectional Encoder Representations from Transformers) architecture and fine-tuned specifically for sentence-level tasks. SentenceBERT, a variant of BERT("stsb-distilbert-base") is capable of capturing semantic similarities between sentences by encoding them into dense vector representations in a high-dimensional space.

With SentenceBERT, textual data can be transformed into fixed-size vector representations, enabling efficient comparison and retrieval of similar sentences or documents. This attribute renders it especially valuable for endeavors like retrieving information, clustering documents, and conducting semantic searches. SentenceBERT excels in capturing subtle nuances and semantic relationships in text, leading to more accurate and contextually relevant search results.

DeepTextSearch:

DeepTextSearch is a Python library designed for fast and accurate text recommendation and search tasks. It leverages advanced text embedding techniques using a variant of BERT("

paraphrase-xlm-r-multilingual-v1") optimized for creating embeddings of more than 50+ languages, to encode textual data into dense vector representations. DeepTextSearch offers features like faster search speed, high accuracy in recommendation and search results, and easy integration with Python-based web applications or APIs.

DeepTextSearch is well-suited for a range of uses such as aggregating news, analyzing social media, and recommending products in e-commerce, catering to developers of all skill levels, from beginners to experts.Its simple installation process via pip and intuitive usage make it accessible to a wide range of users. By utilizing DeepTextSearch, developers can efficiently implement text recommendation and search functionalities in their projects, enhancing user experience and engagement.

Procedure Implemented:

Indexing serves as a crucial step in information retrieval systems, streamlining the process of data retrieval. DFIndexer is a pivotal tool utilized for indexing, simplifying the retrieval process significantly. Additionally, BatchRetrieve, a prominent PyTerrier object, leverages pre-existing Terrier index data structures to facilitate efficient retrieval operations. To proceed with implementation, PyTerrier and Sentence-BERT libraries are employed, aiming to develop an advanced search engine.

The initial setup involves installing necessary packages and libraries, including PyTerrier and Sentence-BERT, to support the implementation process. Additionally, the en_core_web_sm model from spaCy is downloaded to enable further text processing functionalities. Data preprocessing plays a vital role in preparing the dataset for indexing and retrieval tasks. Firstly, the product category tree is cleaned by removing unwanted characters and splitting the categories. Subsequently, redundant columns such as crawl timestamp, product URL, and image URLs are dropped to streamline the dataset.

Duplicate products are removed to ensure data integrity and accuracy during subsequent processing stages. Following this, text data undergoes preprocessing steps such as stop word removal, punctuation elimination, tokenization, and lemmatization to enhance the quality and relevance of keywords extracted from the text. The filter_keywords function is applied to selected columns, including product names, descriptions, and brand names, to extract meaningful keywords.

These extracted keywords are then combined to form a comprehensive set of keywords for each product, enabling effective indexing and retrieval. A 'docno' column is created to serve as a unique identifier for each product, facilitating seamless querying and retrieval operations based on relevant keywords. This preprocessing stage ensures that the dataset is optimized for efficient indexing and retrieval using advanced techniques such as PyTerrier and Sentence-BERT, ultimately enhancing the functionality and performance of the search engine.

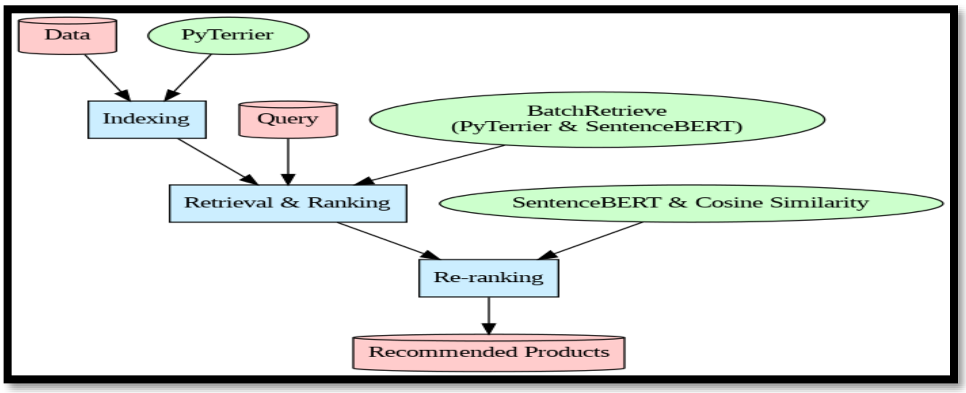


Fig 4: Flow Diagram

Modelling:

Utilizing the DFIndexer object, an index for keywords is generated to facilitate efficient retrieval of relevant products. The unique product dataset is indexed using the DFIndexer, creating an index reference that associates keywords with their respective product identifiers. Subsequently, ranking and retrieval operations are performed using PyTerrier and Sentence-BERT.

The BatchRetrieve function is employed to conduct retrieval based on the TF-IDF weighting model, integrating stop word removal to enhance retrieval accuracy. Querying the indexref with a specified query enables retrieval of relevant products based on their similarity to the query. Next, embeddings are generated using Sentence-BERT to represent each product, and re-ranking is performed using cosine similarity.

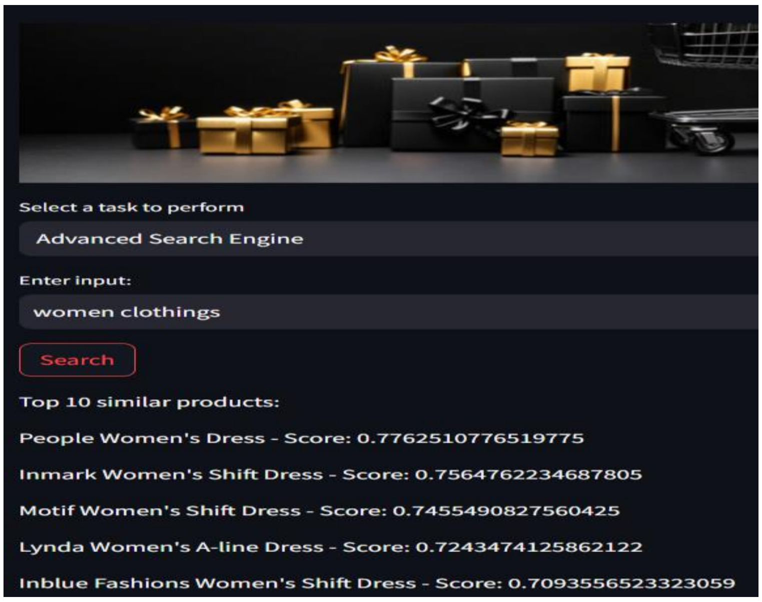
The cosine similarity(Wolfram Research (2007)) of two vectors A and B is given as,

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

Where n is the dimension of the vectors and Ai and Bi are the ith element.

The embeddings are created for both the query and each product, allowing for similarity scores to be calculated using cosine similarity. The resulting scores indicate the relevance of each product to the query, facilitating effective ranking. Finally, the results are presented, showcasing a list of products related to the search query "women clothing" along with their respective relevance scores.

This modeling process demonstrates the effectiveness of PyTerrier and Sentence-BERT in conducting advanced retrieval operations, enabling accurate and relevant product recommendations based on user queries. The resulting list of products reflects high relevance to the search query, showcasing the capability of the system to provide meaningful recommendations to users.



women clothings

Top 10 similar products:

People Women's Dress - Score: 0.7762510776519775

Inmark Women's Shift Dress - Score: 0.7564762234687805

Motif Women's Shift Dress - Score: 0.7455490827560425

Lynda Women's A-line Dress - Score: 0.7243474125862122

Inblue Fashions Women's Shift Dress - Score: 0.7093556523323059

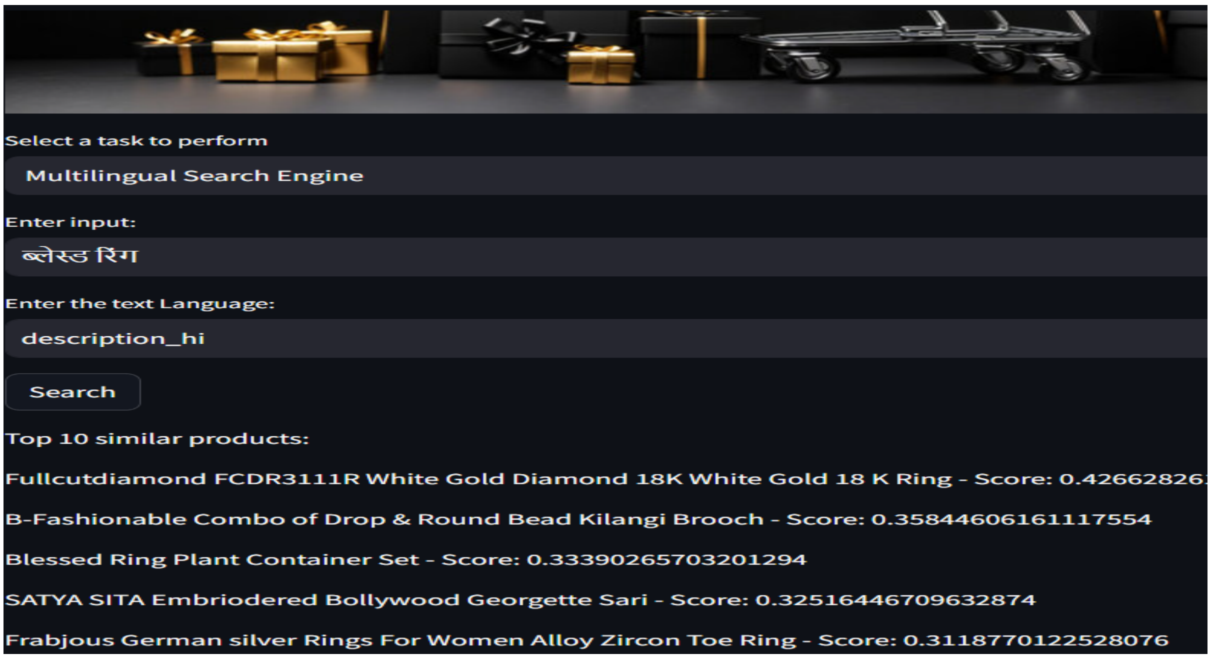
Fig 5: Advanced Search Engine

Incorporating the Multilingual Search Engine into the modeling phase involves several key steps. First, the dataset containing product descriptions is translated into Hindi, Bengali, and

Marathi using the GoogleTranslator API. This ensures that the search engine can cater to users with diverse linguistic preferences. The translated descriptions are then saved to a new CSV file for further processing.

Next, the translated dataset is loaded and preprocessed to prepare it for search operations. This includes normalizing descriptions and product names, as well as embedding the text data using the TextEmbedder module. The corpus embedding is then utilized to perform similarity-based search operations, enabling the retrieval of relevant products based on user queries.During the search process, users input their query, and the search engine retrieves the ten most relevant products based on the similarity score between the query and the product descriptions. Fuzzy matching techniques are applied to ensure accurate matching between descriptions and product names, enhancing the precision of search results.Finally, the search results are presented in a structured format, including the index of each product, the product name, the description, and the similarity score. The columns are appropriately reordered to improve readability, and the top matching products are displayed to the user.

Overall, the Multilingual Search Engine enhances the user experience by enabling users to search for products in their preferred language, thereby increasing accessibility and engagement on the platform.


Select a task to perform

Multilingual Search Engine

Enter input:

ब्लेस्ड रिंग

Enter the text Language:

description_hi

Search

Top 10 similar products:

Fullcutdiamond FCDR3111R White Gold Diamond 18K White Gold 18 K Ring - Score: 0.42662826

B-Fashionable Combo of Drop & Round Bead Kilangi Brooch - Score: 0.35844606161117554

Blessed Ring Plant Container Set - Score: 0.33390265703201294

SATYA SITA Embriodered Bollywood Georgette Sari - Score: 0.32516446709632874

Frabjous German silver Rings For Women Alloy Zircon Toe Ring - Score: 0.3118770122528076

Fig 6: Multilingual Search Engine

Product Categorization

Product categorization plays a pivotal role in the seamless functioning of e-commerce and retail platforms. With the exponential growth of online marketplaces and the continuous influx of new products, manually assigning categories to each item is not only time-consuming but also inefficient. This task demands intelligent and swift solutions to ensure accurate categorization while minimizing costs.

