LECTURE4ALL: A Lightweight Approach to Precise Timestamp Detection in Online Lecture Videos

Viktoria Wrobel, Simon Kazemi, Frank Hammerschmidt, Torben Hannemann, Gregor Stange, Robert Geislinger and **Seid Muhie Yimam**

University of Hamburg
Germany

Correspondence: viktoria.wrobel@studium.uni-hamburg.de

Abstract

This paper presents **LECTURE4ALL**[1], a web application developed to improve the search experience of educational video platforms. Lecture2Go provides a vast collection of recorded lectures, but locating specific content within videos can be time-consuming. **LECTURE4ALL** addresses this issue by leveraging a vector database and a streamlined user interface to enable direct retrieval of precise video timestamps. By enhancing search accuracy and efficiency, **LECTURE4ALL** significantly improves the accessibility and usability of educational video platforms.

1 Introduction

Educational video platforms and courses are part of modern education. Finding specific explanations in university lecture recordings can be time-consuming. Lecture2Go (Kriszat et al., 2010) is an open-source video platform hosting recorded lectures and seminars held at the University of Hamburg. It provides round-the-clock access to academic content. The platform has gained popularity in recent years, particularly during the Covid-19 pandemic, when online education became a necessity (Zawacki-Richter, 2021).

Despite its benefits, Lecture2Go's current search functionality is limited to lecture titles and descriptions, making it difficult to locate specific information within videos. Students must manually navigate through lengthy recordings to find relevant content, which can be time-consuming and inefficient. Without knowledge of the video title or author, searching for video content is not possible. In addition, this process poses accessibility challenges for users with visual impairments, as they must rely on screen readers or external transcription tools to search within videos.

LECTURE4ALL was developed as an open-source, ready to use software solution to address these limitations by providing a more efficient and accessible way to retrieve information from lecture recordings. It is both lightweight and does not require additional storage of video content, instead making use of existing infrastructure. The system introduces several key innovations:

- **Voice-controlled search**: Users can perform searches using voice input, enhancing accessibility and ease of use.
- **Semantic search with vector databases**: Instead of relying on keyword matches in titles or descriptions, **LECTURE4ALL** uses a vector-based retrieval system to find precise timestamps where a query is addressed in the transcript. Searching for concepts contained in the video content is possible without knowledge of title or author.
- **User-friendly interface**: The system supports both voice and text-based input, improving usability across different devices, including mobile platforms. Moreover, searching is multilingual and knowledge of the video language is not necessary to find relevant timestamps.
- **Shorts feature**: Aligns with current trends and allows users to swiftly browse 10-second snippets of the most relevant query results. It provides immediate access to the results with the option to view the full video. Shorts are automatically extracted from the external video source.

LECTURE4ALL integrates state-of-the-art AI models, such as OpenAI's Whisper (Radford et al., 2023), with modern web technologies. Its modular architecture separates the backend, API, and

[1] Demo: `https://lecture4all.demo.hcds.uni-hamburg.de` Source Code: `https://github.com/uhh-hcds/lecture4all`

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations), pages 614–620
July 27 - August 1, 2025 ©2025 Association for Computational Linguistics

