Doctor of Engineering in Data Science

Design and Development of a Lyrics-based Setlist Identification System for Live Performances

Author:
Philipp Stürmer

Prepared for:
Copyright Delta

Date:
2023-06-10

EngD report information

Eindhoven University of Technology
Jheronimus Academy of Data Science 2023
This report is part of the Engineering Doctorate (EngD) Program:
Data Science
Confidentiality status: permanently confidential
EngD-Report Number: 2023/033

Acknowledgement of Code of Scientific Conduct

The design described in this thesis has been carried out in accordance with the TU/e code of Scientific Conduct

**) ISBN information only requested for the report of the final project*

Executive Summary

The music industry's transition from physical sales to the digital realm challenges current royalty infrastructures with a complex transaction chain and insufficient metadata, leading to diminished royalty transparency and unfair payouts. This company targets these issues, ensuring fair royalty distribution for artists and songwriters by harnessing blockchain's transparency. The key project goal is to augment this initiative by crafting a swift, cost-effective Lyrics-based Setlist Identification System (LyricSIS) that accurately identifies songs, artists, and songwriters in live performances.

To accomplish this, I developed a system consisting of two main components: the Automatic Lyric Transcription (ALT) module, transcribing lyrics from raw audio, and a song search module associating songs from the transcribed lyrics with their respective ISWCs, identifying artists and songwriters. The system employs Genius.com's lyrics and MusicBrainz's ISWC databases for metadata retrieval.

The system underwent evaluation, focusing on parameter optimization to maximize correctly identified songs (True Positive Percentage or TPP) and minimize incorrectly identified songs (False Positive Percentage or FPP) and runtime. The system's ALT module outperformed contemporary research, and LyriSIS recorded a TPP of 45.63% and an FPP of 14.46%. Despite superior setlist identification accuracy, it was slower than Shazam, and its performance varied across genres and audio quality levels.

The project proved it's possible to identify setlists of live performances and link them with ISWCs based on lyrics. However, a TPP score below 50%, an FPP of 14.46%, and dependency on audio quality and genre restrict the system's standalone utility for automated setlist identification. Nonetheless, the ALT module's performance suggests potential standalone applications for lyric transcription.

To enhance the ALT module and overcome setlist identification limitations, I propose fine-tuning the Whisper transcription model with singing data and integrating audio fingerprints. Direct lyrics database access and a refined search algorithm could speed up the song search.

Considering the current limitations, I recommend a risk analysis prior to system deployment for setlist identification. The primary focus should be system accuracy enhancement, potentially through the suggested fine-tuning, potentially providing resilience to genre and audio quality variations and preparing the system for commercial applications by enhancing accuracy.

Contents

List of Figures

List of Tables

1 Management Introduction

This chapter presents a comprehensive overview of the project, pinpointing the key aspects detailed in this report. It encompasses several sections, including the project context with the motivation and goals, system design, evaluation encompassing methodology and results, conclusion, and recommendations.

The section on project context clarifies how the project began and the problem it is aiming to solve. The system design part elaborates on the functional and non-functional requirements of the project and the rationale behind the system's design. It includes an implementation section, which details how I executed the project at the programming level. The evaluation section describes the project's success measurement metrics and presents the results from the development and test dataset. The conclusion part encapsulates the evaluation findings, explaining the project limitations, and reflects on their implications on the project's overall success. Lastly, the future work and recommendations section provides insightful suggestions for enhancing the project and guiding the company on utilizing the developed system. It helps ensure the system's continued relevance and potential for growth.

1.1 Project Context

The music industry today is in the throes of significant disruption as a result of rapid technological advancement. Age-old copyright laws, fragmented copyrights, the widespread presence of inaccurate metadata, and a lack of transparency in the chain of royalty distribution are but a few of the many challenges that it faces. The digital transformation has fundamentally altered the landscape of music creation, distribution, and consumption, necessitating new infrastructure models to ensure fair royalty distribution [1].

Accurate metadata of songs is crucial for tracking and allocating royalties, especially in the digital era. Each song has two unique identifiers, the International Standard Recording Code (ISRC) for sound recording and the International Standard Musical Work Code (ISWC) for the underlying work. However, metadata often remains incomplete or incorrect, leading to misallocated or unpaid royalties. Furthermore, song monetization can occur without ISRC or ISWC metadata, particularly on user-generated content platforms like YouTube, which rely on audio fingerprint-based copyright infringement detection systems. This is a significant concern, as variations in song versions can evade detection, leading to missed royalties for artists and songwriters.

Against this background, the company's primary goal is to enhance transparency, fairness, and ethical conduct within the music industry's royalty chain through the Digital Rights eXchange (DRX) network using blockchain technology. DRX is the company's main product and is a decentralized, tamper-proof, data sovereign media and metadata storage network. Each authorised party in the music industry can insert a sound recording and its entire metadata into DRX's blockchain network. This innovation will ensure accurate and prompt payments, and blockchain's inherent transparency eliminates the need for intermediaries, thereby reducing disputes.

The project at hand arises from the vision to innovate automatic song identifications for live performances, which extends to full concert setlist creation. If successful, songs in setlists can be linked to their respective ISWC codes, representing songwriting copyright holders. The primary goal of the project is, therefore, to design and develop a Lyrics-based Setlist Identification System (LyricSIS) for the automatic generation of setlists from raw concert audio files. This approach will ensure fair royalty distributions and recover lost royalties, perfectly aligning with the company's secondary objective: guaranteeing rightful royalty payouts to songwriters for live performances and User Generated Content (UGC). This is accomplished by identifying the underlying works and their copyright holders, the songwriters.

The project's strategy will involve a comprehensive literature review on the current methods for lyrics-based song and setlist identification, the design and implementation of the system based on the literature review and system requirements, and an in-depth system evaluation to assess its accuracy, speed, and requirement meeting.

There are several potential use cases for this system across various markets in the music industry. Live music venues and event organizers could use it for automatic setlist creation from live performance records, which would result in a fairer distribution of royalties, as the manual setlist creation often is inaccurate, incomplete or entirely missing, leading to missed out royalties. Social media platforms could benefit from enhanced copyright detection algorithms by identifying the live versions of songs as well as by an enhanced user experience through displaying the setlists linked to the performances. Moreover, streaming platforms, social media platforms, and lyrics database hosts could use the system's Automatic Lyric Transcription (ALT) module for independent provision of lyrics, thereby enhancing the user experience or expanding their lyric databases. Similarly, collection societies could refine their song metadata databases and resolve inaccuracies in ISWC assignments through lyric comparisons, ultimately ensuring fairer royalty distributions.

In conclusion, the project is set to enhance several markets in the music industry, addressing song usage and copyright owner identification challenges. It aligns perfectly with the company's goals, and its successful implementation will thereby contribute significantly towards making the music industry more equitable and transparent.

1.2 System Design

This project offers an innovative lyrics-based solution to identify setlists of live performances and link the songs to their underlying work by identifying the ISWC. It takes raw audio from a live performance and automatically transcribes it into AI Generated Lyrics (AIGL). The system then uses these AIGL to identify the songs, fetch metadata such as artist name, song name, and the ISWC from a lyrics and a ISWC database, and perform song segmentation for fast performance. Additionally, it creates setlists based on the identified songs and expresses a level of certainty for each song identification.

The design of this system is comprehensive yet straight forward, including sequential components, focusing on various aspects such as robustness, high accuracy, cost-effectiveness, and speedy performance. In terms of accuracy, the system emphasizes on maximizing TPP and minimizing FPP. State-of-the-art technology is utilized for the automatic lyric transcription to meet the requirements, while enterprise-level robust methods handle audio pre-processing to ensure a signal with highest possible audio quality.

The Automatic Lyric Transcription (ALT) module and its component design encompasses key segments: Music source separation, speech enhancement, voice activity detection, and speech-to-text transcription, to extract the vocals, reduce noise, cut out silent parts, and transcribe the lyrics, respectively. Each component plays therefore a vital role in the overall process, enabling the transcription of pre-processed vocals into highly accurate AI Generated Lyrics (AIGL).

The next component is the song search module, which uses these AIGL to match them against Genius.com's extensive lyrics database and the MusicBrainz ISWC database, facilitating song and eventually setlist identification including all necessary metadata.

The implementation of the project is based on Python 3.8. The modular design allows for flexible utilization of the system, enabling setlist identification or focusing on specific tasks such as audio processing or transcription as per requirements.

The project employs a range of datasets to conduct thorough testing and benchmarking. Two datasets are specifically used to benchmark the accuracy of the Automatic Lyric Transcription module against state-of-the-art systems. Additionally, I have a development dataset that encompasses diverse concerts and a comprehensive test dataset for evaluating the system for the setlist identification task. I have also incorporated a Shazam Analysis feature to compare the performance of LyricSIS with a well-established commercial song identification system, Shazam.

In conclusion, the system is designed to offers a cost-effective, highly accurate, and robust solution for setlist identification that comes with an advanced ALT module with audio pre-processing opportunities, making it adaptable for a broad spectrum of use-cases.

1.3 Evaluation

The evaluation of the system was split into two key sections: The ALT module responsible for the transcription of lyrics based on the raw audio input, and the overall system's accuracy and speed in identifying the setlists.

For the ALT module, I used a measurement called Word Error Rate (WER), which is a common way to measure the accuracy of transcription systems. When evaluating the entire system, I used metrics to calculate the percentage of correctly identified songs, TPP, and incorrectly identified songs, FPP. I also considered the speed of the system by measuring each step's runtime relative to the analyzed audio length, allowing to calculate the total relative runtime for each concert and dataset.

The aim during the system development was to achieve the highest TPP while keeping FPP low and maintaining a reasonable system runtime. This was done by evaluating the system on the development dataset for 32 different parameter setting combinations to optimize the system parameters.

After developing the system, I conducted a final evaluation using the optimized parameter settings on a new test dataset. This gave a clearer picture of how the system would perform in a real-world scenario on data it was not optimized on. Additionally I performed the evaluation individually for the ALT module on well-established benchmark datasets for this task.

The final results on the test dataset were promising. For the transcription of lyrics, the system's ALT module proved to be highly effective. Using benchmark datasets, the ALT system outperformed state-of-the-art results, showing great robustness against variations in different languages. The degree of robustness towards audio quality is yet to be investigated, as these benchmark dataset to individually test the ALT module were all high quality.

In terms of overall setlist identification, I compared this project's lyrics-based system with Shazam and leading research in setlist identification. LyricSIS surpassed Shazam in accuracy, achieving a TPP 1.7 times higher and an FPP 2.5 times lower, yielding an overall TPP of 45.63% and an FPP of 15%. Even though Shazam worked faster, LyricSIS proved more successful at correctly identifying the songs performed at the concerts. When I compared LyricSIS to the leading setlist identification systems in research, which relies on a song's harmonics for identification, I found that the researched system outperformed this project's lyrics-based system by 1.5 times in terms of TPP with both systems exhibited similar FPP. Given that both systems still yield inconsistent results and fall short of perfection, this suggests a potential benefit from a combined approach that utilizes various song attributes such as lyrics and harmonics for song identification.

Further, I investigated how factors like genre and audio quality affected the system's accuracy. I found that the system's accuracy deteriorated with poorer audio quality and identifying setlists from rap concerts took about double the time compared to other genres. LyricSIS demonstrated the most effectiveness for high-quality audio and showed topmost accuracy for the pop genre.

In conclusion, LyricSIS has shown highly promising results in the automatic lyric transcription task, outperforming state of the art methods from the research area. Furthermore, it is proven that the setlist identification task for live performances including the ISWCs can be accomplished by using the lyrics of songs. However, there's still room for improvement, especially in accuracy and speed for varied audio qualities. These insights will guide the next steps as to optimize the system for higher TPP while maintaining reasonable FPP and system runtime. Overall, I am optimistic about the potential of LyricSIS to provide a valuable tool for concert setlist identification in the future.

1.4 Conclusion

In this project I developed a lyrics-based system that could identify setlists of live performances, including the ISWCs related to the identified songs. I achieved this by designing a multi-stage pipeline to process raw audio, transform it into lyrics, and use these lyrics to identify the songs performed at the concert, including their metadata. The first step in this process involved the development of an Automatic Lyrics Transcription (ALT) module, which converted raw concert audio into AI-generated lyrics. Despite certain limitations in transcription accuracy, this module outperformed existing state-of-the-art methods. The second step included a song search module that uses the transcribed lyrics as input to search databases like Genius.com and MusicBrainz to identify songs and obtain metadata.

Testing the system on 41 concerts yielded a True Positive rate of 46% and a False Positive rate of 15%. The analysis of the concert took longer than half the duration of the actual event. These results points to some limitations in the system's accuracy and speed. With an accuracy level below 50%, the system's standalone usability is limited. It also processes more slowly compared to other systems like Shazam. The system requires high-quality audio for maximum accuracy, and the accuracy levels fluctuate across different music genres. Though this project's scope only covers concerts featuring songs with lyrics, the system is constrained to identifying songs that include lyrics, which limits its flexibility and versatility for broader use cases.

The dataset for testing was unevenly distributed across genres and audio quality, making it hard to draw definite conclusions on the influence of these two characteristics. Further investigations with a more elaborate dataset would be worthwhile.

Although these limitations regarding the setlist identification exist, the system's ALT module can be a valuable tool for various stakeholders in the music industry, such as social media platforms, streaming services, and collection societies. Whereas the former two might use the ALT module to transcribe lyrics and improve user engagement, the collection societies could use the transcriptions to cross-verify catalogues. This could clarify inaccuracies and resolve conflicts within song entities, including ISWCs.

In conclusion, LyricSIS holds promise, yet it needs enhancement in its accuracy and speed, particularly under diverse audio circumstances, to prepare it for commercial use. I remain optimistic about the value this tool could bring to the music industry in the future.

1.5 Future Work and Recommendations

To optimize the system, a variety of potential improvements have been identified. They are listed here in descending order of the anticipated impact each method might have.

Firstly, I see fine-tuning the Whisper model on a singing dataset as the most promising approach to system enhancement. This strategy holds the potential to significantly boost the system's accuracy by data-driven transitioning of the model from speech to the singing domain. Additionally, it provides the possibility to the user to strike a balance between accuracy and potential speed improvements, depending on the size of the pre-trained baseline Whisper model chosen for fine-tuning.

Secondly, transitioning to an audio fingerprint-based approach for song and setlist identification is another compelling improvement strategy. This transition is anticipated to boost both speed and accuracy. Importantly, it does not discard the use of lyrics but transforms them into a different format, such as a vector representation. This can be seen as a fingerprint of lyrics, which makes this approach an augmentation to this work, rather than making it obsolete. It also enables the identification of songs without lyrics, which could lead to a more robust and versatile setlist identification system with a broader range of use cases.

Thirdly, securing direct access to a lyrics database constitutes another essential step in enhancing the system. Paired with an optimized search algorithm, it could greatly accelerate the song search speed, thereby increasing the efficiency of the existing system.

Lastly, considering novel transcription models such as the Whisper large-v2 model could be beneficial. Although it might potentially decrease speed compared to the use of the medium model, it might significantly enhance accuracy. Thankfully, the exploration of this model is a straightforward task, as I implemented the model size as a user-adjustable parameter, eliminating the need for any additional implementations to conduct this investigation.

Other potential improvements include increasing the dataset size for more statistically significant results, performing software testing for validation of functionality and code robustness, enhancing the search algorithm to identify partially performed songs, and investigating a pre-analysis phase for audio-based song segmentation. The incorporation of non-vocal sources, such as the instrumental source, which is generated during the music source separation step but not used in this project, could also be beneficial for audio-fingerprint-based song identification. To address the issue of poor audio quality, the system could be enhanced through advanced pre-processing and fine-tuning the models with a dataset that includes low audio quality samples.

Moving on to the recommendations, ordered based on my perception of their importance.

First, before the system can be commercially utilized for setlist identification tasks, I recommend focusing on improving the system's accuracy. Due to its current limitations regarding accuracy, a careful consideration of the associated risks, defined by the TPP and FPP, is necessary. I am confident that addressing this issue is possible by fine-tuning the system with singing data and augmenting it with audio-based fingerprints. In my view, these steps are crucial for boosting the system's speed, robustness, and accuracy. Once these improvements are implemented, the system could be effectively tailored for its main task of setlist identification. Thus, I recommend undertaking these initial steps of improvement before the system is deployed for setlist identification.

Additionally, I recommend harnessing the system's ALT module for lyric transcription processes. Given its high accuracy, it could open up new business opportunities. With further fine-tuning of the models, this module could become a very powerful lyric transcription tool compared to competitions in the industry.

Further recommendations include using the system for general audio pre-processing tasks, such as speech enhancement and voice activity detection, and music source separation to generate music sources, also known as stems, potentially expanding its application to licensing stems in addition to the usual licencing of entire songs. Future developments should be strategically aligned with specific use cases, whether the focus is robustness, speed, or accuracy.

2 Project Context

The Project Context chapter begins with the problem background, which serves as the motivation behind the company's goals and the project. It then describes the project's goals and objectives and potential use cases. To give a snapshot of the existing market, four related commercial industry solutions are also presented.

2.1 Problem Background

The music industry currently faces a multitude of challenges due to technological disruptions, outdated copyright laws, fragmentation of copyrights, inaccurate metadata, and a general lack of transparency in the royalty chain. Digital transformation has drastically changed how music is created, distributed, and consumed. The shift from physical sales to digital downloads and streaming services has necessitated a re-evaluation of royalty models. Figure 2.1 demonstrates the extent of this shift. Existing copyright and intellectual property laws, formed prior to the digital era, struggle to navigate the complexities of today's music landscape. This struggle leads to confusion, legal disputes, and challenges in managing copyrights [1].

Rights and Metadata

For any piece of music, two primary copyrights exist: the master copyright, which pertains to the actual sound recording and is typically owned by record labels, producers, or occasionally the performing artists themselves, and the songwriting copyright for the underlying compositional parts such as lyrics and melody, owned by composers, songwriters, or their publishers. The compositional parts are usually referred to as 'work'. Record labels, representing the interests of performing artists and producers, manage the production, distribution, and promotion of sound recordings. On the flip side, publishers represent the interests of songwriters and composers, ensuring that the creators receive royalties for the usage of their work.