In our project, we address this challenge by developing a predictive model for product categorization using machine learning and natural language processing (NLP) techniques. By leveraging supervised classification algorithms, we aim to accurately classify products into their respective categories based on features extracted from product descriptions or images. The primary objective is to achieve high precision in categorization, thereby enhancing the efficiency and effectiveness of e-commerce platforms.

The significance of product categorization lies in its ability to streamline the browsing and search experience for customers. By organizing products into relevant categories, users can easily navigate through the vast inventory, find desired items more efficiently, and make informed purchasing decisions. Additionally, efficient categorization enables personalized recommendations and targeted marketing strategies, leading to improved customer satisfaction and retention.

In our project, we employ a combination of custom-built models, including a CNN-LSTM model and an RNN LSTM model as well as Fine tuned BERT Base model, for text classification and product categorization. These models are designed to effectively process and analyze product descriptions, extracting meaningful features to accurately predict the corresponding categories. By harnessing the power of deep learning and NLP, we aim to optimize the categorization process and enhance the overall functionality of e-commerce platforms.

Preprocessing:

In our project's preprocessing phase, we adhere to a systematic approach aimed at refining the product description data extracted from the Flipkart e-commerce dataset. Initially, upon loading the dataset containing various product attributes such as names, descriptions, and categories, we observe its substantial size, rendering it suitable for model training and testing. A detailed analysis of the dataset reveals that the majority of products fall under popular categories like

Clothing, Jewelry, Footwear, and Mobiles & Accessories. However, certain columns, notably 'brand' and 'retail_price', exhibit missing values, necessitating data cleaning steps.

To address this issue, we embark on a data cleaning journey by eliminating observations with null entries specifically in the 'description' column. This ensures the availability of high-quality data for training models without any incomplete information. Moreover, to enrich the dataset, we introduce a new column computing the word count in each product description before preprocessing. This addition enables us to gauge the distribution of description lengths and make informed decisions throughout the preprocessing phase. Additionally, we employ binning techniques to categorize descriptions based on their word counts, offering deeper insights into the distribution of description lengths.

In our meticulous preprocessing path, we undertake several essential steps to refine the textual data. We start by removing punctuation marks to prevent interference with subsequent analysis or modeling processes. Subsequently, we standardize the text format by replacing multiple whitespaces between terms with a single space and eliminating leading and trailing whitespaces for consistency. Lowercasing all text ensures uniformity and mitigates potential case sensitivity issues. Numeric values such as prices are replaced with a placeholder ('numbr') to standardize the text further. Common English stopwords are then removed to eliminate noise, and single characters are discarded to enhance data quality. Furthermore, domain-related stopwords like 'rs', 'flipkart', and 'buy' are excised to focus solely on relevant information for product categorization.

By meticulously implementing these preprocessing steps, we ensure that the product descriptions are refined, standardized, and optimized for subsequent machine learning model training. This strategic approach enhances the accuracy and efficacy of our models in categorizing products, ultimately leading to more dependable categorization outcomes.

Model building:

In the model building phase, we undertake essential preprocessing steps to prepare the product descriptions for input into our machine learning models. Firstly, we tokenize the product descriptions using the Tokenizer class, which converts text into numerical sequences based on word frequency. This step enables us to represent textual data in a format suitable for model training. The resulting sequences of tokenized words are then subjected to padding, ensuring uniform length across all descriptions. We set a maximum sequence length of 200 tokens to standardize the input size

Furthermore, we encode the target variable, which comprises product categories, using a label encoder to convert categorical labels into numerical representations. This transformation facilitates model training by converting category labels into a format compatible with machine learning algorithms. The number of unique categories determines the size of the output layer in our classification model.

For the train-test split, we partition the dataset into training and testing sets, allocating 80% of the data for training and 20% for testing. This division ensures that our models are trained on a substantial portion of the data while also evaluating their performance on unseen data.

Upon completion of the preprocessing steps, we ascertain the dimensions of the training and testing datasets to ensure consistency and correctness. The training set consists of 14,940 samples, each with a sequence length of 200 tokens, while the testing set comprises 3,736 samples.

Finally, we determine the vocabulary size, which represents the total number of unique words in the tokenized product descriptions. This metric aids in configuring the input layer of our neural network models, ensuring compatibility between the input data and the model architecture.

By meticulously following these steps, we prepare the product description data for model training, ensuring that it is appropriately formatted and ready for input into our machine learning algorithms.

<u>LSTM-RNN Model:</u>

In constructing our model architecture, we define the input layer to accommodate sequences with a length of 200 tokens, corresponding to the maximum length determined during preprocessing. This input layer serves as the entry point for our model, receiving tokenized product descriptions as input data. Next, we integrate an embedding layer, assigning a dense vector representation of 100 dimensions to each token. This layer aids in capturing semantic connections among words, leading to a deeper comprehension of the textual data.Subsequently, we utilize an LSTM (Long Short-Term Memory) layer, comprising 60 units, which processes the embedded sequences and captures long-term dependencies within the data. This LSTM layer is pivotal in capturing sequential patterns and contextual information from the input sequences.

Following the LSTM layer, we employ a dense layer consisting of 30 neurons, applying a ReLU activation function to introduce non-linearity into the model. This dense layer boosts the model's ability to grasp intricate patterns within the data and derive pertinent features from the output of the LSTM. Finally, we append a dense output layer with softmax activation, producing probability distributions over the 15 product categories present in our dataset. This layer computes the likelihood of each category given the input sequence, enabling the model to make accurate predictions. The overall architecture of our model comprises multiple layers interconnected to process the input data and generate predictions. Each layer contributes to the model's capability to extract meaningful features from the input descriptions and make accurate category predictions. After configuring the model, we designate RMSprop as the optimizer and employ sparse categorical cross-entropy as the loss function, which is well-suited for tasks involving multi-class classification. Furthermore, accuracy serves as a metric to assess the model's performance throughout the training process.

To prevent overfitting and ensure that we save the best-performing model, we employ the Model Checkpoint callback. This callback tracks the validation accuracy throughout the training process and stores the model exhibiting the highest accuracy on the validation set.During training, the model is fitted to the training data in batches, with an epoch representing a full pass through the entire dataset. We train the model for 5 epochs while validating its performance on the test set. The training process is monitored and logged to track the model's training and validation performance over time.

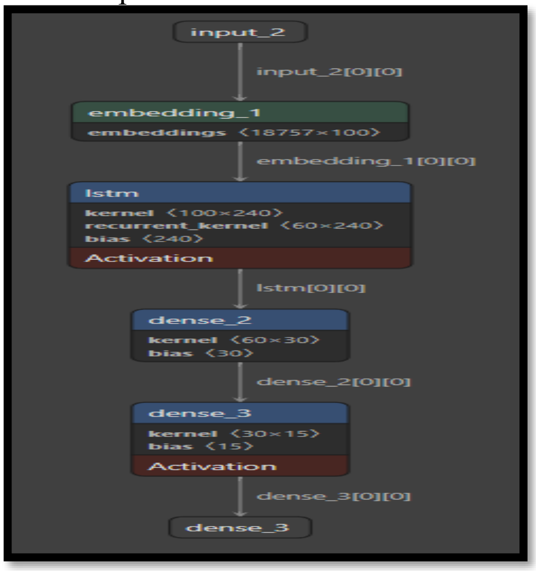
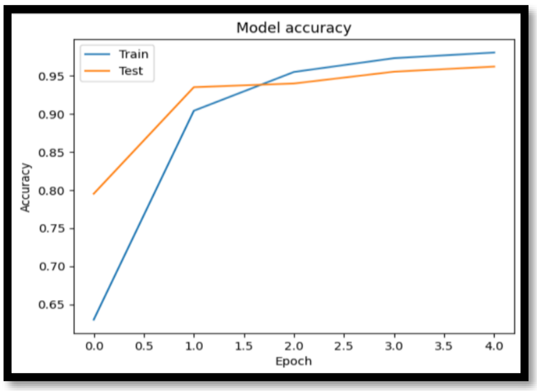


Fig 7:RNN LSTM Model Architecture



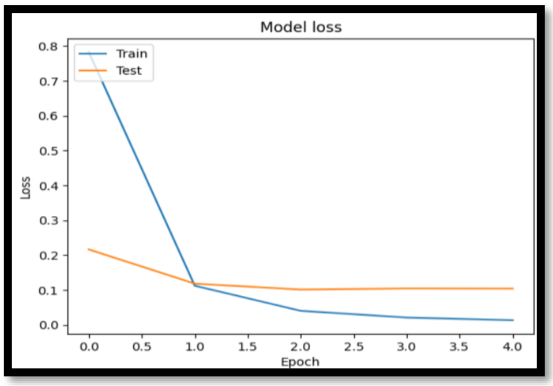


Fig 8:Graphs of Accuracy & Loss vs Epoch

These plots suggest that the model effectively minimizes loss and maximizes accuracy on both training and validation datasets. The quick convergence and the final accuracy values indicate that the model performs well with the given data and parameters.

The embedding layer converts every word in the input sequence into a dense vector of consistent size.

Given:

- `model_inp` is an input sequence of word indices with shape `(max_length)`.
- `vocab_size` is the size of the vocabulary.
- The embedding dimension is 100.

Mathematically:

- E is the embedding matrix of shape `(vocab_size, 100)`.
- X is the one-hot representation of `model_inp` of shape `(max_length, vocab_size)`.
- XembedX_{embed}Xembed is the output of the embedding layer of shape `(max_length, 100)`.

Xembed=X×E

The LSTM layer processes the embedded sequence and captures dependencies over time.

Given:

- The input to LSTM is Xembed with shape `(max_length, 100)`.
- The LSTM has 60 units.

The first Dense layer takes the output of the LSTM and applies a linear transformation followed by an optional ReLU activation.

Given:

- The input is h with shape `(60)`.

The computation is:

$z=W \cdot h+b$

$a=ReLU(z)$

Where:

- W is the weight matrix of shape `(60, 30)`.
- b is the bias vector of shape `(30)`.
- $ReLU(x)=max(0,x)$.

The final Dense layer produces the class probabilities using the softmax activation function.

Given:

- The input is a with shape `(30)`.

The computation is:

$zout=Wout \cdot a+bout$

$ypred=softmax(zout)$

<u>CNN-LSTM:</u>

In our CNN-LSTM model architecture we commence with an input layer configured to handle sequences of 200 tokens, aligning with the maximum length determined during preprocessing. This input layer serves as the gateway for our model, accepting tokenized product descriptions as input data. Subsequently, we introduce an embedding layer, which maps each token to a dense vector representation of dimension 100. This embedding layer plays a crucial role in capturing semantic relationships among words, enriching the model's comprehension of textual data.

After the embedding layer, we integrate a one-dimensional convolutional layer (Conv1D) equipped with 60 filters. This convolutional layer scans the input sequences to detect local patterns, leveraging the sliding window approach to capture features within small windows of text. Additionally, we apply a max-pooling layer (MaxPooling1D), which downsamples the output of the convolutional layer, retaining the most salient features while reducing the dimensionality of the data.

Next, we integrate an LSTM (Long Short-Term Memory) layer with 60 units, which processes the sequences and captures long-term dependencies within the data. This LSTM layer complements the local feature extraction capabilities of the convolutional layer, enhancing the model's ability to capture both local and global patterns within the text data.Subsequently, we include a dense layer comprising 30 neurons, followed by a final dense output layer with softmax activation, generating probability distributions across the 15 product categories in our dataset.

