electronics

MDPI

Article

Automated Research Review Support Using Machine Learning, Large Language Models, and Natural Language Processing

Vishnu S. Pendyala [1,*], Karnavee Kamdar [2] and Kapil Mulchandani [3]

[1] Department of Applied Data Science, San Jose State University, San Jose, CA 95192, USA
[2] Oracle, Austin, TX 78741, USA; kamdarkarnavee@gmail.com
[3] Amazon, Seattle, WA 98170, USA; kapilmulchandani2019@gmail.com
* Correspondence: vishnu.pendyala@sjsu.edu

Abstract: Research expands the boundaries of a subject, economy, and civilization. Peer review is at the heart of research and is understandably an expensive process. This work, with human-in-the-loop, aims to support the research community in multiple ways. It predicts quality, and acceptance, and recommends reviewers. It helps the authors and editors to evaluate research work using machine learning models developed based on a dataset comprising 18,000+ research papers, some of which are from highly acclaimed, top conferences in Artificial Intelligence such as NeurIPS and ICLR, their reviews, aspect scores, and accept/reject decisions. Using machine learning algorithms such as Support Vector Machines, Deep Learning Recurrent Neural Network architectures such as LSTM, a wide variety of pre-trained word vectors using Word2Vec, GloVe, FastText, transformer architecture-based BERT, DistilBERT, Google's Large Language Model (LLM), PaLM 2, and TF-IDF vectorizer, a comprehensive system is built. For the system to be readily usable and to facilitate future enhancements, a frontend, a Flask server in the cloud, and a NOSQL database at the backend are implemented, making it a complete system. The work is novel in using a unique blend of tools and techniques to address most aspects of building a system to support the peer review process. The experiments result in a 86% test accuracy on acceptance prediction using DistilBERT. Results from other models are comparable, with PaLM-based LLM embeddings achieving 84% accuracy.

Keywords: machine learning; peer review; large language models; long short-term memory; support vector machines; natural language processing

Academic Editor: Cecilio Angulo

Received: 28 November 2024
Revised: 30 December 2024
Accepted: 7 January 2025
Published: 9 January 2025

Citation: Pendyala, V.S.; Kamdar, K.; Mulchandani, K. Automated Research Review Support Using Machine Learning, Large Language Models, and Natural Language Processing. *Electronics* **2025**, *14*, 256. https://doi.org/10.3390/electronics14020256

1. Introduction

The pace of innovation in the recent times is astounding. While areas like natural language processing (NLP) and Large Language Models (LLMs) are evidencing an explosive growth, peer reviews are taking several weeks to even months. By the time the review of a submitted research paper is turned around, it is quite likely that the research itself will be obsolete. There is a need to improve the situation [1]. A literature review shows that a number of projects have been implemented to expedite the peer review process. Still, a number of peer reviews take weeks and months, and in the real world, the timeliness of the peer review process has not improved much. The work described in this paper therefore sets out to address the issue using a unique combination of tools and techniques and also make the solution practicable and comprehensive.

Peer reviews keep the quality of the available research literature high and are essential ingredients of the publication process. The entire research community, including the authors, benefit from peer reviews. Based on recent studies [2], the scientific literature is

constantly growing and has a strong correlation with the growth of the economy. There is evidence that the growth of the global scientific publication output for the 33 years from 1980 to 2012 was exponential [3]. To meet the scale of growth, it is imperative that we cut the time for peer reviews and make the process more efficient.

In this age of rapid obsolescence, the inadequacies of the current peer-reviewing process are becoming more evident. The prolonged timeline to publish research can often undermine the very objectives of scholarly work. This drawn-out process exasperates the entire research community, spanning authors, publishers, and readers. Realizing the importance of this process, many researchers [4–6] studied different aspects like consistency, bias, and review quality. These studies indicate the lacunae of the peer review process and inconsistencies in reviews. This process is the foundation of major scientific work, yet it is far from perfect.

Various steps of the paper publication process can be automated and optimized to minimize the time to publish. Unfortunately, these are manually performed to date by the authors, reviewers, and Technical Program Committee (TPC) chairs of conferences or Editors-in-Chief (EICs) of journals. This paper details the efforts at building a system that can assist and enhance the current peer review process. The work aims to automate the process of assigning reviewers for a research paper based on their domain expertise by applying machine learning and natural language processing techniques.

Then, to minimize the review time taken by the reviewers, the system predicts various aspect scores such as impact, clarity, soundness, originality, and acceptance for the research paper using machine learning algorithms. The goal of this work is to reduce the overall review time and improve review quality so that valuable research does not lie unused for several months. For instance, a global repository of profiles of all the researchers who can serve as peer reviewers along with their paper publications can be maintained to help identify potential reviewers for a submitted paper. This can be performed by matching the keywords that are indicative of their research and domain expertise. The process of updating this information manually every time can become monotonous and outdated if not performed proactively. There is a need to automate the process.

As the number of papers being submitted and the number of reviewers joining keeps increasing, the complexity of assignment keeps increasing too, making it essential to be computationally robust and automated for fair assignment. The reviewer assignment problem is further compounded because the following criteria need to be met on assignment:

- There is no conflict of interest between the authors and the reviewers.
- Each paper receives an equal number of reviewers.
- The reviewers are topic experts and assigned papers according to their preference.
- No reviewer is overburdened with a greater number of research papers to review.

This work aims to automate reviewer assignments. To reduce the peer review time, inconsistencies or bias, our project also aims to predict various aspect scores like impact, clarity, soundness, originality, and acceptance for the research paper using machine learning and NLP techniques.

2. Literature Survey

The problem of improving the timelines and quality of peer review is specific to the research domain and therefore received considerable attention in the research literature, so much so that there are now projects [7] to automatically generate the reviews using NLP techniques, specifically a denoising autoencoder called BART. The authors, however, conclude that their solution is only a beginning, and the time has not yet come for automatic review generation. It is an established fact that modern Natural Language Generation techniques such as those used in ChatGPT are prone to hallucination [8] implying their

unsuitability for peer review generation. There is plenty of literature, particularly in the reviewer assignment part of the problem. For brevity, we try to cover a reasonable selection of the literature in the following subsections.

2.1. Dataset

A first step in solving a problem is to determine the materials and methods. In this case, it is the selection of the dataset suitable for the experiments. Kang et al. [9] talk about the experiments performed on a dataset consisting of 14.7 K paper drafts from venues like ACL, NeurIPS, and ICLR and their respective reject or accept decisions as well as textual peer reviews for a subset of 10.7 K written by experts in that domain. The focus of Kang's paper is data collection, and gaining insights into the data collected. Some of the important tasks outlined in this paper include the usage of simple models as opposed to a majority baseline to obtain a 21% error deduction in predicting if a paper is accepted, and determining numerical scores for review aspects like appropriateness, comparison, clarity, impact, substance, soundness, and originality.

CiteTracked [10] is a dataset with papers from the reputed NeurIPS conference, their reviews, and citation statistics. The authors [11] present an interesting way to classify peer reviews for comparison. The authors come up with a taxonomy of categories and use machine learning and Natural Language Processing to learn representations of the peer reviews. For our project detailed in this paper, we model our dataset based on the one detailed in Kang's paper [9].

2.2. Review Assignment

In the given problem, domain review assignment has probably received the most attention in the current literature. The reviewer assignment problem (RAP) has been sufficiently studied and surveyed [12]. Anjum et al. [13] use word embedding-based common topic modeling to find the intersection of topics related to the submission and the reviewers' profiles to make the review assignment. The paper lists a host of previous approaches such as the one using Latent Dirichlet Allocation (LDA), another using Hierarchical Dirichlet Process (HDP), and so on, as baselines and compares them with the proposed solution. The literature survey in this paper illustrates the varied solutions to the problem of the review assignment. For brevity, we do not review each one of them and instead refer the reader to this paper.

To automate the process of paper–reviewer assignment, many exhaustive search and heuristic algorithms have been proposed in the literature in addition to Natural Language Processing (NLP) techniques, Latent Dirichlet Allocation (LDA), Probabilistic Matrix Factorization (PMF), and others. The authors [14] detail the usage of scientometrics, which is the science of measuring and analyzing science, to solve the problem of selecting peer review experts. Their article suggests that the scientific selection of peer reviewers ensures the evaluation of objectiveness, impartiality, and fairness. It uses the application of statistical indicators for evaluating scientific productivity. Some of these methods include citation analysis, co-citation analysis, word frequency analysis, multivariate statistical analysis, and social network analysis. The paper concludes on three main points: (a) analysis for determining the theoretical feasibility of automating the process of peer review expert selection, (b) implementing co-word analysis to suggest the research domain of a reviewer along with exact proof, and (c) using knowledge mapping to find the closest research domain in cross-disciplinary research papers.

The next paper, [15] by the same first author suggests the usage of a set of article data and keywords in interdisciplinary research to find the relevant peer review experts from the disciplines involved in the study. Knowledge mapping is the advancements of scientific knowledge and their structural relationships in a graphical manner. Some of

the theories and methods in knowledge mapping include co-citation analysis, domain-specific ontology, multivariate statistical analysis methods, information visualization, etc. When knowledge mapping and analysis are performed on the interdisciplinary research and the articles cited in other related disciplines, the overlapped research areas can be determined. Li and Hou [16] use Latent Semantic Indexing (LSI), which is essentially Singular Value Decomposition for text, to score the match between a submitted paper and a set of characteristic papers of a given reviewer. They go a step further and use the system they developed, called Erie, for review assignments for an IEEE conference.

Liu et al. [17] use a graph-based approach to assign reviewers automatically. They use a dataset with data from human evaluations to evaluate their model. Authors [18] use the popular expectation–maximization algorithm to compute the profile of a research paper. Based on the computed profiles of the submitted papers and the reviewers, the assignment is performed using multiple approaches previously established in the literature. Most of the work in this area does not consider multiple aspects of the profiles of the reviewers or the submitted papers. Karimzadehgan and co-authors [19] address this deficiency by using a topic modeling technique previously established in the literature called PLSA to model multiple aspects before performing the assignment. The authors [20] use a modified version of an evolutionary algorithm called Bare-Bones Particle Swarm Optimization (BBPSO) to perform the assignment, while LDA is used for reviewer and paper profiling.