Multiple persons or entities could own parts of these two copyrights, and their (initial) shares are determined by their contribution to the creation of the song. Information about the fragmentation of copyrights of a song is crucial to determining the correct royalty split among the artists. This information, such as the contributing artists and their royalty split information, is part of the metadata related to a song.

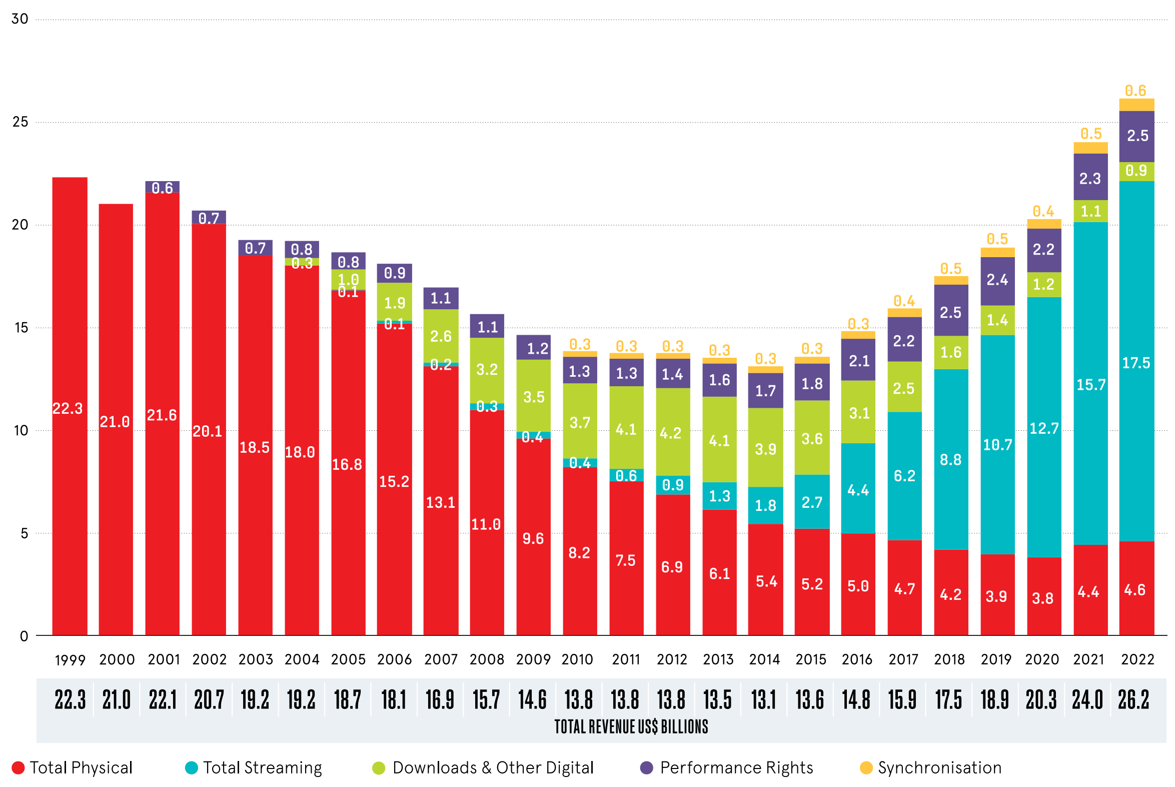


Figure 2.1: Global recorded music industry revenues from 1999 to 2022 in US$ Billions[2]

Every song should have two unique identifiers. The ISRC (International Standard Recording Code) as a unique identifier linked to the actual sound recording and master copyright owners, and the ISWC (International Standard Musical Work Code), which is, on the other hand, linked to the underlying work and its contributors, who hold the songwriting copyright. It is important to note that a recording can only have one ISRC and one ISWC, as these are unique identifiers for the recording and the underlying work, respectively.

However, multiple recordings, such as different versions or remixes of the same work, can share the same ISWC, but they will have different ISRCs as each recording is distinct. To illustrate this, let us consider an example: A band records a song called "Song A" with an underlying work. This recording of the song will have an ISRC (e.g. NL-23-00007) and an associated ISWC (e.g. T-123.456.789-0) for the work. Later, a DJ creates a remix of "Song A," resulting in a different recording. The remix will, therefore, have its own ISRC (e.g., NL-23-00008) because it is a new recording, but it will share the same ISWC (T-123.456.789-0) with the original recording, as the underlying work remains the same.

Challenges in Royalty Attribution

Challenges in royalty attribution arise particularly when versions of an original song are created. This could be a new recording that utilizes the songwriter's lyrics but alters the melody or arrangement - a common occurrence in the music industry. However, in the absence of complete metadata, these adaptations might not be traced back to the original songwriter, resulting in a loss of potential royalties. This problem is especially pronounced when User-Generated-Content (UGC) platforms, which do not require complete metadata for uploading songs, are involved.

These platforms use specific systems, to automatically analyze the uploaded audio and check if it contains copyrighted material. These automatic copyright infringement detection systems are mainly audio fingerprint-based. This means that the systems analyze the unique characteristics of the sound of the song to be uploaded to create a digital 'fingerprint' that can be compared to a database of the digital fingerprints of known copyrighted songs.

The audio fingerprint-based systems used by these platforms to detect copyright infringements might fail to identify the original underlying work due to changes in the audio fingerprint of the new version. This further exacerbates the issue of unfair royalty distribution, especially for songwriters.

Live Performances and Royalty Attribution

Live performances present unique challenges in accurate royalty attribution. Collection societies, which manage artists' and songwriters' rights and ensure they receive their due royalties, rely on the submitted setlists of performed songs to allocate appropriate payments. However, inaccuracies or omissions in these manually generated setlists can lead to errors in royalty distribution.

Further complications arise from the nature of live performances. These often feature varied renditions of the original recordings, including improvisations, remixes, or adaptations, making their identification and correct attribution problematic. Existing audio fingerprint-based song identification methods, used for copyright infringement detection, often struggle to accurately detect these live versions.

These issues result in significant inefficiencies in managing royalty allocation for live performances, leading to an unfair distribution of royalties. It underscores the need for advanced technology solutions to ensure fairness and accuracy in royalty attribution.

The Black Box of Royalties

In instances where the rightful recipients of royalties cannot be immediately identified, their royalties are often held in a so-called "black box" by collection societies and streaming platforms. If the necessary information to distribute the money accurately does not surface after a specific period, the funds are distributed to the rest of the known copyright owners of that song, further contributing to the unfairness in royalty distribution. Figure 2.2 demonstrates this process.

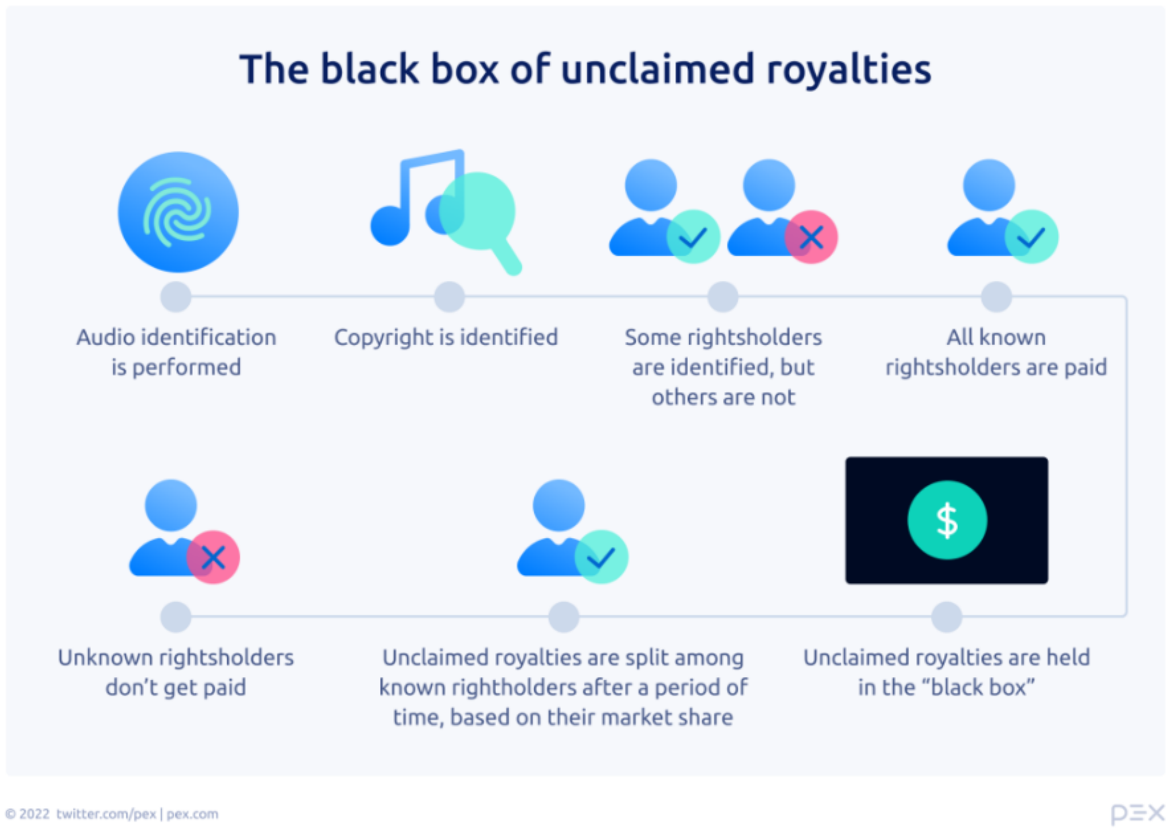

The black box of unclaimed royalties
Audio identification is performed
Copyright is identified
Some rightsholders are identified, but others are not
All known rightsholders are paid
Unknown rightsholders don't get paid
Unclaimed royalties are split among known rightholders after a period of time, based on their market share
Unclaimed royalties are held in the "black box"
© 2022 twitter.com/pex | pex.com
pex

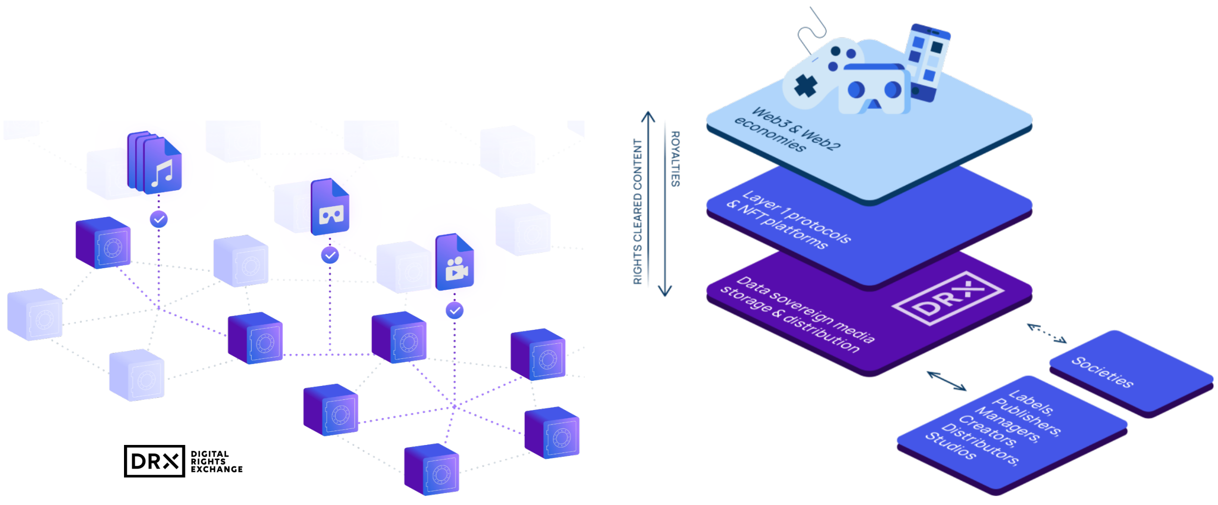
Figure 2.2: Unclaimed royalties due to attribution problem based on lack of song metadata [3]

It is worth noting that the creation of musical versions, or adaptations of original songs, has always been an important part of musical practice. These adaptations, which can include remixes, covers, or re-arrangements, allow songs to remain 'alive' for many decades or even centuries. This makes it all the more important for artists and composers to obtain recognition and credit for their work.

2.2 Company Goals

The company's primary goal is to enhance transparency, fairness, and ethical practices in the music industry's royalty chain by developing their main product called Digital Rights eXchange (DRX) utilizing multiple blockchain technologies.

DRX is a decentralized, tamper-proof, data sovereign media and metadata storage network consisting of digital banking blockchain vaults, as illustrated in 2.3a. Every artist, label, publisher, or other party in the music industry can insert the sound recording and its entire metadata, especially including the royalty split information of all the different copyright holders, into these vaults. Transparency is maintained at the vault level, allowing everyone access to their own vaults and the ability to view the songs hosted in those specific vaults. However, other parties cannot see the detailed insights, such as banking-related information or other sensitive information stored in the vaults, ensuring the perfect balance between transparency, privacy, and security. All vaults are interconnected, facilitating transparent royalty transactions, tracking, fair distribution, and efficient exchange of information, such as metadata of a song, between them. This ensures that artists and rights holders receive accurate and prompt payments. The blockchain's transparency enables all parties to verify transactions, eliminating the need for intermediaries and reducing disputes over royalty payments.


DRX DIGITAL RIGHTS EXCHANGE
RIGHTS CLEARED CONTENT
ROYALTIES
Web3 & Web2 economies
Layer 1 protocols & NFT platforms
Data sovereign media storage & distribution
DRX
Societies
Labels, Publishers, Managers, Creators, Distributors, Studios

(a) DRX as vaults connection **(b)** DRX as root layer

Figure 2.3: Vizualization of DRX as connection between the blockchain vaults and as the root layer of the music industry ecosystem with higher level layers build on top [© Copyright Delta]

DRX and its stored data serve as the root of trust, with songs being uploaded and hosted as NFTs in the blockchain vaults having 100% cleared copyrights and metadata and a unique digital identifier. For clearing the copyrights of a new upload, it is important to automatically detect suspicious uploads regarding copyright infringements. DRX is the root layer 0 upon which multiple higher-level layers can be built, as depicted in 2.3b. Utilizing blockchain technology, it is possible to manage the licenses of the sound recordings via smart contracting on the blockchain. Whenever a license for a musical piece is purchased through smart contracting, the owners of the copyrights for that song will instantly receive their fair share of the payment. The process can be observed in Figure 2.4 for the example of streaming within web3. By leveraging micropayments and smart contracts, DRX can enable more equitable compensation models for streaming services. This would allow artists to be paid fairly based on the actual consumption of their work, rather than relying on the current, often criticized, pro-rata model. Smart Contracting, therefore, increases transparency, fair and fast royalty payouts, and ultimately ensures an ethical distribution of royalties to all involved parties.

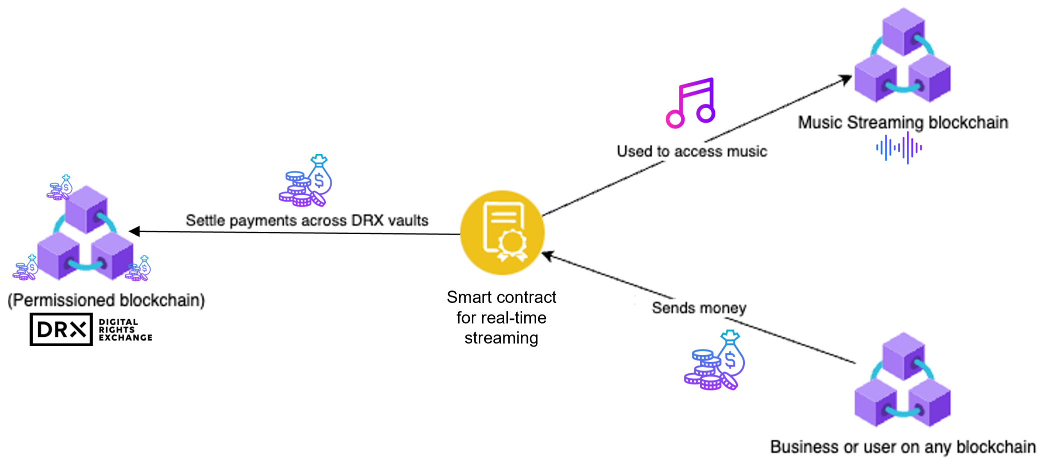

Music Streaming blockchain
Used to access music
Settle payments across DRX vaults
(Permissioned blockchain)
DRX DIGITAL RIGHTS EXCHANGE
Smart contract for real-time streaming
Sends money
Business or user on any blockchain

Figure 2.4: DRX royalty settlement via smart contracting visualized for the example of streaming within web3 © Copyright Delta

To clarify and explain DRX a bit more, let me cite an analogy of a colleague:

> *"We can see the music sector as a forest ecosystem, with its institutional and organizational actors as trees, such as colection societies, labels, publishers and artists. These trees are deeply rooted in the soil of the ecosystem. Well functioning ecosystems are healthy above ground just as they are below the soil. DRX can be seen as the mycelium that connects the roots of the trees in a wood wide web-manner. The Wood Wide Web refers to the finding of a Suzanne Simard describing that trees in a forest could communicate and cooperate through subterranean networks of fungi and are rather altruistic than competitive [4]. The collaboration and symbioses of the tree roots with the mycelium creates value for both species instead of extracting it for only one. It unlocks nutrients that are otherwise not reachable or available for either species. The creation of a wood wide web also enables actors within the system to communicate over vast distances. The current music ecosystem has deeply rooted trees, however it lacks a wood wide web. By inoculating the ecosystem with DRX spores, and by planting some trees ourselves, this can organically grow in scale to a*

fully functioning, healthy and resilient, future proof ecosystem for the music industry." - Frank Kimenai, Chief Public Partnerships & Funding @ Copyright Delta

An important secondary goal is to analytically proof the unfairness of royalty distributions from the artist and songwriters view to show the world that it is time for a change. For that, the company even quantifies the royalties that artists are missing out due to the lack of transparency and copyright infringements. The quantified and proven royalty gaps are then legally claimed to make sure that the artists are given the missing royalties they are entitled to. Thereby, supporting the artists and songwriters and emphasizing the importance and need of a new, revolutionary, disruptive foundation of the music industry's infrastructure, provided by DRX. Overall, the company firmly believes that it is high time for a radical overhaul of the music industry's infrastructure foundation, and is leading this disruptive transformation.

