Nexus ML: A Content-Based Recommendation and Classification Engine for Scientific Abstracts

[1]Himanshu Sharma , [2]Manish Sharma , [3]Prabhakar Sharma
[1] Coder, Team Leader , [2] Paper writer , [3]Assistant Professor
[1]Computer Science and Engineering (Artificial Intelligence)
[1]Shri Shankaracharya Institute of Professional Management and Technology , Raipur , India
[1]manishmanishsharma23@gmail.com [2]hs8962205@gmail.com [3]prabhakar.sharma@ssipmt.com

Abstract

The exponential growth of academic literature across digital repositories has fundamentally altered the landscape of scholarly research. While this proliferation ensures diverse scientific inquiries are documented, it introduces severe information overload for researchers and academicians. Finding highly relevant literature and accurately categorizing new, unlabelled pre-prints have become daunting tasks that traditional search engines handle inefficiently. To mitigate this challenge, this paper presents the design, development, and evaluation of a dual-purpose Research Paper Classifier and Recommender. This integrated machine learning system bridges the gap between document discovery and automated taxonomy by utilizing advanced natural language processing (NLP) and neural network architectures. The proposed system features two primary operational pipelines accessible via a unified web interface. The first pipeline is a multi-label classifier engineered to predict academic subject areas (such as Machine Learning, Artificial Intelligence, and Computer Vision) directly from raw abstract text. By utilizing a customized Term Frequency-Inverse Document Frequency (TF-IDF) vectorization layer integrated directly into a TensorFlow/Keras MultiLayer Perceptron (MLP), the system effectively captures local word contexts and heavily weights domain-specific terminology. The second pipeline is a semantic recommendation engine that transcends simple lexical matching. By analyzing paper titles, it identifies and retrieves contextually adjacent literature, successfully grouping documents by their underlying methodologies and architectural frameworks.

The system was trained and rigorously evaluated on a comprehensive, deduplicated dataset of over 41,000 papers from arXiv.The classification module achieved an exceptional categorical accuracy of 99.45% on the testing subset, demonstrating highly stable convergence and minimal loss. Furthermore, these backend machine learning models were successfully deployed into a streamlined, interactive Streamlit web application. This frontend provides a frictionless experience, allowing researchers to intuitively paste text and receive instantaneous categorizations and curated reading lists

Introduction

In the contemporary academic ecosystem, the dissemination of knowledge occurs at a staggering velocity. Thousands of research papers and pre-prints are published globally every single day across platforms such as arXiv, IEEE, and PubMed. While this rapid exchange of information is the bedrock of scientific advancement, it has paradoxically created a substantial bottleneck in the research lifecycle. Researchers, particularly those entering new domains or conducting interdisciplinary studies, are frequently overwhelmed by the sheer volume of available literature. The traditional approach to literature review relies heavily on manual filtering, which is highly time-consuming, prone to human error, and increasingly inefficient. Users often struggle to formulate the perfect search query, leading to an overwhelming flood of loosely related documents or a restrictive trickle of exact matches that omit crucial foundational work.

Simultaneously, the administration and organization of this literature present a massive challenge. When new papers are drafted, authors and repository managers must accurately categorize them into specific, sometimes overlapping, sub-disciplines. Misclassification effectively buries cutting-edge research. Therefore, the necessity for intelligent, automated tools has never been more critical. While isolated recommendation algorithms or classification models exist, translating their success into a cohesive, user-facing academic domain presents highly complex challenges. Academic literature possesses dense, highly specialized vocabulary and complex semantic structures.

This project focuses on the development of a unified Research Paper Classifier and Recommender, engineered to understand the complex content of scientific documents. The primary objective is to move beyond simple keyword matching toward deep semantic understanding. By deploying a multi-label neural network, the system can instantly analyze an abstract and assign it to multiple appropriate domains (e.g., categorizing a paper under both 'Computer Science - Machine Learning' and 'Statistics - Machine Learning' simultaneously). Parallel to this, the system provides real-time reading recommendations based on input titles.

This paper is structured to provide a comprehensive overview of the design and implementation of this system. Following this introduction, the Literature Review explores the historical progression of information retrieval. The Problem Statement formally defines the technical hurdles of multi-label prediction and semantic gaps. The Theory and

Methodology section details the algorithmic architecture, including text vectorization, neural network topology, and the Streamlit deployment. Finally, the Results and Discussion sections present the quantitative performance of the system, concluding with an analysis of its practical impact on the academic community.