Papalexakis and co-authors [21] propose a sparse matrix factorization approach different from the conventional Latent Dirichlet Allocation (LDA), Singular Value Decomposition (SVD), and Matrix Factorization (NMF) techniques to profile the reviewers. Dated technologies such as an expert system and a Document Management System called DOMINUS along with Latent Semantic Indexing (LSI), which is based on SVD, are used to address the problem [22]. Review assignment is converted into an integer programming problem [23]. The authors develop a language model using a multinomial distribution with Dirichlet smoothing. Collaborative filtering using Probabilistic Matrix Factorization is used to find a suitability matrix. Two of the authors of the previous article in a later work [24] present more details and experiences using their system called "Toronto Paper Matching System".

In more recent work, the authors in [25] also convert the problem into an integer programming problem. Other authors [26] present a literature survey summarizing the various approaches taken to evaluate the effectiveness of review assignments. In their conclusion, they recommend taking a hybrid approach that combines multiple approaches from their summary table. In their survey of the literature from 1992 to 2015, Nguyen et al. neatly summarize various approaches to the reviewer assignment problem in their work [27]. The authors use Ordered Weighted Averaging (OWA) for aggregating scores for various aspects. Similar to other prior works, they too use Latent Dirichlet Allocation (LDA) for modeling the paper profiles.

2.3. Aspect Scoring

Research paper review typically comprises assigning scores for various aspects of the paper such as originality, appropriateness, and clarity. Qiao et al. [28] propose a recurrent convolutional neural network model to represent an academic paper by predicting its aspect scores. The input text is treated as hierarchies of the module document by the model. The model uses "attention pooling CNN and LSTM" to represent the text. Attention-based CNN consists of a convolution layer for extracting local features and an attention-pooling layer to determine the relations between words, sentences, and modules. The output of the model is represented in a linear layer. The authors claim superlative results.

2.4. Submission and Reviewing Systems

The literature review shows that some of the other researchers also took a more comprehensive look at the problem by incorporating solutions to multiple problems that can assist editorial teams or conference managers. As the saying goes, necessity is the mother of invention. Conference organizers in some instances came up with their own conference management systems. Flach [29] described the system that was used for the SIGKDD ' 09 conference. The system included components for bidding for papers, reviewer assignments, and reviewer score calibration. Bidding is facilitated by computing similarity scores between submitted papers and reviewers using the vector space model. Reviewer assignment is converted into an integer assignment problem after generating the bidding and conflict of interest matrices. Graphical models and probabilistic inferencing are used for reviewer score calibration. Towards the end, the paper also details larger efforts grouped under "Submission Sifting" tools to streamline the process.

Text mining techniques such as TF-IDF and Euclidean distance, and tools such as Lucene are used for reviewer assignment [30]. The work also addresses the problem of conflict of interest detection using simple schemes such as avoiding the same last names or email domains for the reviewer that are matched to a specific author. Self-Organizing Map (SOM) is used to assist with poster session setup planning and participant support. A blockchain-based access mechanism for peer review has also been proposed [31].

2.5. Predicting Acceptance

Machine learning has been used for many prediction problems. Researchers [32] have used a popular ensemble machine learning algorithm, Random Forest, to predict the acceptance of a research paper. They use a similar dataset as we do in this paper for their work. In their paper describing a deep learning architecture to exploit the sentiment in the reviews to make better accept/reject decisions [33], Ghosal and co-authors show that their approach reduces error by 29% when compared with the baseline approach. In a more recent work, Bharti, Ghosal, and others use a deep attention network to predict acceptance [34]. Bao et al. [35] use the well-established frequent itemset mining algorithm to predict by constructing decision sets.

Institutions typically want to contribute as much to the research literature as possible. Authors [36] use regression algorithms such as linear regression and gradient boosting to make the prediction. They observe that linear regression outperforms gradient boosting in terms of robustness. For the prediction, they come up with four classes of features such as historical paper acceptance and program committee membership to model an institution. Peer reviews are not only used for publication decisions but also for funding decisions by authorities such as the National Science Foundation (NSF). In their article published in Nature [37], researchers propose an interesting concept called "Distributed Peer Review". Interestingly, the adjective, "distributed" here refers to distribution among peers and not machines. Machine learning is used to match reviewers to proposals.

3. System Architecture

The system primarily comprises three components: (a) automatic reviewer assignment based on domain knowledge, (b) predicting aspect scores of a submitted research paper, and (c) predicting the chances of acceptance of the paper. The overall architecture of the system is given in Figure 1. The user, who can be an author or a reviewer, or a EIC or a TPC chair, interacts with the website to perform various functions such as uploading a paper, reviewing the paper, and accepting or rejecting the paper. The backend comprises a Flask server that has REST API endpoints to perform these functionalities and produces the results using implemented business logic and trained machine learning models to

assign papers to reviewers, generate aspect scores, and predict the acceptance of the research papers. All the data about the users, conferences, and papers are stored in MongoDB, a NOSQL database. The raw research paper files submitted by the authors and the processed parsed JSON files are stored in Amazon S3 buckets.

The MongoDB document-based model in terms of storing data as BSON, a binary representation of JSON, allows one to store research paper data in their natural form without the need for complex joins or transformations. MongoDB is designed to scale out horizontally with ease, using sharding. This is particularly beneficial when dealing with a large number of research papers that may need to be distributed across multiple servers as the operations grow. MongoDB is proven to be effective for read-heavy use cases such as for the task at hand.

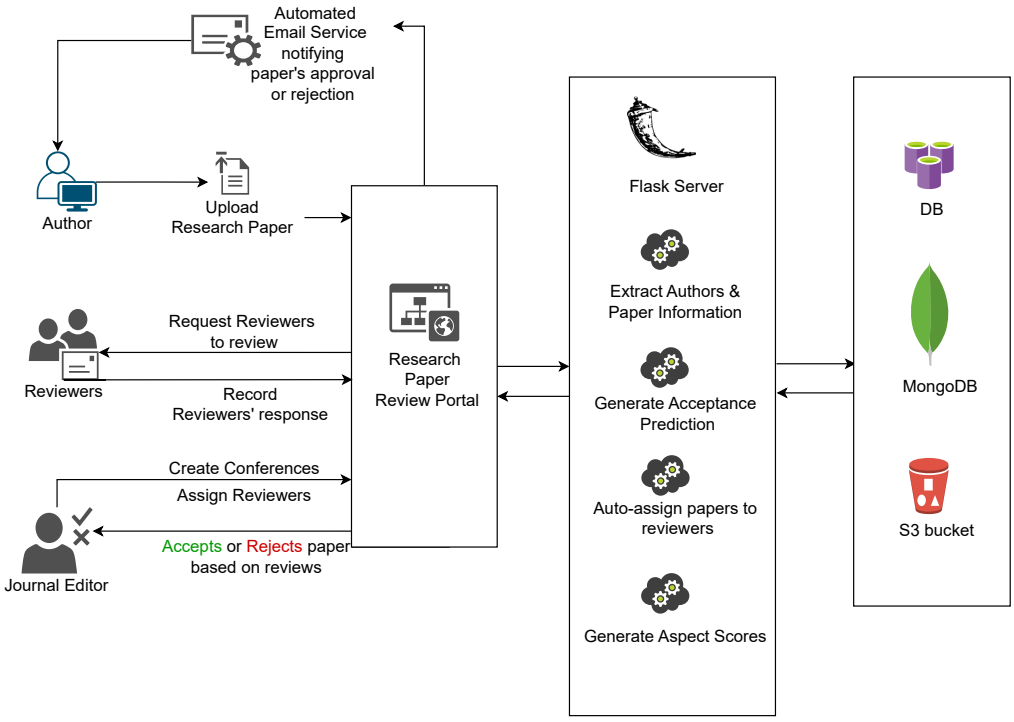

Figure 1. System architecture.

Architecture Subsystems

The system can broadly be divided into three modules as follows:

1. Auto-detection of authors and research paper information;
2. Auto-assigning of papers to reviewers;
3. Prediction of aspect score and acceptance of research paper.

The information contained in the first module includes (a) author names and contact information, and (b) research paper abstract and section extraction. The second module makes sure that each paper receives an equal number of reviews and that each reviewer receives no more than a predefined number of N papers of their domain expertise and preference. The third module predicts aspect scores like impact, clarity, soundness, and originality and also predicts the acceptance or rejection of the submitted research paper.

The entire system is hosted on a Flask server in the cloud to ensure availability, accessibility, and scalability. A simple Python REST API is used to serve the machine learning models. Through Web UI, the user, in this case, the author will be able to sign up and upload their research papers to the selected conference for the publication review process. When the paper is uploaded, the information like author names, contact information, ab-

stract, and references are automatically extracted from the PDF and converted to a JSON file. The paper PDF and JSON are stored in S3, and its file reference URL is stored in the MongoDB database and linked to their profile.

Next, the algorithm to auto-assign the papers to reviewers is executed, and reviewers are notified of the request to accept or reject to review the paper, having given the paper title and abstract. If the reviewer accepts reviewing the assigned paper, then he/she can go ahead with the peer review process; otherwise, the above process of auto-assignment is repeated with the updated reviewer preferences and availability. Furthermore, we perform feature engineering to predict its acceptance and various aspect scores to accelerate the review process and minimize inconsistencies, bias, and/or review time. The aspect scores, author, and paper information are all stored in the database for future reference and modeling.

Finally, once all the papers are peer reviewed, the EIC or Technical Program Committee (TPC) chair is notified of all the predicted aspect scores and reviews, using which he/she can decide on accepting, resubmitting with minor/major modifications, or rejecting the paper. Lastly, the authors are auto-emailed the decision of the EIC or TPC chair and link to the website portal for detailed information.

4. Preparation for Experiments

As in any machine learning application, we collect a dataset, extract features where applicable, split the dataset into training and validation sets, and generate a model based on the dataset.

4.1. Data Collection

The dataset from [9] called PeerRead contains 18,000 research papers, reviews, aspect scores, and accept/reject decisions. As mentioned earlier, we used the approach followed for the PeerRead dataset for our experiments. Our dataset is described in Table 1. As can be seen, it is quite similar to the PeerRead dataset, except that we added more papers from the NeurIPS conference. To download the papers and reviews from the NeurIPS website, we implemented a web scraper in Python using Selenium Web-driver, Requests, BeautifulSoup, and Pandas libraries. This scraper will essentially download the data, i.e., research paper title, author(s) information, abstract, PDF file, and reviewers' comments, and meta-review as well as keeping track of the source links from the NeurIPS website, https://proceedings.neurips.cc/, (accessed on 6 January 2025). The dataset is reflective of the review process in highly reputed forums. The expectation in this work is that the machine learning models emulate this process, which is known for its integrity and quality.

Table 1. Dataset details.