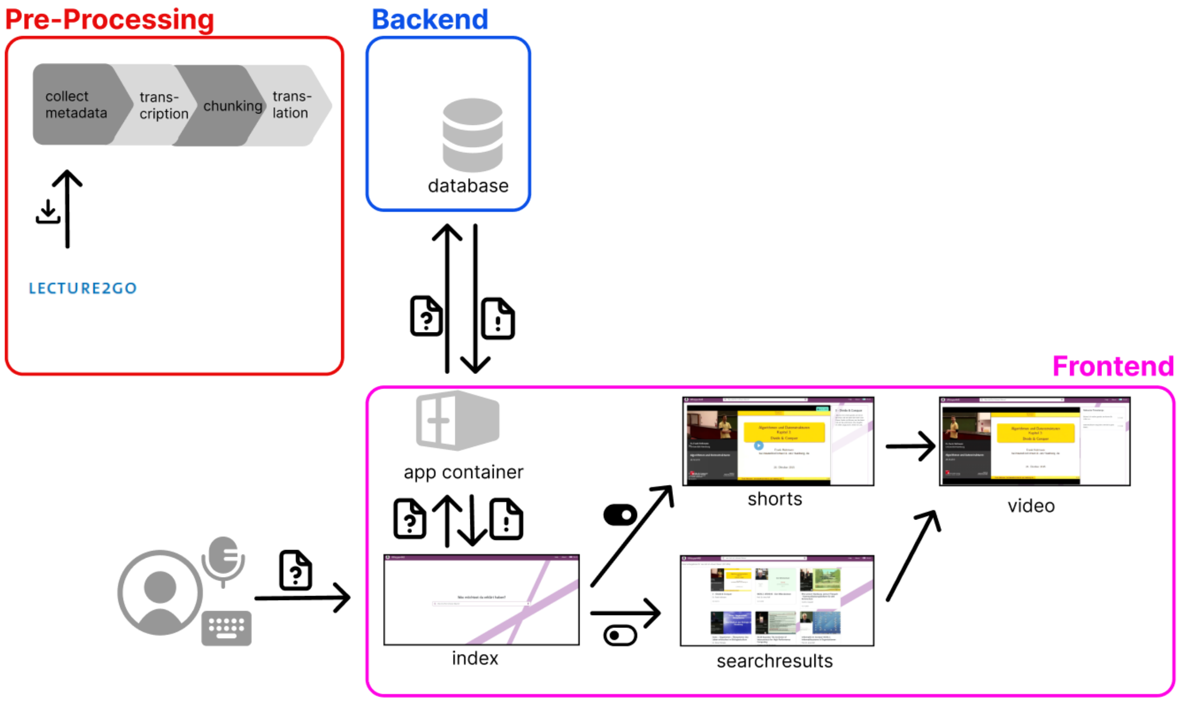


Figure 1: System pipeline. User input (bottom left) creates a query which is sent between Docker containers via Flask. ChromaDB outputs relevant data, which is then sent back to the frontend and displayed according to user's preferences. Top left shows processing pipeline of video data which is executed beforehand.

frontend components, allowing easy adaptation for other video platforms, beyond Lecture2Go. Additionally, the system enhances search accuracy, reduces the time spent locating relevant lecture segments, and promotes inclusivity in digital learning. As an open-source project, it encourages further research and development in educational technology.

To assess its effectiveness, **LECTURE4ALL** was evaluated through a user study, measuring usability as well as search efficiency improvements. The results provide insights into how AI-driven search tools can enhance the user experience of lecture platforms like Lecture2Go.

2 System Architecture

LECTURE4ALL essentially consists of three main parts: Preprocessing pipeline, backend, and frontend. Each part is briefly described below. **LECTURE4ALL** is highly modularized and all components can easily be modified or swapped out for alternatives. A detailed overview can be found in Figure 1.

2.1 Backend

The backend of this system consists of a backend Flask server that is constantly running to process search queries made by the user and retrieve data from the database, and a video transcription pipeline to fill the database. The transcription pipeline consists of multiple steps and generates all necessary files to fill the database using OpenAI's Whisper for transcription of videos while translation of videos to obtain subtitles is done using the MarianMT model (Tiedemann et al., 2023), both of these will run much faster on GPU, so it is recommended to have multiple GPUs available. The database is a ChromaDB[2] vector database that runs on another separate docker container. A vector database allows for context dependent search and does not rely on just metadata and keywords (Taipalus, 2024). The vector database embeddings are generated using the USE (Universal Sentence Encoder) multilingual model (Cer et al., 2018).

While Lecture2Go architecure is utilized to supply the videos, our solution enhances them through the database and new frontend, keeping the system lightweight and making use of existing infrastructure.

[2] https://github.com/chroma-core/chroma

2.2 API

Communication between the frontend and backend is facilitated through Flask. The frontend interacts with the backend by requesting a JSON response that includes relevant text segments, video URLs, and associated metadata. The decision to implement a RESTful interface was made to ensure ease of maintenance and scalability, allowing for seamless integration and expansion of the system during future development. Communication between database and backend is simply done via ChromaDB's built-in HTTP API.

2.3 Frontend

The user interface of LECTURE4ALL consists of four main views:

- **Landing Page**: As can be seen in Figure 2, it is a minimalist interface featuring a search bar with voice recognition and a navigation bar, including a 'Shorts' toggle. Users can type or enter queries via voice input provided by the browser to retrieve results.