2.3 Project Goals and Objectives

This project arose from a shared vision between the company and myself to innovate a solution for a prevalent issue in the music industry: the automatic identification of songs in live performances or concerts. It's generally acknowledged that a song with lyrics can be identified across its varied versions by these lyrics, given that the lyrics remain unchanged [5]. In addition, Automatic Lyric Transcription (ALT) methods can generate song lyrics automatically from raw audio files, although the accuracy of this process isn't always consistent as seen in [5], [6], [7], and [8]. However, it's still unknown whether this lyrics-based song identification approach can be applied to entire concerts, thus enabling the automatic creation of full concert setlists. This project emerged from the mutual belief between the company and myself that this method could possibly extend to setlist identification - an unexplored field. If lyrics-based song identification can indeed be applied to setlist creation, it should be feasible to determine the underlying ISWC code for each song in the setlist. Ideally, this would create a direct link between a concert audio file and a setlist of songs, each associated with their corresponding ISWC codes which represent the songwriting copyright holders.

The primary goal of this project is therefore in direct alignment with the company's ambition to enhance transparency, fairness, and ethical practices in the music industry's royalty chain. The project aims to design and develop a lyrics-based setlist identification system that can generate setlists automatically from raw audio inputs of live performances. The aim is to prove or disprove that setlists can be automatically generated from raw audio inputs of live performances or concerts in a lyrics-based fashion. This would be accomplished by using the lyrics of the performed songs as a key to identify a song and link it back to the ISWC. If this approach is successful, it will revolutionize setlist creation, enabling the identification of songs and their linkage to their respective ISWC codes, representative of songwriting copyright holders.

The successful implementation of this project will contribute significantly to the company's primary product, the DRX, in ensuring a secure and precise database of songs and their metadata.

The application of this system would be instrumental in the company's efforts to minimize copyright infringements, safeguard fair royalty distributions, and recuperate lost royalties for artists.

Additionally, this project contributes to the company's secondary objective by offering the capacity to identify underlying works of songs in User Generated Content (UGC). This capability is particularly crucial for live cover versions of songs or live concerts, where traditional audio fingerprint-based copyright infringement detection systems often fall short. The association between UGC and underlying works uncovers potential copyright infringements and royalty gaps and also promotes the mission of ensuring fair compensation to artists and songwriters.

The primary goal of the project is accomplished by pursuing the following high-level objectives:

1. Conduct a thorough literature review to understand the current state-of-the-art methods applicable for lyrics-based song and setlist identification
2. Design the system with clearly defined requirements and make informed design decisions based on the literature review results and the set requirements.
3. Identify and procure suitable datasets that can be effectively used to develop and evaluate the system's performance.
4. Utilize the state-of-the-art methods identified from the literature review to develop and implement the system.
5. Carry out a comprehensive evaluation of the system to assess its accuracy and speed in different scenarios and its ability to meet the set requirements.
6. Discuss the results and provide a balanced view on the system's strengths and weaknesses. Identify the system's limitations and suggest directions for future work to improve these areas.
7. Derive conclusions on the system's applicability based on the evaluation results and provide clear recommendations to the company on how it can or cannot be utilized effectively.

These objectives are intended to guide the project towards resolving the identified issue in the music industry: the absence of an automatic setlist identification system in live performances or concerts. The successful achievement of these goals would lead to the development of an innovative system that promotes fairness and transparency in the music industry's royalty distribution. The system's success would further fortify the company's primary product, DRX, ensuring a secure and accurate database of songs and their metadata. Ultimately, these objectives directly support the company's mission to minimize copyright infringements, safeguard fair royalty distributions, and recover lost royalties for artists.

2.4 Potential Use Cases

The successful development and accuracy of LyricSIS, which includes an ALT module to transcribe the lyrics from the raw audio files, can result in numerous potential use cases. These might open up various markets within the music industry, from live music venues and streaming platforms to collection societies and social media platforms. The existing challenges, such as difficulties in identifying and tracking song usage and copyright owners across different platforms and formats often lead to lost revenue and unfair compensation for artists and songwriters. The developed system addresses these issues by identifying songs along with their respective ISWCs, facilitating a more efficient and fair distribution of royalties, benefiting all involved stakeholders. Additionally, the ALT module of the system itself can be used for different scenarios.

Below are potential benefits for different parties:

- **Live music venues and event organizers:** This system can help venues generate setlists automatically, replacing the manual creation process. The only requirement for the venues is to provide the recorded live performance. This change not only would save resources, but it also would ensure fairer royalty distribution, making these venues more appealing to artists. By integrating ISWC codes into the setlists, collection societies could distribute performance royalties promptly and accurately to correct copyright holders, including songwriters and composers.
- **Social media platforms:**
  - These platforms currently use systems that rely on audio-fingerprinting technology to determine if user-generated content infringes copyright. This method is generally effective, but it struggles in some cases. For example, if a user uploads a cover of a song that greatly differs from the original, the system may fail to detect copyright infringement. This lyric-based identification system I developed can improve the effectiveness of these platforms' copyright detection algorithms. By analyzing the lyrics, the system also applies to live performances, cover songs, and other recordings that alter the audio, or present a different version of the song, like renditions in unique styles or by different artists. Thereby, the system can identify and flag potentially infringing content which the platforms system might have failed to, allowing the platforms for better compliance with intellectual property laws.
  - Additionally, they could use the system to identify setlists of hosted live performances, providing an enhanced user experience by adding this extra layer of information.
  - Another use case for these platforms results from a successfull ALT module. By using it to automatically transcribe the lyrics of songs or performances, the user experience could be improved by displaying this additional information.
- **Streaming platforms:** Streaming platforms can improve their user experience by using the ALT module of this system to automatically transcribe the lyrics of songs. Currently,

these platforms rely external lyrics database-hosting parties, such as MusixMatch, to provide song lyrics, which limits them to the extent of this external database. By using this system, they could become independent and provide lyrics for any song that contains them. This approach could significantly increase the number of songs displayed with accompanying lyrics, thereby enhancing the user experience.

- **Lyrics database hosters:** Lyrics database providers, like Genius or Musixmatch, can use the ALT module of this system to automatically expand their lyrics database. They could apply it to newly released songs or older songs not yet in their database, considerably improving their catalog. This expansion could increase their business value and give them a competitive edge over other database providers.
- **Collection societies:** Collection societies can improve their song metadata database by using the ALT module of this system. It could be used to link lyrics of multiple registered songs, some of which may be covers of others, but were mistakenly registered as individual works, falsely resulting in multiple ISWCs. By applying the ALT module to extract song lyrics and comparing them, collection societies can resolve these discrepancies and achieve more accurate metadata, which is crucial for fair royalty distribution. For example, the results could be used assign one ISWC for all versions of a song, based on the original. The original song can usually be identified by comparing the registration timestamps if they exist, leading to an overall more accurate music catalogue. Therefore, the ALT module can be employed as a service to assist collection societies in maintaining their catalogs and addressing inconsistencies or inaccuracies in their databases.

However, it is crucial to remember that these are potential use cases. Their feasibility depends on several factors, including the evaluation results regarding this system's accuracy and speed.

Copyright Delta can commercialize the developed system by offering it as a service or licensing the technology to various stakeholders within the music industry. By tailoring the solution to meet the specific needs of each stakeholder group, the company can tap into multiple revenue streams while contributing to a more equitable and transparent music landscape.

2.5 Existing Industry Solutions

This section briefly discusses the existing industry solutions for applied song identification and applied cover song identification, with a focus on four prominent companies: Shazam, SoundHound, DJMonitor, and PEX.

Shazam and SoundHound are both dedicated to song identification besides other tasks, utilizing audio fingerprinting technologies to recognize songs based on their distinct audio features. While the specifics of their fingerprinting algorithms remain proprietary, these companies have achieved significant success in the consumer market by providing user-friendly applications for real-time song identification [9], [10].

DJMonitor and PEX, on the other hand, offer solutions aimed at identifying songs and cover songs for royalty claim purposes. DJMonitor maintains an extensive audio database of songs and analyzes the live performances played by DJs. By applying fingerprinting technology to these recordings and matching them against their database, DJMonitor can identify the songs performed, link this information back to the artists and songwriters, and provide setlists to collection societies for accurate performance royalty payouts [11], [3].

PEX also provides services for artists, labels, publishers, and other stakeholders to identify songs being performed, with a focus on the digital content ecosystem. Possessing their own database and utilizing fingerprinting technologies, PEX claims to be capable of even identifying cover songs by using melody as a matching factor. However, detailed information about their methodology or their accuracy rate is not publicly available.

To evaluate the performance of the LyricSIS, I intend to test it against Shazam's song and setlist identification capabilities as a benchmark on a test dataset of live performances. Shazam was selected for this investigation owing to its readily available API, rendering it suitable for comparison, in contrast to DJMonitor and PEX solutions that lack complimentary access to their song identification software as they offer it as a service. While SoundHound offers a similar service to Shazam, I opted for Shazam because of its widespread recognition and more user-friendly API. This selection facilitates a comprehensive assessment of the system's performance against a well-established industry solution, thereby providing valuable insights into its potential for real-world deployment.

3 System Design

This chapter provides a detailed description of the functional and non-functional requirements of the system, highlights the detailed design choices made with respect to the architecture, component and implementation desing based on the also in here described state of the art methods and explaines design choices made to fulfill these requirements. Additionally it describes the design of the datasets used to be sure the system can be evaluated on the archievement level of the requirements.

3.1 Functional requirements

The system to be developed is a multistep pipeline designed to process an audio file as input and output a setlist of songs. Functional requirements are defined as follows:

- **Automatic Lyric Transcription**
  The approach of this project is to identify songs based on their lyrics. Since the input to the pipeline begins with a raw audio file of a live performance, the first requirement is for the system to automatically transcribe the audio file and generate a transcription, referred to as AI Generated Lyrics (AIGL) throughout this thesis.
- **Lyrics-based Song-Metadata Searching**
  The subsequent step in the pipeline involves identifying songs performed during live events and obtaining their metadata using the AIGL. To achieve this, the system requires access to a comprehensive collection of songs and their metadata, with lyrics being a crucial component of the metadata information. This collection is referred to as the lyrics database. The AIGL is compared to the actual lyrics of songs in the lyrics database using a string comparison method to match the lyrics and identify the songs. Once the song is identified by the Artist and Songname, the system must fetch the ISRC and ISWC metadata from the MusicBrainz database by using their API and based on the artist name and song name. By integrating this additional metadata, the setlist will be in its final format. Considering that the AIGL consists of transcribed lyrics from multiple songs within a single audio file without any knowledge of each song's start and end times, the system must also be capable of identifying the start and end points of individual songs within the AIGL, a process termed song segmentation.

- **Setlist Generation and Evaluation**
  As a final step, the system should compile a list of all identified songs, along with a measure of certainty for each song's identification, expressed as a similarity value resulting from a string comparison between the lyrics of identified songs and the part within the AIGL. For evaluation purposes, when the actual performed songs are known, the setlist should not only include the identified songs and their certainty levels but also classify them as True Positives, False Positives, and False Negatives.

3.2 Non-functional requirements

In addition to the functional requirements, the following non-functional requirements have been established in consultation with stakeholders:

- **Robustness**
  The system should be robust and adaptable across various conditions, ensuring its reliability and effectiveness in different scenarios. The following factors contribute to the robustness of the system:
  - **Multiple Languages and Accents**
    The system should be capable of handling multiple languages and accommodating various accents. This flexibility will enable it to be used in a wide range of settings and cater to diverse user requirements, enhancing its overall utility.
  - **Audio Quality**
    The system should be resilient to poor quality audio, including non-sung vocals such as announcements, applause, or noisy backgrounds. This will ensure that it can still perform effectively even when faced with less-than-ideal input data.
  - **Genre Independence**
    The system should be able to work effectively across multiple genres, demonstrating its versatility and adaptability. This will allow it to cater to a broader range of users and applications, making it more appealing for various use cases.
  - **Audio Length Flexibility**
    The system should be able to handle audio files of varying lengths, from short clips or single songs to complete performances. This robustness to audio length is important because the system shall be able to analyze live performances of just one song as well as complete concerts. Additionally this flexibility is necessary to prevent issues such as Out-Of-Memory errors that may arise when processing large audio files with extensive AI models. By ensuring that the system can efficiently manage its resources and process audio files of different lengths without encountering memory limitations, it becomes more versatile and useful.

  - **Error Handling and Recovery**
    In the event of errors or unexpected situations, the system should be designed to resume processing the pipeline from the point at which the error occurred after debugging the error. One reason to design the system in this way is the potential for a high occurrence of errors, particularly because of relying on an API to access the lyrics database. Issues such as temporary loss of internet connection on the users end or a server crash on the host's end can disrupt the process, making it crucial to have this kind of recovery mechanisms in place. The rationale for resuming at that point is to avoid re-analyzing concerts that have already been processed prior to the error. By resuming the process for the specific concert and at the pipeline-step where the error occurred, the system can save time and computational resources.

- **High Accuracy**
  I will measure the system's accuracy using these metrics: True Positives (songs correctly identified as performed during the live concert), False Positives (songs identified as performed but not actually performed), and False Negatives (songs performed but not identified). It's important to note that True Negatives will not be considered for measuring the system's accuracy. In this context, True Negatives refer to songs correctly identified as not performed. As this includes all other songs globally, the count of True Negatives approximates infinity and does not give a meaningful evaluation. My goal is to maximize True Positives and minimize False Positives.

  I compare the artist's name and the song's name metadata to the ground truth setlist to measure accuracy, as finding this metadata currently holds more importance than finding the ISWC. Ultimately, identifying the ISWC is crucial, but for now, access to ISWC databases is limited. For example, it is possible to find the ISWC based on the artist's name and song title at CISAC, but they do not offer an API for bulk requests. The MusicBrainz database is the only public database that includes the ISWC code in the song metadata. However, this is a community-based database, so it also depends on publicly available data. For the project's completeness, I use this database to fetch the ISWC codes. But it is likely that the ISWC will not exist in the MusicBrainz database for many identified songs.

  There are high chances that the company will gain access to the song metadata catalogues from all collection societies. This access could then be used to fetch most of the ISWCs based on the artist's name and song name. Therefore, I am currently evaluating the project using only True Positives and False Positives based on the artist's name and song name. I am not considering the number of ISWCs identified.

- **Cost-effective**
  While CPD's priority is to evaluate the system's accuracy while keeping high speed and low cost, the development process will rely exclusively on open-source products and avoid

spending money on external services. This approach is reasonable for a proof-of-concept stage and helps maintain cost-effectiveness during the initial development phase.

- **Fast Performance**
  While the system's speed is a low-priority requirement, it should still perform reasonably quickly to maintain a positive user experience, particularly for tasks such as copyright infringement detection in UGC. The main priority, however, is to establish a trustworthy source of copyright-cleared songs, making accuracy more important than speed. In the context of live music performances, in further work the system could be implemented to analyze performances in a streaming manner, allowing the setlist to be generated as the performance is ongoing. Speed will be measured as the ratio of runtime to audio length, both for the entire pipeline and for each individual step, to analyze the impact of different pipeline components on the systems speed.

3.3 Architecture Design

In this section, I provide a detailed design of the architecture for LyricSIS, addressing the functional and non-functional requirements highlighted in the preceding sections. I explain the decisions made during the design process, which were informed by a thorough review of relevant literature. The final system architecture, as illustrated in Figure 3.1, comprises multiple sequential modules. Each of these modules plays an essential role in achieving the system's intended functionality.

Setlist Identification

I initiated the process by reviewing the literature on setlist identification (SLI), defined as a music recognition scenario aimed at retrieving metadata and timestamps for each song played in a music set. This yields a list of songs, referred to as the setlist, that includes metadata such as the artist's name and song title, along with the start and end times of each song within the set. These sets could take the form of DJ sets, live music concerts, or any other performance involving multiple songs [12]. However, for the purposes of this thesis, the timestamps are not essential; the focus is solely on identifying the metadata for songs played. Consequently, I use the term 'setlist' in this context to signify a list of songs performed during a concert.

Research on SLI is relatively scarce and, to my knowledge, has only been explicitly addressed by Yesiler et al. [12] and Wang et al. [13]. The setlist identification process in both researchers' systems operates on the basis of a song identification system. This system works by partitioning the audio into sections or by sliding an overlapping window across the audio, and then identifying the song in that section. The setlist is compiled by executing song identification on all portions of the concert and adding each identified song to the list.

The overall state of the art method for the song identification task in general is to use audio fingerprints [12]. Even though the focus of this project is on employing plain song lyrics for song identification, understanding the audio fingerprint-based song identification method is crucial. This is because Yesiler used this method for setlist identification, and its limitations were part of the motivation for this project.