Section	No. of Papers	No. of Reviews	Accept/Reject
NeurIPS 2013–2020	5746	15,804	2420/0
ICLR 2017	427	1304	172/255
ACL 2017	137	275	88/49
CoNLL 2016	22	39	11/11
arXiv 2007–2017	11,778	-	2891/8887

4.2. Automatic Information Extraction

Whenever a user uploads a research paper for conference or journal submission from the Author's Explore screen shown in Figure 2, a unique paper_id is generated and stored in MongoDB. The raw PDF file is stored in the S3 bucket and used as a source to extract the authors and paper's information. The science-parse-api library (alternatively science

parse jar file) is used to parse the PDF and extract the information. The output is stored in a JSON file format in another S3 bucket. The papers collection will have unique paper_ids for each uploaded paper and the corresponding URL of the unaltered PDF file and parsed JSON file. If one of the required fields is missing, the system will not accept the submission and display the required fields missing error to the user. Additionally, the system will roll back any changes stored in the database if required. An example of the auto-extracted information is shown in Figure 3.

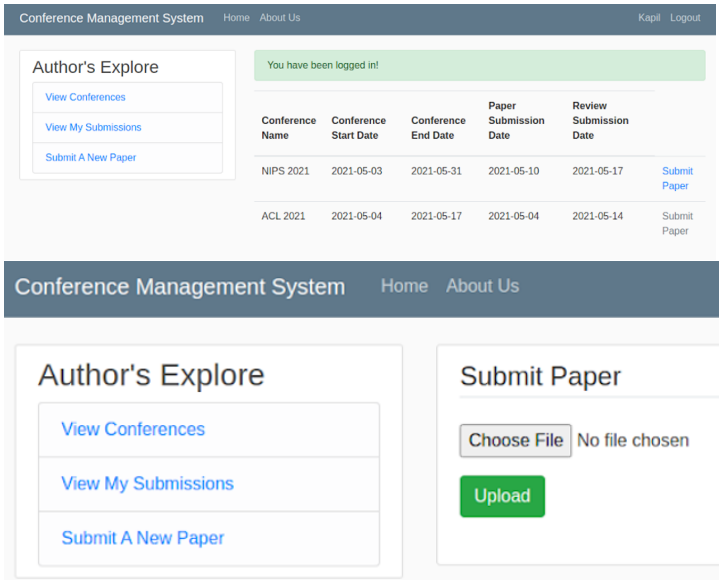

Figure 2. Conferences view from Author's Explore.

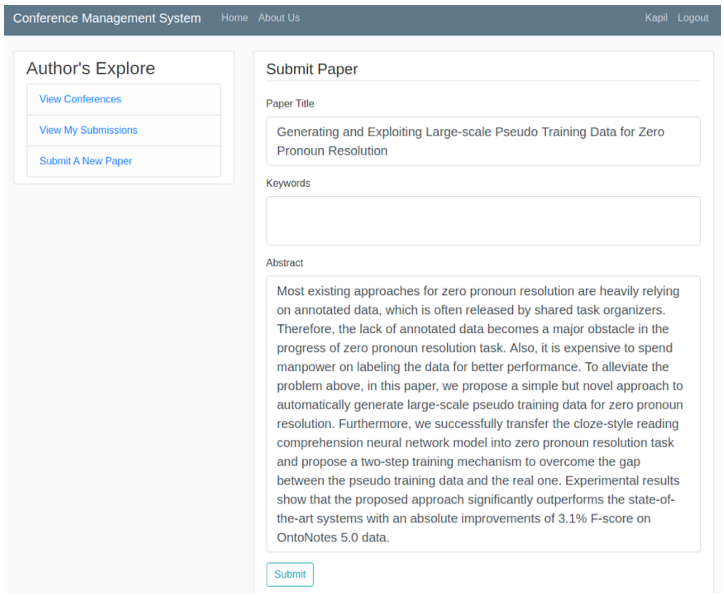

Figure 3. Auto-extraction from uploaded paper.

If the user chooses to edit their submission prior to the submission deadline, the old paper information will be lost, and the paper_id will be updated with the new paper information, i.e., at no point in time will the user be able to track the history of their submissions or access their previously submitted files from the system. The users' collection will have a unique user_id and store attributes like name, email, and password. A user can possess and be granted various roles like the author, reviewer, and TPC chair/EIC. The type of role a user has can be found in the role_type attribute in the users' collection. Based on the user permissions, they can switch between different roles and gain access to various

role-based functionalities. If a user has submitted any research paper for a conference then information pertaining to that context, in this case, a list of paper_ids is associated with the author's profile (document) and shown when the user logs in.

4.3. Database Design

Figure 4 illustrates the database entity–relationship diagram presenting the different entities of the system and the relationships among them. The diagram is self-explanatory. Corresponding MongoDB representations are given in Figure 5. The expected parsed output format is given below.

```
{name: pdf_name.pdf,
metadata: { source: CRF,
title: [Paper title],
authors: [author names],
emails: [email addresses],
sections: [{heading: Introduction, text: Introduction here}, ],
references: [{title: paper title, author: [author name], venue: location}, ]
}
}
```

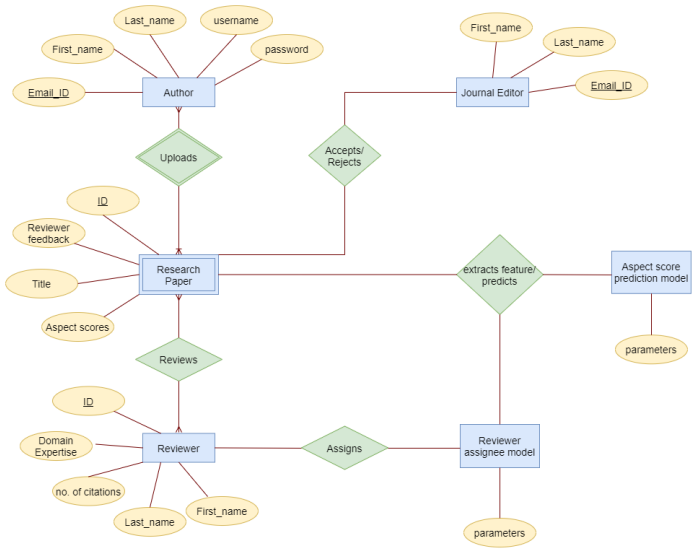

Figure 4. Database Design Diagram.

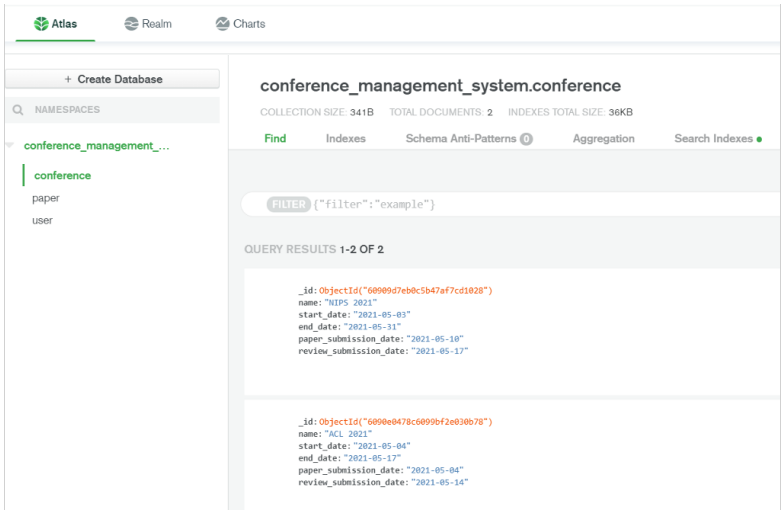

Figure 5. *Cont.*

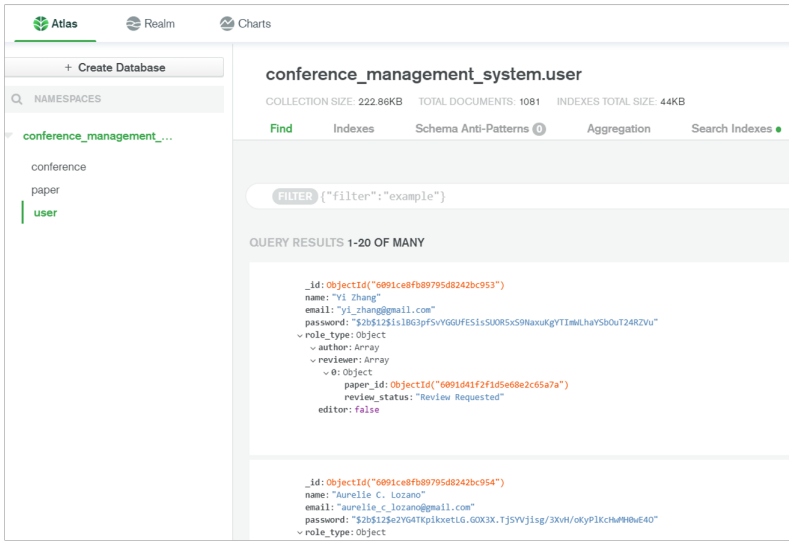

Figure 5. MongoDB Documents.

4.4. Data Preprocessing

The dataset contains research papers from different conferences. Each conference datum consists of two folders:

1. Research paper content in JSON format;
2. Reviews in JSON format.

Each research paper consists of different sections, a different number of reviews, and different attributes. All the records must be in a uniform format to train a model. The following steps are performed to create a clean dataset:

- Combined research paper content and reviews of the research paper into one data frame.
- Combined research papers of different conferences into one data frame.
- Created a new record for each review text using Left Outer Join on the combined data frame, as each research paper contains multiple reviews.
- Drop duplicate records.
- Removed HTML tags, punctuations, stop-words, nan values, and digits from the review text.
- Converted each word into lowercase.
- Performed lemmatization on each word.

4.5. Feature Engineering

Text data in the research papers need to be converted to numbers. This process is often referred to as vectorizing. The following steps are performed to vectorize the data to prepare them for the machine learning processes:

- Handling Imbalanced Data: There are a disproportionate number of observations in each class (1–5) of aspect scores. One of the techniques to solve this problem is to combine minority classes. We combined the scores with values 0–1 in one class and those with values 4–5 into another class. Also, we used under-sampling and removed random observations of the majority class.
- Creating Vocabulary Index: A vocabulary is created from all the words present in the dataset, and a unique integer value is assigned to each word. Later, this index transforms a word into a vector of integers.
- The text in the research papers is converted into vectors of floating point numbers using the following frameworks.