- **Shorts View**: If enabled through the shorts toggle which can be seen in 3 (1), this view displays a large video preview panel that plays short, auto-extracted clips matching the input query, allowing users to quickly scan and identify relevant content (Violot et al., 2024). A button allows the user to navigate to the corresponding full-length lecture, as can be seen in Figure 3.

- **Search Results View**: If Shorts is disabled, users see a list of relevant lecture videos ranked by relevance. Each result includes metadata such as lecture title, duration, and key timestamps.

- **Video View**: Selecting a lecture opens a detailed video player with highlighted timestamps and an interactive sideboard for rapid navigation. The videos are delivered by the existing backend for a fast response even with higher load.

The Shorts Toggle was implemented to allow users unfamiliar with this format to rely on a more traditional video browsing layout. The front-end is implemented using **Bootstrap (CSS framework), HTML5** and **JavaScript**, ensuring a responsive design for both desktop and especially mobile users. **Flask** handles routing between pages. A persistent navigation bar provides quick access to the Help and About pages for user guidance.

3 Integration and Containerization

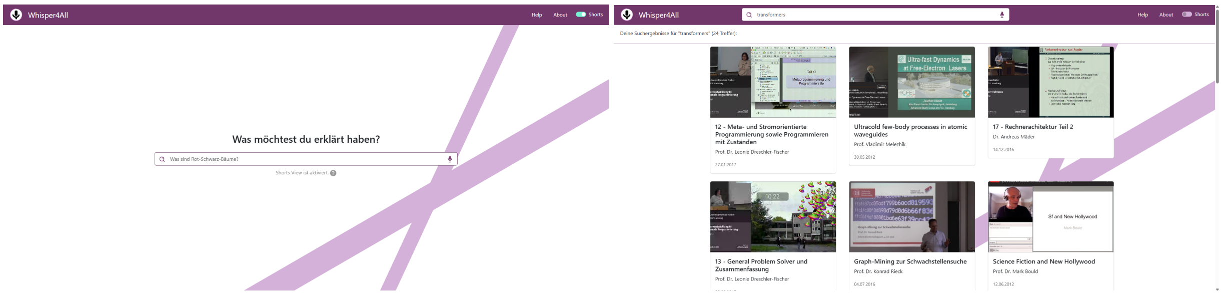
The running system is based on three docker containers, which are connected via a docker network and also have ports for access outside of the net. For the vector database container the chromaDB docker image from the docker hub is used. The other two containers are built by self-created docker images. The database container hosts the database environment, called db-env, which handles search requests passed by the frontend. The app container, called l4a-app, is part of the frontend, forwarding the input and visualizing the output. All three containers can be started via the docker compose file. We chose containerization to facilitate deployment of the running system. Aside from running the docker compose commands, the only required configuration is entering a database path in the environment file.

4 Data Processing Pipeline and Vector Database

The following sections outline how we transcribe video content and prepare it for semantic search. We first describe the transcription pipeline based on Whisper, followed by an explanation of how the processed data is embedded and stored in a vector database for efficient retrieval.

4.1 Transcription

In this project, we developed a comprehensive suite of Python scripts designed to facilitate the transcription of video content using OpenAI's Whisper model. Each script takes in a specific ID range (the identifier Lecture2Go uses for their videos) and performs its designated task on all videos within that range. The workflow begins with the download script by iterating through its specified range of video IDs and utilizing the yt-dlp (GitHub Contributors, 2025) package to download **m3u8-playlists** that are then assembled to mp4 files. The M3U files are encoded with UTF-8, which allows for better handling of special characters. Once the videos are stored, Whisper timestamped (Louradour, 2023; Giorgino, 2009; Radford et al., 2023) is employed to obtain JSON transcripts with timestamps for each single word, so that we can do our own chunking, for ideal



Whisper4All
Help
About
Shorts
Was möchtest du erklärt haben?
Was sind Rot-Schwarz-Bäume?
Shorts View ist aktiviert.
transformers

Deine Suchergebnisse für "transformers" (24 Treffer):

12 - Meta- und Stromorientierte Programmierung sowie Programmieren mit Zuständen
Prof. Dr. Leonie Dreschler-Fischer
27.01.2017

Ultracold few-body processes in atomic waveguides
Prof. Vladimir Melezhik
30.05.2012

