SmartReviewHub: An AI Based Scientific Conference Management System

Mihail Popescu
University of Craiova
Craiova, Romania
mihaixz4@gmail.com

Paul-Stefan Popescu
University of Craiova
Craiova, Romania
stefan.popescu@edu.ucv.ro

Marian Cristian Mihaescu
University of Craiova
Craiova, Romania
cristian.mihaescu@edu.ucv.ro

ABSTRACT

Conference management systems are one of the most essential tools for evaluating papers submitted to conferences or journals. These systems evolved over the years, providing better functionalities to facilitate the conference management process. The problem tackled by this paper is to provide a system that enhances the user experience and ease of use for all the actors involved. The approach was to integrate machine learning and deep learning features in order to provide features like concept extraction, plagiarism detection, spell checker or even sentiment analysis. The result is an intelligent conference system that can provide all the classic and new functionalities for an optimised user experience.

Author Keywords

Conference system, machine learning, deep learning, user experience.

DOI: 10.37789/ icusi.2024.6

INTRODUCTION

Scientific events rely on their management systems for reviews, evaluation and paper ranking. Such systems mainly include basic functionalities like paper uploading along with an abstract, title, keywords, and authors; then, reviewers can assign them automatically, or they can be assigned randomly from a predefined list. After the evaluation part is completed, a paper ranking is available, and we have the next step of announcing the authors about the decision. Then, the camera-ready paper is prepared, and in the end, we have the final papers. This kind of functionality is quite common in many paper management systems and provides the baseline on which we propose our system. We also need to mention here that there are several roles in such systems, and the most common are author, program committee and organiser, and each of them has several actions that can be performed. Most of the systems like EasyChair [1], ConfBay, OpenConf [2], Microsoft CMT, COMS, Papercept, ConfSys [3,4] and EDAS implement the main functionalities.

Nowadays, machine learning and deep learning algorithms have become more and more popular and are integrated into a variety of systems, but most scientific management systems don't benefit from what intelligent algorithms have to offer. We propose a system that provides functionalities that help all the main actors from such a system directly and indirectly.

The system presented in our paper addresses the idea that we can merge standard conference systems with machine learning and deep learning technologies in order to provide an up-to-date system that includes new functionalities. These functionalities are designed to increase productivity and reduce the time for all the users of conference management systems.

The standard functionalities of the system include setting up a conference, paper uploading, reviewer assignment, paper evaluation, etc., and covering most of the features available in standard conference management systems.

The advanced functions included in SmartReviewHub are:

- Concept extraction/topic detection. This is very useful for keywords and more accurate reviewer matching.
- Plagiarism detection This can be performed using third-party applications, but having an integrated feature makes the process more convenient.
- Spell checkers help evaluate and improve the language because most authors may not speak native English. Like the previous one, there are third-party apps that can be used, but having it implemented can increase the user's experience.
- Sentiment analysis may not seem the most relevant for scientific papers, but measuring how neutral a paper is may provide insight into its quality.

The system's goal is to provide many functionalities and enhance the user's experience and ease of use because the interface is optimised to deliver the features exactly where they are needed. We also consider optimising the number of actions an author needs to do to accomplish their goal. Here, the system adds a straightforward benefit because authors don't have to log into a different system, upload their paper there, and re-upload it in the standard conference management system.

RELATED WORK

Research in the area of conference management systems plays an important part and is quite actual, as presented in the survey [5], which analyses EasyChair, ConfBay, OpenConf, Microsoft CMT, COMS, and EDA as tools commonly employed by conference organisers. The survey examines the strengths and features of each system, with a focus on critical aspects such as paper submission, the review process, registration, agenda and program management, virtual conference support, proceedings, and e-mail communication. The analysis serves as a valuable resource for navigating the complex landscape of CMS platforms, guiding organisers toward optimal choices aligned with their unique event management needs.

However, the interest in paper management systems is not new, as we can see in [6], which presents a Conference Management Online System that provides an easier way of managing events when conducting a conference or in [7].

One key feature previously discussed was the automatic assignment of reviewers to papers as presented in [8], which provides an algorithm for an automatic assignment that takes into account all - selected keywords, reviewers' bids and conflicts of interest and tries to find the most accurate assignment while maintaining load balancing among reviewers.