1. TF-IDF Vectorizer: The main characteristic of the TF-IDF vectorizer is its unique nature of transforming word representations into numerical representations based on Term Frequency and Inverse Document Frequency (IDF). The count vectorizer gives the frequency of each word in the dataset with respect to the index of the vocabulary, whereas the TF-IDF makes use of the overall documents of weights of the words. Equations (1) and 2 state how TF-IDF is computed for a given word:

$$TF - IDF(t,d) = tf(t,d) * idf(t) \tag{1}$$

$$idf(t) = \log(\frac{N}{df(t)}) \tag{2}$$

where tf is the term frequency for the word t, N is the total number of documents in the corpus, and $df(t)$ is the number of documents in which the word t appears.
2. Word2Vec Embeddings: The Word2Vec NLP framework [38] represents words in continuous vector spaces, capturing contextual and semantic relationships between them. Words with similar meanings in the given context are positioned close to each other in the vector space, allowing for semantic relationships to be captured mathematically.
3. GloVe Model: GloVe [39] stands for Global Vectors. Vectors here are numerical representation of words, also called embeddings. GloVe is a pre-trained word embedding model that focuses on the co-occurrence of the words. It contains encoded vectors of different sizes: 50-D, 100-D, 200-D, and 300-D.
4. BERT Embeddings: The Bidirectional Encoder Representations from the Transformers [40] framework produces contextualized numerical representations of the words in the input text, based on massive pre-training. In the process, both the left and the right contexts of a word are embedded into its fixed-size, vectorized numerical representation.
5. DistilBERT Embeddings: DistilBERT [41] is a relatively smaller and more efficient version of BERT. The model's training involves a distillation process, which results in fewer model parameters, thereby requiring fewer resources to execute.
6. FastText Representation: The unique algorithm used in FastText [42] bunches the sequence of subwords into groups called n-grams to produce embeddings. This strategy helps to capture the semantic sense of rare or unseen words. Like DistilBERT, FastText is computationally efficient, hence deriving its name.
7. PaLM 2 LLM embeddings: PaLM 2 [43] and its earlier version, PaLM [44] are recent Large Language Models (LLMs) from Google. PaLM 2 is particularly relevant for this project because it was pre-trained on a more comprehensive dataset that included scientific papers and web pages that contain mathematical expressions. Its performance is comparable to GPT-4's in certain aspects. Specifically, the Gecko model in the family of PaLM 2 framework has been used to generate embeddings.

5. Methodology

The methodology adopted aligns with the following intended contributions of the paper.

5.1. Contribution of This Paper

After carefully examining the advantages and limitations of various techniques used in the literature, including those of the latest transformer architecture [45] that power Large Language Models (LLMs) and ChatGPT, we chose a unique blend of tools and techniques to overcome some of the shortcomings in the existing literature and achieve an acceptable accuracy of prediction. The paper's objective is to investigate the following research questions:

RQ1: How mature are the recent Large Language Models (LLMs) such as PaLM 2 [43], other transformer architecture-based models such as BERT [40] and DistilBERT [41], and recent frameworks for quick natural language experimentation such as FastText [42] for peer review support, which is an important natural language understanding task?

RQ2: Do the relatively inexpensive techniques, TF-IDF, GloVe, and Word2Vec, serve as efficient representation learning approaches in comparison to the LLMs and Transformer-based architectures for peer review support?

RQ3: Does implementing a system architecture with specific components result in a more efficient peer review process compared to existing systems, while also facilitating future enhancements such as semantic search using a vector database?

RQ1 is investigated by applying various machine learning classification algorithms on word embeddings generated by PaLM 2, BERT, DistilBERT, and FastText and measuring their key performance indicators. For RQ2, we performed similar experiments using TF-IDF, GloVe, and Word2Vec text embeddings. To answer RQ3, we incorporated a frontend and a backend using NOSQL tools to build a product that is ready to be used. Based on our literature survey, to the best of our knowledge, this work is unique in proposing a comprehensive peer review system that uses Support Vector Machines and sequential neural network classifiers for research article acceptance prediction; GloVe vectorizer and LSTM for an aspect score prediction; and a nearest neighbors-based approach for reviewer assignment that is cloud based and uses NOSQL at the backend. The results from our implementation prove that the approach is practicable.

The embeddings generated for the text in the research papers using the vectorization frameworks listed in Section 4.5 are used for classification using a variety of machine learning techniques.

It is a common observation that human beings spend no more than 5 to 10 min a day to comprehend all the relevant news items. This is performed internally by focusing on the most important words that contribute to the theme rather than stop words and other less important words. Human beings ignore or forget less important words, something that we consciously do for precise writing as well. As the name indicates, Long Short-term Memory (LSTM) [46] models this behavior. We used LSTM for some of our experiments and obtained encouraging results.

In view of the limitations of LSTM [47], the TFAutoModelForSequenceClassification class that is based on the transformer architecture, and fully connected Artificial Neural Network (ANN), Support Vector Machines (SVM) are also used for classification.

5.2. Aspect Score Prediction

The Aspect Score Prediction module covers the generation of numerical scores for the various factors of the reviews submitted by the reviewers. Based on these aspect scores, the EIC or TPC chair can understand the quality of the paper. The authors can see the predicted aspect scores. They can improvise their submitted paper based on these factors. We used LSTM, a sequential model to predict a score for each aspect. The aspects used for our experiments are clarity, impact, soundness, and originality of the research paper. The words in the document are vectorized using the GloVe framework and input to the LSTM model as a sequence.

LSTM is a special kind of Recurrent Neural Network (RNN) that can consider the long-term dependencies among the words during training. This model contains a memory state which helps to remember or forget things, similar to how reading comprehension works in human beings. This capability enables LSTM to work well with sequential data such as text. The key aspect of LSTM is to split the hidden state into two. The first part is for long-term retention in a "memory cell" and the second part is short term. Some

information about the words is selectively read, some is remembered, and some of it is forgotten or ignored, quite analogous to how reading comprehension works in the human mind. All this is accomplished using gates and states. The gate (Equations (3)–(5)) and state (Equations (6)–(8)) update equations are given below:

$$o_t = \sigma(W_o h_{t-1} + V_o x_t + b_o) \tag{3}$$

$$i_t = \sigma(W_i h_{t-1} + V_i x_t + b_i) \tag{4}$$

$$f_t = \sigma(W_f h + V_f x_t + b_f) \tag{5}$$

$$\tilde{s}_t = \sigma(W h_{t-1} + V x_t + b) \tag{6}$$

$$s_t = f_t \odot s_{t-1} + i_t \odot \tilde{s} \tag{7}$$

$$h_t = o_t \odot \sigma(s_t) \tag{8}$$

where we have the following:

s_t is the state of the RNN at timestep t;
x_t is the new information at timestep t;
o_t is the output gating vector at timestep t that decides how much of the state needs to be handed over to the next state;
i_t is the input gating vector at timestep t that implements selective read;
f_t is the forget gate vector that implements selective forget;
W_i,W_o, W_f are the parameters that need to be learned for the input, output, and forget gates respectively to weigh the hidden state from the previous timestamp, t−1;
V_i,V_o, V_f are the parameters that need to be learned for the input, output, and forget gates, respectively, to weigh the input at timestamp t;
b_i,b_o, b_f are the respective biases;
σ is the sigmoid function;
h_t is the portion of the hidden state at timestep t that moves forward;
$\tilde{s_t}$ is the state before selective read.

5.3. Acceptance Prediction

The acceptance prediction module predicts whether the research paper will be published or not. When an author logs into the research paper review portal and uploads the research paper, a form will come up with all the details of the research paper automatically extracted from the pdf file, and once the author submits the research paper for review, the frontend sends a request to the backend which in turn initiates this model. The acceptance prediction model which is already trained with the past research papers dataset extracts the features from the parsed_pdf, which become the input to the classifier. The features that are used by the classifier are the various sections of the paper, including title, abstract, methodology, and so on. The text in these sections is modeled by the vectors generated in Section 4.5.

After the data are collected, extracted, cleaned, and preprocessed, the data are converted into vectors of floating point numbers used for training the model. Based on this, a model is built using word vectors for the text classification task in which we attempt to predict the classification of the research paper. We use three classifiers and compare the results. The classifiers are (a) Support Vector Machine (SVM) classifier, (b) sequential artificial neural network classifier, and (c) the transformer architecture-based TFAutoModelForSequenceClassification classifier in the transformers library.

The SVM model that we use implicitly transforms the features (word vectors) of the papers in the training set to a higher-dimensional space using a polynomial kernel and

finds a linear decision boundary that maximizes the margin between the accepted and rejected papers in this higher-dimensional space. To be flexible, we use a soft-margin SVM with a regularization parameter. The SVM margin between the two classes of papers that needs to be maximized is given in Equation (9). The denominator is the norm of the weight vector, w. The maximization is constrained by the inequality given by (10) below. Here, y_i are the target variable values (1 = accept and 0 = reject), x_i are the feature vectors, w is the vector of the weights for each of the features, and b is the bias:

$$M = \frac{2}{||w||} \tag{9}$$

$$y_i(w.x_i + b) >= 1 \tag{10}$$

Noting that maximizing the margin in Equation (9) is the same as minimizing its reciprocal, with the norm squared, in the case of soft margin SVM, the above two equations change to Equations (11) and (12), respectively. In a perfect SVM, the data should be linearly separable, and there would be no need for slack variables. However, in many real-world situations, the data are not linearly separable, and some misclassifications are tolerated. The slack variables represent the amount by which the data points violate the margin constraints [48]. ζ_i are slack variables introduced to accommodate this slack in the margins of the two classes of the papers, namely, accepted and rejected:

$$Minimize \frac{||w||^2}{2} + C\Sigma\zeta_i \tag{11}$$

$$y_i(w.x_i + b) >= 1 - \zeta_i \tag{12}$$

Using Lagrange multipliers to solve the above constrained optimization problem, and then using its dual, we end up in a classifier that only contains dot products. Therefore, if we can compute only the dot products in a higher-dimensional space, we do not need to literally transform the features. Kernels provide us with a feasible way of computing the dot products in a higher-dimensional space. For our problem, we use a cubic polynomial for the kernel. The general form of a polynomial kernel is given in Equation (13), where x and x' are the feature vectors of any two research papers from the training dataset, and c is a constant:

$$K(x, x') = (x^T x' + c)^d \tag{13}$$

The alternate approach we use is a sequential neural network classifier. As the name indicates, the layers in the neural network work sequentially, as contrasted to, say, a residual neural network. The features of the research papers in the training dataset are passed as inputs to the sequential neural network, and the weights (parameters) for the features are determined by backpropagation. The feature vectors of the research papers and gradients essentially flow sequentially through the layers of the neural network and their activation functions. More details about the hyperparameters we use for both SVM and the sequential neural network classifier are provided in the results section.