Our model architecture embodies a synergistic combination of convolutional and recurrent layers, designed to effectively process textual data and make accurate predictions. Each layer contributes to the model's proficiency in extracting meaningful features from input descriptions and facilitating precise category predictions.

Additionally, CNNs (Convolutional Neural Networks) have gained popularity in processing textual data due to their ability to automatically learn hierarchical feature representations. CNNs excel in capturing local patterns and features within text, making them well-suited for tasks such as text classification, sentiment analysis, and document summarization. Moreover, CNNs require fewer parameters compared to recurrent architectures like LSTMs, resulting in faster training times and reduced computational complexity. This efficiency, coupled with their effectiveness in capturing textual features, has led to the widespread adoption of CNNs in natural language processing tasks.

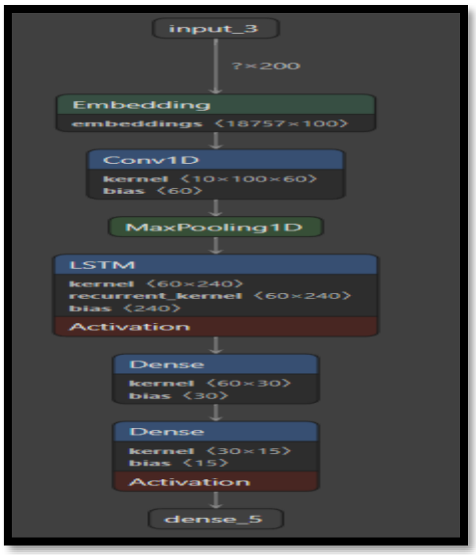
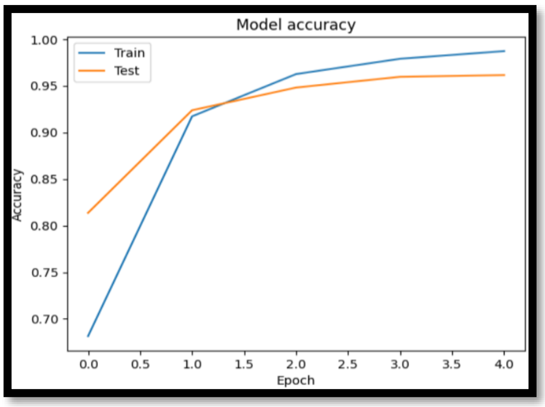


Fig 9: CNN-LSTM Model Architecture

Fig 9: CNN-LSTM Model Architecture



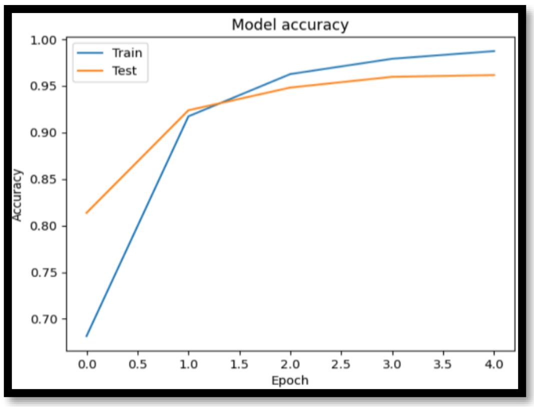


Fig 10: Graphs of Accuracy & Loss vs Epoch

These plots suggest that the model effectively minimizes loss and maximizes accuracy on both training and validation datasets. The quick convergence and the final accuracy values indicate that the model performs well with the given data and parameters.

The input layer is defined as: x∈Rmax_length

where `max_length` is the length of the input sequence. The input consists of a series of integers, with each integer corresponding to a word within the vocabulary.

The embedding layer maps each word in the input sequence to a dense vector in a higher-dimensional space: E∈Rvocab_size×100

zt=E[xt]

Here,E is the embedding matrix of shape `(vocab_size, 100)`, and xt is the t-th word in the input sequence. The output is: Z=(z1,z2,…,zmax_length) with shape `(max_length, 100)`.

The convolutional layer applies a set of filters to the input sequence. Each filter W of size (k,d).

Here k is the filter width (kernel size), and d is the depth of the input, produces an output by sliding over the sequence and performing dot products.

For filter W:

(W∗Z)[t]=σ(∑i=1 to k(Wk[i]⋅Z[t+i−1,:]+b))

where:

- Wis the weight matrix of the filter.
- b is the bias term.
- σ\sigmaσ is the activation function (here, it's linear by default).

The LSTM layer processes the sequence data and captures temporal dependencies. Each LSTM cell has the following equations:

The output of the LSTM layer is a sequence of hidden states with the final state being used as the output, with shape `(60)`.

The Dense layers apply a linear transformation followed by an activation function. For the first Dense layer:

y=W*h+b

n this context, W represents the weight matrix, b denotes the bias term, and h signifies the input originating from the LSTM layer. The shape of f{y}:y is `(30)`.

The final Dense layer produces the class probabilities using the softmax activation function.

Given:

- The input is a with shape `(30)`.

The computation is:

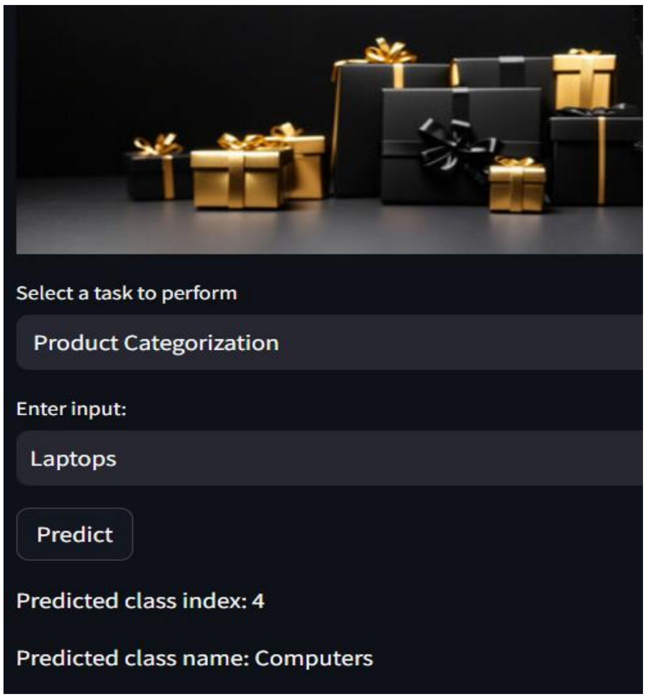
zout=Wout·a+bout

ypred=softmax(zout)

<u>Fine tuned BERT Base Model:</u>

The BERT (Bidirectional Encoder Representations from Transformers) base model is a groundbreaking innovation in natural language processing (NLP) and serves as a pivotal component of our product categorization system. Created by Google, BERT has revolutionized various NLP tasks with its bidirectional context understanding, enabling it to interpret words based on the complete sentence both forwards and backwards.

BERT's architecture is built on the transformer model, utilizing self-attention mechanisms to process text. This design allows BERT to manage long-range dependencies more effectively than traditional Recurrent Neural Networks (RNNs). Unlike sequential models, BERT's bidirectional training approach processes the entire sequence of words at once, capturing the context of a word from both preceding and following words, which results in a deeper understanding of the text.


Select a task to perform

Product Categorization

Enter input:

Laptops

Predicted class index: 4

Predicted class name: Computers

Fig11: Product Categorization

A significant aspect of BERT is its tokenization process, which uses WordPiece tokenization. This method breaks words into smaller subword units, allowing the model to handle a large vocabulary and manage rare or unknown words efficiently. For example, the word "unhappiness" might be split into ["un", "##happy", "##ness"], enabling the model to grasp its meaning even if the whole word is rare in the training data. This detailed tokenization is essential for accurately capturing the semantic nuances of product descriptions.

BERT is pre-trained on a massive text corpus using two unsupervised tasks: Masked Language Model (MLM) and Next Sentence Prediction (NSP). Within the MLM framework, certain input tokens are concealed, prompting the model to forecast these hidden tokens using the nearby context. Meanwhile, in NSP, BERT grasps the connection between two sentences, offering assistance in tasks that involve sentence pairs. Subsequent to pre-training, BERT's adaptability extends to task-specific objectives, such as our product categorization, by incorporating a task-specific output layer. The BERT base model consists of 12 layers (transformer blocks), 12 attention heads, and a hidden size of 768, resulting in 110 million parameters. This configuration balances complexity and performance, making it suitable for a variety of tasks without excessive computational requirements.

In this project, we fine-tuned a BERT base model to classify descriptions into specific categories. We began by loading the dataset, ensuring that all descriptions were valid, and converting them to string format. The labels were encoded using Label Encoder to facilitate the subsequent processing stages. We then split the data into features and labels, where descriptions formed the feature set and categories constituted the label set.

Utilizing the BERT tokenizer, we tokenized the descriptions, padding and truncating them to a maximum length of 512 tokens. This tokenization process yielded input IDs and attention masks, which were then used to generate embeddings. The pre-trained BERT base model (BERT-base-uncased) was employed to create these embeddings in smaller batches to efficiently handle memory constraints.

Given the nature of the dataset, which exhibited class imbalance, we applied the Synthetic Minority Over-sampling Technique (SMOTE) to ensure a balanced representation of all categories in the training data. This oversampling resulted in resampled embeddings and corresponding labels.

Subsequently, we saved the embeddings, labels, label encoder, tokenizer, BERT model, and the custom embedding creation function for future use. The dataset was then divided into training and validation sets. Both sets were converted into TensorFlow datasets, with appropriate shuffling and batching applied to prepare them for model training.

The fine-tuning involved defining a custom neural network model, consisting of an input layer for the BERT embeddings, followed by a dropout layer to mitigate overfitting, a dense layer with ReLU activation to capture non-linear relationships, and an output layer with softmax activation to provide probability distributions over the categories. The model was compiled with the Adam optimizer and sparse categorical cross-entropy loss, incorporating accuracy as a performance metric. This fine-tuning approach aimed to leverage the pre-trained capabilities of BERT while adapting it to the specific classification task at hand.

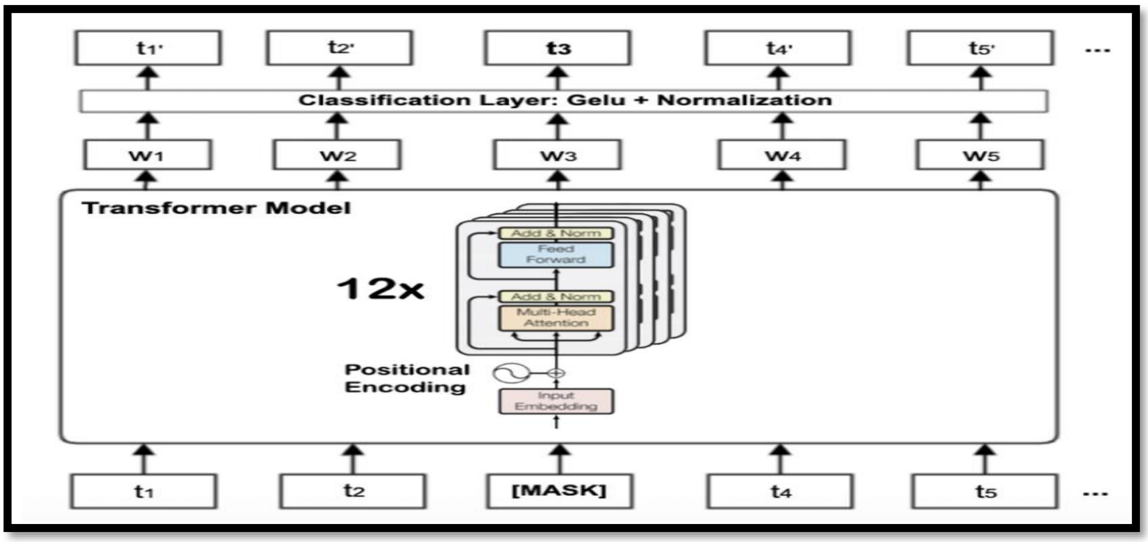


Fig 12: BERT Base Model Architecture (Source: Khalid et al.(2021))