Another relevant task for conference management systems is the detection of conflicts of interest, as presented in [9]. The authors propose an utterly novel framework that can be practically implemented to improve the performance of existing systems. They map the reviewer assignment problem to an equilibrium multi-job assignment problem. Moreover, they propose a meta-heuristic greedy solution to solve it using weighted matrix factorisation.

Other systems are constantly developed and kept up to date, like MyConfree presented in [10], which was created 15 years ago but is still in use.

SYSTEM DESIGN

The system is designed so that multiple types of roles with their specific attributes can interact seamlessly through the web application. Figure 1 shows the main actions each role can take.

An author can upload documents, perform various tests and other processing actions on them, or send them directly for review to the conference they choose. They will also see all the information about the documents, tests, and results clearly and straightforwardly.

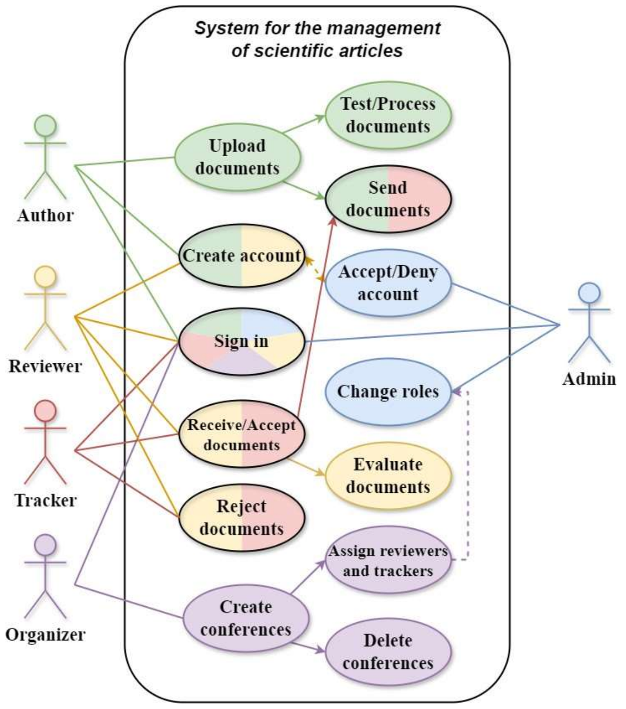


Figure 1. Use case diagram

A reviewer receives documents only for the conferences to which they are assigned and has all the tools needed for evaluation, including viewing the documents and returning the results.

A tracker, in the same way, receives documents for the conferences where they are assigned and has all the information necessary to match the documents with the reviewers who are qualified to assess them through a simple UI.

An organiser can create conferences with the most relevant information and then assign specific people as reviewers or trackers to them. They also have the option to delete the conferences they made when they end or if an issue arises.

The admin can accept or deny requests for reviewer accounts and change the roles of any user on the platform. Additionally, they are provided with relevant information for all ongoing reviews to resolve any issues.

A person can hold different roles at different conferences, so an account should be able to have multiple roles simultaneously. A custom user model was created in Django to accommodate this, where each role status is saved as a Boolean value. The admin role, however, is managed in a separate Django model.

Consolidating all the primary roles into a single model simplifies the interconnection of all the models required for other functionalities. As shown in Figure 2, most tables are connected to the CustomUser table, with only the table related to the admin role functionality being separate.

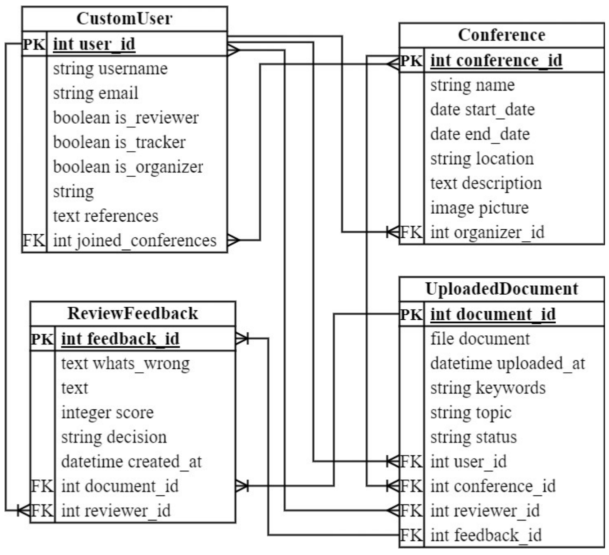


Figure 2. Database design

Another core component of the system design is how documents are sent from one role to another. Instead of having a single copy of the document in the database shared between different roles, different copies were created. This way, when an author sends a document for review, the tracker receives a copy of it saved under their ID in the database. The exact process is followed when the tracker sends it to the reviewers later on. This method uses more memory space but enhances the safety of the reviewing process.