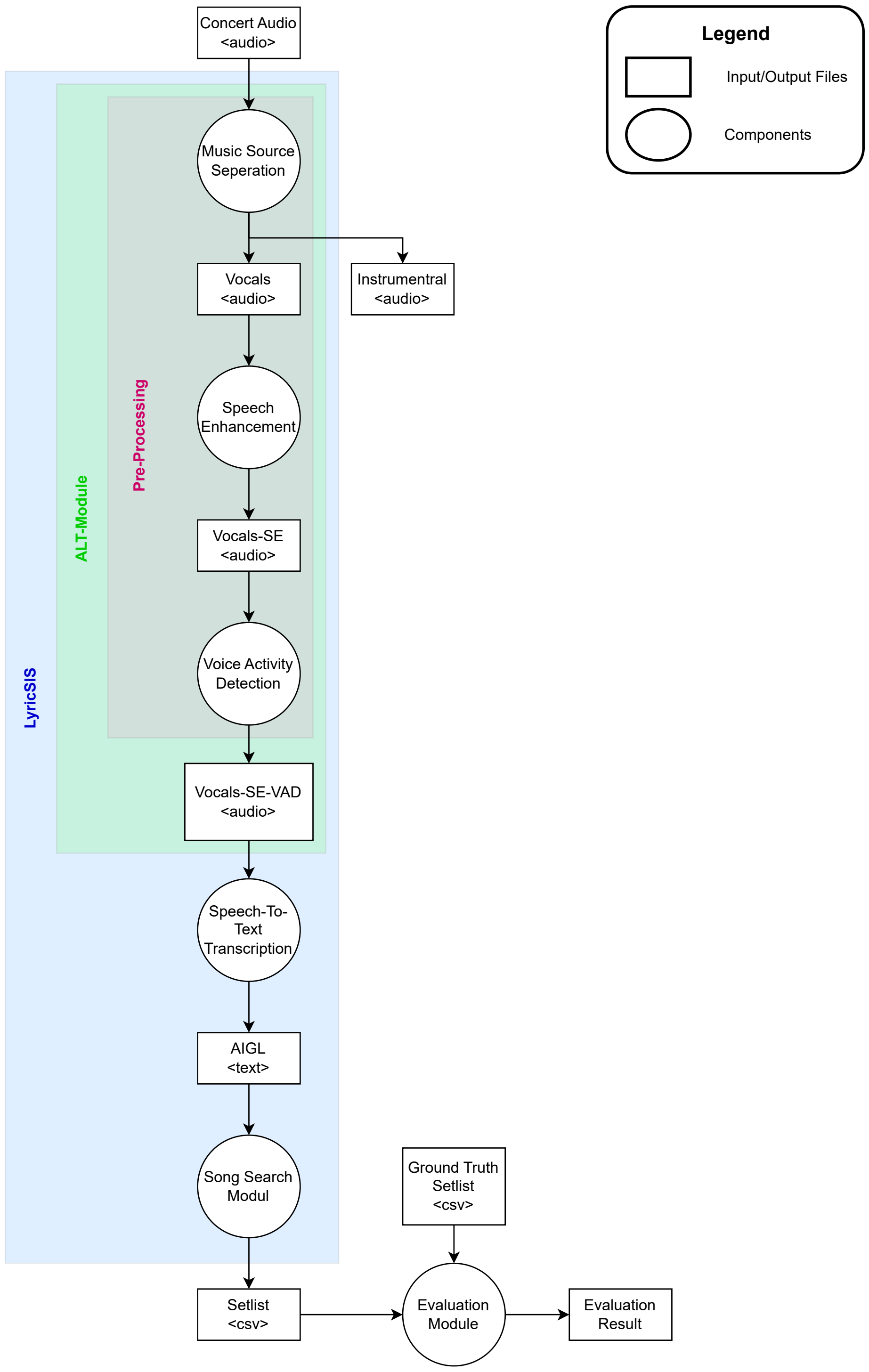


Figure 3.1: LyricSIS Architecture Overview

Audio Fingerprinting in Song Identification

Audio fingerprints are unique features extracted from a song's audio waveform, creating a distinct digital fingerprint for that song. These features encompass elements such as spectral content, temporal patterns, and tonal characteristics. The song identification method hinges on matching the audio fingerprints from the song under examination with the reference audio fingerprint of the actual song.

Audio fingerprinting, while effective for identifying original songs, often struggles when it comes to recognizing cover or live versions of songs [12]. This limitation arises because these variations typically display significant differences in their audio features, making their fingerprints unlike those of the original versions. This constraint has given rise to the research field known as Cover Song Identification (CSI). CSI aims to identify different versions of a song, including cover and live versions, by extracting audio fingerprints that remain comparatively consistent across these variations. The features commonly examined in CSI encompass representations of melody, rhythm, harmony, and lyrics. However, the lyrics in this context are not used in their plain style; instead, they're transformed into lyric-representing fingerprints [14].

Lyric-Based Song Identification

To the best of my knowledge, the only research utilizing plain lyrics for CSI is by Vaglio et al. [5], which presents a method for identifying songs based on their lyrics using a combination of text-based and audio-based features. Throughout this project, using plain lyrics to identify songs or cover songs will be termed as "lyric-based song identification". Although these methods demonstrate a certain level of effectiveness in performing the CSI task, they have not been applied to the SLI task, with the exceptions of the aforementioned methods by Yesiler and Wang. Yesiler's method in particular focuses solely on audio fingerprints capturing the harmony of songs.

A crucial aspect and limitation of all approaches to audio fingerprint song identification is the essential need for access to an audio database to generate the reference audio fingerprints of all songs to be potentially identified. Consequently, there is no feasible way to create and use audio fingerprints as the basis for CSI within this project. Additionally, the potential of the lyrics-based song identification from Vaglio inspired this project's lyrics-based approach for the setlist identification task.

Automatic Lyrics Transcription

Lyric-based song identification is a key component of lyric-based SLI, which necessitates the incorporation of an Automatic Lyrics Transcription (ALT) module in the system architecture design. This is due to the fact that lyrics are pivotal in the identification of songs. One of the critical aspects of this project is therefore acquiring accurate lyrics from the input audio. Unlike SLI, ALT is a well-researched area.

Various approaches can be employed for ALT while the state of the art methods are generally AI-based and differ based on the input data. There are primarily two approaches to ALT with respect to the input data used: polyphonic audio and monophonic audio input.

Polyphonic vs Monophonic Audio in ALT Polyphonic input audio involves working with the complete audio recording of a song, where the vocals are embedded within the mix alongside other instruments and sounds. In contrast, monophonic input audio refers to cases where the extracted vocals or the singing voice are unaccompanied by other music components, such as instrumental tracks or background harmonies, that make up the complete song.

Researchers focusing on the monophonic type of input data work on transcribing the isolated vocal tracks This task can be more straightforward, as the vocal content is not obscured by other instruments or sounds. However, this approach depends on the availability of pre-separated vocal tracks, which often requires additional pre-processing methods to extract and pre-process them for the subsequent transcription process.

These pre-processing methods are also needed when using the model for inference in terms of subscribing song lyrics, as most use cases transcribe the song itself so that the audio to be analyzed is in its original polyphonic shape. These methods typically involve several key steps to ensure an accurate transcription in the subsequent phase.

State of the art methods in ALT Researchers, independent of using polyphonic or monophonic audio as input data, have explored two main strategies for ALT. One approach involves training a new AI model from scratch, using the audio signal as input and the corresponding lyrics as output [7], [6]. This method requires a significant amount of labeled training data and is time and computationally-intensive. An alternative approach involves adapting existing speech-to-text models to the task of ALT [8]. By fine-tuning these pre-trained models on singing data, researchers can leverage the existing knowledge of the models to create a ”singing-to-text” transcription system. This method capitalizes on the advancements made in speech recognition technology.

The current state of the art results suggest the use of a singing datasets to fine-tune a wav2vec based speech-to-text model to transfer its usability from the speech to the singing domain, while keeping polyphonic audio input [8]

Although this potentially reduces the amount of training data and computational resources needed to achieve accurate lyric transcription compared to train a model from scratch, the absolute of data, time and computational resources needed it still a huge limitation, especially because it includes shaping the input data to the format that the model required while being trained from scratch. Due to limited time and computational resources, as well as the unavailability of open-source pre-trained models for ALT, these options are not suitable for this project.

Using the Whisper Model for ALT

Instead, I have opted to use the Whisper model, a recently released, open-source, language-robust Speech-to-Text AI transcription model from OpenAI [15]. Whisper demonstrates superior accuracy to other speech-to-text transcription models on widely recognized benchmark datasets. Given that fine-tuning is not feasible for this project and that the characteristics of a singing voice and speech can differ in many terms [16], the model's resilience to accents and languages renders it an ideal choice, as this robustness may extend to handling singing as opposed to speech. Additionally, the availability of various model sizes enables users to balance speed and accuracy, further solidifying its suitability for this project.

Considering the use of the Whisper model, I decided to employ monophonic audio as input. The rationale behind this decision is to shape the input signal as close to natural speech as possible, increasing the likelihood of accurate transcription. The hypothesis is that a non-fine-tuned speech-to-text model, trained on speech data, would perform better on a singing-to-text task when the input resembles speech as much as possible. As a result, I adopt several audio pre-processing techniques, such as music source separation, speech enhancement, and voice activity detection (VAD) to convert the polyphonic audio into monophonic audio.

Pre-processing Module

The first step in audio pre-processing is the music source separation, to transform the polyphonic to a monophonic audio by extracting vocals. Several open-source models are available for this task, with varying speed and accuracies, with the latter being measured by the signal-to-distortion ratio. As the influence of the source separation technique on the final setlist identification task is not certain, it might be necessary to choose a slow but accurate model. It could also be that the quality of the results of a fast but not as accurate model is enough. Therefore, I chose the open-source model Spleeter from Deezer (fast and less accurate) and the Hybrid Transformer Demucs model (htdemucs) (quite fast and accurate) from Meta, as well as another version of the Hybrid Transformer Demucs model that is additionally fine-tuned on more data (htdemucs_ft) (slow and highly accurate), to determine their impact on the pipeline's overall accuracy and speed and finally make the best design choice for this component.

Next, I perform speech enhancement to minimize residual noise and artifacts present in the extracted vocals. I have chosen the denoiser from Meta for this task, which relies also on the underlying Demucs model as the model used in the music source separation stage. This model outperforms many other researchers in benchmark tests, while simplifying the pipeline by reusing the same model for source separation and speech enhancement [17].

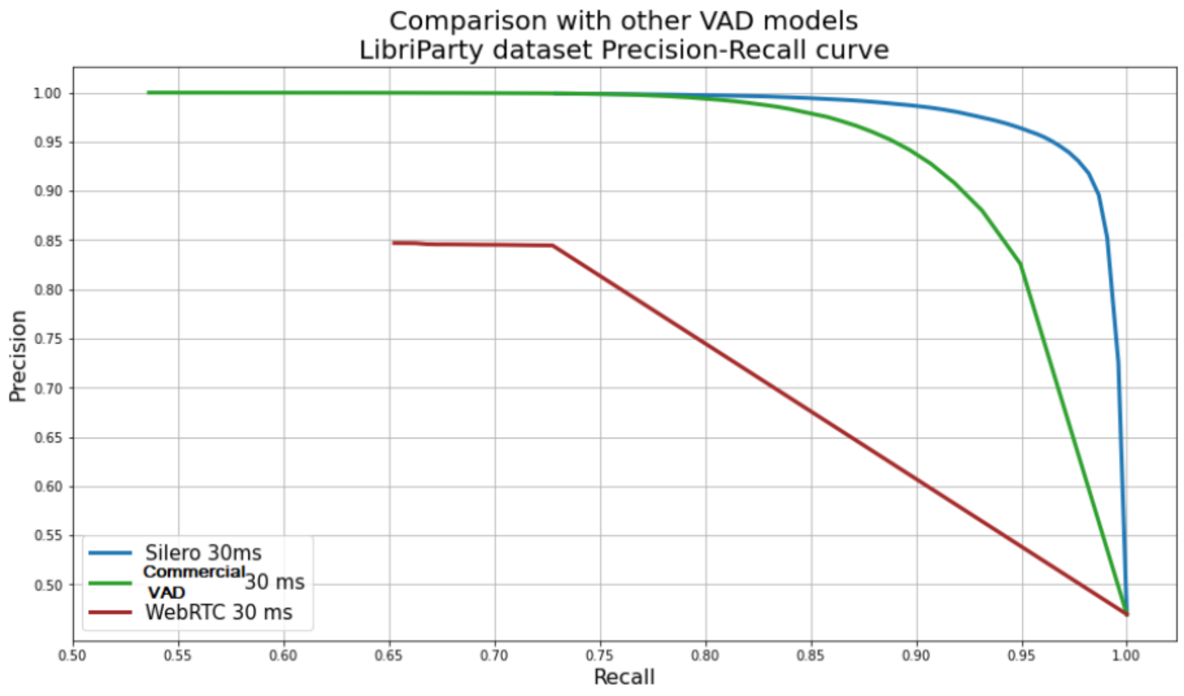


Figure 3.2: Silero-VAD performance benchmark compared to competitors

Following speech enhancement, I apply the Silero-VAD for voice activity detection (VAD) to identify and remove silent or non-vocal segments from the extracted and noise reducted vocal audio. This part is essential to reduce the transcription time and accuracy, as the analysis of silent parts is unnecessary and therefore a waste of resources slowing down the process, as well as prone to generate wrong transriptions for consecutive vocal parts.[15] By refining the input signal in this way, the transcription models can focus exclusively on relevant sections of the audio, improving transcription efficiency and accuracy. The Whisper model has its own internal VAD implemented, but it tends to be inaccurate, as one of the co-authors of Whisper indicated at a discussion within the GitHub project [18].

Silero-VAD is an enterprise-level, open-source, fast, and accurate model that outperforms competitors. I didn't find any existing competitive method or solution for this specific task, except the ones, that the Silero-VAD is benchmarked against, which can be seen in Figure 3.2. The precision-recall curve is shown to showcase Silero-VAD's performance, demonstrating its superiority compared to alternative VAD systems. Using Silero-VAD ensures that the Whisper model's internal VAD is not needed, or at least not the only responsible VAD to cut out silence, helping the reduction of transcription issues such as repeat loops and hallucinations as well as speed up the transcription process.

Lyrics-based Song Search Module

After completing the audio pre-processing steps (source separation, speech enhancement, and VAD), the pre-processed vocal audio is transcribed using the Whisper model, generating the

AI Generated Lyrics (AIGL). The AIGL serves as input for the next step in the pipeline, the lyrics-based song search module, which I call the song search module as of now.

Ideally, this step would directly access an extensive database containing all songs, their metadata (including artist name, song name, and lyrics, and ISWCs), and employ a fast algorithm like TFIDF, as used by [5], to compare lyrics and identify the songs performed in the live concert. However, such a database is unavailable for this project. Therefore, I depend on an external, free-of-charge, and easily accessible database with a comprehensive song catalog. The Genius.com lyrics database (genius.com), searchable via an API, fulfills these requirements, enabling the search for songs based on their lyrics.

The song search module accepts the AIGL as input and processes it iteratively in chunks of length of approximately six words to identify the setlist. It requests song information matching these lyric chunks from the Genius API. Mostly, several songs match these short lyric chunks, which all are songs that have potentially been performed. As next step, I compare the lyrics of these potential songs proposed by the Genius API with a strategicalle chosen part of the AIGL to identify the performed songs. This comparison operates at the string-wise similarity level. If the algorithm finds a match based on a similarity threshold, it adds the song to the final setlist.

I then use the metadata of the songs found in the Genius lyrics database to fetch the ISWC. This information is accessed via an API to the MusicBrainz metadata database (musicbrainz.org). Finally, the fetched ISWCs are added to generate the final version of the setlist.

Optional Evaluation

As an optional step, the generated setlist by LyricSIS can be evaluated, if the ground truth setlist of the live concert is available. This evaluation generates a final setlist containing the results of the setlist identification process, including the classification of each song as TP, FP, the TPP, FPP, and runtimes, so that accuracy and speed of the system can be evaluated. Additionally, if genre and quality of the concerts are available, the evaluation visualizes the evaluation results on a genre and quality level.

In summary, this chapter presents a detailed system architecture design for the lyric-based setlist identification system, covering research supported decision making of designing the architecture that consists of various components with their roles in fulfilling the functional and non-functional requirements.

3.4 Components Design

In this section, I will delve into the various components that form the architecture of the lyric-based setlist identification system that meets the functional and non-functional requirements specified earlier. As discussed in the the precious Section 3.3, the system consist of several components: Music source separation, speech enhancement, voice activity detection, speech-to-text transcription, the song search module, and an additional evaluation module. In addition to these core steps, I have developed different methods for side-tasks like Shazam-based setlist identification and evaluation, and a method to compile a final evaluation result for a dataset containing multiple concerts for both, LyricSIS and Shazam. The evaluation module also covers a method to create the graphs for the evaluation. Moreover, it incorporates several utility functions, such as one that creates the directory structure for analysis based on the source Excel workbook, one that breaks an audio file into smaller evenly-sized pieces, or a basic string cleaning method necessary for string effective matching, to name just a few. To maintain the report's brevity, I don't delve into the specifics of these within this document.

The first four components constitute the ALT module, which turns the concert audio file into AI Generated Lyrics (AIGL). The ALT module has been designed for robustness and efficiency, accommodating different languages and varying audio qualities while ensuring high accuracy and reasonable speed.

3.4.1 Music Source Separation

The task of music source separation addresses separating individual sound sources, from a mixture of sounds. In terms of this project that means that the music source separation component aims to isolate the vocals track from the complex mixture of sounds in a polyphonic audio recording, resulting in a monophonic signal containing only the vocals.

Various techniques exist for vocal extraction and music source separation, each with different speed and accuracy characteristics. In recent years, AI-based methods have made substantial advancements in this field. Typically, these methods involve analysing different song representations, such as waveforms in the time domain or spectrograms in the frequency domain. The latter is the most common approach used in this field [19], [20], [21].

One such frequency domain model is Deezer's Spleeter [21]. This model is fast but not as accurate compared to state-of-the-art solutions. The Band-Split RNN, developed by Luo in 2022, represents some of the most advanced results in this field [22].

Recently, there has been a focus on time-domain systems [23]. An example of this is Facebook AI Research's Demucs, which achieved state-of-the-art results upon its release [24]. Impressively, it was the first model in the time domain to surpass the results of systems operating in the frequency domain at the time of its release.