For experiments using DistilBERT word vectors, TFAutoModelForSequenceClassification class from the transformers library is used for classification. This class automatically identifies and uses the appropriate pre-trained transformer model for classification based on the configuration provided. The model adds a classification head on top of the transformer base to adapt the model for classification tasks.

5.4. Reviewer Assignment Module

The reviewer assignment module facilitates the allocation of peer reviewers for a paper submitted to the conference. This is a crucial as well as a complex task in any conference,

as there are hundreds of papers to allocate amongst hundreds of reviewers. The NeurIPS dataset which is used for reviewer assignment consists of 1048 published authors and their previously submitted work. The functioning of the Reviewer assignment module involves the vectorization of abstract and text of the papers in the NeurIPS dataset using the TF-IDF transformation. In this process, the words in a document are assigned a value that is directly proportional to the number of words in that document and inversely proportional to the frequency of documents in the dataset as explained in Section 4.5. A popular machine learning library in Python, scikit-learn, is used for this purpose. A TF-IDF matrix is generated by transforming the abstract and paper texts of the conference papers.

Following the vectorization of data, the nearest neighbors technique is used on the TF-IDF matrix to identify the papers that are closest or most similar to the data point in question. Similar papers suggest that the authors of these papers have expertise in the same domain. Hence, for a given paper, the authors of the papers which are most similar to it are assigned as its reviewers. The metric we use for distance calculations between nearest neighbors is the Euclidean distance. Some other papers in our literature survey have also used this metric to obtain good results. The Euclidean distance between two research papers with TF-IDF vectors, x and x', respectively, is computed as

$$dist(x, x') = \sqrt{\Sigma(x_i - x_i')^2} \tag{14}$$

5.5. Web Application Development

The web application is built using the Flask micro web framework in Python. The frontend of the website is built using HTML5, Bootstrap version 4.0.0, CSS version 4.0.0, jQuery version 3.2.1, Popper.js version 1.12.9, and JavaScript version 4.0.0. We use Jinja2 to create HTML templates and forms to avoid code redundancy and dynamically populate the user screen with information specific to them. A user of our application could have one or more roles from the following and access to the following role-specific REST APIs:

1. Author

a. Add/Edit/Delete Research Paper for a conference prior to the submission deadline;
b. Add comments on the feedback provided by the reviewers;
c. View the status of the paper—Submitted/Under Review/Reviewed/Accepted/Rejected, etc.

2. Reviewer

a. Accept/reject paper review requests;
b. Provide and view comments on the reviewed paper.

3. Technical Program Chair

a. Create/edit conference timelines;
b. View assignments of papers to reviewers and reassign if needed;
c. View/provide comments to the papers;
d. View aspect scores generated by ML model based on the reviewer comments;
e. Update paper status to accepted/rejected/resubmit with major or minor modifications.

Other commonly used APIs accessible to the users irrespective of their above roles from the web interface are as follows:

- User registration and login;
- View conference details.

All the papers, users, and conference information is stored in MongoDB and S3 buckets as covered earlier. The machine learning models to auto-assign papers to reviewers,

generate aspect scores, and predict paper acceptance are deployed on a Flask server and available as a service.

5.6. Deployment

Our Research Paper Review Portal provides the author, EIC/TPC chairs and reviewers with a web application portal, in which the author can upload the paper, check the fields, and submit it for review. On the same website when the EIC/TPC chair logs into the website, all the papers that are pending for publication can be viewed, with a number of options to act on. Moreover, when the reviewer logs into the website, all the papers that are pending for review or that have been reviewed are displayed. For deployment of this application, Heroku and Amazon Web Services (AWS) are used.

The system designed for this paper is a prototype, and the choice of the technology stack is only indicative. Other alternatives for the tools in the stack can be used without a huge loss of functionality. For this prototype, AWS is used, as it has a mature ecosystem of third-party tools, platforms, and libraries, making it easier to integrate into existing pipelines and for future enhancements. AWS S3 provides robust security features, including encryption at rest and in transit (AES-256), IAM-based access control, Bucket Policies, and ACLs. It has a wide range of compliance certifications, making it suitable for the scientific publication industry. The servers are containerized using Docker and used as Docker hosts. Private and public subnets are used for security purposes. Load Balancers along with auto-scaling groups are used to distribute the number of requests hitting the servers.

Client: The website uses HTML, CSS, Javascript, and a Flask server. So, for deploying the client, Heroku on the public network is used. This is connected to the backend server using a public facing load balancer in AWS.

Backend: The Backend server runs on Flask and is deployed using AWS EC2 instances in private subnets with the help of docker. These instances are connected to the public-facing load balancer that acts as a link between the client server and the backend server. Auto-scaling groups are implemented to cater to a huge number of requests. The EC2 instances will have docker hosts running, and once the CPU utilization of the instances goes above 40%, a new instance will be added automatically so that one server is not overloaded.

6. Testing and Verification

We perform an exhaustive suite of tests and verifications. The following are the test objectives:

- To ensure that the research review support system works as expected;
- To ensure that the paper–reviewer assignment meets all the expected criteria;
- To ensure that the generated aspect scores and prediction of publication of the submitted paper is promisingly accurate.

The testing includes unit testing, integration testing, and system testing using conventional methods and involving all components of the system.

7. Experiments and Results

We evaluate each of the modules in the system separately using the appropriate metrics and methods as described below.

7.1. Aspect Score Prediction

For Aspect Score Prediction, word embeddings are generated using the GloVe framework, and the following LSTM parameters are used for prediction:

- The first layer consists of the embedded layer having 100 length vectors representing each word of the review.
- There are two layers of LSTM with 64 memory units.
- Dropouts are set to 0.5 after each LSTM layer to avoid overfitting.
- The output layer creates 3 values (0–2), one for each class.

The hyperparameters for the LSTM model are set as follows:

- Activation function—Softmax;
- Loss function—categorical_cross-entropy (due to multi-class classification problem);
- Optimizer—RMSProp;
- Number of epochs—25;
- Batch size—64.

The LSTM model summary is shown in Figure 6.

```
Model: "model_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_10 (InputLayer)        [(None, None)]            0
_________________________________________________________________
embedding_4 (Embedding)      (None, None, 100)         805400
_________________________________________________________________
lstm_8 (LSTM)                (None, None, 64)          42240
_________________________________________________________________
dropout_13 (Dropout)         (None, None, 64)          0
_________________________________________________________________
lstm_9 (LSTM)                (None, 64)                33024
_________________________________________________________________
dropout_14 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_14 (Dense)             (None, 3)                 195
_________________________________________________________________
activation_4 (Activation)    (None, 3)                 0
=================================================================
Total params: 880,859
Trainable params: 75,459
Non-trainable params: 805,400
_________________________________________________________________
```

Figure 6. Model summary of the LSTM architecture used for Aspect Score Prediction.

Since it is a multi-class classification problem, the categorical cross entropy (CCE) loss function is used to measure the performance. Equation (15) gives the expression for computing CCE:

$$CCE(t,s) = -\Sigma t_{i,c} log(s_{i,c}) \quad (15)$$

where $t_{i,c}$ and $s_{i,c}$ are the ground truth and predicted aspect scores of the research paper for each class c in C.

Table 2 summarizes the CCE for various aspects.

Table 2. Aspect Score Prediction model results.

Aspect	Training Loss	Validation Loss	Test Loss
Clarity	0.4953	0.6592	0.6196
Impact	0.5052	0.6843	0.7515
Soundness	0.5063	0.6649	0.6453
Originality	0.4697	0.8754	0.7515

We apply sigmoid activation to the scores before the CCE. We are able to achieve good results for the clarity and soundness aspects of the review compared to the impact and

originality. Figure 7 presents the training and validation loss of models over 25 epochs. Loss for impact and originality starts increasing at the end of the epochs. We use the early stopping technique to stop the model training whenever there is a rise in the validation loss.

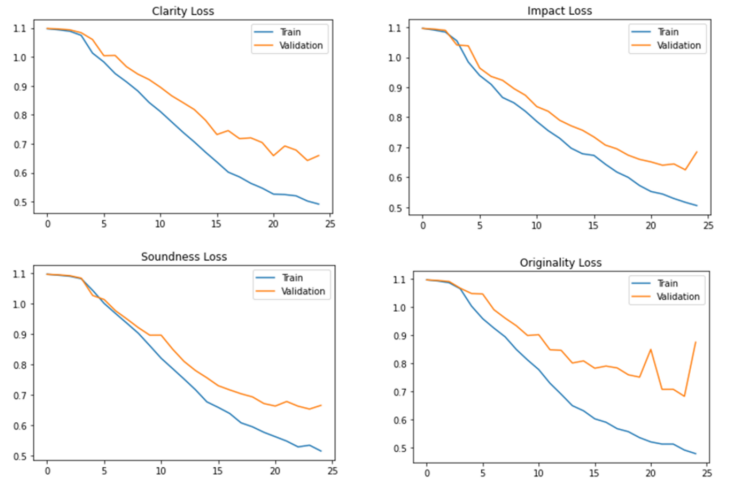

Figure 7. Training and validation loss of the LSTM model for Aspect Score Prediction.

7.2. Acceptance Prediction

For acceptance prediction, three classifiers are used and evaluated: the Support Vector Machine (SVM) classifier, a sequential neural network classifier, and a transformer architecture-based classifier. The SVM model used, with "poly" kernel, degree = 3, regularization parameter C = 1.0, cache_size of 200 and kernel coefficient, gamma "scale", produces a test accuracy of 61.85%. The losses are tabulated in Table 3. Figures 8–10 illustrates the classification process with PaLM 2 embeddings and Figures 11 and 12 with BERT embeddings using ANN and LSTM classifiers, respectively.

For the second approach, a sequential neural network with the Adam optimizer along with a 0.0001 learning rate, Softmax and Sigmoid activation, 25 epochs, and batch size of 64 is used for training the model. The sparse categorical cross entropy loss function is used to calculate the loss. The sequential neural network produces a test accuracy of 72.05%. As we can see from Table 3, the sequential neural network classifier fares better on the dataset. Table 4 summarizes the experiments using the embeddings generated from neural NLP models.

Table 3. Acceptance prediction model results using TF-IDF.