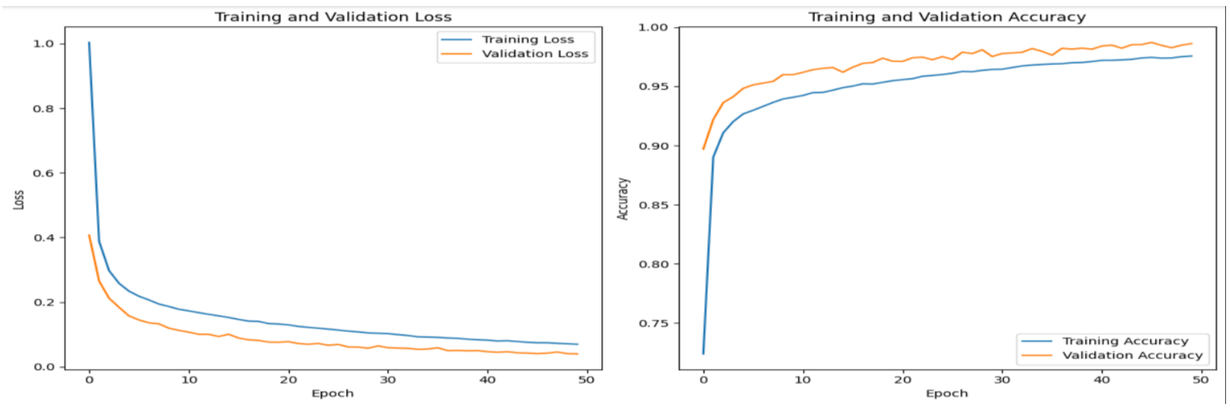


Fig 13: Graphs of Loss & Accuracy vs Epoch

The training loss (blue line) and validation loss (orange line) both show a steep decline in the initial epochs, indicating that the model is learning effectively and quickly reducing its error. After the initial drop, the losses continue to decrease gradually and stabilize, showing no significant signs of overfitting, as the validation loss does not increase.

The training accuracy (blue line) and validation accuracy (orange line) demonstrate a similar trend, with rapid improvement in the early epochs, followed by a gradual increase that plateaus towards the later epochs. The high final values for both training and validation accuracy suggest that the model generalizes well to unseen data.

Model Training and Evaluation

In our project, "Enhancing Ecommerce with Recommendation System, Advanced Search Engine, and Product Categorization," we developed and evaluated four neural network models: a Recurrent Neural Network with Long Short-Term Memory (RNN-LSTM), a Convolutional Neural Network combined with LSTM (CNN-LSTM), and a fine-tuned BERT base model. Each model was trained on tokenized product descriptions to classify products into 15 categories.

For training, we used the RMSprop optimizer and categorical cross-entropy loss function. The models were trained with a batch size of 64 for 5 epochs, using a 25% validation split. To monitor and save the best model based on validation accuracy, we employed the ModelCheckpoint callback. The training process included shuffling the data to ensure better generalization.

The fine-tuned BERT base model underwent extensive training, completing nearly 50 epochs, and achieved an accuracy of 98.62%.

After training, the models were evaluated on a validation set. Accuracy is a metric that measures the proportion of correctly predicted instances out of the total instances in the dataset. It is calculated using the following formula:

Accuracy = (Number of Correct Predictions) / (Total Number of Predictions)

The validation accuracies achieved by the models were as follows:

- Recurrent Neural Network with LSTM (RNN-LSTM): 96.12%
- Convolutional Neural Network combined with LSTM (CNN-LSTM): 96.22%
- Fine-tuned BERT base model: 98.62%

These results demonstrate that while all models performed well, the fine-tuned BERT base model achieved the highest accuracy in classifying product categories, surpassing the performance of the other models.

RESULTS

Our project has successfully integrated several advanced techniques to enhance the ecommerce platform. We developed a content-based filtering recommendation system, an advanced search engine, and predictive models for product categorization. Below are the detailed results for each component.

The content-based filtering recommendation system utilizes user preferences and product attributes to deliver personalized recommendations. By analyzing features of items with which users have interacted, the system suggests similar items, thereby increasing user engagement and satisfaction. This method ensures that users receive relevant products tailored to their interests, enhancing their overall shopping experience.

Using PyTerrier and Sentence-BERT, we created an advanced search engine capable of handling multilingual queries. This search engine retrieves relevant results based on semantic understanding, significantly enhancing the user search experience. Additionally, integrating DeepTextSearch has further improved search capabilities, ensuring accurate and contextually appropriate search results across multiple languages. This multilingual support broadens the platform's accessibility, providing a seamless search experience regardless of the user's language.

We developed and assessed three deep learning models for product categorization: a Recurrent Neural Network with Long Short-Term Memory (RNN-LSTM) and a Convolutional Neural Network combined with LSTM (CNN-LSTM). These models were trained on tokenized product descriptions to classify products into 15 categories. The validation accuracies achieved by the models were as follows: the RNN-LSTM model achieved 96.14%, and the CNN-LSTM model achieved 96.22%. Notably, our Fine-Tuned BERT Base model achieved the highest accuracy at 98.62%. These results demonstrate that the Fine-Tuned BERT model provided the highest accuracy, while the RNN-LSTM and CNN-LSTM models also performed strongly. The predictive models significantly contribute to the accuracy and efficiency of the product categorization system, ensuring correct classification of products, thereby improving the overall user experience on the ecommerce platform.

Overall, the implementation of these advanced techniques has resulted in a more robust, user-friendly ecommerce system, offering personalized recommendations, efficient multilingual search capabilities, and accurate product categorization. This comprehensive approach not only enhances the user experience but also supports the platform's ability to cater to a diverse and global customer base.

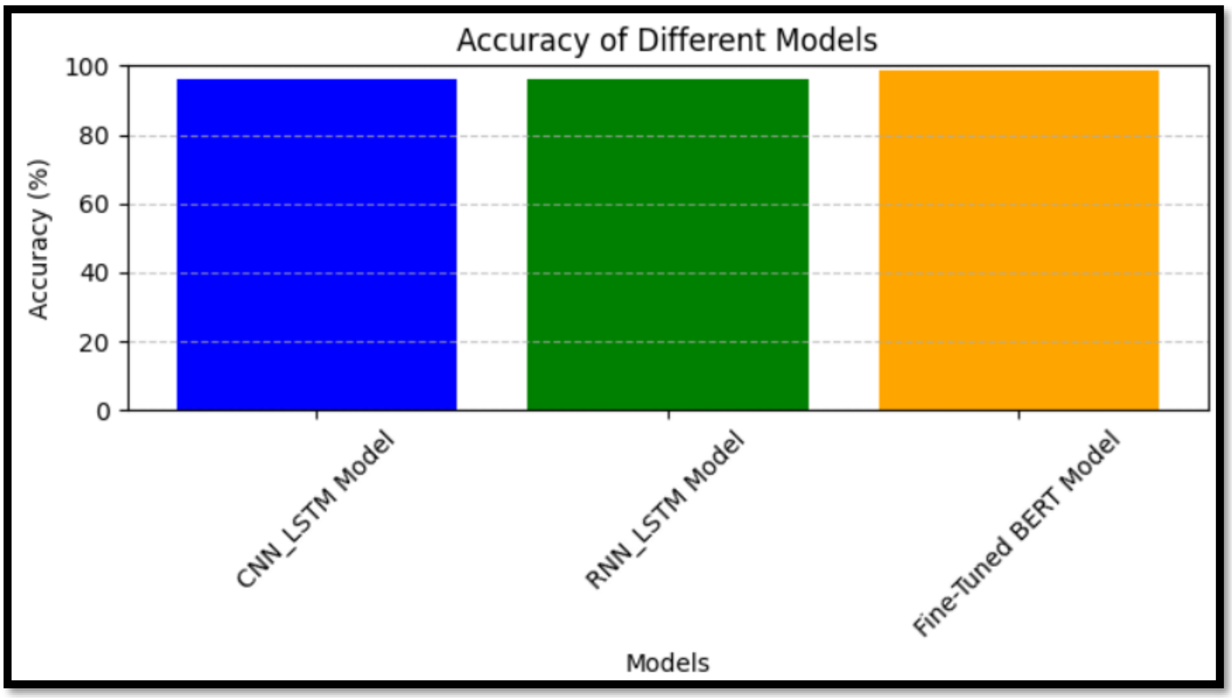


Fig 14: Comparison of accuracies of different models used

DISCUSSION

Our project aimed to enhance an e-commerce platform by developing advanced systems for personalized recommendations, sophisticated search, and efficient product categorization, all leveraging deep learning.

We created a content-based recommendation system using TF-IDF vectors to analyze product descriptions and provide personalized suggestions, improving user engagement. The advanced search engine employs word embeddings like Word2Vec to understand search query context, delivering accurate results. Integration of PyTerrier and Sentence-BERT further enhances search and recommendation accuracy.

Supporting multilingual queries through Deep Text Search broadens accessibility, catering to a diverse user base. For product categorization, we developed predictive models to automate classification, reducing manual effort and costs. Our Fine-Tuned BERT Base model achieved the highest accuracy at 98.62%, outperforming RNN-LSTM and CNN-LSTM models.

System performance optimization was crucial, ensuring efficiency and scalability to handle large data volumes and peak usage times. Overall, these advancements have created a robust, user-friendly platform, enhancing user experience, engagement, and sales, while catering to a global audience.

CONCLUSION

In conclusion, our project has successfully implemented advanced techniques to significantly enhance the functionality and user experience of an e-commerce platform. The content-based recommendation system, utilizing TF-IDF vectors, effectively delivers personalized product suggestions, thereby increasing user engagement and satisfaction. The sophisticated search engine, leveraging word embeddings and models like PyTerrier and Sentence-BERT, ensures contextually accurate and relevant search results, improving user satisfaction and retention.

The integration of multilingual support through Deep Text Search has expanded the platform's accessibility, making it more inclusive for a global audience. The development of predictive models for product categorization, particularly the Fine-Tuned BERT Base model with a 98.62% accuracy, has streamlined the classification process, reducing manual efforts and operational costs.

By optimizing system performance for efficiency and scalability, the platform can handle large volumes of data and peak user activity without compromising responsiveness. Overall, these advancements have created a robust, user-friendly e-commerce system, enhancing the user experience, driving higher engagement, and boosting sales. This comprehensive approach positions the platform to effectively cater to a diverse and global customer base.