Recent developments have centered around hybrid models that merge both time-domain and frequency-domain approaches. An example of this is the Hybrid Demucs from Meta AI [25], and its transformer-based variant, Hybrid Transformer Demucs [26]. The incorporation of transformer layers ensures longer time-wise segments are captured to enhance the results. This model, alongside the Band-split RNN, has taken over the state-of-the-art benchmarks, depending on the source [22].

While the Band-split RNN slightly outperforms the Hybrid Transformer Demucs model in terms of vocal accuracy, it is not publicly accessible. Therefore, I chose to include the publicly available Hybrid Transformer Demucs model in this project as a potential component for source separation, given its highly accurate results.

I evaluated various approaches and ultimately selected the most appropriate method based on a balance between speed and sufficient accuracy to ensure high overall setlist identification precision. The methods I opted to test are both robust and efficient, guaranteeing superior vocal extraction for the next processing stages. The function of the component remains constant; only the AI model utilized for audio separation varies among spleeter, htdemucs, and htdemucs_ft. While htdemucs_ft yields highly accurate results, the spleeter model, though less precise, tends to be much quicker. To avoid OOM errors, the live concert audio for analysis is divided into smaller audio chunks prior to vocal extraction. The source separation takes place chunk by chunk, and the extracted vocals are likewise stored, ready for the subsequent pre-processing stage: speech enhancement.

3.4.2 Speech Enhancement

Speech enhancement involves the improvement of speech signals' quality and intelligibility, which might have been compromised due to background noise or other elements. In the context of this project, this could be residual artefacts from other sources in the vocals post extraction. The primary goal here is to diminish background noise while minimizing distortion, thereby enhancing the quality of the vocals.

There is an array of speech enhancement techniques, encompassing traditional statistical model-based methods and deep learning approaches.

Traditional techniques often involve statistical model-based speech enhancement methods that utilize estimators such as Wiener filters, Minimum Mean Square Error (MMSE) short-time spectral amplitude [27], and MMSE log-spectral amplitude [28]. While these have shown decent performance in estimating the clean speech signal, they require a priori and a posteriori signal-to-noise ratio estimates [29].

Deep learning methods, which have been increasingly studied for speech enhancement, have shown significant performance improvements [30]. These methods usually estimate target masks from noisy speech and apply them to noisy spectra.

One approach worth mentioning involves the use of the Demucs model, initially designed for music source separation by Facebook AI Research, as discussed in the previous section. This model was adapted into a causal speech enhancer that can handle real-time processing on consumer-level CPUs, achieving state-of-the-art results at its release.

Later studies have fused deep learning methods with statistical frameworks, such as Deep Xi and iDeepMMSE [31], [29]. These models leverage both traditional statistics and deep learning approaches to estimate the speech power spectral density, speech presence probability, and the a priori signal-to-noise ratio, thereby estimating the clean speech signal. The iDeepMMSE model achieves new state-of-the-art measures, slightly outperforming the Demucs model.

In summary, I chose to use the Demucs model for the speech enhancement task [17]. This model provides both speed and near state-of-the-art accuracy. Utilizing the same model for both source separation and speech enhancement ensures processing consistency, simplifies the pipeline and its maintenance, and offers open-source accessibility via the denoiser python package. The package includes an easy method for model fine-tuning, making it adaptable for potential future enhancements, such as shifting from the speech to the singing domain. To prevent memory exhaustion during the speech enhancement, this process is applied chunk-wise to each of the vocal chunks and merged back together at the end. The chunk-wise, speech-enhanced vocals are not stored since the following pipeline step, VAD, can handle long audio files without running into out-of-memory errors.

3.4.3 Voice Activity Detection

The VAD component serves to identify and eliminate silence or background noise from the enhanced vocals. A typical VAD system involves feature extraction followed by classification. Acoustic features like zero crossing rate, pitch, and signal energy, along with features extracted through deep learning methods, are employed. This is then followed by classification using methods such as Gaussian mixture models, Hidden Markov models, or other machine learning techniques [32].

In this project, the VAD component takes speech-enhanced vocals as input and produces a silence-edited version, distilling the signal to its essential parts. This is achieved by classifying sections with vocals and those without. These sections are defined by start and end timestamps for the vocal parts, which are then extracted and pieced back together. An effective VAD model should not be overly stringent, which could lead to the removal of vital vocal elements, nor too lax, risking the inclusion of noise [32]. The Silero-VAD meets these requirements, recognized for its enterprise-level speed and accuracy. Currently, there are hardly any high-quality, modern, open-source, and publicly available VAD systems exist, apart from the WebRTC VAD, which has many false positives, as illustrated in Figure 3.2 [33]. Consequently, due to its fast and accurate performance as well as the lack of competitive alternatives, I selected Silero-VAD for this project.

However, I observed that the output signal seemed to have the start and end of each vocal passage slightly cut. Therefore, I introduced an extra parameter representing the number of milliseconds to add to the end of each timestamp and to subtract from each starting timestamp. After some manual experimentation with this parameter, I discovered that it occasionally led to marginally improved overall results. These improvements were minimal enough for me to set this parameter to zero by default and not to include it as a selectable parameter in the design of experiments. Nonetheless, I retained it for potential future investigations.

3.4.4 Speech-To-Text Transcription

Speech-to-text research has seen significant progress in recent years, primarily due to advancements in artificial intelligence. Notably, OpenAI's Whisper model is a state-of-the-art speech transcription system, showing excellent accuracy across different languages, dialects, and acoustic environments [15].

Prior state-of-the-art research in the speech-to-text task includes XLS-R and mSLAM, both contributing substantially to the field [34], [35].

However, the Whisper model has exceeded their accuracy, as demonstrated in a research paper comparing Whisper with prior work on multilingual speech recognition. The exceptional performance of the Whisper model marks a significant milestone in speech-to-text research, setting a new standard for future advancements in the field.

The speech-to-text transcription component transcribes the pre-processed vocals into AIGL and stores it in a text file. As mentioned in previous sections, I employ OpenAI's Whisper model due to its state-of-the-art results, and its robustness and efficiency in various languages and accents. The hypothesis is that this robustness might extend to the singing domain.

In using the Whisper model, apart from the primary goal of transcribing lyrics for setlist identification, I aim to contribute to ALT research by investigating the potential of OpenAI's Whisper model for the ALT process. This will involve assessing the ALT module's accuracy in transcribing lyrics from the pre-processed vocals and determining its suitability for ALT. This project's exploration of the state-of-the-art Whisper model for the ALT task could advance the research field and provide insights into the model's effectiveness in handling the challenges associated with ALT across multiple languages.

3.4.5 Lyric-based Song Search Module

In this phase, I use the AI Generated Lyrics (AIGL) to identify the songs performed at the live event. The system interfaces with the comprehensive lyrics database of Genius.com via their API, extracting songs whose lyrics potentially match the AIGL based on a lyric chunk query. To evaluate the similarity between the lyrics of potential songs and the AIGL, I use two metrics: the Word Information Preserved (WIP) and the Word Error Rate (WER). Both metrics are designed to assess the effectiveness of an automated speech recognition system, or in this case, an automatic lyric transcription system, making them suitable measures for this task.

The WIP can be interpreted as the proportion of word information communicated within the transcription compared to the reference [36]. It calculates the sum of two ratios when comparing a reference word sequence with a predicted one. The first ratio is between the correctly predicted words, denoted as C, and the number of words in the reference, N. The second ratio is also related to the correctly predicted words C, but here they are compared to the number of words in the predicted sequence, P. A higher WIP value indicates a more accurate transcription, with the ideal score being a WIP of 1. The WIP is computed according to Formula 3.1.

The WER measures the cost of restoring the predicted word sequence to the reference sequence. It's defined as the ratio of the total number of insertions I, substitutions S, and deletions D applied to the predicted sequence that are necessary to restore the reference sequence, to the total number of words in the reference N, as indicated in Formula 3.2. Thus, lower values indicate a more precise transcription than higher values, with 0 being the perfect transcription. WER is a more standard and widely used performance measure for evaluating automatic speech recognition systems [37], making it a suitable metric for assessing the performance of the ALT module and the comparison between AIGL and actual song lyrics.

$$WIP = \frac{C}{N} + \frac{C}{P} \tag{3.1}$$

$$WER = \frac{S + D + I}{N} \tag{3.2}$$

In this step, I align the lyrics found in the database with a strategically selected chunk of the AIGL by minimizing the WER across all possible alignment positions of the lyrics within the AIGL chunk. This process helps identify the most likely matches along with their corresponding metadata, such as artist and song names.

Utilizing this metadata, I perform a query on the MusicBrainz database to obtain the International Standard Musical Work Code (ISWC), if available. The matches I identify form the basis for generating the setlist. MusicBrainz was chosen as the ISWC source because it is the only freely accessible public database for music metadata that includes ISWC.

Another method to obtain ISWC based on the artist name and song title is to use the web portals of collection societies that have registered the songs. However, as these societies are distributed globally, a more reliable and comprehensive alternative source of work metadata is the International Confederation of Societies of Authors and Composers (CISAC). CISAC maintains a database known as the ISWC Network, which holds metadata from numerous global collection societies. However, access to their API for bulk requests is not free, making it unsuitable within the scope of this project, given the project requirements.

3.4.6 Evaluation Module

When the ground truth setlist is available, for example, in a CSV file with the columns 'Artist' and 'Song Title,' the identified setlist can be evaluated. This stage involves comparing the CSV file containing the predicted setlist with the ground truth setlist based on title matching. The output is a consolidated final result CSV file, classifying identified songs as True Positives, False

Positives, and False Negatives. It also includes columns indicating the system's confidence in identifying songs, which is informed by the calculated string similarity.

This final setlist is user-friendly, detailing the specifics, metadata, and identification confidence level for each song. The classification aids in evaluating system accuracy and performance, providing valuable insights for future system improvements. Additionally, it includes an aggregation of these results across all concerts in a dataset, stored in a final evaluation CSV file. When evaluating multiple concerts, these files can be easily merged and aggregated across concerts, facilitating evaluations of entire datasets.

3.4.7 Shazam Analysis

In accordance with prior statements, Shazam, an industry-standard solution for song identification, has been chosen for setlist identification and serves as a benchmark for this lyrics-based system. As a result, two functions have been implemented: one function analyzes the concert's MP3 files to generate the setlist, and another evaluates the resulting setlist when ground-truth setlists are available. Similar to LyricSIS, the evaluation results include TPP, FPP, and runtime for each concert. Furthermore, an aggregated version of these results is calculated for multiple concerts, such as the test dataset, and is available for benchmarking on datasets.

3.5 Implementation Details

This chapter presents a comprehensive discussion of the implementation details of the software product. It starts with an overview of the software tools and technologies utilized, then proceeds to outline the overall structure of the code and data, before providing an in-depth analysis of each component's implementation. To ensure alignment with the functional and non-functional requirements outlined earlier, several key considerations have been taken into account during implementation, which are also elaborated in this chapter.

3.5.1 Software Tools and Technologies

The software was exclusively developed using Python, version 3.8. A variety of different libraries and third party packages have been deployed, which are shown in Table 3.1. This table provides information about the package, including its name, version, and the specific purpose it plays in this project. In line with the project's requirements, I have utilized pre-trained open source models and libraries to ensure cost-effectiveness and ease future development. The Whisper model was selected to ensure robustness and versatility across various languages.

Table 3.1: Python libraries and third party packages used for this project

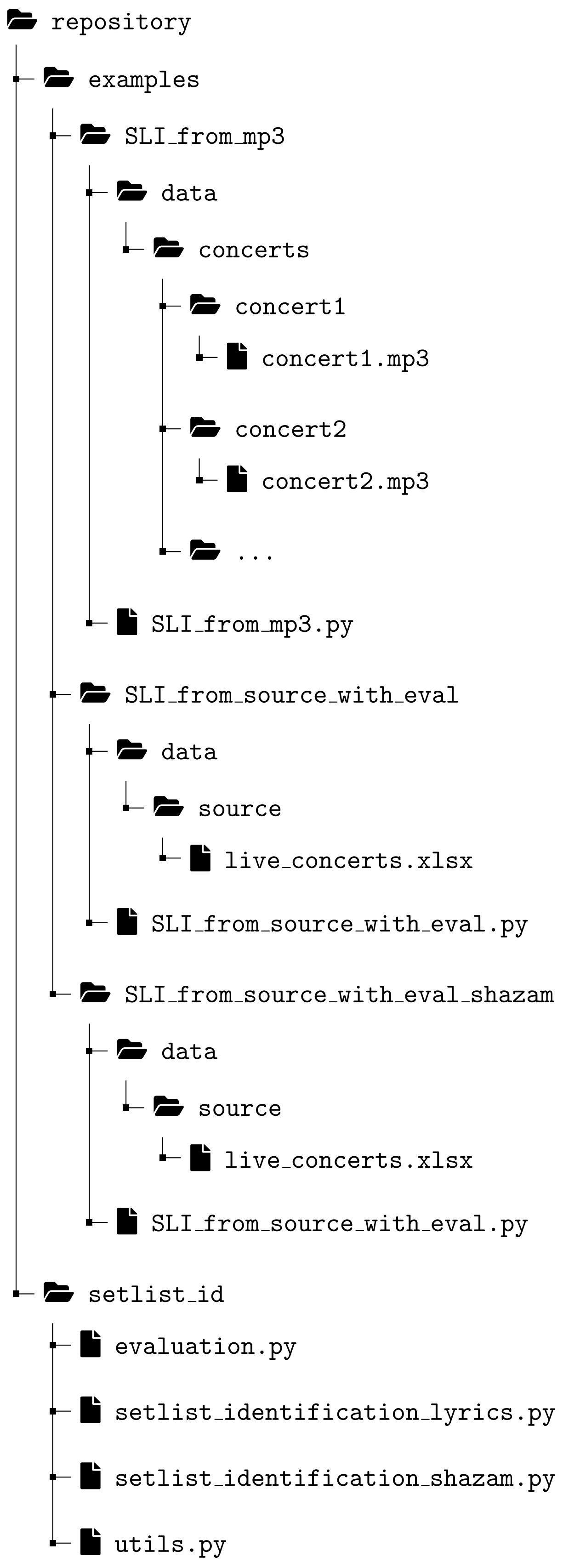
Name	Version	Purpose
demucs	4.0.0	music source separation
spleeter	2.3.2	music source separation
denoiser	0.1.5	speech enhancement
whisper	github.com/openai/whisper	transcribe audio
lyricsgenius	3.0.1	access Genius' database
musicbrainzngs	0.7.1	access MusicBrainz' database
ShazamAPI	0.0.2	access Shazam
torch	1.13.0	demucs, denoiser, and silero-vad models
tensorflow	2.11.0	spleeter model
jiwer	2.5.1	WER and WIP calculation
difflib	python-included	sequence comparison for title matching
numpy	1.22.4	math operations
pandas	1.5.2	data handling
soundfile	0.11.0	read and write audio files
pydub	0.25.1	read, write and manipulate audio files
ffmpeg-python	0.2.0	audio handling backend

3.5.2 Overall Structure and Code Organization

The software product is packaged as a GitHub repository. Figure 3.3 illustrates the repository's overall structure. The core code is found within the Python files in the setlist_id directory. It includes separate Python files for the lyrics-based SLI, the Shazam-based SLI, the evaluation,which covers both lyric- and Shazam-based SLIs, and another file with various utility functions.

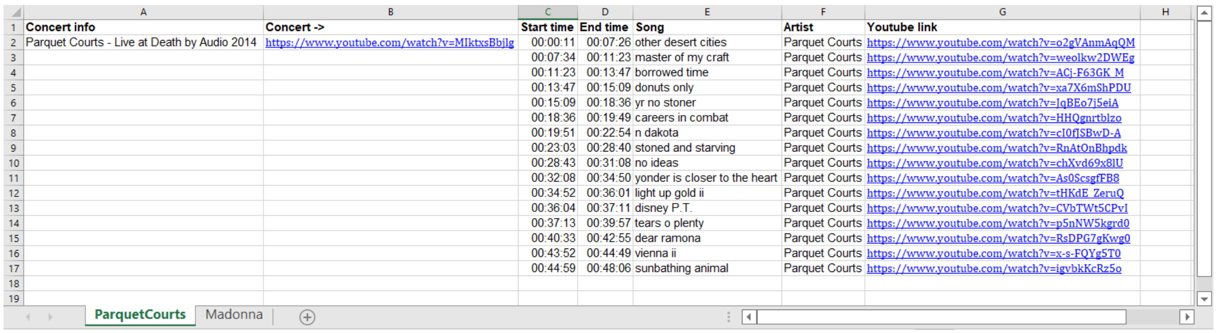
The examples directory contains examples of three distinct use cases of the software. (The SLI_from_mp3 example covers the use case, when a setlist shall be created for one or multiple concerts based on the mp3 files. The SLI_from_source_with_eval covers the use case, when the actual ground truth setlist for a video on youtube and it's URL is known.)