Classifier	Training Accuracy	Validation Accuracy	Test Accuracy
Support Vector Machine	70.21%	64.71%	61.85%
Sequential Neural Network	81.69%	73.78%	72.05%

Table 4. Acceptance prediction model results using neural embedding schemes.

Embedding Technique	Classification Model	Accuracy
PaLM 2 Gecko	Fully Connected Sequential ANN	84%
Word2Vec	LSTM	72%
BERT	LSTM	82%
BERT	Fully Connected Sequential ANN	84%
DistilBERT	Classification head on top of transformer base	86%
DistilBERT	Support Vector Machines	70%
FastText	Support Vector Machines	84%

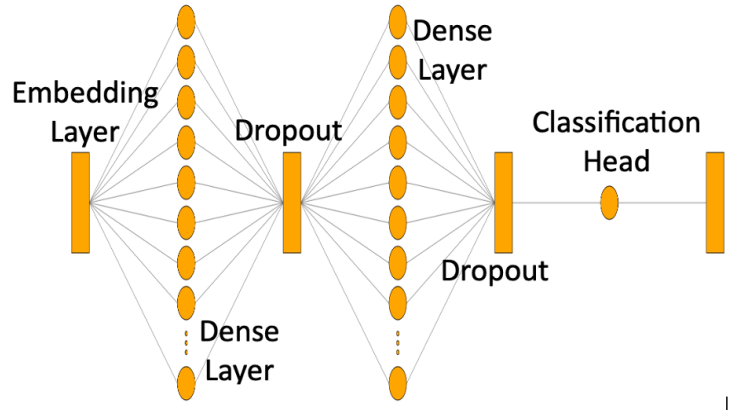

Figure 8. PaLM 2 embeddings: model schematic

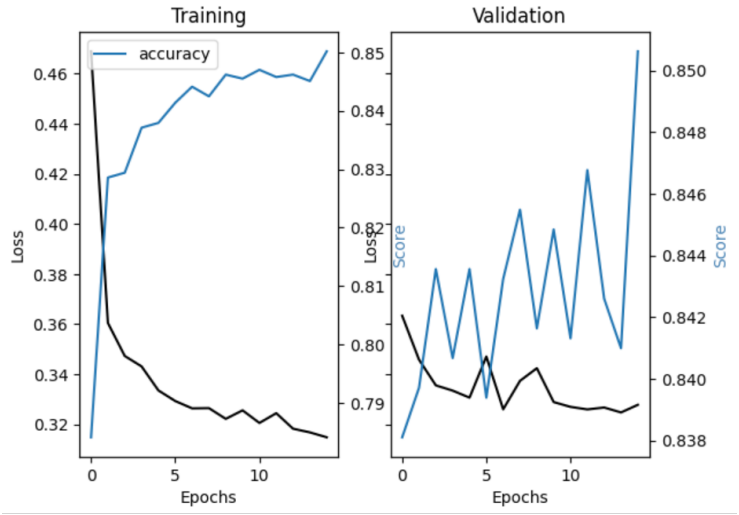

Figure 9. PaLM 2 embeddings: training and validation loss vs. accuracy for the ANN model

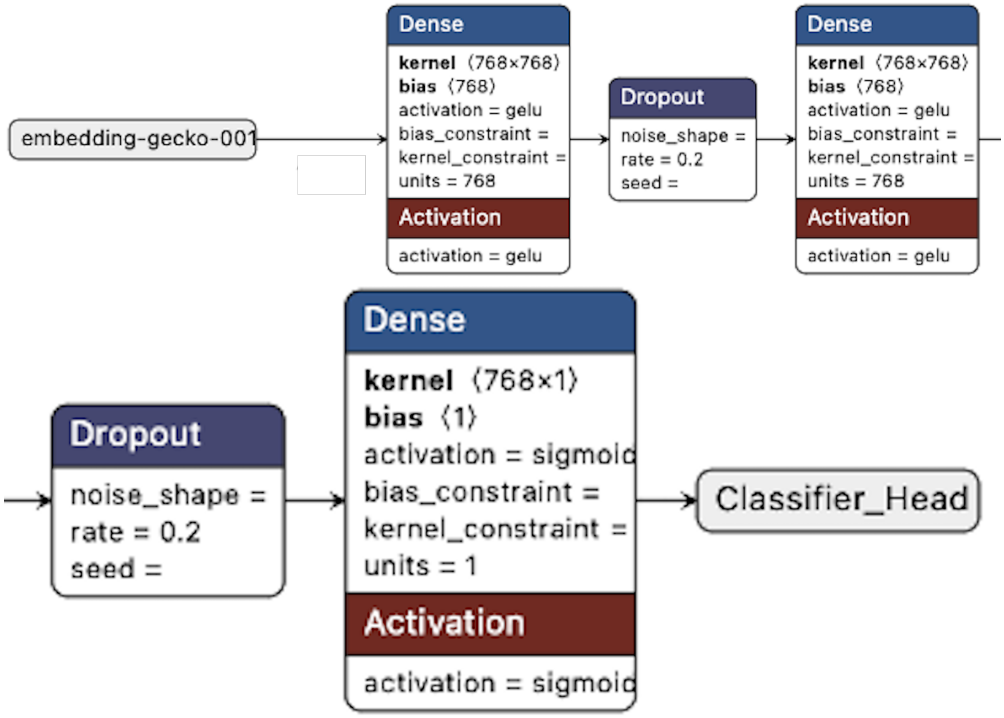

Figure 10. PaLM 2 embeddings: model configuration.

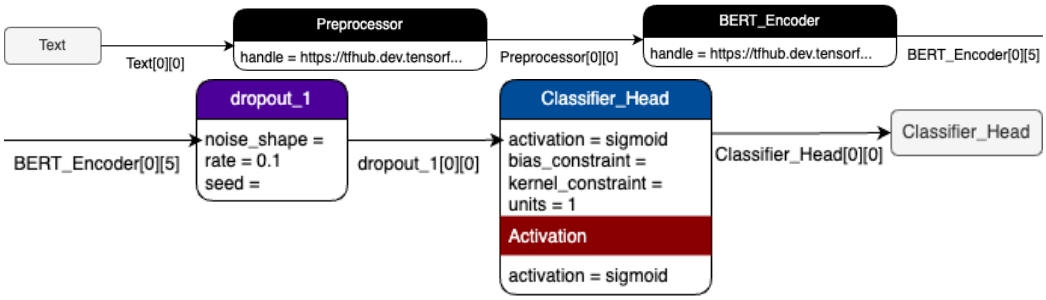

Figure 11. Model Configuration with BERT embeddings using Sequential ANN.

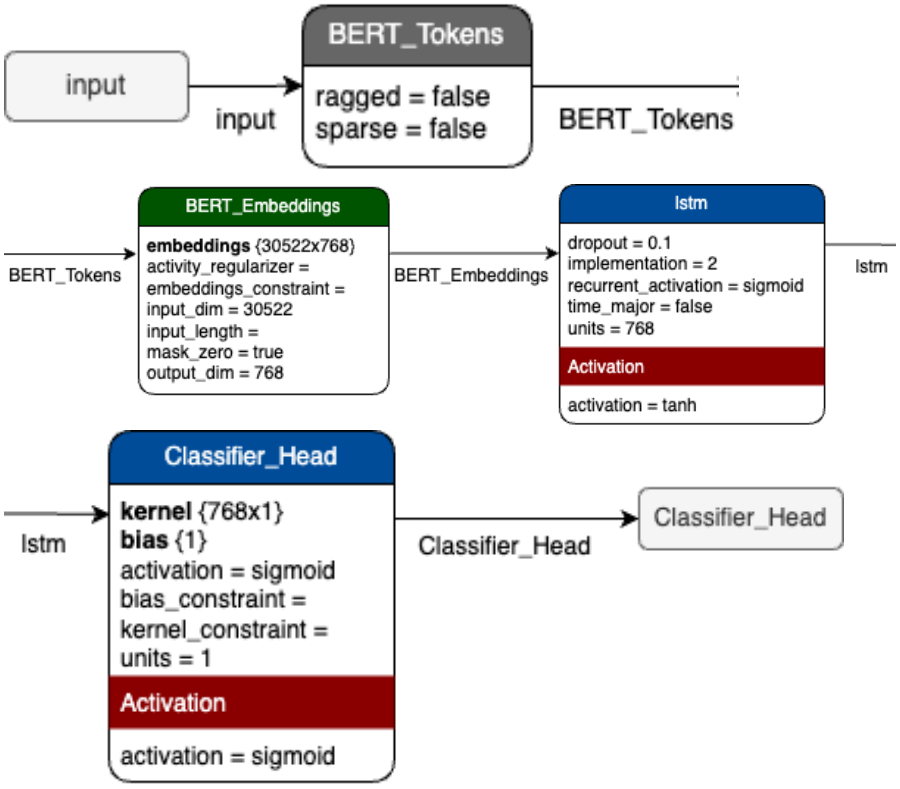

Figure 12. Model configuration with BERT embeddings using LSTM.

More detailed performance analysis is provided in Figures 13–16.

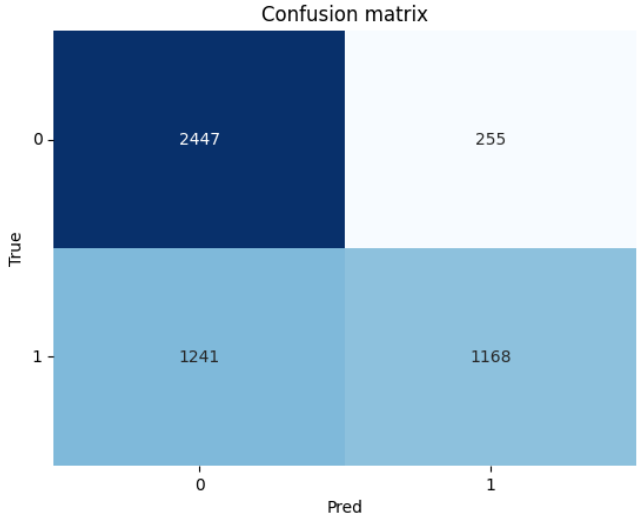

Figure 13. Confusion matrix for the results from applying SVM on DistilBERT embeddings.

```
Accuracy: 0.84
Detail:
              precision    recall  f1-score   support

           0       0.80      0.93      0.86      2691
           1       0.90      0.75      0.82      2420

    accuracy                           0.84      5111
   macro avg       0.85      0.84      0.84      5111
weighted avg       0.85      0.84      0.84      5111
```

Figure 14. All computed metrics for classification using ANN on BERT embeddings.