REFERENCES:

Meshal Alfarhood, Jianlin Cheng(2018). "DeepHCF: A Deep Learning Based Hybrid Collaborative Filtering Approach for Recommendation Systems". DOI: DOI:10.1109/ICMLA.2018.00021

Poonam Sharma, Lokesh Yadav(2020). "Movie Recommendation System Using Item Based Collaborative Filtering". DOI: https://doi.org/10.21276/ijircst.2020.8.4.2

Yunjiang Jiang,Yue Shang,Hongwei Shen, Wenyun Yang,Yun Xiao(2020). "Fine-tune BERT for E-commerce Non-Default Search Ranking". DOI: https://doi.org/10.48550/arXiv.2008.09689

Yunjiang Jiang, Yue Shang, Ziyang Liu B, Hongwei Shen, Yun Xiao,Wei Xiong, Sulong Xu, Weipeng Yan, Di Jiny(2020). "BERT2DNN: BERT Distillation with Massive Unlabeled Data for Online E-Commerce Search". DOI: https://doi.org/10.48550/arXiv.2010.10442

Lakshya Kumar, Sagnik Sarkar(2022). "ListBERT: Learning to Rank E-commerce products with Listwise BERT". DOI: https://doi.org/10.48550/arXiv.2206.15198

Wenqi Fan, Zihuai Zhao, Jiatong Li, Yunqing Liu, Xiaowei Mei, Yiqi Wang, Zhen Wen, Fei Wang, Xiangyu Zhao, Jiliang Tang, and Qing Li(2024). "Recommender Systems in the Era of Large Language Models (LLMs)". DOI:https://doi.org/10.48550/arXiv.2307.02046

Xu Huang, Jianxun Lian, Yuxuan Lei, Jing Yao, Defu Lian, Xing Xie(2024). "Recommender AI Agent: Integrating Large Language Models for Interactive Recommendations". DOI:https://doi.org/10.48550/arXiv.2308.16505

Jianghong Zhou and Bo Liu and Jhalak Nilesh Acharya Yao Hong and Kuang-chih Lee and Musen Wen(2023). "Leveraging Large Language Models for Enhanced Product Descriptions in eCommerce". DOI: https://doi.org/10.48550/arXiv.2310.18357

Jing Yao, Wei Xu, Jianxun Lian, Xiting Wang, Xiaoyuan Yi and Xing Xie(2023). "Knowledge Plugins: Enhancing Large Language Models for Domain-Specific Recommendations". DOI: https://doi.org/10.48550/arXiv.2311.10779

Dario Di Palma(20217). "Retrieval-augmented Recommender System: Enhancing Recommender Systems with Large Language Models". DOI: https://doi.org/10.1145/3604915.3608889

Xuejiao Wang and Chao Liu(2023). "Design of Personalized News Recommendation System Based on an Improved User Collaborative Filtering Algorithm". DOI: 10.1155/2023/9898337

Zeyu Chena, Kailun Jianb(2023). "Design of Digital Media Recommendation System Based on User Behavior Analysis and Collaborative Filtering Algorithm". DOI: 10.1109/ICNETIC59568.2023.00164

Illia Balush, Victoria Vysotska and Solomiia Albota(2021). "Recommendation System Development Based on Intelligent Search, NLP and Machine Learning Methods".

Ms. Shakila Shaikh Dr. Sheetal Rathi,Asst Prof. Prachi Janrao(2017). "Recommendation System in E-Commerce Websites: A Graph Based Approached". DOI: DOI:10.1109/IACC.2017.0189

Soodabeh Sarafrazi, Darwin Wheeler , David Garcia, Shane Henrikson, Naveed Sharif, Hui Wu(2023). "Enhancing Search Engine Optimization in Healthcare and Clinical Domains with Natural Language Processing and Graph Techniques". DOI: 10.1007/978-3-031-52216-1_1

Wolfram Research (2007), CosineDistance, Wolfram Language function, https://reference.wolfram.com/language/ref/CosineDistance.html.

Khalid, Usama,Beg, Mirza,Arshad, Muhammad,(2021).RUBERT: A Bilingual Roman Urdu BERT Using Cross Lingual Transfer Learning.

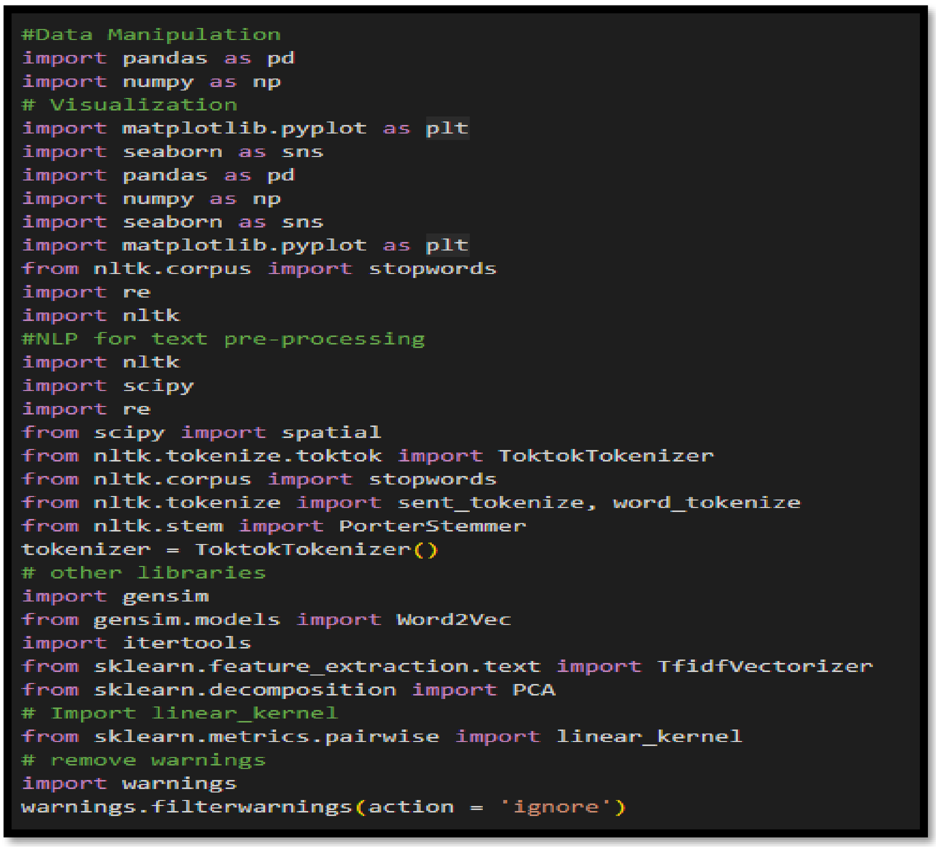
Hyeyoung Ko , Suyeon Lee ,Yoonseo Park, Anna Choi (2021), A Survey of Recommendation Systems: Recommendation Models, Techniques, and Application Fields DOI: https://doi.org/10.3390/electronics11010141

James F. Allen(2003), Natural language processing DOI: https://dl.acm.org/doi/abs/10.5555/1074100.1074630

Akiko Aizawa (2003), An information-theoretic perspective of tf–idf measures,DOI: https://doi.org/10.1016/S0306-4573(02)00021-3

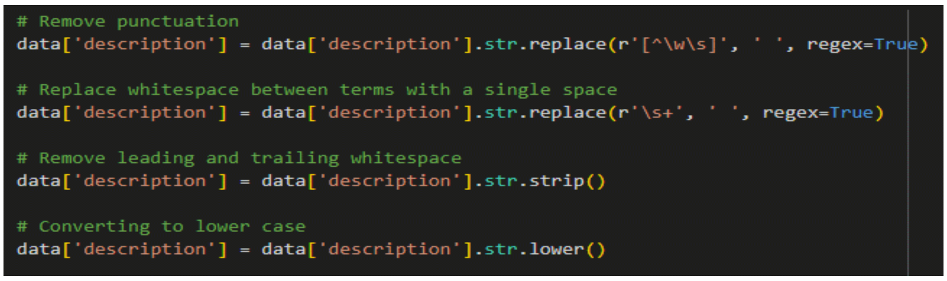
APPENDIX

Importing Libraries for search engine and recommendation system

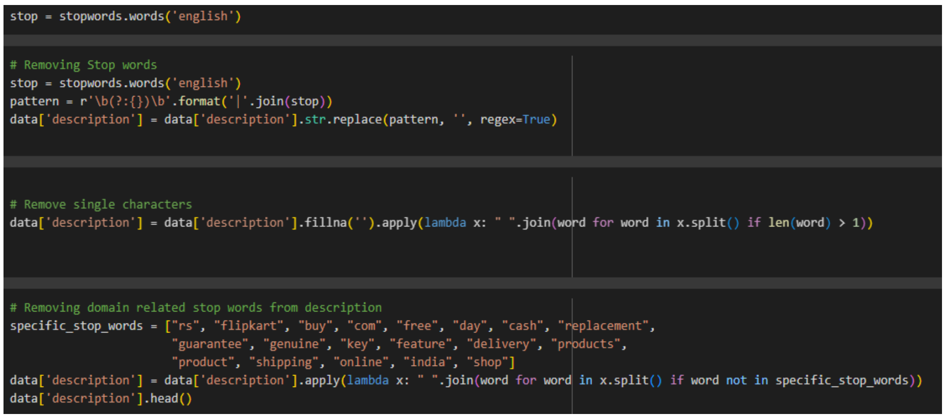


```
#Data Manipulation
import pandas as pd
import numpy as np
# Visualization
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from nltk.corpus import stopwords
import re
import nltk
#NLP for text pre-processing
import nltk
import scipy
import re
from scipy import spatial
from nltk.tokenize.toktok import ToktokTokenizer
from nltk.corpus import stopwords
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.stem import PorterStemmer
tokenizer = ToktokTokenizer()
# other libraries
import gensim
from gensim.models import Word2Vec
import itertools
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import PCA
# Import linear_kernel
from sklearn.metrics.pairwise import linear_kernel
# remove warnings
import warnings
warnings.filterwarnings(action = 'ignore')
```

Preprocessing

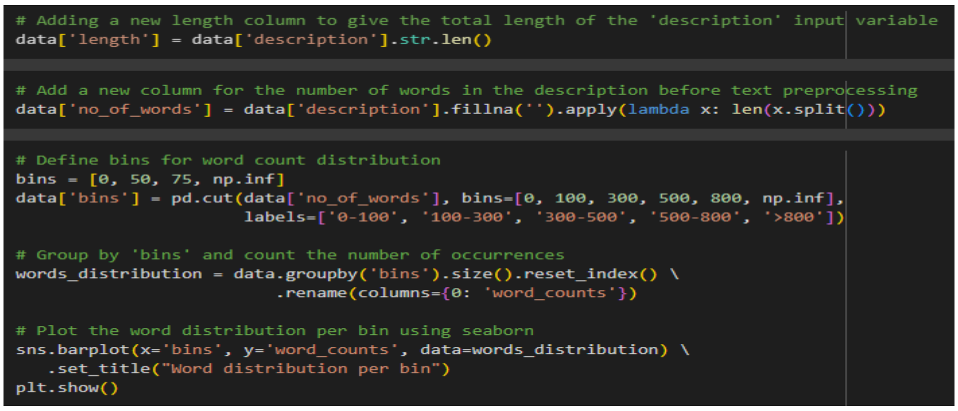


```
# Remove punctuation
data['description'] = data['description'].str.replace(r'[^\w\s]', ' ', regex=True)

# Replace whitespace between terms with a single space
data['description'] = data['description'].str.replace(r'\s+', ' ', regex=True)

# Remove leading and trailing whitespace
data['description'] = data['description'].str.strip()

# Converting to lower case
data['description'] = data['description'].str.lower()
```



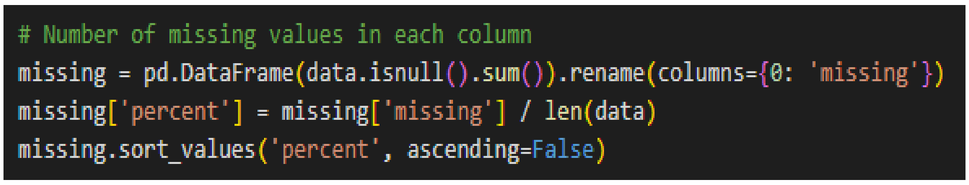
```
stop = stopwords.words('english')

# Removing Stop words
stop = stopwords.words('english')
pattern = r'\b(?:{})\b'.format('|'.join(stop))
data['description'] = data['description'].str.replace(pattern, '', regex=True)

# Remove single characters
data['description'] = data['description'].fillna('').apply(lambda x: " ".join(word for word in x.split() if len(word) > 1))

# Removing domain related stop words from description
specific_stop_words = ["rs", "flipkart", "buy", "com", "free", "day", "cash", "replacement",
                       "guarantee", "genuine", "key", "feature", "delivery", "products",
                       "product", "shipping", "online", "india", "shop"]
data['description'] = data['description'].apply(lambda x: " ".join(word for word in x.split() if word not in specific_stop_words))
data['description'].head()
```