The live_concerts.xlsx file (shown in Figure 3.4) contains information on the Parquet Courts and Madonna concerts. Each concert has its own sheet, the concert's YouTube URL must be on B2, and columns for 'Song' and 'Artist' must contain the concert's song list and respective artist names. The remaining columns, 'Start time', 'End time', 'Youtube link', are not used in this project and can be ignored or deleted. Note that the 'Youtube link' column refers to the links for individual songs, not the entire concert link. (The additional information is there, because the dataset used in this project is build upon the ASID dataset from Yesiler, who included this information. The structure is aquivilent to the one from Yesiler who created the ASID dataset, which lays the foundation for the dataset used in this project.)



```
repository
├─ examples
│  ├─ SLI_from_mp3
│  │  ├─ data
│  │  │  └─ concerts
│  │  │     ├─ concert1
│  │  │     │  └─ concert1.mp3
│  │  │     ├─ concert2
│  │  │     │  └─ concert2.mp3
│  │  │     └─ ...
│  │  └─ SLI_from_mp3.py
│  ├─ SLI_from_source_with_eval
│  │  ├─ data
│  │  │  └─ source
│  │  │     └─ live_concerts.xlsx
│  │  └─ SLI_from_source_with_eval.py
│  └─ SLI_from_source_with_eval_shazam
│     ├─ data
│     │  └─ source
│     │     └─ live_concerts.xlsx
│     └─ SLI_from_source_with_eval.py
└─ setlist_id
   ├─ evaluation.py
   ├─ setlist_identification_lyrics.py
   ├─ setlist_identification_shazam.py
   └─ utils.py
```

Figure 3.3: Directory tree of the repository



	A	B	C	D	E	F	G	H
1	Concert info	Concert ->	Start time	End time	Song	Artist	Youtube link	
2	Parquet Courts - Live at Death by Audio 2014	https://www.youtube.com/watch?v=MIktxsBbjlg	00:00:11	00:07:26	other desert cities	Parquet Courts	https://www.youtube.com/watch?v=o2gVAnmAqQM	
3			00:07:34	00:11:23	master of my craft	Parquet Courts	https://www.youtube.com/watch?v=weolkw2DWEg	
4			00:11:23	00:13:47	borrowed time	Parquet Courts	https://www.youtube.com/watch?v=ACj-F63GK_M	
5			00:13:47	00:15:09	donuts only	Parquet Courts	https://www.youtube.com/watch?v=xa7X6mShPDU	
6			00:15:09	00:18:36	yr no stoner	Parquet Courts	https://www.youtube.com/watch?v=JqBEo7j5eiA	
7			00:18:36	00:19:49	careers in combat	Parquet Courts	https://www.youtube.com/watch?v=HHQgnrtblzo	
8			00:19:51	00:22:54	n dakota	Parquet Courts	https://www.youtube.com/watch?v=cI0fJSBwD-A	
9			00:23:03	00:28:40	stoned and starving	Parquet Courts	https://www.youtube.com/watch?v=RnAtOnBhpdk	
10			00:28:43	00:31:08	no ideas	Parquet Courts	https://www.youtube.com/watch?v=chXvd69x8lU	
11			00:32:08	00:34:50	yonder is closer to the heart	Parquet Courts	https://www.youtube.com/watch?v=As0ScsgfFB8	
12			00:34:52	00:36:01	light up gold ii	Parquet Courts	https://www.youtube.com/watch?v=tHKdE_ZeruQ	
13			00:36:04	00:37:11	disney P.T.	Parquet Courts	https://www.youtube.com/watch?v=CVbTWt5CPvI	
14			00:37:13	00:39:57	tears o plenty	Parquet Courts	https://www.youtube.com/watch?v=p5nNW5kgrd0	
15			00:40:33	00:42:55	dear ramona	Parquet Courts	https://www.youtube.com/watch?v=RsDPG7gKwg0	
16			00:43:52	00:44:49	vienna ii	Parquet Courts	https://www.youtube.com/watch?v=x-s-FQYg5T0	
17			00:44:59	00:48:06	sunbathing animal	Parquet Courts	https://www.youtube.com/watch?v=igvbkKcRz5o	
18								
19								

Figure 3.4: Structure of the Excel file containing the concert information, including one sheet for one concert, for each sheet the Youtube-URL on B2, a column called *Song* that has the ground truth of the played songs in the concert, as well as a column called *Artist* containing the artist name for each song, and the general concert information on A2.

The process begins with an Excel file from which a directory named *concerts* is generated. This directory contains individual subdirectories for each concert, each holding a corresponding CSV file (**_concert.csv*) that represents the respective concert's details extracted from the original Excel file. The YouTube link embedded within the CSV file is used to download the audio from the respective YouTube video in .mp3 format, which is temporarily stored within the concert directory.

It is important to note that all audio files are stored only temporarily during the analysis phase, and they are subsequently deleted to avoid any potential copyright issues during this project. The .mp3 file forms the basis for the execution of the entire pipeline to generate the lyrics-based setlist, followed by the evaluation of the results.

For comparative purposes, in the final example, SLI_from_source_with_eval_shazam, the Shazam-based SLI is applied. The process is similar to the previous one, with the main difference being that Shazam is utilized to create the setlist.

Figure 3.5 presents an illustrative overview of the concert directory and its file structure. This snapshot was captured after the execution of LyricSIS's complete pipeline, including the evaluation process, on a given concert directory.

Within each concert directory, there exists an Excel file which contains a sheet extracted from the source Excel file, representing the specific concert. Accompanying this are various text files associated with each stage of the pipeline - source separation, speech enhancement, voice activity detection, transcription, and lyrics-based song search. Each of these files records the runtime of its corresponding step.

The transcribed lyrics are stored in the AI_transcription.txt file. Meanwhile, the end results of the lyrics-based SLI are held in setlist_final.csv. This is a table comprising multiple columns with the artist name, song title (detailing the songs that have been identified by the system), the WIP and WER scores, which reflect the confidence level in the song identification, and the

```
concert1
├─ concert1_concert.csv
├─ runtime_source_separation.txt
├─ runtime_speech_enhancer.txt
├─ runtime_vad.txt
├─ runtime_transcription.txt
├─ AI_transcription.txt
├─ runtime_song_search.txt
├─ setlist_final.csv
├─ setlist_alignment_data.json
├─ all_potential_hits_with_wip_and_maybe_wer.csv
├─ all_analyzed_songs_with_lyrics.pkl
├─ class_config.json
├─ eval_concert_performance_table.csv
└─ eval_concert_results_table.csv
```

Figure 3.5: Directory tree of a post-analysis and post-evaluation concert directory

information from the ISWC fetching process, which is the MusicBrainz-ID (MB_ID), the ISRC, and the related ISWC.

The file setlist_alignment_data.json contains supplementary alignment scores, providing insight into the algorithm's capability to identify the optimal alignment score of a matched song within the overall lyrics, thereby determining the song's position in the concert.

Two additional files, all_potential_hits_with_wip_and_maybe_wer.csv and all_analyzed_songs_with_lyrics.pkl, provide information about the potential songs that have been analyzed. The former stores artist names, song titles, WIP, and, if the WIP threshold is surpassed, the WER. The latter file contains the artist name, song title, and lyrics.

The class_config.json file stores all attributes of the class object, thus ensuring that the SLI settings are retained and accessible during the evaluation process.

The final two CSV files store the results of the evaluation. The eval_concert_results_table.csv file lists all the songs played during the concert and those that were identified as a match. This file provides a clear link between the detected song and the actual song that was played, also detailing the WIP, WER, and SIM values to represent the certainty of matching. The SIM value denotes the title similarity indicative of a match. Furthermore, it specifies the type of match for each song, which can be a TP, FP, or False Negative (FN). Additional information for each of the TPs or FPs is of obviously the MB_ID, the ISRC, and the ISWC.

The second evaluation file, eval_concert_performance_table.csv, is a single-row table that contains more comprehensive information about the SLI's performance for this specific concert. This information includes the following:

- Artist - Name of the performing artist(s)
- Genre - Musical genre performed by the artist
- Quality - Audio quality of the concert
- SongsPerformedCount - Number of songs performed
- TruePositivesCount - Number of matches of type TruePositive
- FalseNegativesCount - Number of matches of type FalseNegatives
- FalsePositivesCount - Number of matches of type FalsePositives
- TruePositives[%] - Percentage of matches of type TruePositive relative to number of songs performed
- FalseNegatives[%] - Percentage of matches of type FalseNegatives relative to number of songs performed
- FalsePositives[%] - Percentage of matches of type FalsePositives relative to number of songs performed

- TotalRuntimePerAudioLength - Sum of ratios between runtime and the concert audio length for each step in the pipeline
- `<step>`RuntimePerAudioLength - A column for each step of the pipeline with ratio between runtime and the concert audio length for each step
- AudioLength[min] - Concert audio length
- TitleMatchingSimilarity - Similarity threshold for the title matching between the list of performed songs and the list of identified songs
- `<parameter setting>` - A column for every information about the parameter settings used for the SLI process during analyzis.
- ISWCPercentage - Percentage of found ISWC codes relative to the songs identified (TP+FP)
- TransFailurePercentage - Percentage of audio chunks with failed transcription relative to the total number of audio chunks

In scenarios where multiple concerts are being analyzed, such as an entire dataset of concerts, these tables can be seamlessly merged to form a consolidated table. This table retains the same columns, with each row corresponding to a specific concert. This simplifies the process of acquiring a high-level perspective of the system's performance across the entire datasets, by facilitating further aggregation of these values.

3.5.3 Implementation details of each component

The design of this project follows an object-oriented programming approach. Core Python files contain classes necessary for conducting both lyrics-based and Shazam-based SLI tasks, as well as for evaluation. This section will primarily focus on the class for the lyrics-based SLI implementation and evaluation, as it forms the crux of this work. The Shazam-based SLI and evaluation will be briefly explained towards the end of this chapter.

The SLI class is designed to analyze one concert at a time. Therefore, for every concert, a new class object should be created. Since every component is implemented as a class method, which accesses class attributes, a list of the parameters that need to be specified by the user when creating a class object is necessary. These parameters, shown in Table 3.2, will then, along with others, become class attributes.

Music Source Separation

The method source_separation() is the key implementation for source separation in this system. At the beginning of the method, user-defined options are given to the method by accessing the class attributes, which include the source separation technique to be used. Based on this, either the spleeter or demucs method gets called. They work the same way except for the different

Table 3.2: Parameters for lyrics-based setlist identification class

Name	**Meaning**	**Type**	**Default Value**
output_dir	Path to output directory	Path	NO DEFAULT
path_to_concert_mp3	Path to concert mp3 file to be analyzed	Path	NO DEFAULT
to_exclude	strings in song titles to be excluded from genius.com response	list of str	None
source_separation_technique	technique for source separation	str	'hudemucs'
speech_enhancement_technique	technique for speech enhancement	str	'denoiser'
vad_technique	technique for voice activity detection	str	'silero'
wip_threshold	song identification threshold of WIP value	float	0.195
wer_threshold	song identification threshold of WER value	float	0.89
nsongs_per_request	number of songs proposed by genius.com per API request	int	5
transcription_model_size	size of Whisper model	str	'medium'
genius_api_token	API key to access genius.com	str	'XXXXXXX'
read_timeout	seconds to wait for genius.com response	int	20
sleep_time	Seconds to wait between genius.com API requests	float	0.3
nwords_per_request	number of words per genius.com API request	int	6
no_speech_threshold	threshold of the probability for non-speech by whispers internal VAD	float	0.6
condition_on_previous_text	previous transcribed 30sec window influences transcription of next window	boolean	True
vad_smoothing_seconds	smoothen out the Silero VAD detected cut-off timestamps by +- seconds	float	0.0
language_detection_segments	number of segments of a mp3 file analyzed to identify language	int	20

model type, so let me explain based on the demucs: Acceptable audio file formats for processing as MP3 file and output preferences for audio files as WAV file are hardcoded within this function and could easily be made available as user-defined inputs as class parameters. In this particular method, either the Hybrid Transformer Demucs model ('ht_demucs'), or the fine-tuned Hybrid Transformer Demucs model ('ht_demucs_ft') are available for this source separation method, called _source_separation_demucs().

The source separation process initiates by trying to split the input concert audio into smaller chunks that are managable without running into OOM issues. These are stored in a directory labeled 'raw_audio_chunks'. If the splitting is not possible during this stage, a corresponding error message is displayed and the process is stopped.

Next, a directory named 'vocal_chunks' is created (if it doesn't already exist) to store the outputs of the separation process. The Demucs model is loaded and moved to the GPU if one is available, enhancing the speed and efficiency of the enhancement process. If the concert audio has not already been split and processed, the function loops through each audio chunk and separates the vocals using the Demucs model.

The Demucs separation process is achieved by executing an operating system command that calls the demucs.separate function, which produces an audio file containing the isolated vocals of the audio chunk. The function includes internal checks to verify the validity of the audio files being processed and the success of each separation command. Following the separation, the individual vocal chunks are then recombined to form a single, continuous vocal audio file.

Upon completion of the source separation process, the function records the total runtime and stores it in a text file for future reference. In situations where the concert has already been processed, the function skips and alerts the user with a information message.

Finally, the method calls _class_to_dict() to write the class configurations to a JSON file. This ensures that the settings of the SLI task are preserved and can be accessed later, such as during evaluation. This method will be called after completion of every following step to update the configuration file in case a subsequential step is completed.

Speech Enhancement

The method speech_enhancer() is the primary function responsible for the enhancement of speech or vocals within our audio dataset. The method involves pre-processing the separated vocal tracks using a pre-trained Demucs model for speech enhancement.

The paths to the previously separated vocal chunks are identified at the beginning of this function. The output of the speech enhancement process is stored as 'vocals_SE.wav', and the runtime information is saved to 'speech_enhancer.txt'.

The speech enhancement process begins with loading the pretrained Demucs model. This model is moved to the GPU if one is available, enhancing the speed and efficiency of the enhancement process.

Afterward, the method iterates through the directories containing the vocal chunks, enhancing each chunk in turn. For each chunk, the method loads the audio file, adjusts its sample rate and shape to match the requirements of the Demucs model, and passes it through the model for denoising.

The enhanced audio is then converted back to the original format and saved. The method also merges the processed chunks back into one continuous audio file. This operation follows the same order as the initial splitting of the audio to ensure the integrity of the audio track is maintained. Throughout the process, memory management is handled by calls to the Python garbage collector and clearing the PyTorch cache to prevent memory overflow.

Upon completion, the function calculates the total runtime of the enhancement process and records this in a text file 'runtime_speech_enhancement.txt'

In cases where this output file already exists, the function avoids repeating the enhancement process, saving computational resources. It then clears any remaining memory allocations, ensuring optimal resource usage for subsequent operations.

The implementation of the speech enhancement method is optimized for efficiency and accuracy, taking full advantage of advanced deep learning models and Python's memory management capabilities. The robust design of this method allows it to effectively handle large audio datasets, making it an essential component of the SLI process.

Voice Activity Detection

Voice Activity Detection is designed to isolate vocal segments from prolonged silence. This process relies on the open source Silero-VAD model, widely acknowledged for its speed, precision, and enterprise-level performance.

The primary method, voice_activity_detection(), serves as the key implementation of this process. This method initiates by defining the paths for input and output audio files. The input is the speech-enhanced vocal audio file, named vocals_SE.wav, and the output is the silence-cut speech-enhanced vocal audio file, dubbed vocals_SE_VAD.wav, as well as stored as audio chunks in a subdirectory created during the VAD process.

To ensure that an already analyzed concert is not analyzed twice, a check for the presence of a 'voice_activity_detection.txt' file is made. If this file exists, it signals that the silence has already been cut from the song and the function concludes its operation.

Upon the nonexistence of the 'voice_activity_detection.txt' file, the VAD process starts. The Silero-VAD model is loaded using torch library, providing the necessary utilities for the task.

These utilities include get_speech_timestamps for identifying vocal timestamps, read_audio for loading vocal audio, save_audio for storing the resultant audio, and VADIterator and collect_chunks for managing audio chunks.

The audio is loaded and speech timestamps are identified within the full audio file. Notably, the model's results are adjusted to either side of the identified speech chunks, accounting for an over-aggressiveness observed in the model, if desired by the user. This adjustment is done by a constant value defined by vad_smoothing_seconds, and it ensures no significant vocal elements are lost during the silence removal process.

The adjusted speech chunks are merged into a single audio file, which is then stored as vocals_SE_VAD.wav. Subsequently, the audio file is split into equal-sized chunks of one minute each and saved in the VAD_audio_chunks directory.

Performance metrics, including the audio length after VAD and the function's runtime, are captured and saved in the 'voice_activity_detection.txt' file. This action serves the dual purpose of documenting the process metrics and indicating that the VAD process has been performed on the specific audio file.

Finally, memory housekeeping is performed by clearing redundant memory allocations, allowing for efficient resource utilization in subsequent iterations.

Speech-To-Text Transcription

The overall process is encapsulated within the song_transcriber() method. It first checks if the transcription of the audio file already exists in the specified output directory. If it does, it skips the transcription process; else, it proceeds to transcribe the audio.

The audio files to be transcribed are sorted in a natural order, crucial when transcribing a sequence of audio files that together form a complete song. This process is followed by the transcription of each individual audio file. For each file, a new instance of the Whisper model is loaded to ensure the resetting of the model's memory, thus preventing the repetition of audio loops.

One unique feature of the transcription process is the language identification which is accomplished using the Whisper model. Initially, the Whisper model would analyse the first 30 seconds of the audio to identify the language. However, this method posed a challenge for live concerts which sometimes extend for hours and might begin with an introduction in a different language from the one the band performs in. This can lead to transcription errors as the Whisper model would assume the language of the entire audio based on the first 30 seconds.

To mitigate this, the language identification process was modified to make it more robust. The Whisper model was adjusted to analyse 20 evenly distributed 30-second windows across the entire audio, choosing the language that appeared most frequently. Furthermore, the concert was split into smaller audio chunks for transcription, allowing for a more reliable transcription even when the language of performance varied from song to song within a concert. This measure also served to prevent the Whisper model from getting stuck in a repetitive loop or hallucinate, where it repeats the same transcription over and over or outputs a transcript entirely unrelated to the actual audio, respectively. [15]

Post transcription, the transcribed text is saved in chunks corresponding to each audio file, ensuring that transcriptions are preserved even in case of interruptions. Furthermore, the runtime taken for the transcription process, and the final class configurations are stored for reference and documentation.

Additionally, the implemented audio transcription method allows for the tuning of the model size based on the user's requirements, enabling a trade-off between performance and accuracy. A smaller model would facilitate faster transcription but compromise on the accuracy of the outcome, and vice versa.. This flexibility empowers the user to balance between performance and accuracy based on their specific needs.

Lyrics-based Song Search Module

The song search module is designed to efficiently parse through a concert's entire AIGL. The algorithm's crux is an elaborate series of operations involving song lyrics request, sequence comparison, and sequence alignment computations, ultimately resulting in a detailed setlist of songs performed during the concert.

Initially, the algorithm uses the AI-generated concert lyrics, from which it creates n-word chunks. Each chunk is then used to request potential matching songs and their respective lyrics from Genius.com using their API service. This process results in a list of potential songs that share some similarity with the given n-word chunks, and therefore concert's lyrics.