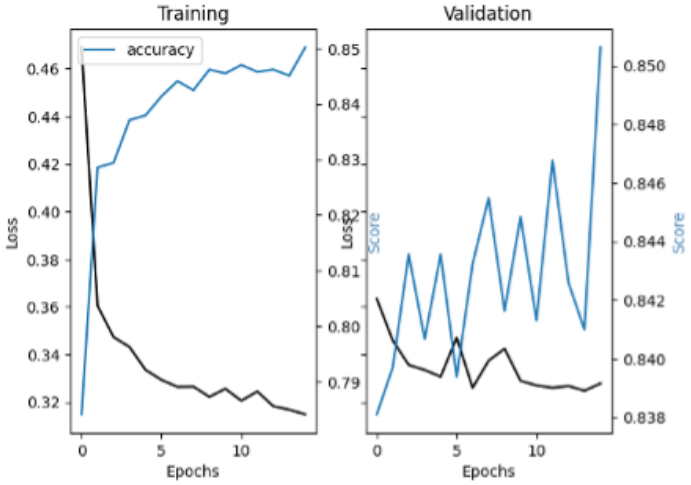

Figure 15. Training and test loss and accuracy progression plot using ANN on PaLM 2 embeddings.

```
Accuracy: 0.84
Detail:
              precision    recall  f1-score   support

           0       0.79      0.94      0.86      2688
           1       0.92      0.73      0.81      2423

    accuracy                           0.84      5111
   macro avg       0.85      0.83      0.84      5111
weighted avg       0.85      0.84      0.84      5111
```

Figure 16. All computed metrics for classification using ANN on PaLM 2 embeddings.

7.3. Reviewer Assignment

The reviewer assignment module is mainly evaluated in two subjective ways. First is manual testing. The set of reviewers assigned for each paper is analyzed by their previous work to determine if the assignment is justifiable. The model is also evaluated using nearest neighbors. The nearest neighbors for each paper in the dataset are found using the TF-IDF vectorizer used for the assignment problem. The group of similar papers hence found is manually observed for common topics by which they are associated. Most of the groups of nearest neighbors have at least a few key entities in common. Therefore, using these two human-in-the-loop methods, the performance of the reviewer assignment model is evaluated and found to be assigning relevant reviewers. We find that the nearest neighbor algorithm performs considerably well.

8. Discussion

Overall, the experiments are substantially successful, and the research questions are answered sufficiently. As expected, embeddings generated from neural NLP models perform substantially better than the TF-IDF framework. PaLM 2-based embeddings do

not exhibit any better performance than BERT, which is a surprise. DistilBERT embeddings used with an automatically selected model classifier based on the transformer architecture obtain the best results. To answer RQ1, it can be concluded from the experiment results that neural NLP and LLM-based embeddings are reliable enough to be used in the peer review process to support reviewers.

The TF-IDF, Word2Vec, and GloVe approaches to representation learning are inexpensive, and through the experiments detailed in this paper, reasonably help in natural language understanding. RQ2 is therefore answered appropriately.

The architecture detailed in this paper includes efficient components from NOSQL, Cloud Computing, and web application building tool sets, and provides an answer to RQ3. The architecture is efficient and practicable. It is hoped that this further facilitates adoption in real-world scenarios. Moreover, with the recent developments in NOSQL databases, the system can be enhanced to leverage vector search for reviewer assignment. This can be one of the future directions.

The prediction of acceptance based on LSTM does not work as well as the sequential ANN model possibly because of the long documents that research papers typically are. The usual length of a research paper is several words more than what LSTM can model as a sequence. Still, as can be seen from the results, it is a good compromise.

In terms of embeddings, even the pre-trained transformer architecture-based language models such as BERT are not able to handle the length, and the performance still suffers [7]. PaLM 2-based embeddings are expected to perform better in this regard because of the claim that "PaLM 2 was trained to increase the context length of the model significantly" [43]. However, the performance difference is not noticeable.

Another issue we have is with the nature of our dataset. Initially, we are not able to achieve good results because of the imbalanced data. For instance, the left side bar chart in Figure 17 shows the reviewers' scores for "Originality" assigned by the reviewers for various papers. As can be seen, the scores are highly imbalanced. Most of the papers are either rated "1" or "5" by the reviewers. To solve this problem, we use two techniques. The first combines minority classes, and the second is the downsampling technique. The score distribution before and after fixing the data imbalance problem is shown in Figure 17. As can be seen, these two techniques effectively result in a more balanced dataset and significant improvement in the performance of the models.

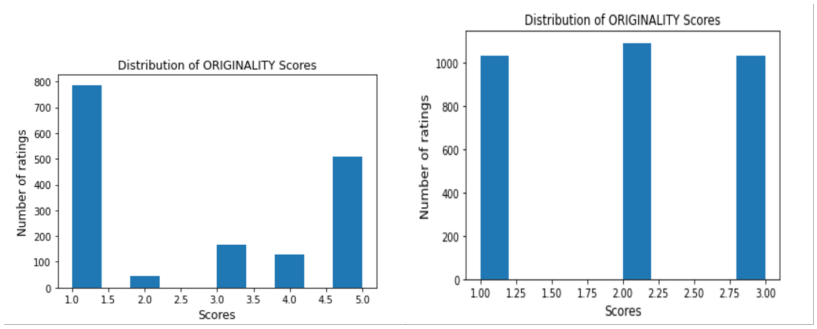

Figure 17. Score distribution before and after fixing the data imbalance problem.

Limitations

For evaluating the reviewer assignment module, manual intervention was needed. In a blind review process, the names of the reviewers is never revealed, so there was no way to automatically test how well the algorithm performed. The only alternative left was to evaluate the model's performance was to examine the reviewer assignments manually. The aim of the project is not to entirely replace the human reviewers but to support their efforts as much as possible.

The techniques used in the paper for correcting the data imbalance problems can impact the model performance. However, currently, there is no other technique or tool to address the issue. Using oversampling techniques with synthetic data in the form of fake research papers and their synthesized scores for balancing the dataset may make the problem worse. This limitation notwithstanding, it must be noted that the performance is satisfactory.

9. Conclusions and Future Directions

The system developed in this paper provides a platform for EIC/TPC chairs, authors, and reviewers to expedite the process of research paper publication. The system proposed in this paper is complementary and prior to human judgment. It does not aim to replace human judgment. With this portal, the EIC/TPC chair can obtain a list of recommended reviewers based on the submitted research paper and the reviewer's previous publications, obtain the aspect scores instantly from the reviews, and act upon it. Moreover, authors can instantly know their research paper acceptance probability once they submit the research paper to a conference.

For the aspect score prediction model, LSTM with softmax activation function and categorical_cross-entropy loss function returned good results. The sequential neural network with four hidden layers, Adam optimizer and sparse categorical cross entropy returned an accuracy of 74% on the test dataset. The nearest neighbors technique on the TF-IDF matrix for the reviewer assignment module proved to be quite successful as well. This system expedites the paper publication process for each user involved in the process.

The work also demonstrated the maturity of language models in assisting research paper reviewers and editors to indicate the acceptability of the contributed paper. In terms of just the embeddings generated, recent LLMs such as PaLM 2 do not seem to offer any significant performance gains over their predecessors such as BERT. PaLM 2 uses the Pathways architecture, which aims to improve efficiency and scalability across tasks. PaLM 2 is a generative model and performs very well on both generative tasks like text generation and discriminative tasks like classification. However, from the results obtained for this work, the embeddings generated by BERT are as good as those from PaLM 2 for downstream tasks such as the classification of the papers. It confirms that for many standard downstream tasks, fine-tuned BERT embeddings can already provide strong performance, and in some cases, they might be competitive with newer models like PaLM 2.

Further research could include predicting the aspect scores without the reviews and by just using the research paper sections and extracted features from the research paper. This can enhance the usability of the application to a large extent, as just by submitting the research paper, the author will be able to find the scores on each of the sections that the research paper is being judged on. Another direction for future research is to provide the reasons why the research paper is rejected or accepted.

Explainability is an important emerging field in machine learning and can substantially enhance applications in this domain. It is helpful if the editors or the reviewers can obtain an idea of the specific sections that highlight or degrade the quality of the research paper. As part of future work, we also plan to explore the explainability of the results and the MongoDB vector search to better support the needs of its users. Given that the existing work is only a prototype, it can be extended into an industry-strength solution by addressing the model inference latency and concurrency, more efficient model deployment, and an exhaustive analysis of the technology stack alternatives.

Author Contributions: Conceptualization, V.S.P.; Methodology, V.S.P.; Software, V.S.P., K.K. and K.M.; Validation, V.S.P., K.K. and K.M.; Formal analysis, V.S.P.; Investigation, V.S.P., K.K. and K.M.; Resources, K.K. and K.M.; Data curation, K.K. and K.M.; Writing—original draft, V.S.P., K.K. and K.M.; Writing—review & editing, V.S.P. and K.K.; Visualization, K.K. and K.M.; Supervision, V.S.P.; Project administration, V.S.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on the NeurIPS Conference website at https://proceedings.neurips.cc/, (accessed on 6 January 2025) and the public dataset at https://paperswithcode.com/dataset/peerread, (accessed on 6 January 2025) reference number [9]. These data were derived from the following resources available in the public domain: https://proceedings.neurips.cc/, (accessed on 6 January 2025).

Acknowledgments: The authors acknowledge the help of Sathya Sri Pasham and Shivani Shivanand Suryawanshi for their inputs and contributions to this project. Special thanks to Christopher Eliot Hall for working on some of the experiments and making an impact to this project in a short amount of time.