Literature Review

The evolution of document retrieval and classification systems spans several decades, mirroring broader advancements in computer science and artificial intelligence. Early information retrieval systems were predominantly based on Boolean logic and simple keyword matching. While foundational, these systems suffered from severe limitations, primarily their inability to rank documents by relevance or classify text with overlapping domains. The introduction of the Vector Space Model (VSM) marked a significant paradigm shift. By representing documents and queries as vectors in a multidimensional space, VSM allowed for the computation of continuous similarity scores. This era popularized the Term Frequency-Inverse Document Frequency (TF-IDF) weighting scheme, which remains a resilient baseline in modern text mining due to its efficiency in highlighting discriminative terms within a corpus.

As digital libraries expanded, the need for automated document categorization grew. Early machine learning approaches utilized algorithms like Naive Bayes and Support Vector Machines (SVMs) for text classification. However, academic papers frequently belong to multiple disciplines—a single paper might encompass robotics, computer vision, and artificial intelligence. Traditional single-label classifiers are ill-equipped for this. The literature subsequently shifted toward Multi-Label Classification frameworks, utilizing techniques like Binary Relevance or Classifier Chains to predict multiple concurrent categories.

To address the limitations of purely lexical approaches, contemporary research has embraced deep learning. Multi-Layer Perceptrons (MLPs) and Convolutional Neural Networks (CNNs) have proven highly adept at uncovering non-linear patterns in highdimensional text data. More recently, the field has been revolutionized by transformer architectures, which excel at understanding deep contextual meaning. In parallel, recommendation systems have evolved from simple collaborative filtering to contentbased semantic matching. Despite these advancements, a gap remains in practical, deployed applications. Much of the literature focuses on theoretical offline metrics, ignoring the end-user experience. Consequently, contemporary research advocates for integrated systems that combine robust backend classification (using optimized NLP vectorization and neural networks) with accessible, realtime frontend interfaces. This project builds upon these foundations, creating a hybrid pipeline that is both highly accurate and immediately usable.

Problem Statement

Despite the wealth of available academic literature and continuous advancements in machine learning, designing an effective, integrated Research Paper Classifier and Recommender involves navigating a series of complex technical challenges. The overarching problem is how to accurately predict multiple subject areas for unstructured scientific abstracts while simultaneously surfacing the most relevant research papers for a user, minimizing the time spent on manual literature searches. This broad challenge can be decomposed into specific algorithmic and architectural problems.

First is the issue of Multi-Label Taxonomy and high-dimensional data. Academic preprints, such as those on arXiv, rarely fit into a single distinct category. A paper on graph neural networks might belong to computer science,

statistics, and systems engineering simultaneously. The system must utilize an architecture capable of predicting a multihot encoded array of labels without suffering from the curse of dimensionality or catastrophic forgetting.

The second major hurdle is the "Semantic Gap." Traditional keyword-based retrieval systems fail to understand the contextual meaning of scientific text. If a system relies purely on raw word counts, it fails to capture the relationships between adjacent words. The text preprocessing pipeline must bridge this gap by capturing local context (such as bigrams) and appropriately weighting rare scientific jargon against common English stop words, ensuring the neural network receives high-quality numerical representations of the text.

Thirdly, the system must contend with computational scalability and real-time inference latency. Generating predictions and recommendations requires processing dense mathematical arrays. If the system is to be deployed as a web application, it cannot afford extended loading times. The architecture must employ efficient vector indexing and optimized data structures to ensure recommendations and classifications are delivered with minimal latency.

Finally, there is the challenge of accessibility. A highly accurate machine learning model is virtually useless to the broader academic community if it remains confined to a Jupyter Notebook. The problem is not merely building the algorithm, but successfully exporting the model architectures, serialized weights, and vocabulary states into a seamless, interactive frontend deployment that any researcher can utilize without writing code.

Theory/Methodology

The architecture of the Research Paper Classifier and Recommender is designed as a robust, dual-pipeline machine learning system utilizing TensorFlow and Keras. The methodology is structured into rigorous data preprocessing, text vectorization, neural network classification, and the deployment of an interactive web frontend via Streamlit.

1. Data Acquisition and Preprocessing:

The foundational dataset utilized is the arxiv_data_210930-054931.csv, initially comprising 56,181 records with three primary features: terms (subject labels), titles, and abstracts. Initial exploratory data analysis confirmed zero null values. To ensure model integrity, a strict deduplication process was executed based on the titles column, reducing the corpus to 41,105 unique papers. Because the system performs multi-label classification, the string representations of the labels were parsed into Python lists using the ast.literal_eval library. The dataset was further filtered to remove extremely rare terms (occurrences) to prevent the model from learning noise. The deduplicated, filtered dataset was rigorously split using ScikitLearn's train_test_split. A stratified split preserved the distribution of the multi-label terms, resulting in a robust training set of 34,741 papers, a validation set of 1,930 papers, and a testing set of 1,931 papers.