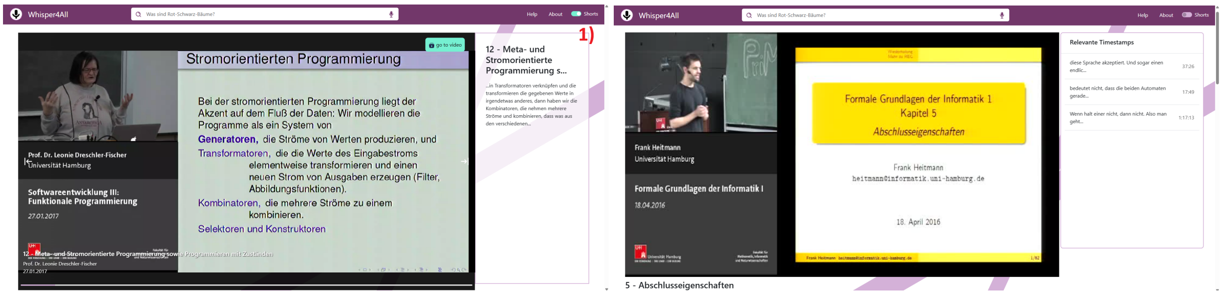
17 - Rechnerachitektur Teil 2
Dr. Andreas Mäder
14.12.2016

13 - General Problem Solver und Zusammenfassung
Prof. Dr. Leonie Dreschler-Fischer

Graph-Mining zur Schwachstellensuche
Prof. Dr. Konrad Rieck
04.07.2016

Science Fiction and New Hollywood
Prof. Dr. Mark Bould
12.06.2012

Figure 2: Main Page and search result overview



Stromorientierten Programmierung

Bei der stromorientierten Programmierung liegt der Akzent auf dem Fluß der Daten: Wir modellieren die Programme als ein System von
Generatoren, die Ströme von Werten produzieren, und
Transformatoren, die die Werte des Eingabestroms elementweise transformieren und einen neuen Strom von Ausgaben erzeugen (Filter, Abbildungsfunktionen).
Kombinatoren, die mehrere Ströme zu einem kombinieren.
Selektoren und Konstruktoren

Prof. Dr. Leonie Dreschler-Fischer
Universität Hamburg
Softwareentwicklung III: Funktionale Programmierung
27.01.2017

12 - Meta- und Stromorientierte Programmierung sowie Programmieren mit Zuständen
Prof. Dr. Leonie Dreschler-Fischer
27.01.2017

12 - Meta- und Stromorientierte Programmierung s...

...in Transformatoren verknüpfen und die transformieren die gegebenen Werte in irgendetwas anderes, dann haben wir die Kombinatoren, die nehmen mehrere Ströme und kombinieren, dass was aus den verschiedenen...

Frank Heitmann
Universität Hamburg
Formale Grundlagen der Informatik I
18.04.2016

Formale Grundlagen der Informatik 1
Kapitel 5
Abschlusseigenschaften

Frank Heitmann
heitmann@informatik.uni-hamburg.de

18. April 2016

Relevante Timestamps

diese Sprache akzeptiert. Und sogar einen endlic...	37:26
bedeutet nicht, dass die beiden Automaten gerade...	17:49
Wenn halt einer nicht, dann nicht. Also man geht...	1:17:13

5 - Abschlusseigenschaften

Figure 3: Shorts and regular Video View

chunk size with regards to the vector database and user experience. For this step explicitly it is highly recommended to use GPU support. To obtain the best results We recommend using the largest recent Whisper model Whisper Large v3 which requires around 7.5 GB VRAM to run properly. The metadata of the videos is then retrieved from Lecture2Go and assembled into a specifically structured JSON file per video. Utilizing the word timestamps, another processing script brings the raw Whisper output and metadata JSON files into the JSON structure we later use to feed into the database. Our custom chunks have a length of at least 12 seconds to make sure enough context is captured for proper search results in the database and also such that the video chunks are a good length for the shorts feature. In the future dynamic chunking based on semantic boundary detection or pause detection could help capture context in chunks and lead to better search results. Though determining the ideal chunk size with regards to user experience will require further testing and user feedback. The outputs are then organized into a dedicated directory for streamlined access and integration into the database. Finally, at the end of the data processing pipeline, a script for subtitle generation is run. Using the MarianMT model, it goes through the transcripts output by Whisper and generates .srt files in English for all German and English videos.