Having all the roles in one model and creating a copy of the documents for each role added a level of complexity when creating some functionalities and, for example, differentiating between the documents uploaded by the author and those received by the tracker and reviewer when they are being sent, rejected, tested on, etc.

Figure 3 presents the central system architecture along with the functionalities implemented in the system. The system processes author-uploaded documents to perform spell-checking, plagiarism detection, sentiment analysis, and entity extraction. In the first step, we have the User Dashboard Request, which is the initial step in which an author makes a request from the dashboard interface. This could be a request to upload a document for a conference. Django Development Server is the next step where a Django development server receives and handles the request. Django serves as the web framework to manage incoming requests, interact with the database, and coordinate the processing pipeline. The text extraction (Fitz/Python-docx) extracts the text document content. Then, the documents, including their metadata and possibly the extracted text, are stored in an SQLite3 database. This allows for persistent storage and easy retrieval of documents. The text Preprocessing (NLTK) module includes tasks such as tokenisation, removing stop words, and other normalisation processes to prepare the text for analysis. The spell Checking (PyEnchant) checks the text for English spelling errors using PyEnchant and outputs Suggestions or corrections for any detected spelling errors. Plagiarism Detection (Gensim) is another functionality that checks for plagiarism. Gensim is a library often used for topic modelling and similarity detection. The Sentiment Analysis (Transformers) module performs sentiment analysis using transformer models, which are state-of-the-art in natural language processing (NLP) for understanding the sentiment or emotions expressed in the text. Entity extraction is the last functionality, which is conducted using spaCy to identify entities, such as names, dates, locations, etc., within the text and outputs a list of recognised entities with their types. The results from all the analyses (spell checking, plagiarism detection, sentiment analysis, and entity extraction) are compiled and returned to the author.

Web App Features

Every role has its own dashboard page where specific functionalities are available. These are accessible through buttons on the header that appear only if the user has the required roles.

The author dashboard has a section where you can upload documents in either DOC or PDF formats, along with their topic and keywords. After a document is uploaded, it appears in the uploaded documents section, where the author can see its status and perform multiple actions.

Each document has a preview button. When pressed, it extends a section under the document to show the contents. For PDF documents, it uses the built-in PDF viewer for browsers, and for Word documents, it extracts the text from the document and displays it directly.

Four functionalities are designed to aid the author in testing and processing their document before sending it for review.

One functionality is spell-checking, triggered by a button with the same name. It only supports the english language. The text is extracted using one of the functions for extracting text from either PDF or DOC. Then, the spell check function uses the PyEnchant library to create an English dictionary and iterates through each word to check if it is correctly spelt. The results are returned on a new page where the author is redirected. All the text from the document is displayed, with only the mistakes marked in yellow.

Another functionality intended to aid the author in extracting relevant entities from their document through a natural language model. This can help them find new or more fitting keywords for their paper. The model is implemented through the spaCy library, which provides the en_core_web model in three sizes: small (sm), medium (MD), and large (LG). Their main difference is how much computing power they will use, resulting in more or less accurate outputs. The model first tokenises the text by splitting it into basic units such as words and punctuation, then reduces the words to their base form (lemmatisation) and identifies the parts of speech for each word through POS tagging (part-of-speech tagging). Lastly, it extracts named entities, such as people, works of art, historical events, languages, laws, etc. This is done after the 'keywords helper' button is pressed, and the results are returned as a list with the words and their tags. The results are displayed on a new page within columns for each category. A word cloud of the most common words is generated using the WordCloud library and the results list for better visualisation. The output is then converted into a base64 image to be displayed on the webpage.

The sentiment check functionality is designed to use a model to examine a document and determine its overall sentiment—negative, positive, or neutral. A scientific paper should be as neutral as possible; this test can give the author an impartial perspective on their paper.