2. Feature Engineering and Text Vectorization:

To convert the unstructured abstracts into a machine-readable format, the system leverages the Keras TextVectorization layer. Instead of basic word counts, the vectorizer was configured with ngrams=2 (capturing bigrams to maintain localized word context) and an output_mode="tf_idf". This applies Term Frequency-Inverse Document Frequency weighting directly within the TensorFlow pipeline, ensuring that specific scientific jargon is weighted more heavily than common stop words. The vectorizer was adapted exclusively on the training dataset to prevent data leakage, and its configuration and weights were serialized (text_vectorizer_config.pkl, vocab.pkl) for frontend integration.

3. Model Architecture (The MLP Classifier):

For the classification pipeline, a Shallow Multi-Layer Perceptron (MLP) was constructed using the keras.Sequential API. The architecture relies on a dense hidden layer consisting of 512 neurons utilizing the Rectified Linear Unit (ReLU) activation function. To combat overfitting on high-dimensional TF-IDF vectors, Dropout layers were strategically integrated. The model was trained to minimize loss while predicting the multi-hot encoded categories. An EarlyStopping callback was implemented during training, monitoring validation metrics to halt training once performance plateaued, thereby preserving the optimal model weights (saved as model.h5 and model.keras).

4. Frontend Deployment and Inference:

To bridge the gap between backend machine learning and user accessibility, the system is deployed using Streamlit (app.py). The web application provides a graphical user interface where researchers can input an abstract for real-time multilabel classification or a title to receive semantic recommendations via precalculated embeddings. The frontend utilizes tf.saved_model.load and custom inversion functions (invert_multi_hot) to translate raw tensor outputs back into human-readable arXiv taxonomies (e.g., cs.LG), featuring robust error handling and asynchronous loading states for a frictionless experience.

Results

The quantitative and qualitative evaluation of the dual-purpose system demonstrates highly robust, real-time performance across its multi-label prediction and recommendation pipelines, validating the chosen TensorFlow/Keras methodology.

In the domain of abstract classification, the MLP model exhibited exceptional precision. Upon evaluating the finalized model against the hold-out datasets, the system achieved a categorical accuracy of 99.45% on the test set and a corresponding 99.47% on the validation set. This incredibly tight alignment between testing and validation accuracy indicates that the model generalizes exceptionally well to unseen data, successfully avoiding overfitting. The training dynamics substantiate the model's stability; the binary accuracy curve exhibited a rapid ascent within the first epoch, stabilizing near the 0.995 threshold, while the validation loss remained remarkably stable around the 0.018 mark across 5 epochs.

Qualitatively, the system handles complex, unstructured text with high fidelity. For instance, when tested against a dense, domain-specific abstract detailing "Graph Neural Networks (GNNs)" and "Multi-Level Attention Pooling," the deployed model accurately inverted the multi-hot encoded predictions to classify the paper under 'cs.LG' (Computer Science - Machine Learning) and 'stat.ML' (Statistics - Machine Learning). This inference was executed efficiently, averaging approximately 278ms per step.

Parallel to the classification engine, the recommendation module demonstrated significant contextual awareness. When querying the system with the seminal title "Attention is All you Need," the generated outputs successfully surfaced highly relevant, theoretically adjacent literature, such as "Attention that does not Explain Away" and "Area Attention." Similarly, inputs related to BERT architectures successfully retrieved specialized, derivative models like "BEiT: BERT Pre-Training of Image Transformers."

Furthermore, the successful integration of these models into the Streamlit web-based user interface represents a major functional achievement. The system not only categorizes unseen literature with high mathematical accuracy but actively guides the user through the semantic neighborhood of their specific research interests via a highly accessible, lag-free frontend platform.

Discussion

The development and evaluation of this Research Paper Classifier and Recommender reveal critical insights into the intersection of natural language processing and academic information retrieval. The superior performance of the MLP model—achieving nearly 99.5% accuracy—underscores the efficacy of pairing a carefully calibrated TF-IDF vectorizer (with bigram preservation) with dense neural network architectures. This approach successfully navigates the sparsity and vocabulary mismatches inherent in specialized scientific datasets. By processing text as TF-IDF arrays within the Keras pipeline, the system accurately highlights discriminative jargon while discarding structural noise.