4.2 Vector Database

We use a vector database to perform a semantic search on the transcribed videos with the help of embeddings. ChromaDB is easy to use and has already implemented many functions. It also offers the possibility to easily exchange the embedding model for the vectors, which gave us the opportunity to try different models and makes it easy to customize the system in the future. After an evaluation of different embedding models, we decided to use USE (Universal Sentence Encoder) multilingual version 3 (Cer et al., 2018). The big advantage of this model is that, unlike many other models that were either trained mostly to find words or sentences of similar context, this model is suited for both sentence-type queries and word-type queries. This allows for consistently good results, allowing for flexible querying. The multilingual version additionally offers support for multiple languages, allowing for search queries in English to find suitable German videos and vice versa. Further testing is required to determine the quality of search results for highly specific topics, which in the context of university lectures might very well be relevant. Knowing how to query the database and future features allowing for keyword search or other methods of filtering could also help in finding suitable videos. This helps break the language barrier and makes lectures videos in different languages searchable by everyone. The choice of this model is crucial for the user experience, because it determines the quality of the search results. The processed transcripts are loaded into the database via a Python script, which runs through the directory with these transcripts. For each chunk in a video file, a new database entry with metadata is created. The chunks are stored independently of the video, so it is important that the metadata include the video ID and timestamps to provide a reference to the origin of the chunk. Every chunk entry contains the references to the video, the chunk text and its embedding and remaining important information for the frontend presentation, like the video link or the title of the video. End-to-end latency measured in a browser, querying a 1.7 gigabyte database (corresponding to roughly 2000 videos) ranges from 300 to 400 milliseconds and code-measured time for querying the database resulted in around 52 milliseconds.

5 Evaluation

The evaluation was conducted using a survey that included the NASA-TLX and the System Usability Scale (SUS), both standardized assessment tools for workload and usability, respectively. A total of 43 participants were divided into three conditions and assigned one of two possible tasks from different domains of education. The conditions were: (1) Lecture4All with regular search results only, (2) Lecture4All with 'Shorts' enabled, and (3) Lecture2Go (control group). The participants consisted of students from the University of Hamburg between the ages of 18 and 44 with a majority identifying as male (59%) or female (38%), and one non-binary respondent.

The survey results demonstrate that Lecture4All outperformed the traditional Lecture2Go system in key usability and efficiency metrics. The comparative analysis revealed a clear performance gap between the systems. While Lecture4All enabled 60% of users to successfully complete their tasks, Lecture2Go users showed significantly higher failure rates, with only 40% finding satisfactory answers. This 20-percentage-point performance difference was further exacerbated by qualitative findings - even successful Lecture2Go responses often required substantially more effort (typically 4+ searches vs. Lecture4All's 1-2 searches) and longer completion times. Successful Lecture4All users typically located answers within 1–2 searches, whereas Lecture2Go often required 4 or more attempts, reflecting its less intuitive search functionality.

Condition	Mean SUS	SD
Condition 1	67.9	16.0
Condition 2	81.2	9.88
Condition 3	49.4	15.6

Table 1: SUS scores and standard deviation

A descriptive evaluation of the SUS-scores also showed clear differences between the three conditions. Condition 2 reached the highest mean SUS-score (Mean = 81.2, Standard Deviation = 9.88), followed by condition 1 (M = 67.9, SD = 16.0). Condition 3 achieved the lowest SUS-score (M = 49.4, SD = 15.6). A one-way ANOVA confirmed significant group differences (F(2, 40) = 16.14, p < .001, η^2 = 0.45 [large effect]). Post-hoc tests (Tukey HSD) revealed the following: Condition 2 performed significantly better than condition 1 (p = .043, d = 0.99). Condition 3 scored significantly lower than Condition 1 (p = .003, d = 1.25). The difference between condition 3 and condition 2 was most pronounced (p < .001, d = 2.31). The residuals were normally distributed (Shapiro-Wilk-Test: W = 0.99, p = .963).