This feature can have deeper implications on what words or phrases are considered neutral, positive or negative based on what data it is trained on. That is why it is left up to the author to decide whether they want to use it and consider the result or not before they send the paper up for review. The results should be treated as a suggestion rather than a factual conclusion.

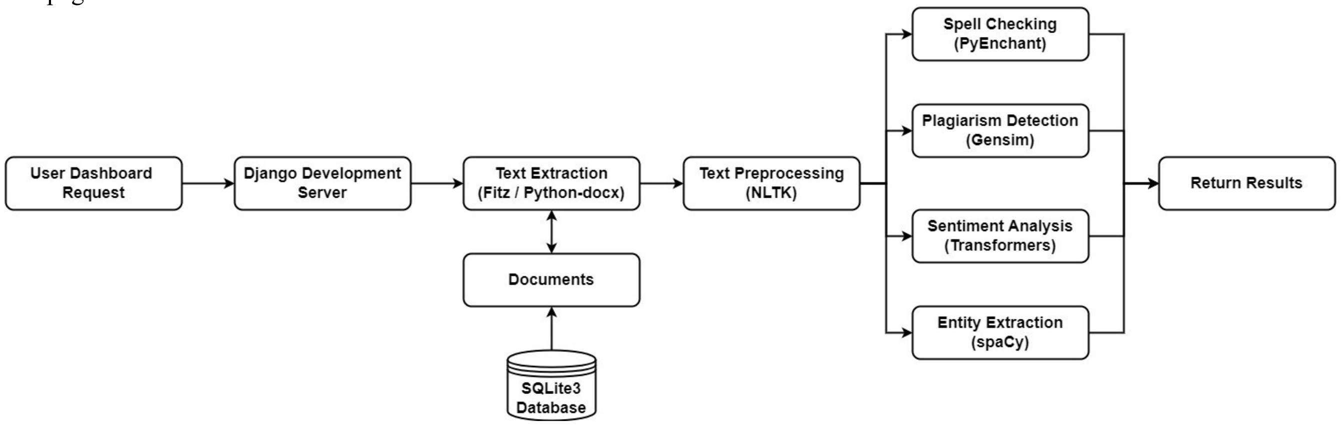

User Dashboard Request
Django Development Server
Text Extraction (Fitz / Python-docx)
Documents
SQLite3 Database
Text Preprocessing (NLTK)
Spell Checking (PyEnchant)
Plagiarism Detection (Gensim)
Sentiment Analysis (Transformers)
Entity Extraction (spaCy)
Return Results

Figure 3. Main system design

The Transformers library created and maintained by Hugging Face was used for this functionality. It contains multiple models, one of which is BERT [11] (Bidirectional Encoder Representations from Transformers). BERT can learn the context of all words in a sentence by analysing them bi-directionally, and it can be fine-tuned for various specific tasks such as text classification, sentiment analysis, question answering, and more. BERT excels at understanding the complex context of words and phrases, making it ideal for tasks that require a deep understanding of natural language and an excellent choice for sentiment analysis of scientific papers.

In the implementation, the text is extracted from the document for which the 'sentiment check' button was pressed. It is then sent to a function where the text is tokenised and analysed by the BERT model, which returns one of the three possible sentiments. The result is displayed on the author dashboard page in a section under the chosen document.

The final functionality of this type is comparing the chosen document with all the other documents uploaded by authors on the site. This is achieved using a Python library called Gensim, which implements multiple algorithms and models. For example, Latent Dirichlet Allocation (LDA) [12] is used to discover hidden topics in a set of documents, and Word2Vec is an algorithm for learning vector representations of words in a multidimensional space.

To implement this, the document for which the 'compare documents' button is pressed and all the other documents are retrieved. The text is extracted and preprocessed by the 'preprocess_text' function, and then a dictionary is created by the 'corpora. The function of the dictionary (processed_texts) is where each word is mapped to a unique ID. Next, a corpus is created by converting each preprocessed text to a bag-of-words representation using the 'dictionary.doc2bow(text)' function. The similarities between the selected document and all the other documents are then calculated using 'index[corpus[0]]', where 'corpus[0]' is the bag-of-words representation of the selected document. The similarities are sorted in descending order, and the top similar documents are selected by slicing the sorted list and retrieving the corresponding documents and their similarity scores. The results are returned as a list of tuples containing a document and its similarity score for the top similar documents.