To ascertain the relevance of these potential matches, the algorithm undertakes a comparison of the potential song lyrics against a specific part of the AI-generated concert lyrics. The selected part from the concert's lyrics for this comparison is not arbitrary; it is strategically chosen based on the length of the potential song's lyrics. The starting position of this comparison segment in the concert lyrics is determined by subtracting the word length of the potential song from the position of the lyric chunk used for the song request. Simultaneously, the ending position is computed by adding the word length of the potential song to the position of the same lyric chunk. The rationale behind this is, that the n-word chunk could be the inital or the final n-words from the lyrics of potential songs.

The process of identifying a song based on lyrics involves a two-stage string comparison between the lyrics of the potential song and a selected part of the AI-generated lyrics (AIGL).

In the first stage of the song identification process, I calculate the WIP for a potential song's lyrics and the reference AIGL using Formula 3.1 and compare it against a preset WIP threshold. If the computed WIP exceeds this threshold (indicating that the chunks go beyond a similarity threshold), it initiates the second stage of the identification process. This involves aligning the potential song's lyrics with the selected part of the AIGL. This stage essentially acts as a lyrics-based song segmentation, determining the song's location within the concert based on the best alignment position of the lyrics in the AIGL.

The second stage involves aligning these sequences: the lyrics of the potential song and the AIGL chunk. This step is primarily driven by the calculated WER, and its main goal is to locate the most probable position of the potential song within the concert's lyrics. This is based on the WER between the potential song's lyrics and the aligned segment of the concert lyrics.

I iteratively calculate the WER score for each sequence of the selected part of the AIGL that has the same number of words as the potential song's lyrics. The lowest WER from this iteration indicates the best alignment position of the potential song in the AIGL. The WER is a vital metric in establishing the accuracy of the song identifications. If the computed minimal WER for a potential song is below a certain WER threshold, the song is confirmed as a match. This means the song is presumed to have been performed during the concert and is added to the final setlist.

Based on the lowest WER, I specify the alignment of the lyrics of the identified song within the AIGL. The next n-word chunk for the next potential match is determined by the end of lyrics within the AIGL. This ensures that the lyrics of the just-identified song are not reused for finding potential matches. Hence, the algorithm identifies the concert's complete setlist based on AI-generated lyrics, using the WIP and WER parameters.

During the implementation phase, I manually experimented with three different metrics - WIP, WER, and TF-IDF, which Vaglio used in his lyrics-based song identification method [5]. These were experimented for both stages in the identification process. I found that using WIP for the first stage and WER for the second stage led to the highest TPP.

The final step of the song search module involves associating the identified songs with their corresponding ISWC. To find the ISWC for the identified songs, the algorithm uses the identified artist's name and song title to perform a match.

The add_ISWCs() function takes in a the final identified setlist as DataFrame, which contains the information about the identified songs. This DataFrame is then used to retrieve the ISWC codes for each identified song, if available. The function is designed to interact with the MusicBrainz database via the musicbrainzngs Python library, which provides a convenient way to request music metadata using the MusicBrainz API.

The function first initializes the MusicBrainz service with application details, setting a rate limit to prevent overloading the service with requests. This configuration is crucial to respect the terms of use of the MusicBrainz API and ensure sustainable usage.

Next, the function applies a series of operations on the DataFrame, using each song's artist name and title to query the MusicBrainz database for the corresponding MusicBrainz ID (MB_ID) and International Standard Recording Code (ISRC). This step is a necessary intermediate step to get the ISWC from the MusicBrainz database based on the MB_ID. These operations are encapsulated in the _get_mb_id_and_isrc() helper function, which performs the actual MusicBrainz database query, iterates over the returned results, and matches the artist and song title names to find the first corresponding ISRC code. A delay is induced before each query to respect the rate limit.

Once the MB_ID and ISRC are retrieved and appended to the DataFrame, the function proceeds to the next step, fetching the ISWC codes. The _get_iswc_from_mb_id() helper function takes the MB_ID as input and uses it to look up the recording in the MusicBrainz database, retrieving the related works including the ISWC. If the ISWC is found, it's returned and appended to the DataFrame.

In case no songs are identified (i.e., the DataFrame is empty), the function adds columns for MB_ID, ISRC, and ISWC, setting their values to 'None'. This ensures the DataFrame structure remains consistent for further evaluation processing, regardless of the ISWC fetching outcome.

This ISWC retrieval step is essential as it completes the setlist identification process, providing comprehensive metadata for each song, including the unique ISWC code. It ensures that the results can be used for further processing and copyright-related use-cases, such as accurate performance royalty distribution.

3.6 Datasets

Designing appropriate datasets is a pivotal aspect of this project. These datasets must be suitable for probing the system requirements. I will utilize four datasets for two distinct purposes throughout this project.

Two of the datasets will comprise multiple individual song audio files, alongside their corresponding lyrics. These will serve as a benchmark to evaluate the ALT module of the system against the current state-of-the-art ALT systems within the research domain.

The remaining two datasets, namely the development and test datasets, will contain live performance audio files and their corresponding ground truth setlists. These datasets will be used for developing and testing the entire system. I will use the development dataset (dev dataset) for refining the system and determining the optimal parameter settings. Given the myriad of experiments required during system development, this dataset will remain small and diverse to save time and ensure that the system meets the requirements during the development phase. A larger test dataset, consisting also of live performances and the corresponding setlists, will be used to evaluate the finalized system with its best settings, as determined from experiments on the dev dataset.

3.6.1 Automatic Lyric Transcription

ALT is a prominent research area. This field is dedicated to identifying optimal methods for transcribing lyrics from raw song audio, and several benchmark datasets exist for this purpose. These datasets allow researchers to assess their results and compare them with those of others. Three widely used benchmark datasets in the research community — Jamendo[38], Hansen [39], and Mauch[40] — serve as standards for comparing research results and competing with other researchers. In this project, I will employ two of these datasets, Jamendo and Hansen, for comparative analysis. Regrettably, I was unable to acquire the Mauch dataset, as it is not publicly available, and my multiple attempts to contact its creator, Mathias Mauch, remained unanswered.

The Jamendo dataset comprises 20 contemporary music recordings, which are released under Apache License 2.0, along with the lyrics of these recordings. The Hansen dataset contains 10 music recordings of widely recognized pop songs and their corresponding lyrics. I categorize the quality of songs for both datasets as high-quality audio.

3.6.2 Entire Pipeline

I designed the dev- and test datasets to facilitate efficient experimentation and testing against various characteristics and requirements within the project's timeframe. Both datasets are based on the ASIS dataset created by author Yesiler et al.[12]. Their work is unique as they assessed a cover version identification system for setlist identification use cases, and they created the ASID dataset as a benchmark for the setlist identification task. The ASID dataset forms the foundation for the dev- and test datasets used in the evaluation of this project's setlist identification system. Out of the 75 concerts in the ASIS dataset, I selected 10 for the dev dataset, and 41 for the test dataset. The audio for the remaining concerts couldn't be acquired or was corrupted, making analysis of these concerts impossible.

I based the selection of the dev dataset on multiple criteria. It is intended to run numerous experiments during the development phase to identify the best system parameters and to ensure robustness requirements are met. Therefore, this dataset had to be diverse in terms of quality, genre, and language. Unfortunately, the language diversity of concerts within the ASIS dataset is limited. Consequently, I acquired two additional concerts, one in Dutch and one in Polish, which are the languages of the countries where the company was conducting business and partnership discussions at the time of this project. Therefore, the dev dataset consists of 12 concerts. Figure 3.3 displays the number of concerts according to Audio Quality (AQ) and genre. The quality is grouped into three categories: 'AQ-A contains high-quality recordings, mainly coming from broadcast recordings or official releases. AQ-B contains professionally recorded concerts, mainly from small venues (in general, we observe that the mixing/mastering quality for concerts in AQ-B is inferior to the ones in AQ-A). Lastly, AQ-C contains smartphone or video camera recordings from varying-size venues/events.' [12]. Examples of the qualities A, B, and C are the concerts of the following artists:

- **Audio-Quality A:** J Balvin (https://www.youtube.com/watch?v=99KBfcFg7Vo)
- **Audio-Quality B:** Nirvana (https://www.youtube.com/watch?v=I1yTkY2bSfs)
- **Audio-Quality C:** Sobs (https://www.youtube.com/watch?v=qtJ7wjtipkY)

Key decisions for the dev dataset include:

- Increasing the diversity of concerts while maintaining the overall audio length of the dataset. Instead of full concerts, I used five to eight tracks (17 to 43 minutes in length), enabling more experimentation while gaining more insights into the system's robustness.
- Ensuring language robustness by including two additional concerts, one Dutch and one Polish, and ten English concerts to analyze influences disregarding language.
- Analyzing genres within the ten English concerts, which cover five genres (Pop, Rock, Indie, Rap, Electronic), with two concerts representing each genre.
- While creating such a small dataset covering different concert characteristics was challenging, an equal distribution of the three different qualities across the concerts was not feasible. Instead, the concerts only have a qualities within A and B classes, and the lowest quality class C is not represented.

The specifications for the test dataset, used only once at the end, are not as important in terms of size as those for the dev-set. Additionally, including more concerts improves the reliability of the statistical evaluation for specific subgroups of concerts, for example, with respect to genre. Therefore, I used all remaining concerts from the ASIS dataset, except those where the audio couldn't be acquired or was corrupted, or those where more than 15% of the songs performed during a concert were not found in the genius.com database. Unfortunately, there were quite a few of these, so the final test dataset comprises 41 concerts, represented by 54.5 hours of audio and 742 songs. The languages in the test dataset were English, Finnish, Spanish, and Italian, represented by 37, one, one, and one number of concerts, respectively. Figure 3.4 depicts the distribution across genre and quality.

Table 3.3: Number of concerts in development dataset per audio quality and genre

Genre	**AQ-A**	**AQ-B**	**AQ-C**	**Total**
Electronic	2	0	0	2
Indie/Alternative	1	1	0	2
Pop	3	0	0	3
Rap/Hip-Hop	1	2	0	3
Rock/Metal	1	1	0	2
Total	8	4	0	12

Table 3.4: Number of concerts in test dataset per audio quality and genre

Genre	**AQ-A**	**AQ-B**	**AQ-C**	**Total**
Electronic	2	0	0	2
Indie/Alternative	2	5	2	9
Pop	6	2	3	11
Rap/Hip-Hop	3	1	1	5
Rock/Metal	8	4	2	14
Total	20	13	8	41

4 Evaluation

This Chapter details the testing and evaluation of the system, following the requirements specified in previous sections. Here, I introduce the chosen evaluation metrics and delineate the strategy for testing the system. Additionally, this Chapter presents the results of the system's evaluation, emphasizing the influence of concert characteristics such as audio quality and genre on these results. This consideration enables a more detailed understanding of the findings and helps identify possible limitations. Moreover, I analyze the results on a concert-level basis to provide a thorough understanding of the system's behaviour and constraints.

Broadly, the evaluation of the system is divided into two sections: the first section focuses on the initial steps of the system utilized for the ALT task, while the second part evaluates the whole system's speed and accuracy in the lyrics-based setlist identification task. This split evaluation approach is necessary due to the limited research and benchmarks specifically available for setlist identification.

4.1 Evaluation Metrics

The evaluation metrics differ based on whether I am assessing the ALT module or the system's overall accuracy and speed in setlist identification. These metrics are defined in the sections below.

4.1.1 Automatic Lyric Transcription

For the ALT task, I employ the WER [%] according to Formula 3.2, which is as mentioned in Section 3.4.5 the most standard metric to measure the accuracy of automatic speech recognition systems, and therefore has itself established as the most common metric to measure the accuracy of ALT systems as well.

4.1.2 Entire System

When evaluating the accuracy of the entire system in terms of setlist identification, I will use - similarly to Yesiler et al. [12] - True Positives (TP) and False Positives (FP). Songs that the system correctly identifies and were performed during the concert are marked as TP. Conversely, songs that the system identifies but were not performed during the concert are marked as FP.

Yesiler et al. report TP in absolute values, which is not ideal for result comparison when using a different dataset, as I am in this project. To overcome this and for better clarity, I have chosen to use the True Positive Percentage (TPP) and False Positive Percentage (FPP). These metrics calculate TP or FP relative to the total number of songs performed during the concert, as defined in Formulas 4.1 and 4.2.

$$TPP = \frac{\#TP}{\#songs} \tag{4.1}$$

$$FPP = \frac{\#FP}{\#songs} \tag{4.2}$$

By using these relative measures on a concert-level, I can also evaluate on a dataset-level by simply calculating the mean of the values. This provides two practical metrics for measuring accuracy across datasets with N concerts, as defined in Formula 4.3 and 4.4.

$$TPP_{mean} = \frac{\sum_{n=1}^{N} TPP_n}{\#concerts} \tag{4.3}$$

$$FPP_{mean} = \frac{\sum_{n=1}^{N} FPP_n}{\#concerts} \tag{4.4}$$

To assess the system's speed, I measure the runtime of each step in the pipeline, which includes source separation, speech enhancement, voice activity detection, lyric transcription, and song search, both in absolute terms and relative to the concert audio length. Additionally, by summing up the absolute and relative runtimes of each step, I calculate the absolute and relative total runtime of the entire system. The relative total runtime values per concert can be aggregated by calculating the mean value over all concerts, allowing me to evaluate the system's speed across a dataset with multiple concerts, akin to the method used in Formula 4.3 and 4.4.

4.2 Methodology

This section delves into the strategy utilized for evaluating the system, focusing on the ALT module and particularly the entire system as a whole. The methodology highlights the evaluation of LyricSIS in terms of setlist identification accuracy and speed.

4.2.1 Automatic Lyric Transcription

To evaluate the ALT module of the system, I employ established benchmark datasets, namely Jamendo and Hansen. These datasets allow me to measure the ALT module's accuracy via the WER calculation, providing a comparative standpoint with the current state-of-the-art results.

This approach facilitates a comprehensive evaluation of the ALT module's effectiveness within the system.

However, it's important to note that the ALT module's parameters have not been individually optimized to achieve the lowest possible WER on a dataset. Instead, I seek the combination of system parameter settings that results in the highest TPP on the development dataset for the setlist identification task. Therefore, I only evaluate these parameter settings.

I measure the accuracy of the ALT module by comparing the AIGL of the songs in the benchmark datasets with their original lyrics. For this comparison, I use the WER and benchmark the results against various research outcomes, including those that achieve state-of-the-art results.

This method provides an understanding of the ALT module's accuracy, enabling me to assess its individual effectiveness.

4.2.2 Entire System

I utilize both development and test datasets for system evaluation, with the former to identify optimal component design and system parameter settings that maximize TPP while keeping FPP and runtime low, and the latter to evaluate the system's accuracy and speed on a fresh dataset.

Developing the system

The entire system consists of ten user-adjustable system parameters. Some parameters have a finite set of possible values, while others have an infinite range of potential settings. Due to computational, cost, and time constraints, and the impracticality of testing all possible combinations of parameter settings, designing experiments became a crucial part of this project during the development phase and will intensively be elaborated in this section.

To evaluate the entire system, I had to run it fully for each experiment. While individual components could be assessed using specific metrics, these couldn't reflect the overall system accuracy due to complex inter-component interactions influencing the TPP.

Hence, I rely on the metrics measuring the entire system's accuracy and speed to evaluate different parameter combinations.

During the implementation phase, I initially performed unstructured manual test runs to understand the system's parameters' behavior and sensitivity. This process identified critical and sensitive parameters and established a baseline for effective parameter settings (Table 4.1). These baseline settings defined the settings for not-yet-evaluated parameters during the experiments and limited the potential value range for parameters with infinite settings.

Testing every parameter combination would necessitate 1080 experiments, which was too expensive and time-consuming. So, I evaluated parameters one by one, while keeping others constant, splitting the pipeline into three modules: Pre-processing, Transcription, and Search Algorithm. Each has its own vital steps and significantly influences the final output. This strategy reduced the number of necessary experiments from 1080 to 43.

However, even this reduced number of experiments is still too high for the scope of this project. Therefore, I further subdivided some sections into smaller parts, enabling evaluation at the parameter level. This means that to evaluate a specific parameter, I ran the entire pipeline multiple times with different settings for that parameter while keeping all other parameters constant, except the NWordsPerRequest and NSongsPerRequest were evaluated together to understand their interaction. This approach allowed me to determine optimal parameter settings for each component while minimizing necessary experiments.

The sequence of experiments and modules followed the pipeline stages' progression, starting with Pre-processing.

Pre-processing Parameter As stated in Chapter 3, audio pre-processing is vital for various audio-related tasks, including speech recognition and music transcription. However, pre-processing steps may not always benefit automatic lyric transcription, possibly leading to loss of vital vocal artifacts, affecting transcription accuracy. Despite resource limitations and inaccessible pre-trained models, I wanted to test this approach with an experiment where the pipeline starts with the transcription process on the raw polyphonic concert audio.

The speech enhancement and voice activity detection models in this system generally have fixed parameters as they are each based on one fix pre-trained network. The Silero-VAD voice activity detection step has an adjustable parameter, vad_smoothing_seconds, adding some time to the suggested timestamps before cutting out silence. Preliminary tests indicated its minimal impact on results compared to the default settings.

To evaluate the impact of pre-processing on the systems accuracy, I will keep all other parameters constant at their baseline settings and vary the pre-processing parameters. This approach will result in four different parameter combinations to evaluate, which can be seen in the Table 4.2, which displays the entire design of experiments that I conducted. For visual clarity, I lowered the font size and abbreviated the column names.

The column names were abbreviation as follows:

$$\begin{aligned}
SourceSeparationTechnique &= SST \\
SpeechEnhancementTechnique &= SET \\
VADTechnique &= VADT \\
TranscriptionModelSize &= Model \\
NoSpeechThreshold &= NoSpTh \\
ConditionOnPreviousText &= CondPrTe \\
NWordsPerRequest &= NWords \\
NSongsPerRequest &= NSongs \\
WIPThreshold &= WIPTh \\
WERThreshold &= WERTh
\end{aligned}$$

The 'best' parameter setting is the configuration that resulted in the highest TPP compared to all other experimented settings for this parameter.