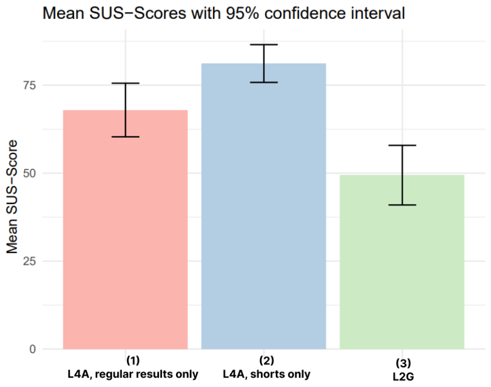


Figure 4: Distribution of SUS-scores (0–100) by experimental condition.

NASA-TLX scores were rated on a 5-point likert scale to improve design and user-friendliness of the study. The raw data was then linearly transformed to a 0-100 scale, higher values indicating higher workload. An analysis of the scores showed distinct differences between the three conditions.

Condition 1 induced a moderate to small workload (M = 34.6, SD = 18.9), with condition 2 showing the smallest workload (M = 18.3, SD = 16.6) and condition 3 showing the highest workload (M = 47.4, SD = 19.7). A one way ANOVA confirmed group differences (F(2, 40) = 8.13, p = .001, η^2 = 0.29). Post-hoc-tests (Tukey HSD) revealed a significant difference between condition 3 and condition 2 (p = .001).

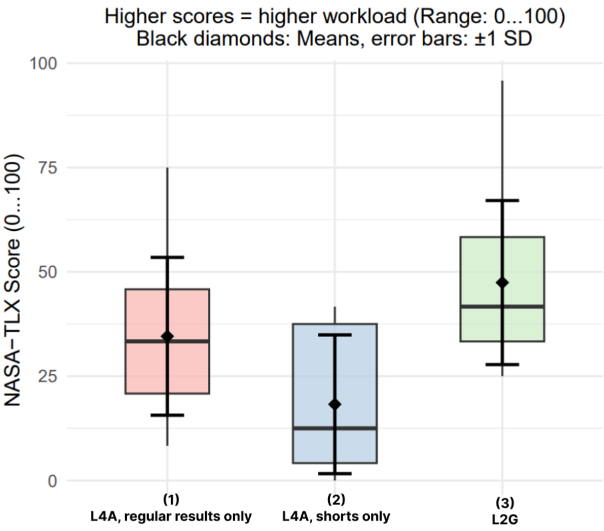


Figure 5: Distribution of NASA-TLX scores (0 - 100) by experimental condition

User feedback further underscored Lecture4All's advantages. Participants praised its speed and ease of use, particularly when answers were readily accessible. In contrast, Lecture2Go was frequently criticized for unclear video titles, inefficient navigation, and the difficulty of pinpointing relevant content in lengthy lectures. These observations were supported by NASA-TLX workload scores, which indicated higher mental demand and frustration with Lecture2Go. System Usability Scale (SUS) ratings also favored Lecture4All, with users rating it as more intuitive and less cumbersome.

Findings suggest that Lecture4All offers a more effective solution for retrieving lecture-based information. Its streamlined interface and faster response times align better with user expectations, positioning it as a superior alternative to traditional lecture platforms.

6 Related Work

Video search functionality has been explored in previous works, with YouTube being a widely used but closed-source platform that lacks precise timestamp detection. The user is provided with auto-generated chapters and data for popular sections, but no precise content-focussed timestamps are provided. Its search accuracy is further constrained by reliance on automatically generated captions. Other tools, such as TalkMiner (Adcock et al., 2010), have been discontinued. TalkMiner focusses on slide content only, thereby omitting any spoken information in lectures.

CONQUER (Hou et al., 2021) is another system designed to retrieve and rank audio content from videos. Compared to Lecture4All it does not include a frontend and was not designed with educational content in mind. In contrast, Lecture4All is fully open-source and provides precise timestamp retrieval. The role and impact of artificial intelligence in education have been highlighted Holmes et al. (2019).

Conclusion

Lecture4All presented lightweight solution with modern machine learning models to improve the user experience within Lecture2Go, offering enhanced accessibility and precision in video retrieval. The approach described in this paper is not only applicable to Lecture2Go but can also be extended to other video platforms, facilitating easier access to educational content through voice input and cross-language capabilities without costs for licensing.