Conflicts of Interest: Author Karnavee Kamdar was employed by the company Oracle. Author Kapil Mulchandani was employed by the company Amazon. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Pendyala, V.S. Why Scientific Publishing Culture Needs a Major Overhaul in the Age of Generative AI, VISIBLE Magazine. 2024. Available online: https://visiblemagazine.com/why-scientific-publishing-culture-needs-a-major-overhaul-in-the-age-of-generative-ai/ (accessed on 6 January 2025).
2. Bornmann, L.; Haunschild, R.; Mutz, R. Growth rates of modern science: A latent piecewise growth curve approach to model publication numbers from established and new literature databases. *Humanit. Soc. Sci. Commun.* **2021**, *8*, 1–15. [CrossRef]
3. Bornmann, L.; Mutz, R. Growth rates of modern science: A bibliometric analysis based on the number of publications and cited references. *J. Assoc. Inf. Sci. Technol.* **2015**, *66*, 2215–2222. [CrossRef]
4. García, J.A.; Rodriguez-Sánchez, R.; Fdez-Valdivia, J. Confirmatory bias in peer review. *Scientometrics* **2020**, *123*, 517–533. [CrossRef]
5. Tomkins, A.; Zhang, M.; Heavlin, W.D. Reviewer bias in single-versus double-blind peer review. *Proc. Natl. Acad. Sci. USA* **2017**, *114*, 12708–12713. [CrossRef] [PubMed]
6. Yue, M.; Tang, H.; Liu, F.; Ma, T. Consistency index: Measuring the performances of scholar journal reviewers. *Scientometrics* **2021**, *126*, 7183–7195. [CrossRef]
7. Yuan, W.; Liu, P.; Neubig, G. Can we automate scientific reviewing? *J. Artif. Intell. Res.* **2022**, *75*, 171–212. [CrossRef]
8. Ji, Z.; Lee, N.; Frieske, R.; Yu, T.; Su, D.; Xu, Y.; Ishii, E.; Bang, Y.J.; Madotto, A.; Fung, P. Survey of hallucination in natural language generation. *ACM Comput. Surv.* **2023**, *55*, 1–38. [CrossRef]
9. Kang, D.; Ammar, W.; Mishra, B.D.; Van Zuylen, M.; Kohlmeier, S.; Hovy, E.; Schwartz, R. A dataset of peer reviews (PeerRead): Collection, insights and NLP applications. In Proceedings of the NAACL HLT 2018—2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, New Orleans, LO, USA, 1–6 June 2018; Volume 1, pp. 1647–1661.
10. Plank, B.; van Dalen, R. CiteTracked: A Longitudinal Dataset of Peer Reviews and Citations. In Proceedings of the BIRNDL@ SIGIR 2019, Paris, France, 25 July 2019; pp. 116–122.
11. Singh, S.; Singh, M.; Goyal, P. COMPARE: A Taxonomy and Dataset of Comparison Discussions in Peer Reviews. In Proceedings of the ACM/IEEE Joint Conference on Digital Libraries, Champaign, IL, USA, 27–30 September 2021; pp. 238–241. [CrossRef]
12. Zhao, X.; Zhang, Y. Reviewer assignment algorithms for peer review automation: A survey. *Inf. Process. Manag.* **2022**, *59*, 103028. [CrossRef]
13. Anjum, O.; Gong, H.; Bhat, S.; Xiong, J.; Hwu, W.M. Pare: A paper-reviewer matching approach using a common topic space. In Proceedings of the EMNLP-IJCNLP 2019, Hong Kong, 3–7 November 2019; pp. 518–528.

14. He, Y. The study on expert selection in peer-review based on knowledge management-the new application of scientometrics. In Proceedings of the 2009 First International Conference on Information Science and Engineering, Nanjing, China, 26–28 December 2009; pp. 4605–4608.
15. He, Y.; Tian, K. Peer-review experts selection for evaluating interdisciplinary studies based on scientific knowledge mapping. In Proceedings of the 2018 4th International Conference on Information Management (ICIM), Oxford, UK, 25–27 May 2018; pp. 275–279.
16. Li, B.; Hou, Y.T. The new automated IEEE INFOCOM review assignment system. *IEEE Netw.* **2016**, *30*, 18–24. [CrossRef]
17. Liu, X.; Suel, T.; Memon, N. A robust model for paper reviewer assignment. In Proceedings of the 8th ACM Conference on Recommender Systems, Foster City, CA, USA, 6–10 October 2014; pp. 25–32.
18. Kou, N.M.; U, L.H.; Mamoulis, N.; Li, Y.; Li, Y.; Gong, Z. A topic-based reviewer assignment system. *Proc. VLDB Endow.* **2015**, *8*, 1852–1855. [CrossRef]
19. Karimzadehgan, M.; Zhai, C.; Belford, G. Multi-aspect expertise matching for review assignment. In Proceedings of the 17th ACM Conference on Information and Knowledge Management, Napa Valley, CA, USA, 26–30 October 2008; pp. 1113–1122.
20. Yang, C.; Liu, T.; Yi, W.; Chen, X.; Niu, B. Identifying expertise through semantic modeling: A modified BBPSO algorithm for the reviewer assignment problem. *Appl. Soft Comput.* **2020**, *94*, 106483. [CrossRef]
21. Papalexakis, E.E.; Sidiropoulos, N.D.; Garofalakis, M.N. Reviewer profiling using sparse matrix regression. In Proceedings of the 2010 IEEE International Conference on Data Mining Workshops, Sydney, Australia, 13–17 December 2010; pp. 1214–1219.
22. Ferilli, S.; Di Mauro, N.; Basile, T.M.A.; Esposito, F.; Biba, M. Automatic topics identification for reviewer assignment. In Proceedings of the International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, Annecy, France, 27–30 June 2006; pp. 721–730.
23. Charlin, L.; Zemel, R.; Boutilier, C. A framework for optimizing paper matching. In Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence, Barcelona, Spain, 14–17 July 2011; pp. 86–95.
24. Charlin, L.; Zemel, R. The Toronto paper matching system: An automated paper-reviewer assignment system. In Proceedings of the International Conference on Machine Learning (ICML), Atlanta, GA, USA, 16–21 June 2013.
25. Jin, J.; Niu, B.; Ji, P.; Geng, Q. An integer linear programming model of reviewer assignment with research interest considerations. *Ann. Oper. Res.* **2020**, *291*, 409–433. [CrossRef]
26. Patil, A.H.; Mahalle, P.N. Trends and Challenges in Measuring Performance of Reviewer Paper Assignment. *Procedia Comput. Sci.* **2020**, *171*, 709–718. [CrossRef]
27. Nguyen, J.; Sánchez-Hernández, G.; Agell, N.; Rovira, X.; Angulo, C. A decision support tool using Order Weighted Averaging for conference review assignment. *Pattern Recognit. Lett.* **2018**, *105*, 114–120. [CrossRef]
28. Qiao, F.; Xu, L.; Han, X. Modularized and attention-based recurrent convolutional neural network for automatic academic paper aspect scoring. In Proceedings of the International Conference on Web Information Systems and Applications, Taiyuan, China, 14–15 September 2018; pp. 68–76.
29. Flach, P.A.; Spiegler, S.; Golénia, B.; Price, S.; Guiver, J.; Herbrich, R.; Graepel, T.; Zaki, M.J. Novel tools to streamline the conference review process: Experiences from SIGKDD'09. *ACM SIGKDD Explor. Newsl.* **2010**, *11*, 63–67. [CrossRef]
30. Pesenhofer, A.; Mayer, R.; Rauber, A. Improving scientific conferences by enhancing conference management systems with information mining capabilities. In Proceedings of the 2006 1st International Conference on Digital Information Management, Bangalore, India, 6–8 December 2006; pp. 359–366.
31. Tenorio-Fornés, Á.; Tirador, E.P.; Sánchez-Ruiz, A.A.; Hassan, S. Decentralizing science: Towards an interoperable open peer review ecosystem using blockchain. *Inf. Process. Manag.* **2021**, *58*, 102724. [CrossRef]
32. Skorikov, M.; Momen, S. Machine learning approach to predicting the acceptance of academic papers. In Proceedings of the 2020 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT), Bali, Indonesia, 7–8 July 2020; pp. 113–117.
33. Ghosal, T.; Verma, R.; Ekbal, A.; Bhattacharyya, P. A sentiment augmented deep architecture to predict peer review outcomes. In Proceedings of the 2019 ACM/IEEE Joint Conference on Digital Libraries (JCDL), Champaign, IL, USA, 2–6 June 2019; pp. 414–415.
34. Bharti, P.K.; Ghosal, T.; Agarwal, M.; Ekbal, A. PEERRec: An AI-based approach to automatically generate recommendations and predict decisions in peer review. *Int. J. Digit. Libr.* **2023**, *25*, 55–72. [CrossRef]
35. Bao, P.; Hong, W.; Li, X. Predicting Paper Acceptance via Interpretable Decision Sets. In Proceedings of the Companion Proceedings of the Web Conference 2021, New York, NY, USA, 19–23 April 2021; pp. 461–467.
36. Xie, J. Predicting institution-level paper acceptance at conferences: A time-series regression approach. In Proceedings of the KDD Cup Workshop, San Francisco, CA, USA, 13–17 August 2016; pp. 1–6.
37. Kerzendorf, W.E.; Patat, F.; Bordelon, D.; van de Ven, G.; Pritchard, T.A. Distributed peer review enhanced with natural language processing and machine learning. *Nat. Astron.* **2020**, *4*, 711–717. [CrossRef]

38. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient Estimation of Word Representations in Vector Space. In Proceedings of the International Conference on Learning Representations (ICLR), Scottsdale, AZ, USA, 2–4 May 2013.
39. Pennington, J.; Socher, R.; Manning, C.D. Glove: Global vectors for word representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; pp. 1532–1543.
40. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the 2018 Conference on Neural Information Processing Systems, Montreal, QC, Canada, 3–8 December 2018; pp. 4171–4186.
41. Sanh, V.; Debut, L.; Chaumond, J.; Wolf, T. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. In Proceedings of the NeurIPS 2019, Vancouver, BC, Canada, 8–14 December 2019.
42. Bojanowski, P.; Grave, E.; Joulin, A.; Mikolov, T. Enriching Word Vectors with Subword Information. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Copenhagen, Denmark, 9–11 September 2017; pp. 2369–2379.
43. Anil, R.; Dai, A.M.; Firat, O.; Johnson, M.; Lepikhin, D.; Passos, A.; Shakeri, S.; Taropa, E.; Bailey, P.; Chen, Z.; et al. Palm 2 technical report. *arXiv* **2023**, arXiv:2305.10403.
44. Chowdhery, A.; Narang, S.; Devlin, J.; Bosma, M.; Mishra, G.; Roberts, A.; Barham, P.; Chung, H.W.; Sutton, C.; Gehrmann, S.; et al. Palm: Scaling language modeling with pathways. *arXiv* **2022**, arXiv:2204.02311.
45. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 5998–6008.
46. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]
47. Siami-Namini, S.; Tavakoli, N.; Namin, A.S. The performance of LSTM and BiLSTM in forecasting time series. In Proceedings of the 2019 IEEE International conference on big data (Big Data), Los Angeles, CA, USA, 9–12 December 2019; pp. 3285–3292.
48. Cortes, C. Support-Vector Networks. *Mach. Learn.* **1995**, *20*, 273–297. [CrossRef]