The document names, authors and their contents are kept anonymous because some documents that are checked can still be in the process of reviewing, or their authors may not have made the work public. This differs from mainstream plagiarism checkers, which use databases of finished works where the authors consent to be checked upon. So, it would raise some issues if authors could have a way to access any information about the documents other than theirs in the database.

After completing all the checks, an author can read all the information about ongoing and upcoming conferences on the 'Conferences' page, which helps them decide the best conference for their paper. Notably, the conferences on this page are automatically moved from the 'Planned' to the 'Ongoing' section by a function that compares the current date with the start and end dates of the conferences.

Once the author has decided where to send their paper, they can use the 'Send to review' button. This opens a form in a section under the document where they can update the keywords and topic and select the conference to which they want to send the document. After making the desired changes and pressing the 'Send' button, the 'send_to_review' function sends a copy of the document to all the trackers assigned to the chosen conference by saving copies of the chosen document under the IDs of the trackers. The status of all those documents, including the author's original version, is then changed to "Submitted".

The status is a field in the document model that can be either *Uploaded, Submitted, Under Review, Reviewed, or Rejected*. It is used to indicate the review stage of the document. When the author uploads a document, it is saved with the status 'Uploaded'. Once successfully sent to a conference, its status changes to 'Submitted'. When at least one reviewer accepts the document for evaluation, the status changes to 'Under_Review'. Once all the reviewers return their evaluations, the status changes to 'Reviewed'.

When the author receives at least one result from a reviewer, the 'View Feedback' button appears under the document that received feedback. When pressed, the 'get_feedback' function returns all the results in a section under the document.
The feedback form fields that the reviewer must fill out and the author will see are: 'What's wrong:' (text type), 'What can be improved' (text type), 'Score' (number type), and 'Decision'. The 'Decision' field is a dropdown with the following options: 'Reject', 'Accept with small revisions', 'Accept with major revisions', and 'Accept'. These fields provide enough space for the reviewer to write their observations, resulting in a meaningful review process.

An organiser can access specific functionalities for their role on the organiser dashboard page. This page has three sections: one for creating new conferences, another for assigning trackers or reviewers, and the last one for viewing already created conferences with the option to delete them. A conference is created through a form with the following fields: Conference name, Start date, End date, Location, Description, and Picture. After making at least one conference, the organiser can use a search bar to find users to assign as either reviewers or trackers to their conference. When assigning a user, the 'assign_user_to_conference' function includes checks to ensure a user isn't both a reviewer and a tracker at the same conference.

Tracker can find functionalities on the tracker dashboard page that are specific to their role. The first section shows the conferences where the tracker has been assigned. The following section displays the ongoing reviews in these conferences, defined as reviews where a reviewer has accepted a document. Another section lists the reviewers and their relevant information for the conferences the tracker is overseeing. The last section shows the documents sent for review and relevant information about the authors who submitted them. This setup allows the tracker to match documents to reviewers based on their specialisations, workplaces, and achievements. After selecting a reviewer from the dropdown menu, the tracker presses the 'Match Reviewer' button, which calls the 'match_reviewer' function and makes a copy of the document under the reviewer's ID. The tracker can repeat this process multiple times, and when no more matches are possible, they can use the remove button to delete the document for themselves. If no suitable match is found, the tracker can reject the document, which deletes it for itself and notifies the author on the author dashboard page that the document was rejected.

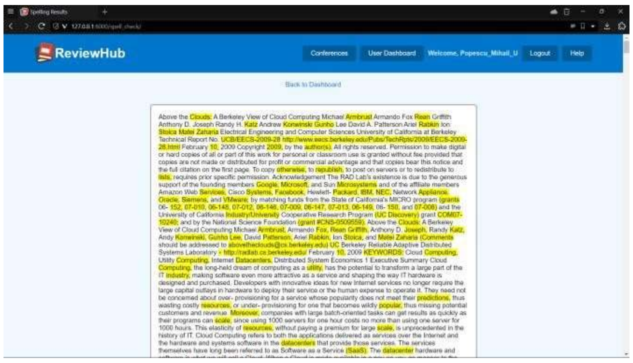
A reviewer can find the main functionalities of their role on the reviewer dashboard page. The first section displays the conferences where they have been assigned as a reviewer. The next section shows all the documents assigned to them, with a search bar to help them search by topic or keywords. A document can only be previewed, accepted, or removed in this section. If a reviewer accepts a document, the status of the reviewer's copy and the author's document is changed to 'Under_Review,' and it is moved to the last section for documents under review. Here, reviewers can download the document or return it with feedback. When the 'Return Document' button is pressed, a form opens under the document where the reviewer can write feedback, provide a score, and give their decision on acceptance.