Checking the word distribution in every document



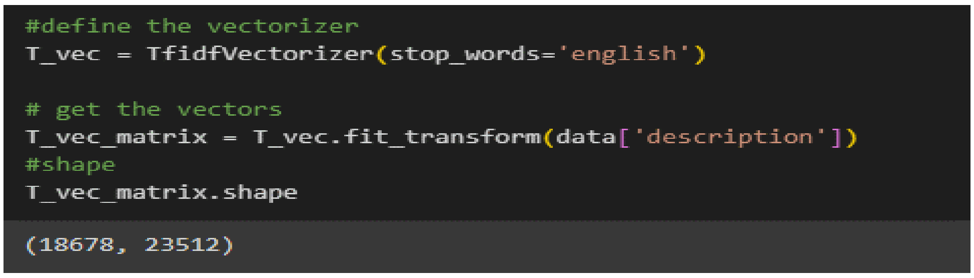
```python
# Adding a new length column to give the total length of the 'description' input variable
data['length'] = data['description'].str.len()

# Add a new column for the number of words in the description before text preprocessing
data['no_of_words'] = data['description'].fillna('').apply(lambda x: len(x.split()))

# Define bins for word count distribution
bins = [0, 50, 75, np.inf]
data['bins'] = pd.cut(data['no_of_words'], bins=[0, 100, 300, 500, 800, np.inf],
                      labels=['0-100', '100-300', '300-500', '500-800', '>800'])

# Group by 'bins' and count the number of occurrences
words_distribution = data.groupby('bins').size().reset_index() \
                         .rename(columns={0: 'word_counts'})

# Plot the word distribution per bin using seaborn
sns.barplot(x='bins', y='word_counts', data=words_distribution) \
   .set_title("Word distribution per bin")
plt.show()
```

Checking the missing Value percentage for every column



```python
# Number of missing values in each column
missing = pd.DataFrame(data.isnull().sum()).rename(columns={0: 'missing'})
missing['percent'] = missing['missing'] / len(data)
missing.sort_values('percent', ascending=False)
```

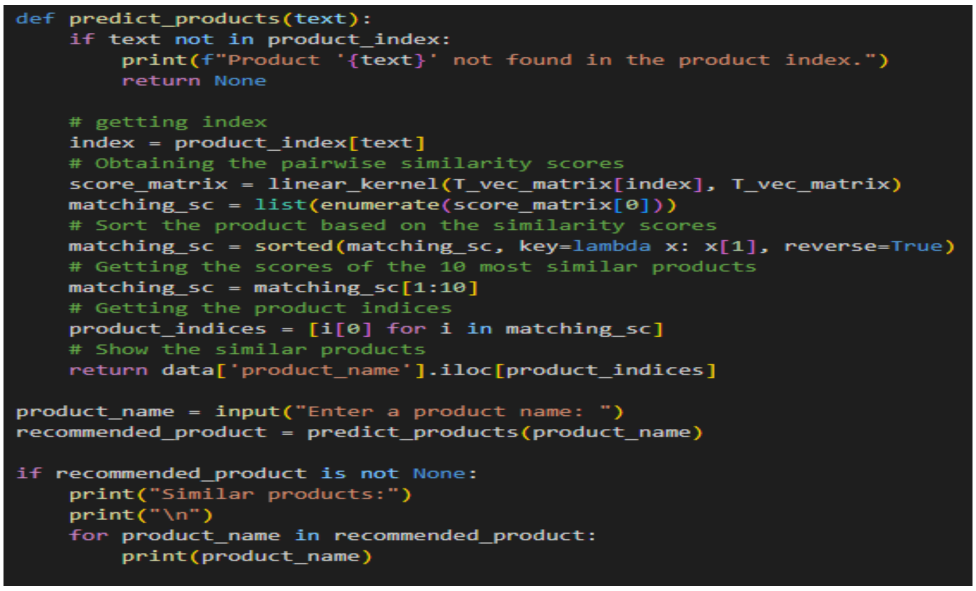
Defining the Tf-idf Vectorizer



```python
#define the vectorizer
T_vec = TfidfVectorizer(stop_words='english')

# get the vectors
T_vec_matrix = T_vec.fit_transform(data['description'])
#shape
T_vec_matrix.shape
```

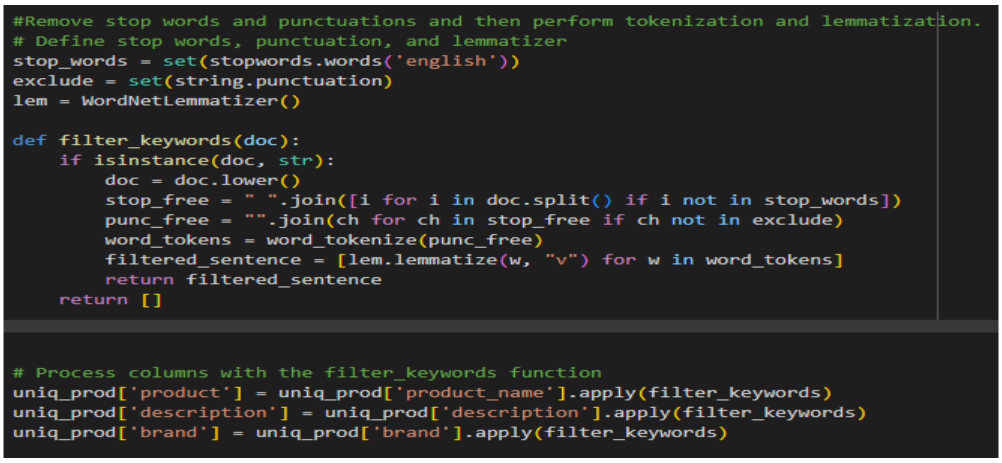
```
(18678, 23512)
```

Recommending the similar products



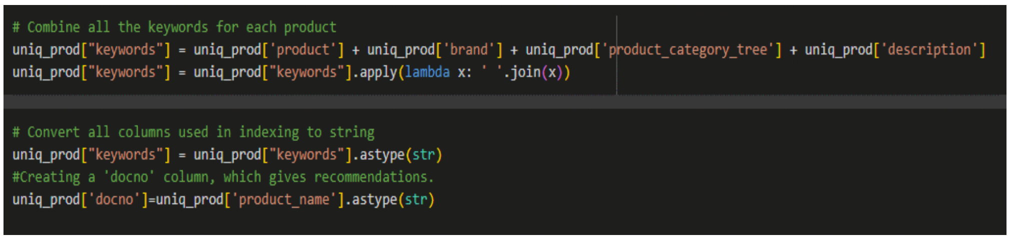
```python
def predict_products(text):
    if text not in product_index:
        print(f"Product '{text}' not found in the product index.")
        return None

    # getting index
    index = product_index[text]
    # Obtaining the pairwise similarity scores
    score_matrix = linear_kernel(T_vec_matrix[index], T_vec_matrix)
    matching_sc = list(enumerate(score_matrix[0]))
    # Sort the product based on the similarity scores
    matching_sc = sorted(matching_sc, key=lambda x: x[1], reverse=True)
    # Getting the scores of the 10 most similar products
    matching_sc = matching_sc[1:10]
    # Getting the product indices
    product_indices = [i[0] for i in matching_sc]
    # Show the similar products
    return data['product_name'].iloc[product_indices]

product_name = input("Enter a product name: ")
recommended_product = predict_products(product_name)

if recommended_product is not None:
    print("Similar products:")
    print("\n")
    for product_name in recommended_product:
        print(product_name)
```

Preprocessing for advanced search engine



```python
#Remove stop words and punctuations and then perform tokenization and lemmatization.
# Define stop words, punctuation, and lemmatizer
stop_words = set(stopwords.words('english'))
exclude = set(string.punctuation)
lem = WordNetLemmatizer()

def filter_keywords(doc):
    if isinstance(doc, str):
        doc = doc.lower()
        stop_free = " ".join([i for i in doc.split() if i not in stop_words])
        punc_free = "".join(ch for ch in stop_free if ch not in exclude)
        word_tokens = word_tokenize(punc_free)
        filtered_sentence = [lem.lemmatize(w, "v") for w in word_tokens]
        return filtered_sentence
    return []
```

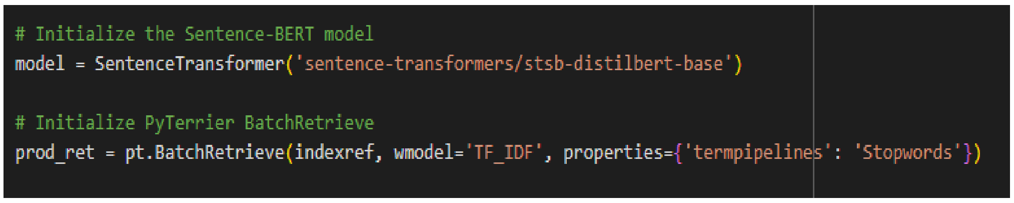
```python
# Process columns with the filter_keywords function
uniq_prod['product'] = uniq_prod['product_name'].apply(filter_keywords)
uniq_prod['description'] = uniq_prod['description'].apply(filter_keywords)
uniq_prod['brand'] = uniq_prod['brand'].apply(filter_keywords)
```



```python
# Combine all the keywords for each product
uniq_prod["keywords"] = uniq_prod['product'] + uniq_prod['brand'] + uniq_prod['product_category_tree'] + uniq_prod['description']
uniq_prod["keywords"] = uniq_prod["keywords"].apply(lambda x: ' '.join(x))
```

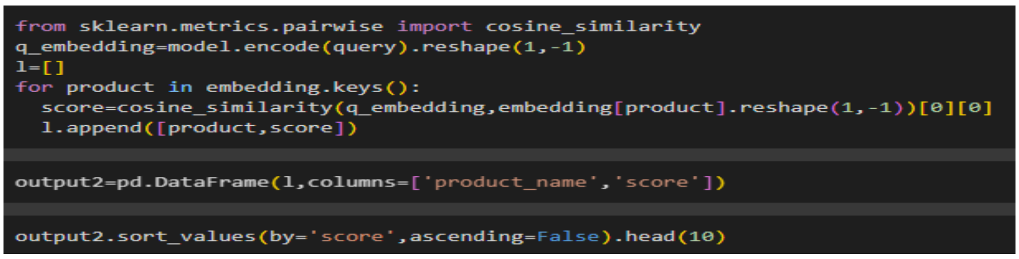
```python
# Convert all columns used in indexing to string
uniq_prod["keywords"] = uniq_prod["keywords"].astype(str)
#Creating a 'docno' column, which gives recommendations.
uniq_prod['docno']=uniq_prod['product_name'].astype(str)
```

Initializing Sentence BERT and Pyterrier



```python
# Initialize the Sentence-BERT model
model = SentenceTransformer('sentence-transformers/stsb-distilbert-base')

# Initialize PyTerrier BatchRetrieve
prod_ret = pt.BatchRetrieve(indexref, wmodel='TF_IDF', properties={'termpipelines': 'Stopwords'})
```

List of products based on the search item

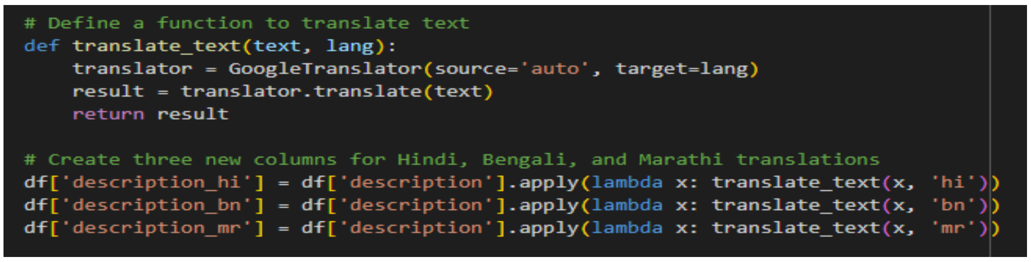


```python
from sklearn.metrics.pairwise import cosine_similarity
q_embedding=model.encode(query).reshape(1,-1)
l=[]
for product in embedding.keys():
  score=cosine_similarity(q_embedding,embedding[product].reshape(1,-1))[0][0]
  l.append([product,score])
```

```python
output2=pd.DataFrame(l,columns=['product_name','score'])
```