The results also highlight the successful resolution of the multi-label classification problem. Rather than forcing a paper into a single category, the system's ability to predict a multi-hot encoded array reflects the true interdisciplinary nature of modern research. The output formatting, which calculates confidence thresholds and presents the most probable labels alongside their percentage likelihoods, provides transparency to the end-user.

However, the findings also highlight certain operational dependencies and trade-offs. The system relies heavily on static, serialized assets (vocab.pkl, embeddings.pkl, model.keras). Consequently, as new scientific terms are coined and novel research fields emerge, the current static vocabulary may become outdated. The presence of dedicated retraining scripts (retrain_model.py and train.py) within the system architecture anticipates this limitation, allowing for scheduled ingestion of new arXiv data. Ultimately, the discussion confirms that the proposed architecture provides an incredibly accurate, lightweight, and deployable solution, successfully balancing computational overhead with high-fidelity classification and recommendation.

Conclusion and Future Scope

The rapid proliferation of academic literature mandates a shift from traditional search mechanisms to intelligent, proactive discovery systems. This project successfully designed, implemented, and deployed a dual-purpose Research Paper Classifier and Recommender capable of understanding complex scientific text. By synthesizing the strengths of bigram TFIDF vectorization with a robust Multi-Layer Perceptron, the system accurately predicts multilabel subject areas with over 99% accuracy. Furthermore, the integration of these models into an interactive Streamlit frontend validates the system's viability as a powerful, userfriendly tool designed to accelerate the research lifecycle and mitigate information overload.

Looking toward the future, several promising avenues exist to further enhance the system's architecture. First, migrating the classification pipeline from a TF-IDF/MLP structure to a finetuned Large Language Model or Transformer architecture (such as SciBERT) could provide an even deeper semantic understanding of complex mathematical formulations within abstracts. Second, implementing a live web-scraping pipeline or utilizing the arXiv API directly within the Streamlit app would allow the system to continuously update its database, ensuring recommendations remain state-of-theart without requiring manual retraining. Finally, integrating an interactive graph visualization of the recommended papers' citation networks within the frontend would create a truly comprehensive academic mapping tool.

Acknowledgements

This capstone research endeavor, undertaken to fulfill the requirements for my B.Tech degree in Computer Science and Data Analytics, would not have been possible without the support and guidance of numerous individuals. I would like to express my deepest gratitude to my project supervisor and the department faculty for their invaluable technical insights, strict academic standards, and continuous encouragement throughout the development of this machine learning system. I also extend my thanks to the open-source community, particularly the developers of TensorFlow, Keras, and Streamlit, whose extensive documentation and robust libraries served as the foundational infrastructure for this work. Finally, I am profoundly grateful to my family for their unwavering support and patience during the intensive coding, model training, and writing phases of this project.

Appendix

System Architecture and Project Structure The deployed system architecture relies on a structured repository of scripts, data, and serialized assets to function in real-time. The core project directory is organized as follows:

- app.py: The main Streamlit frontend application script handling user interaction and interface rendering.
- research_paper.ipynb: The comprehensive Jupyter Notebook utilized for Exploratory Data Analysis, model construction, and evaluation metric generation.
- recommendation.py: The backend logic module responsible for executing semantic similarity searches.
- train.py / retrain_model.py: Modular Python scripts designed to facilitate the updating and retraining of the neural network upon the ingestion of new datasets.
- arxiv_data_210930-054931.csv: The primary raw dataset containing scientific abstracts and titles.
- Serialized Assets (/models/, /saved_model/):
  - model.keras / model.h5: The saved architecture and weights for the trained Sequential MLP classifier.
  - labels.pkl: The inverted dictionary utilized to convert multi-hot probability arrays back into readable string taxonomies (e.g., cs.LG).
  - text_vectorizer_config.pkl & text_vectorizer_weights.pkl: The saved configurations required to process unseen text consistently.
  - embeddings.pkl: The serialized vector space utilized for real-time paper recommendations.

References

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). "Attention is all you need." *Advances in neural information processing systems*, 30.

[2] Salton, G., & McGill, M. J. (1983). *Introduction to Modern Information Retrieval*. McGrawHill.

[3] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Zheng, X. (2016). "TensorFlow: A system for large-scale machine learning." In *12th USENIX symposium on operating systems design and implementation (OSDI 16)* (pp. 265-283).

[4] Tsoumakas, G., & Katakis, I. (2007). "Multi-label classification: An overview." *International Journal of Data Warehousing and Mining (IJDWM)*, 3(3), 1-13.

[5] Beel, J., Gipp, B., Langer, S., & Breitinger, C. (2016). "Researchpaper recommender systems: a literature survey." *International Journal on Digital Libraries*, 17(4), 305-338.