Looking ahead, we aim to integrate metadata into the search process and incorporate image recognition to enable searches for slide content displayed within the videos, similar to TalkMiner. The modular architecture of Lecture4All allows for usage in non-education settings, too. Existing video databases which feature long-form videos based around spoken language might benefit from incorporating a timestamp search feature. While the current implementation demonstrates a strong user experience, a more comprehensive evaluation of the search results would enable further refinement and optimization of the database model for better performance. The University of Hamburg has shown interest in officially integrating Lecture4All into Lecture2Go.

Acknowledgments

We appreciate the University of Hamburg for supporting this work and the participants who contributed to our user study. We also thank the hub of Computing & Data Science (HCDS) at the University of Hamburg for hosting our demo and for the GPU servers to run our computation. AI assistance was used for grammer checks, as well as for debugging of the LECTURE4ALL code.

7 Limitations

Due to a small sample size ranging from 13 to 17 participants per condition, generalizability of our findings might be limited. Additionally, the reliance on subjective self-report measures (NASA-TLX, SUS) might introduce potential biases. We were able to demonstrate longer survey completion times for Lecture2Go users but this included the survey itself. Incorporating quantifiable behavioral data - such as completion time of the task itself, could provide a more comprehensive assessment of LECTURE4ALL's efficiency. While the user study showed improvements in subjective workload (NASA-TLX) and usability (SUS), these metrics do not directly measure retrieval effectiveness. To better assess retrieval precision, we plan to incorporate standard IR metrics for timestamped video segments. This will help isolate the contribution of the underlying vector-based retrieval from UI-related improvements. Furthermore, speech recognition is only supported by Chromium-based browsers. Lastly, the measured system latency and querying time demonstrate the success of our lightweight approach with regards to performance.

References

John Adcock, Matthew Cooper, Laurent Denoue, Hamed Pirsiavash, and Lawrence A. Rowe. 2010. Talkminer: a search engine for online lecture video. In *Proceedings of the 18th ACM International Conference on Multimedia*, page 1507–1508, New York, NY, USA. Association for Computing Machinery.

Daniel Cer, Yinfei Yang, Sheng yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder. *Preprint*, arXiv:1803.11175.

Toni Giorgino. 2009. Computing and visualizing dynamic time warping alignments in r: The dtw package. *Journal of Statistical Software*, 31(7):1–24.

GitHub Contributors. 2025. yt-dlp. `https://github.com/yt-dlp/yt-dlp`. Version 2025.03.29.

Wayne Holmes, Maya Bialik, and Charles Fadel. 2019. *Artificial Intelligence in Education: Promises and Implications for Teaching and Learning*. Center for Curriculum Redesign, Boston, MA.

Zhijian Hou, Chong-Wah Ngo, and W. K. Chan. 2021. Conquer: Contextual query-aware ranking for video corpus moment retrieval. In *Proceedings of the 29th ACM International Conference on Multimedia*, page 3900–3908, New York, NY, USA. Association for Computing Machinery.

Martin Kriszat, Iavor Sturm, and Jan Torge Claussen. 2010. Lecture2Go – von der Vorlesungsaufzeichnung ins World Wide Web. *Digitale Medien für Lehre und Forschung*, pages 25–38.

Jérôme Louradour. 2023. whisper-timestamped. `https://github.com/linto-ai/whisper-timestamped`.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2023. Robust speech recognition via large-scale weak supervision. In *Proceedings of the 40th International Conference on Machine Learning*, pages 28492–28518, Honolulu, HI, USA.

Toni Taipalus. 2024. Vector database management systems: Fundamental concepts, use-cases, and current challenges. *Cognitive Systems Research*, 85:1–8.

Jörg Tiedemann, Mikko Aulamo, Daria Bakshandaeva, Michele Boggia, Stig-Arne Grönroos, Tommi Nieminen, Alessandro Raganato Yves Scherrer, Raul Vazquez, and Sami Virpioja. 2023. Democratizing neural machine translation with OPUS-MT. *Language Resources and Evaluation*, (58):713–755.

Caroline Violot, Tuğrulcan Elmas, Igor Bilogrevic, and Mathias Humbert. 2024. Shorts vs. regular videos on youtube: A comparative analysis of user engagement and content creation trends. In *Proceedings of the ACM Web Science Conference (WEBSCI '24)*, Stuttgart, Germany.

Olaf Zawacki-Richter. 2021. The current state and impact of covid-19 on digital higher education in germany. *Human Behavior and Emerging Technologies*, 3(1):218–226.