An admin can access the main functionalities of their role on the admin dashboard page. The first section shows all the reviewer account requests along with important information. These requests can be approved or denied. The second section is for changing user roles. The admin can search for a user using the search bar, and the result will show their name, e-mail, and checkboxes for each role. If a checkbox is checked, the user has that role; if not, they don't. An admin can check or uncheck these boxes to change roles. The last section displays all ongoing reviews for all conferences, showing authors' and reviewers' names and e-mails. This helps the admin resolve any potential issues that might arise during the review process. The source code can be found here:
https://github.com/Mihail-Popescu/Sistem-pentru-gestiunea-articolelor-stiintifice

RESULTS

For testing purposes, two accounts were created for each role, including two accounts for the author role. On each author's account, three random scientific papers were uploaded on the internet, each with a considerable number of pages, words and characters. This allowed me to test the functionalities of the author dashboard more comprehensively.

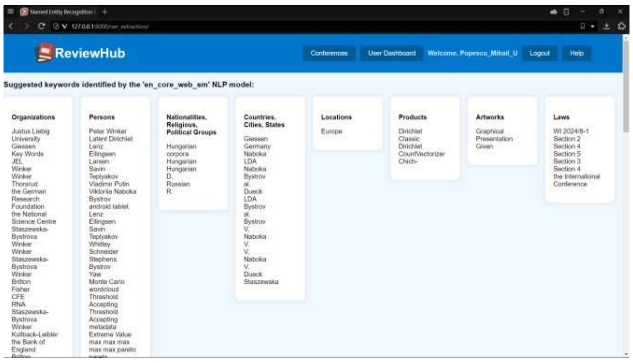
Figure 4 shows a test done for the spell-checking functionality used on the 'Cloud Computing' document. It illustrates two critical observations about the library's workings and my implementation.



Above the Clouds: A Berkeley View of Cloud Computing Michael Armbrust Armando Fox Rean Griffith Anthony D. Joseph Randy H. Katz Andrew Konwinski Gunho Lee David A. Patterson Ariel Rabkin Ion Stoica Matei Zaharia Electrical Engineering and Computer Sciences University of California at Berkeley Technical Report No. UCB/EECS-2009-28 http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html February 10, 2009 Copyright 2009, by the author(s). All rights reserved. Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission. Acknowledgement The RAD Lab's existence is due to the generous support of the founding members Google, Microsoft, and Sun Microsystems and of the affiliate members Amazon Web Services, Cisco Systems, Facebook, Hewlett- Packard, IBM, NEC, Network Appliance, Oracle, Siemens, and VMware; by matching funds from the State of California's MICRO program (grants 06- 152, 07-010, 06-148, 07-012, 06-146, 07-009, 06-147, 07-013, 06-149, 06- 150, and 07-008) and the University of California Industry/University Cooperative Research Program (UC Discovery) grant COM07-10240; and by the National Science Foundation (grant #CNS-0509559). Above the Clouds: A Berkeley View of Cloud Computing Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia (Comments should be addressed to abovetheclouds@cs.berkeley.edu) UC Berkeley Reliable Adaptive Distributed Systems Laboratory • http://radlab.cs.berkeley.edu/ February 10, 2009 KEYWORDS: Cloud Computing, Utility Computing, Internet Datacenters, Distributed System Economics 1 Executive Summary Cloud Computing, the long-held dream of computing as a utility, has the potential to transform a large part of the IT industry, making software even more attractive as a service and shaping the way IT hardware is designed and purchased. Developers with innovative ideas for new Internet services no longer require the large capital outlays in hardware to deploy their service or the human expense to operate it. They need not be concerned about over- provisioning for a service whose popularity does not meet their predictions, thus wasting costly resources, or under- provisioning for one that becomes wildly popular, thus missing potential customers and revenue. Moreover, companies with large batch-oriented tasks can get results as quickly as their programs can scale, since using 1000 servers for one hour costs no more than using one server for 1000 hours. This elasticity of resources, without paying a premium for large scale, is unprecedented in the history of IT. Cloud Computing refers to both the applications delivered as services over the Internet and the hardware and systems software in the datacenters that provide those services. The services themselves have long been referred to as Software as a Service (SaaS). The datacenter hardware and

Figure 4. Spell checking example

One observation is that some words are marked as wrong because they are combined with punctuation marks or are names that aren't in the library's dictionary. This leaves much room for improvement, which can be addressed by better extraction and separation of the words or by checking for false positives.