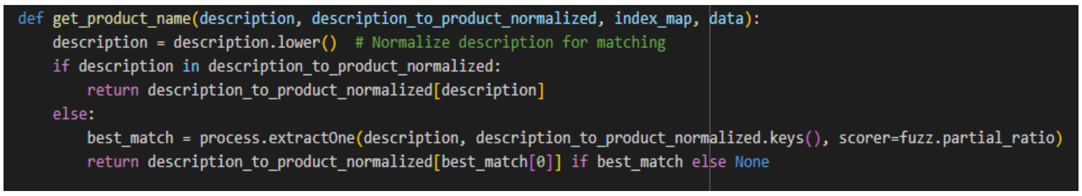
```python
output2.sort_values(by='score',ascending=False).head(10)
```

Preprocessing for Multilingual Search engine



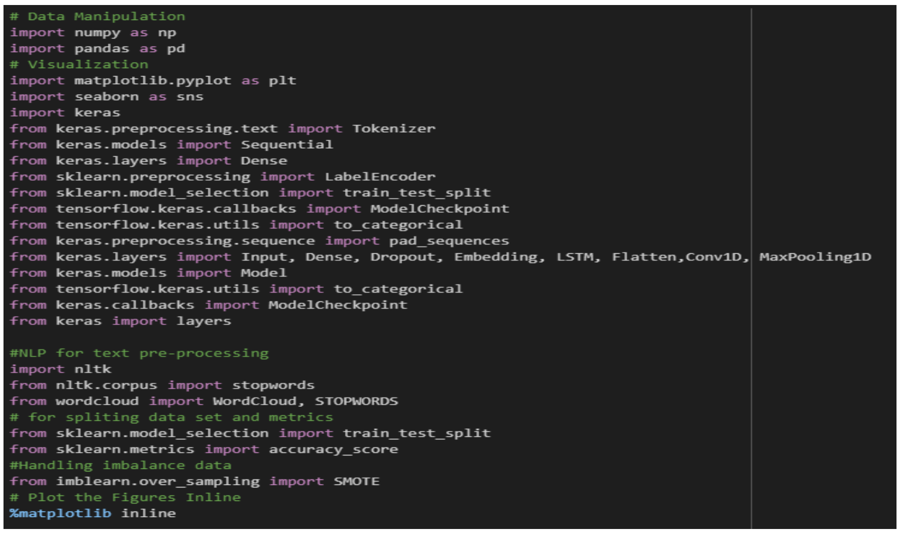
```python
# Define a function to translate text
def translate_text(text, lang):
    translator = GoogleTranslator(source='auto', target=lang)
    result = translator.translate(text)
    return result

# Create three new columns for Hindi, Bengali, and Marathi translations
df['description_hi'] = df['description'].apply(lambda x: translate_text(x, 'hi'))
df['description_bn'] = df['description'].apply(lambda x: translate_text(x, 'bn'))
df['description_mr'] = df['description'].apply(lambda x: translate_text(x, 'mr'))
```

List of products based on multilingual search



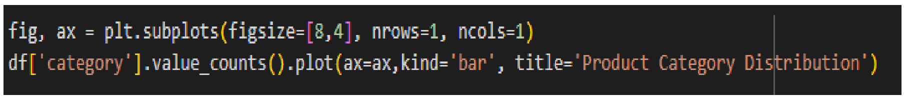
```
def get_product_name(description, description_to_product_normalized, index_map, data):
    description = description.lower()  # Normalize description for matching
    if description in description_to_product_normalized:
        return description_to_product_normalized[description]
    else:
        best_match = process.extractOne(description, description_to_product_normalized.keys(), scorer=fuzz.partial_ratio)
        return description_to_product_normalized[best_match[0]] if best_match else None
```

Importing libraries for product categorization



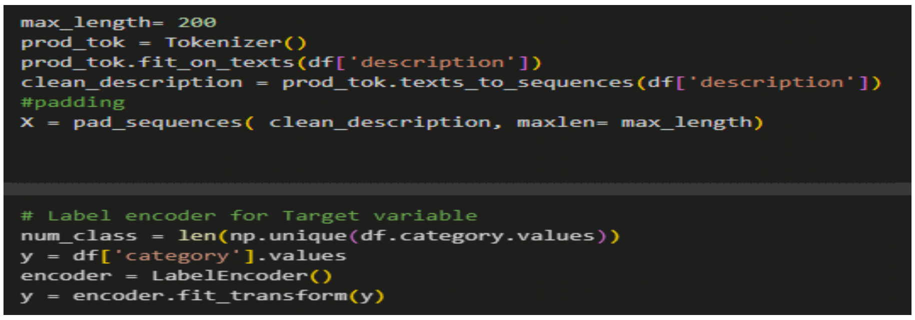
```
# Data Manipulation
import numpy as np
import pandas as pd
# Visualization
import matplotlib.pyplot as plt
import seaborn as sns
import keras
from keras.preprocessing.text import Tokenizer
from keras.models import Sequential
from keras.layers import Dense
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.utils import to_categorical
from keras.preprocessing.sequence import pad_sequences
from keras.layers import Input, Dense, Dropout, Embedding, LSTM, Flatten,Conv1D, MaxPooling1D
from keras.models import Model
from tensorflow.keras.utils import to_categorical
from keras.callbacks import ModelCheckpoint
from keras import layers

#NLP for text pre-processing
import nltk
from nltk.corpus import stopwords
from wordcloud import WordCloud, STOPWORDS
# for spliting data set and metrics
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
#Handling imbalance data
from imblearn.over_sampling import SMOTE
# Plot the Figures Inline
%matplotlib inline
```

Checking the number of products in each category



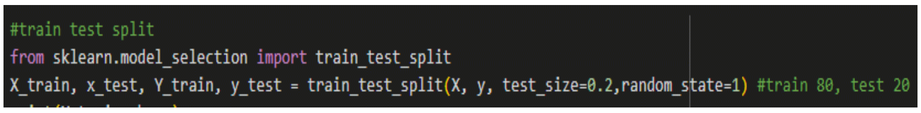
```
fig, ax = plt.subplots(figsize=[8,4], nrows=1, ncols=1)
df['category'].value_counts().plot(ax=ax,kind='bar', title='Product Category Distribution')
```

Preparing data for model training



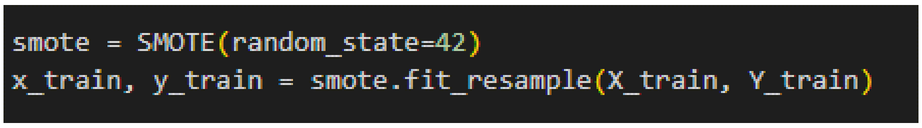
```
max_length= 200
prod_tok = Tokenizer()
prod_tok.fit_on_texts(df['description'])
clean_description = prod_tok.texts_to_sequences(df['description'])
#padding
X = pad_sequences( clean_description, maxlen= max_length)
```

```
# Label encoder for Target variable
num_class = len(np.unique(df.category.values))
y = df['category'].values
encoder = LabelEncoder()
y = encoder.fit_transform(y)
```



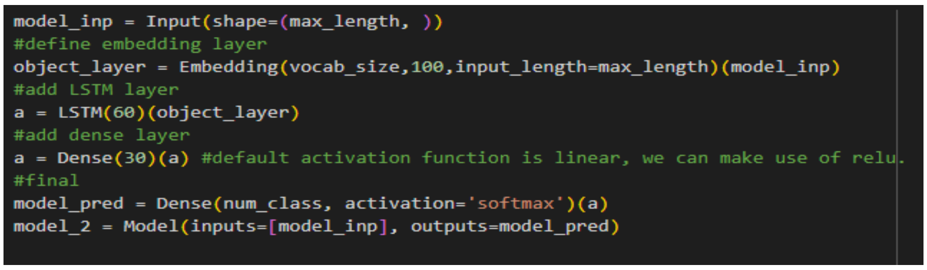
```
#train test split
from sklearn.model_selection import train_test_split
X_train, x_test, Y_train, y_test = train_test_split(X, y, test_size=0.2,random_state=1) #train 80, test 20
```

Applying SMOTE to balance the dataset



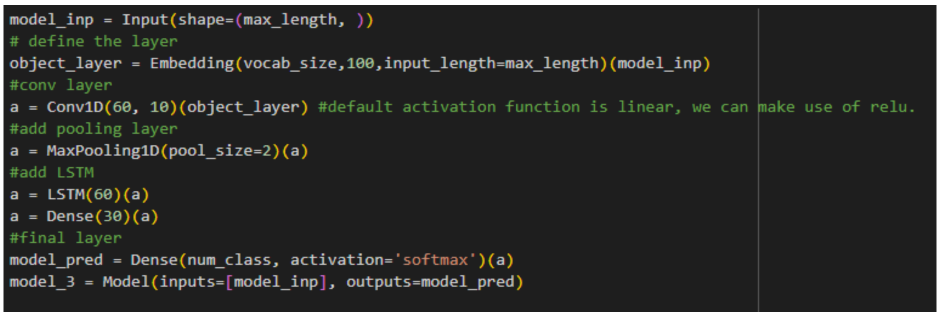
```
smote = SMOTE(random_state=42)
x_train, y_train = smote.fit_resample(X_train, Y_train)
```

LSTM-RNN



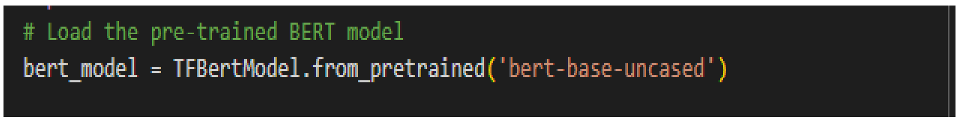
```
model_inp = Input(shape=(max_length, ))
#define embedding layer
object_layer = Embedding(vocab_size,100,input_length=max_length)(model_inp)
#add LSTM layer
a = LSTM(60)(object_layer)
#add dense layer
a = Dense(30)(a) #default activation function is linear, we can make use of relu.
#final
model_pred = Dense(num_class, activation='softmax')(a)
model_2 = Model(inputs=[model_inp], outputs=model_pred)
```

CNN-LSTM

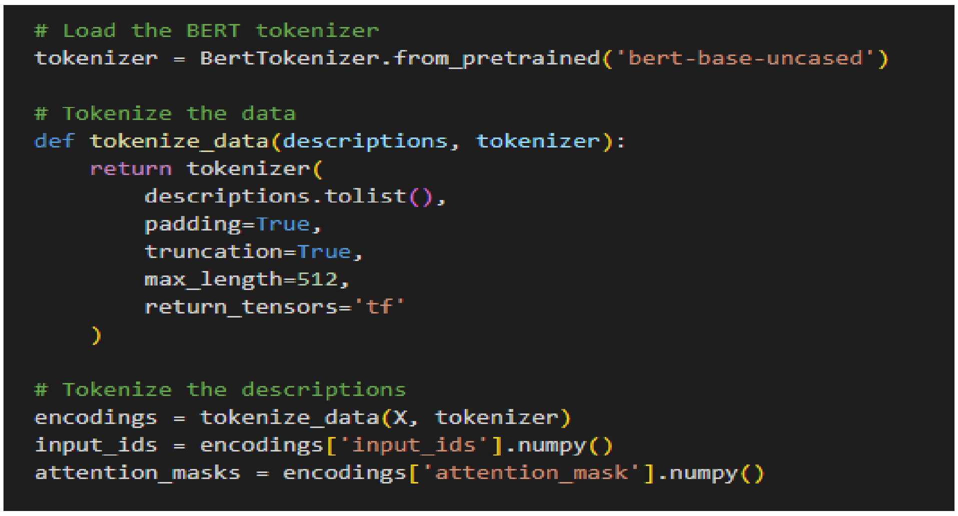


```
model_inp = Input(shape=(max_length, ))
# define the layer
object_layer = Embedding(vocab_size,100,input_length=max_length)(model_inp)
#conv layer
a = Conv1D(60, 10)(object_layer) #default activation function is linear, we can make use of relu.
#add pooling layer
a = MaxPooling1D(pool_size=2)(a)
#add LSTM
a = LSTM(60)(a)
a = Dense(30)(a)
#final layer
model_pred = Dense(num_class, activation='softmax')(a)
model_3 = Model(inputs=[model_inp], outputs=model_pred)
```