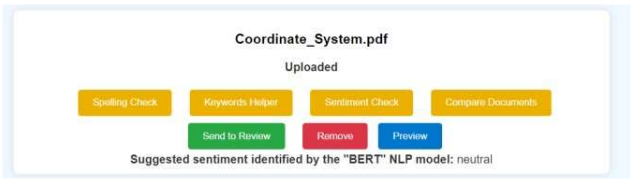
Suggested keywords identified by the 'en_core_web_sm' NLP model:

Organizations	Persons	Nationalities, Religious, Political Groups	Countries, Cities, States	Locations	Products	Artworks	Laws
Justus Liebig	Peter Winker	Hungarian	Giessen	Europe	Dirichlet	Graphical	WI 2024/8-1
University	Latent Dirichlet	corpora	Germany		Classic	Presentation	Section 2
Giessen	Lenz	Hungarian	Naboka		Dirichlet	Given	Section 4
Key Words	Ellingsen	Hungarian	LDA		CountVectorizer		Section 5
JEL	Larsen	D.	Naboka		Chich-		Section 3
Winker	Savin	Russian	Bystrov				Section 4
Winker	Teplyakov	R.	al.				the International
Thorsrud	Vladimir Putin		Dueck				Conference
the German	Viktoriia Naboka		LDA				
Research	Bystrov		Bystrov				
Foundation	android tablet		al.				
the National	Lenz		Bystrov				
Science Centre	Ellingsen		V.				
Staszewska-	Savin		Naboka				
Bystrova	Teplyakov		V.				
Winker	Whitley		V.				
Winker	Schneider		Naboka				
Staszewska-	Stephens		V.				
Bystrova	Bystrov		Dueck				
Winker	Yaw		Staszewska				
Britton	Monte Carlo						
Fisher	wordcloud						
CFE	Threshold						
RNA	Accepting						
Staszewska-	Threshold						
Bystrova	Accepting						
Winker	metadata						
Kulback-Leibler	Extreme Value						
the Bank of	max max max						
England	max max pareto						

Figure 5. Concept extraction example

The other observation is that the spell-checking library works as intended overall. Most words are correctly checked, and the inconsistencies mentioned earlier provide good examples of words being marked as wrong.

Figure 5 shows a test for the keywords finder functionality used in the 'Visualising topic uncertainty in topic modelling' document. Multiple observations can be made about the output given by the



Suggested sentiment identified by the "BERT" NLP model: neutral

model.

One is that there are both relevant and irrelevant results. Most irrelevant results are repetitions or simply words that don't fit their category. As was mentioned in more detail in the system design, there are multiple sizes of the 'en_core_web' model. The smallest version was used for testing purposes, while in a real-case scenario, the most extensive version can be used for the most accurate results.

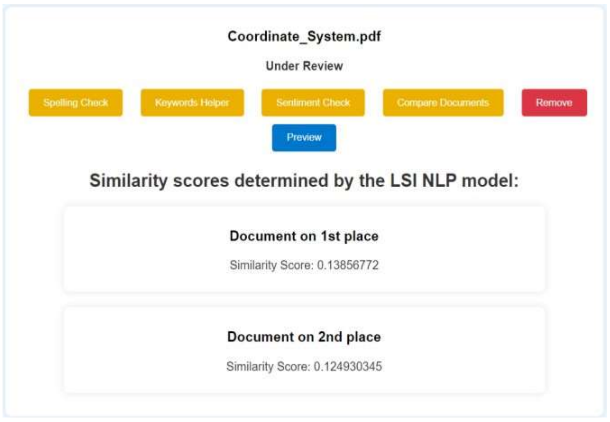
If wanted, duplications can be removed by filtering the model results. It can still be useful for some people to see how many times a keyword appears in their paper.