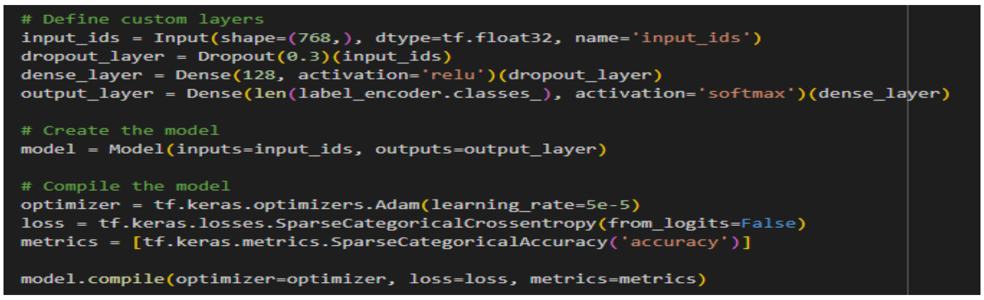
BERT Base Model



```
# Load the pre-trained BERT model
bert_model = TFBertModel.from_pretrained('bert-base-uncased')
```

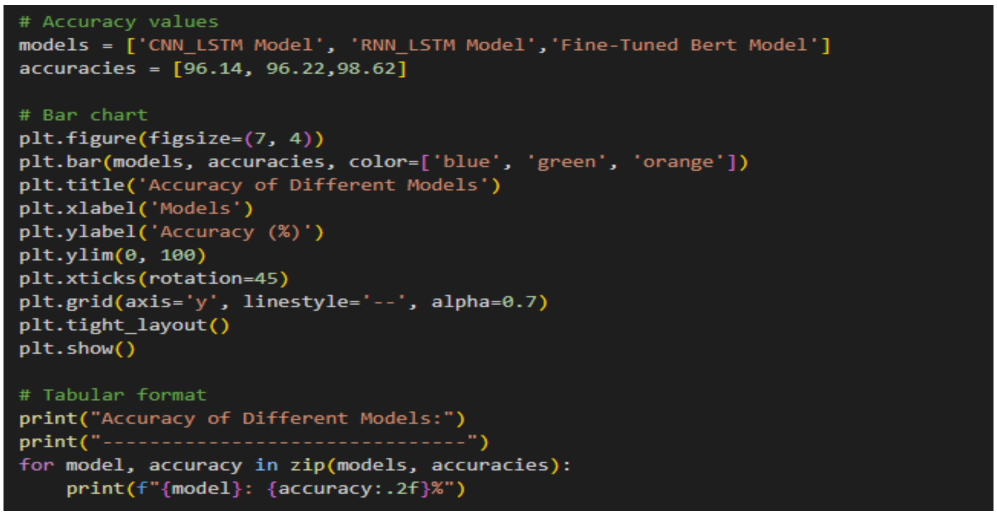


```
# Load the BERT tokenizer
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')

# Tokenize the data
def tokenize_data(descriptions, tokenizer):
    return tokenizer(
        descriptions.tolist(),
        padding=True,
        truncation=True,
        max_length=512,
        return_tensors='tf'
    )

# Tokenize the descriptions
encodings = tokenize_data(X, tokenizer)
input_ids = encodings['input_ids'].numpy()
attention_masks = encodings['attention_mask'].numpy()
```



```
# Define custom layers
input_ids = Input(shape=(768,), dtype=tf.float32, name='input_ids')
dropout_layer = Dropout(0.3)(input_ids)
dense_layer = Dense(128, activation='relu')(dropout_layer)
output_layer = Dense(len(label_encoder.classes_), activation='softmax')(dense_layer)

# Create the model
model = Model(inputs=input_ids, outputs=output_layer)

# Compile the model
optimizer = tf.keras.optimizers.Adam(learning_rate=5e-5)
loss = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False)
metrics = [tf.keras.metrics.SparseCategoricalAccuracy('accuracy')]

model.compile(optimizer=optimizer, loss=loss, metrics=metrics)
```

Comparing the accuracies of all models



```
# Accuracy values
models = ['CNN_LSTM Model', 'RNN_LSTM Model','Fine-Tuned Bert Model']
accuracies = [96.14, 96.22,98.62]

# Bar chart
plt.figure(figsize=(7, 4))
plt.bar(models, accuracies, color=['blue', 'green', 'orange'])
plt.title('Accuracy of Different Models')
plt.xlabel('Models')
plt.ylabel('Accuracy (%)')
plt.ylim(0, 100)
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()

# Tabular format
print("Accuracy of Different Models:")
print("--------------------------------")
for model, accuracy in zip(models, accuracies):
    print(f"{model}: {accuracy:.2f}%")
```

Proforma 4

Undertaking from the PG student while submitting his/her final dissertation to his respective institute

Ref. No. _________

I , the following student

Sr. No.	Sequence of students names on a dissertation	Students name	Name of the Institute & Place	Email & Mobile
1.	First Author	Kinjal Bandopadhyay	SIG	Email:22070243028@sig.ac.in Mobile:+918620886966

Note: Put additional rows in case of more number of students

hereby give an undertaking that the dissertation Vision Based Driver Behaviour Monitoring for Automotive System _been checked for its Similarity Index/Plagiarism through Turnitin software tool; and that the document has been prepared by me and it is my original work and free of any plagiarism. It was found that:

1.	The Similarity Index (SI) was: *(Note: SI range: 0 to 10%; if SI is >10%, then authors cannot communicate ms;* ***attachment of SI report is mandatory****)*	5%
2.	The ethical clearance for research work conducted obtained from: *(Note: Name the consent obtaining body; if 'not appliable' then write so)*	NA
3.	The source of funding for research was: *(Note: Name the funding agency; or write 'self' if no funding source is involved)*	Self
4.	Conflict of interest: *(Note: Tick √ whichever is applicable)*	No
5.	The material (adopted text, tables, figures, graphs, etc.) as has been obtained from other sources, has been duly acknowledged in the manuscript: *(Note: Tick √ whichever is applicable)*	Yes

In case if any of the above-furnished information is found false at any point in time, then the University authorities can take action as deemed fit against all of us.

Full Name &
Signature of the student

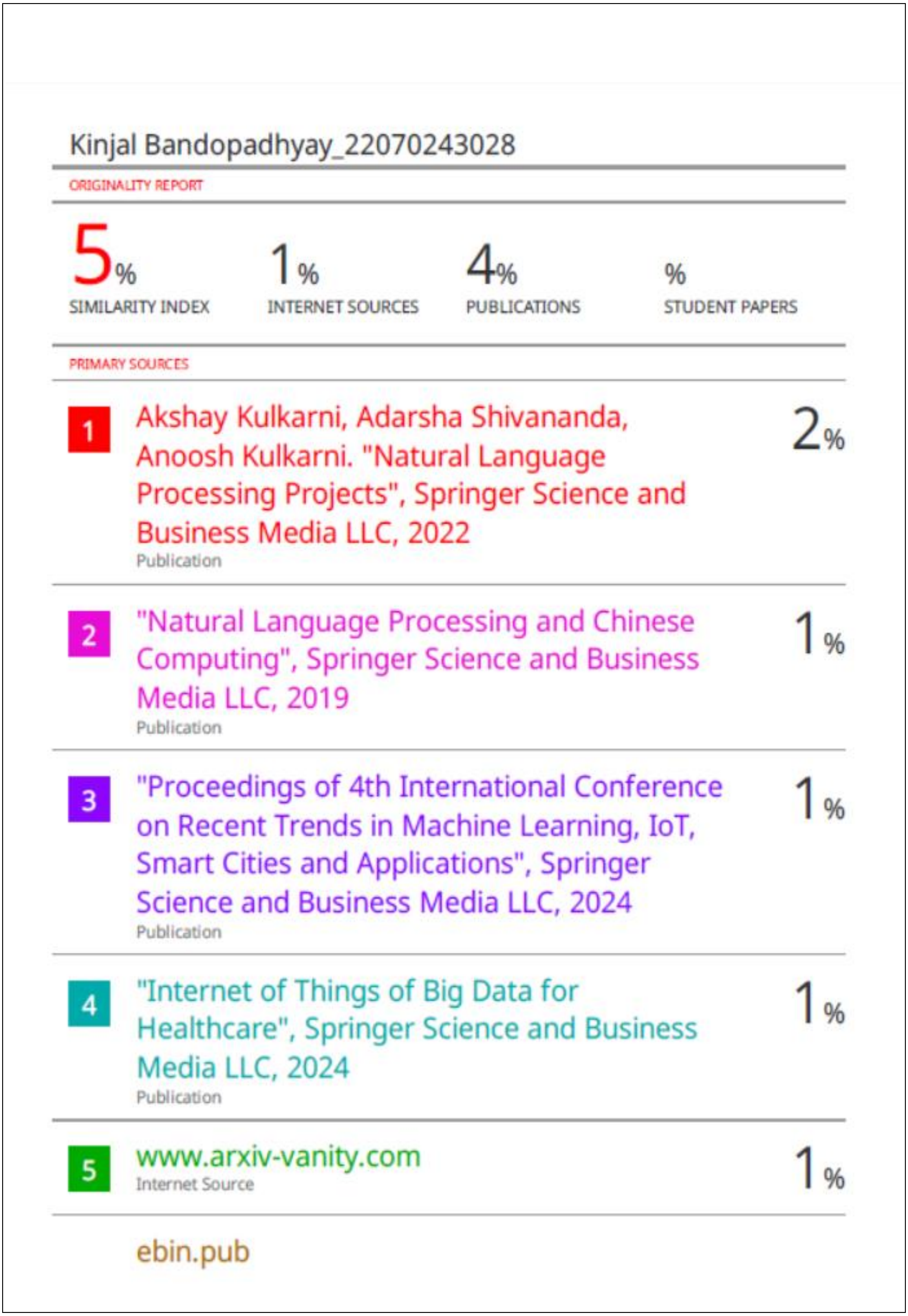
Name &
Signature of SIU Guide/Mentor

Date:

Endorsement by
Academic Integrity Committee (AIC)

Place: Pune

Note: It is mandatory that the Similarity Index report of plagiarism (only first page) should be appended to the UG/PG dissertation



Kinjal Bandopadhyay_22070243028

ORIGINALITY REPORT

5%	1%	4%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Akshay Kulkarni, Adarsha Shivananda, Anoosh Kulkarni. "Natural Language Processing Projects", Springer Science and Business Media LLC, 2022 Publication	2%
2	"Natural Language Processing and Chinese Computing", Springer Science and Business Media LLC, 2019 Publication	1%
3	"Proceedings of 4th International Conference on Recent Trends in Machine Learning, IoT, Smart Cities and Applications", Springer Science and Business Media LLC, 2024 Publication	1%
4	"Internet of Things of Big Data for Healthcare", Springer Science and Business Media LLC, 2024 Publication	1%
5	www.arxiv-vanity.com Internet Source	1%
	ebin.pub	