As for the relevant results, many fit their sections and could be useful in a real-case scenario for an author looking to find more or better keywords.

Figure 6 displays an example of a sentiment check test used on the 'Coordinate System' document, which came out as neutral. Every time the sentiment check button is pressed, the model processes the text and generates a new result. Assuming the scientific papers used for testing are neutral, the model's output should be neutral most of the time. For the 'Coordinate System' document, the model returned eight results as neutral and two as negative out of ten tests.

Figure 6. Sentiment analysis example

Figure 7 shows the result of the' compare documents' functionality used in the 'Coordinate System' document. The scores for the two most similar documents are displayed in descending order. Both the documents and their authors are kept anonymous.



Similarity scores determined by the LSI NLP model:

Document on 1st place
Similarity Score: 0.13856772

Document on 2nd place
Similarity Score: 0.124930345

Figure 7. Similarity score example

The similarity scores range from zero to one, with zero representing very low similarity and one representing very high similarity. All documents used for testing had their highest scores under 0.3, representing less than 30% similarity, in the low to moderate similarity range. Also, while testing, comparing the same document with itself gave a similarity score of 0.9 - 1.0. The preprocessing function can be fine-tuned to achieve even more accurate results.

CONCLUSION

The development and implementation of this web app have proven to be a valuable exercise in integrating public libraries that provide natural language models for document analysis. By creating a robust system that accommodates multiple user roles, we have enhanced the design and reliability of conference paper submissions and reviews.

The app's functionalities, such as spell-checking, sentiment analysis, entity extraction, and document comparison, provide authors with comprehensive tools to refine their submissions before review. The automated status tracking and detailed feedback system also streamline the review process, ensuring clear communication and constructive feedback between authors and reviewers.

Future improvements could focus on refining the models' preprocessing functions and expanding the range of functionalities where those models are used. Moreover, the app's design allows for the potential integration of new features and enhancements based on user feedback and technological advancements.

Overall, this project underscores the potential of combining modern web development frameworks with advanced natural language processing libraries to create efficient and user-friendly solutions for academic and professional environments.

.

REFERENCES

1. Goghate, A., Yerlekar, A., Turkar, H., Nanotkar, A., Dhok, A., & Sakhare, N. (2024). Easy Chair–Research Paper process Handling in Salesforce. Educational Administration: Theory and Practice, 30(4), 5674-5679.

2. Hasan, L. R., & Abuelrub, E. (2013). Usability Testing for IAJIT OpenConf Journal Management System. J. Softw., 8(2), 387-396.

3. Yadav, Y., & DESAI, D. B. C. (2023, May). ConfSys-An Intelligent Conference Management System. In Proceedings of the 27th International Database Engineered Applications Symposium (pp. 127-130).

4. Yadav, Y. O. (2024). ConfSys 4: An Advanced Conference Management System with Automatic Semantic Header Generation (Doctoral dissertation, Concordia University).

5. Ishak, W. H. W., Yamin, F. M., Mohsin, M. F. M., & Mansor, M. F. (2023). A Comparative Review of Conference Management System. Journal of Technology and Operations Management, 18(2), 87-93

6. Ahmad, K., Abdullah, A. A., & Zeki, A. M. (2012, November). Web-based conference management system for higher learning institutions. In 2012 International Conference on Advanced Computer Science Applications and Technologies (ACSAT) (pp. 340-343). IEEE.

7. Bioco, J., & Rocha, A. (2019). Web application for management of scientific conferences. In New Knowledge in Information Systems and Technologies: Volume 1 (pp. 765-774). Springer International Publishing..

8. Kalmukov, Y. (2011). Architecture of a conference management system providing advanced paper assignment features. arXiv preprint arXiv:1111.6934..

9. Pradhan, D. K., Chakraborty, J., Choudhary, P., & Nandi, S. (2020). An automated conflict of interest based greedy approach for conference paper assignment system. Journal of Informetrics, 14(2), 101022.

10. Santiputri, M., Agustin, N. S., & Delimayanti, M. K. (2018, October). MyConfree: a web-based conference management system. In 2018 International Conference on Applied Engineering (ICAE) (pp. 1-4). IEEE

11. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

12. Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. Journal of machine Learning research, 3(Jan), 993-1022