Transcription Parameter Upon identifying optimal parameters for pre-processing through four experiments, I pivoted to the transcription section. The audio pre-processing and search algorithm parameters remained at their optimal and baseline values, respectively. I explored the transcription section's parameter space by testing five model sizes, three non-speech thresholds, and two condition-on-previous-text settings as detailed in Table 4.2. Each parameter was evaluated independently to ensure minimal interaction, beginning with the medium baseline model size.

The non-speech threshold impacts Whisper's internal voice activity detection. Although I disabled Whisper's internal VAD through the Silero-VAD in pre-processing, I examined a default and low non-speech threshold setting. As per Whisper's developers, this threshold prevents the model from repetitive loops or hallucinations [15].

Next, I inspected the condition-on-previous-text parameter at its optimal non-speech threshold setting. I then assessed the five model sizes. This process narrowed the transcription phase's experiments from 30 to 7, including tests with baseline values.

Search Algorithm Parameter After the transcription phase, I solidified optimal parameters and proceeded with the search algorithm evaluation. Four variables were key: words per request, songs per request, WIP threshold, and WER threshold. I experimented with words per request at three settings: six (baseline), three (lower limit), and ten (upper limit). The songs per request variable, dictating potential songs per Genius API request, was evaluated at five (baseline), twenty (upper limit), and one (lowest). I performed an exhaustive exploration of all possible combinations, resulting in nine experiments.

Lastly, I studied the WIP threshold and WER threshold parameters by setting them to zero, ensuring zero songs identified. This experiment guaranteed a TPP of zero as no songs were found, allowing each lyric chunk to request potential songs from Genius API. Due to no songs being discovered, WIP and WER values were computed and stored for each potential song. Post-experiment, this song set's analysis enabled me to determine the optimal WIP threshold by calculating the potential maximal TPP and FPP for different thresholds. The WIP threshold helps filter out potential FPs, while letting pass all potential TP to the next WER threshold stage, reducing computational demand for the WER calculation during alignment, with the primary goal of maximizing TPP.

It is important to understand that the TPP and FPP for different WER thresholds are calculated based on the song set that results from the experiment where no song was actually identified. Therefore, all lyric chunks are used to find potential songs, indicating the potential maximum of TPP and FPP. This isn't the case when applying the system in a real scenario, as songs, TP as well as FP, are found during analysis and some lyric chunks are skipped due to these song identifications. This means the actual TPP and FPP will both be lower than the depicted potential maximums for TPP and FPP.

Although this experiment's evaluation suggests the potential area where the best WER threshold might lie, it is not sufficient to draw conclusions based solely on this experiment. Consequently, I conducted 12 more experiments, choosing a suitable value range based on this preliminary, yet non-real-scenario-representative experiment results. By evaluating the results of these 12 real-scenario-representing experiments, I could determine the best WER threshold, depending on user needs, by observing the influence of the WER threshold on TPP and FPP values.

Altogether, pre-processing, transcription, and song search sections necessitated four, seven, and 21 experiments respectively, totaling 32 experiments.

Testing the system

After the comprehensive system development phase utilizing the dev dataset to identify the optimal combination of parameter settings, I conducted the final evaluation of the entire system using precisely these best parameter settings on the basis of the test dataset. I analyzed the TPP, FPP, and relative runtimes, considering the influences of genre and audio quality. In addition, I utilized Shazam for the SLI task on the test dataset, comparing the accuracy and speed metrics between the two methods. Furthermore, I added the SLI results of the best performing systems from Yesiler et al. [12]. This served as a benchmark to gauge how well the lyrics-based system can generate setlists for live music concerts in comparison to a commercial song identification system as well as to the research area.

Table 4.1: Baseline and evaluation settings and number of settings to be evaluated per parameter per section

Section	Parameter	Baseline Settings	Evaluation Settings	Number of Settings
Pre-processing	SourceSeparationTechnique	htdemucs	spleeter, htdemucs, htdemucs_ft, None	4
	SpeechEnhancementTechnique	denoiser	denoiser, None	2
	VADTechnique	silero	Silero, None	2
Transcription	TranscriptionModelSize	medium	tiny, small, base, medium, large	5
	NoSpeechThreshold	1	0.1, 0.6, 0.99	3
	ConditionOnPreviousText	TRUE	TRUE, FALSE	2
Search Algorithm	NWordsPerRequest	6	3, 5, 10	3
	SongsPerRequest	5	1, 5, 10	3
	WIPThreshold	0.25	0	1
	WERThreshold	0.8	0, 0.78-0.95	1

Table 4.2: Design of experiments conducted with different settings

No.#	Section	SST	SET	VADT	ModelSize	NoSpTh	CondPrTe	NWordsReq	NSongs	WIPTh	WERTh
1	Pre-processing	**None**	**None**	**None**	medium	0.999	TRUE	6	5	0.25	0.8
2		**spleeter**	denoiser	Silero	medium	0.999	TRUE	6	5	0.25	0.8
3		**htdemucs**	denoiser	Silero	medium	0.999	TRUE	6	5	0.25	0.8
4		**htdemucs_ft**	denoiser	Silero	medium	0.999	TRUE	6	5	0.25	0.8
5	Transcription	**best**	denoiser	Silero	medium	**0.6**	TRUE	6	5	0.25	0.8
6		**best**	denoiser	Silero	medium	**0.25**	TRUE	6	5	0.25	0.8
7		**best**	denoiser	Silero	medium	**best**	**FALSE**	6	5	0.25	0.8
8		**best**	denoiser	Silero	**tiny**	**best**	**best**	6	5	0.25	0.8
9		**best**	denoiser	Silero	**small**	**best**	**best**	6	5	0.25	0.8
10		**best**	denoiser	Silero	**base**	**best**	**best**	6	5	0.25	0.8
11		**best**	denoiser	Silero	**large**	**best**	**best**	6	5	0.25	0.8
12	Song Search	**best**	denoiser	Silero	**best**	**best**	**best**	**6**	**1**	0.25	0.8
13		**best**	denoiser	Silero	**best**	**best**	**best**	**6**	**20**	0.25	0.8
14		**best**	denoiser	Silero	**best**	**best**	**best**	**3**	**1**	0.25	0.8
15		**best**	denoiser	Silero	**best**	**best**	**best**	**3**	**5**	0.25	0.8
16		**best**	denoiser	Silero	**best**	**best**	**best**	**3**	**20**	0.25	0.8
17		**best**	denoiser	Silero	**best**	**best**	**best**	**10**	**1**	0.25	0.8
18		**best**	denoiser	Silero	**best**	**best**	**best**	**10**	**5**	0.25	0.8
19		**best**	denoiser	Silero	**best**	**best**	**best**	**10**	**20**	0.25	0.8
20		**best**	denoiser	Silero	**best**	**best**	**best**	**best**	**best**	**0**	**0**
21		**best**	denoiser	Silero	**best**	**best**	**best**	**best**	**best**	**best**	**0.78**
22		**best**	denoiser	Silero	**best**	**best**	**best**	**best**	**best**	**best**	**0.80**
23		**best**	denoiser	Silero	**best**	**best**	**best**	**best**	**best**	**best**	**0.82**
24		**best**	denoiser	Silero	**best**	**best**	**best**	**best**	**best**	**best**	**0.84**
25		**best**	denoiser	Silero	**best**	**best**	**best**	**best**	**best**	**best**	**0.86**
26		**best**	denoiser	Silero	**best**	**best**	**best**	**best**	**best**	**best**	**0.88**
27		**best**	denoiser	Silero	**best**	**best**	**best**	**best**	**best**	**best**	**0.90**
28		**best**	denoiser	Silero	**best**	**best**	**best**	**best**	**best**	**best**	**0.91**
29		**best**	denoiser	Silero	**best**	**best**	**best**	**best**	**best**	**best**	**0.92**
30		**best**	denoiser	Silero	**best**	**best**	**best**	**best**	**best**	**best**	**0.93**
31		**best**	denoiser	Silero	**best**	**best**	**best**	**best**	**best**	**best**	**0.94**
32		**best**	denoiser	Silero	**best**	**best**	**best**	**best**	**best**	**best**	**0.95**

4.3 Results and Analysis

This section describes the experimental results obtained from the dev dataset, the ALT module applied for transcribing the lyrics from the Jamendo and Hansen datasets, the final outcomes of the entire system on the test dataset, and all analyses of these results.

4.3.1 Automatic Lyric Transcription

Table 4.3 presents the results of the ALT module when applied to the Jamendo and Hansen datasets.

Table 4.3: WERs [%] of most recent state of the art-ALT systems on the Jamendo and the Hansen dataset. **Bold** indicates the best result, an underline marks the second best result on a dataset. '-' refers to 'non-applicable'.

Method	Jamendo	Hansen
Stoller [38]	77.80	-
Demirel [41]	34.94	36.78
Gao [7]	43.44	36.34
Ou [8]	<u>33.13</u>	**18.71**
LyricSIS	**29.53**	<u>21.30</u>

The creators of the Jamendo dataset themselves back in 2019 achieved a WER of 77.80% [38]. Since then, significant improvements can be observes, particularly those made by Demirel et al. in 2021 with their MSTRE-NET method [41]. Subsequent similar or even better results were produced by Gao et al. in 2022 on the Hansen dataset [7], and as far as I am aware, the most recent state-of-the-art results are from Ou et al. in 2022 [8], who scored 33.13% and 18.71% on the Jamendo and Hansen datasets, respectively. This displays considerable enhancements, especially for the Hansen dataset.

The ALT module of the system developed in this project outperforms all existing methods on the Jamendo dataset and achieves the second-best score on the Hansen dataset, coming significantly closer to Ou's state-of-the-art scores on the Hansen dataset than earlier work. Ou et al.'s method involves applying transfer learning to a wav2vec 2.0 model to transition from the speech-to-text domain to the singing-to-text domain, and the model is fine-tuned on singing data. Even though the transcription model Whisper was not specifically fine-tuned with singing data, my approach achieves state-of-the-art results on Jamendo. Thus, using the pre-processing steps and the Whisper model, which is trained on speech data and not singing data, and designed for the speech-to-text domain, leads to better results than the methods and models used in recent research, which are developed specifically for the ALT task. These results can probably be explained by the fact that Whisper has the most recent state-of-the-art scores on multiple benchmark datasets for speech-to-text transcription tasks. Furthermore, it is trained on a large training dataset, incorporating various languages, and employs a unique weak-supervised training strategy designed to make it robust against accents, and potentially, the differences between spoken and sung words. This suggests it is a suitable model for transfer learning, and fine-tuning it with singing datasets could potentially enhance the project's ALT module results. Furthermore, the songs in both dataset are high-quality audio files, so it would be interesting to see how the ALT module performs on a dataset with lower quality songs to evaluate this module more in detail.

4.3.2 Entire System

During the development phase, I evaluated the entire system based on the dev dataset to identify the best components and system parameters. Numerous experiments were conducted on this dev dataset to determine the optimal system parameters, prioritizing maximizing TPP while keeping FPP and runtime low. The best system parameters resulting from these experiments were then tested on the test dataset in a final test run. This section is therefore divided into two subsections corresponding to the two datasets and their respective uses.

Dev dataset

Table 4.4 presents the results of the experiments on the dev-set, which aimed to identify the best system parameters leading to the highest TPP_{mean}. By comparing the differences of the TPP_{mean} within each level of evaluation, the influence of various parameter settings can be determined.

Both Demucs models with experiment TPP scores of approx. 68%, for instance, work better for source separation than the Spleeter model, which scored 52.33% in the experiment. Interestingly, feeding the polyphonic audio directly into the transcription step without any pre-processing yields a higher TPP score of 63.17% than when spleeter is utilized, though not as good as either Demucs model, making them a solid choice. The standard, non-fine-tuned Demucs model performs slightly better, even though it operates 3-4x faster. This makes this model an excellent choice for the source separation task. The NoSpeechThreshold parameter can influence the TPP scores by up to 5%. If the Whisper-internal VAD is configured to be quite aggressive by setting the NoSpeechThreshold to low values like 0.25, lower TPP result can be seen. This is likely due to voices being misclassified as silence, resulting in the loss of important lyrics. The impact of ConditionOnPreviousText can account for almost a 10% difference in TPP, making it a crucial parameter. As lyrics usually form a story, it makes sense that without the information from previously sung or transcribed lyrics, the next chunk of transcription is more likely to be mistranscribed.

Of all parameters evaluated the model size has the most significant impact on TPP, with values ranging from 28.42% using the tiny model to 70.92% using the medium model. Counterintuitively, the highest value was achieved using the medium model, not the largest one. Upon

Table 4.4: TPP$_{\textbf{mean}}$ of experiments conducted with different settings on dev dataset. The **bold** marks the best result, the underline the second best

No.#	Section	SST	SET	VADT	ModelSize	NoSpTh	CondPrTe	NWordsReq	NSongs	WIPTh	WERTh	TPP$_{\text{mean}}$
1	Pre-processing	None	None	None	medium	0.999	TRUE	6	5	0.25	0.8	63.17
2		spleeter	denoiser	Silero	medium	0.999	TRUE	6	5	0.25	0.8	52.33
3		htdemucs	denoiser	Silero	medium	0.999	TRUE	6	5	0.25	0.8	68
4		htdemucs_ft	denoiser	Silero	medium	0.999	TRUE	6	5	0.25	0.8	67.9
5	Transcription	htdemucs	denoiser	Silero	medium	0.6	TRUE	6	5	0.25	0.8	70.92
6		htdemucs	denoiser	Silero	medium	0.25	TRUE	6	5	0.25	0.8	66.67
7		htdemucs	denoiser	Silero	medium	0.6	FALSE	6	5	0.25	0.8	61.17
8		htdemucs	denoiser	Silero	tiny	0.6	TRUE	6	5	0.25	0.8	28.42
9		htdemucs	denoiser	Silero	small	0.6	TRUE	6	5	0.25	0.8	41.75
10		htdemucs	denoiser	Silero	base	0.6	TRUE	6	5	0.25	0.8	53.25
11		htdemucs	denoiser	Silero	large	0.6	TRUE	6	5	0.25	0.8	70
12	Song Search	htdemucs	denoiser	Silero	medium	0.6	TRUE	6	1	0.25	0.8	69.25
13		htdemucs	denoiser	Silero	medium	0.6	TRUE	6	20	0.25	0.8	69.25
14		htdemucs	denoiser	Silero	medium	0.6	TRUE	3	1	0.25	0.8	63.5
15		htdemucs	denoiser	Silero	medium	0.6	TRUE	3	5	0.25	0.8	68
16		htdemucs	denoiser	Silero	medium	0.6	TRUE	3	20	0.25	0.8	69.25
17		htdemucs	denoiser	Silero	medium	0.6	TRUE	10	1	0.25	0.8	64.08
18		htdemucs	denoiser	Silero	medium	0.6	TRUE	10	5	0.25	0.8	65.75
19		htdemucs	denoiser	Silero	medium	0.6	TRUE	10	20	0.25	0.8	67.42
20		htdemucs	denoiser	Silero	medium	0.6	TRUE	6	5	0	0	0
21		htdemucs	denoiser	Silero	medium	0.6	TRUE	6	5	0.195	0.78	65.25
22		htdemucs	denoiser	Silero	medium	0.6	TRUE	6	5	0.195	0.80	67.58
23		htdemucs	denoiser	Silero	medium	0.6	TRUE	6	5	0.195	0.82	70.92
24		htdemucs	denoiser	Silero	medium	0.6	TRUE	6	5	0.195	0.84	69.92
25		htdemucs	denoiser	Silero	medium	0.6	TRUE	6	5	0.195	0.86	68.00
26		htdemucs	denoiser	Silero	medium	0.6	TRUE	6	5	0.195	0.88	67.58
27		htdemucs	denoiser	Silero	medium	0.6	TRUE	6	5	0.195	0.90	67.58
28		htdemucs	denoiser	Silero	medium	0.6	TRUE	6	5	0.195	0.91	70.08
29		htdemucs	denoiser	Silero	medium	0.6	TRUE	6	5	0.195	0.92	**71.33**
30		htdemucs	denoiser	Silero	medium	0.6	TRUE	6	5	0.195	0.93	68.50
31		htdemucs	denoiser	Silero	medium	0.6	TRUE	6	5	0.195	0.94	68.25
32		htdemucs	denoiser	Silero	medium	0.6	TRUE	6	5	0.195	0.95	66.58

further investigation, I noticed that transcriptions from the large model sometimes included music-indicating characters, such as note symbols like '♩♬'. Even though the Whisper model's training data was pre-processed to avoid non-alphanumeric occurrences, I suspect that some might have been missed. This would explain why the large model can identify singing as music and transcribe it as a note symbol. The medium model might be too small to do so. Regardless, the large and medium models both yield high TPP values, and for this project, the medium model is deemed the best, largely due to its significantly faster transcription times compared to the large model, although high accuracy is prioritized over speed.

After establishing the best parameter settings for the pre-processing and transcription sections, the impact of the song search parameters NWordsPerRequest and SongsPerRequest on TPP can be seen, which ranges from 63.5 to 70.92%.
SongsPerRequest has a marginal impact with minor TPP increase for higher values, but an insignificant difference between 5 and 20. The key shift happens between 1 and 5, where TPP slightly rises. Given the disproportionate runtime increase for analyzing 20 songs per lyric chunk, the optimal SongsPerRequest is 5, providing an acceptable TPP increase from 1 to 5.
For NWordsPerRequest, a 'sweet spot' at the middle value 6 exists when SongsPerRequest is constant. Boundary values of 3 and 10 consistently yield lower TPP, confirming my hypothesis. Too small a value generates excess potential song matches from Genius.com, whereas a high value risks transcription inaccuracies leading to insufficient potential song matches, possibly excluding the correct song. Thus, NWordsPerRequest optimally sits at 6.

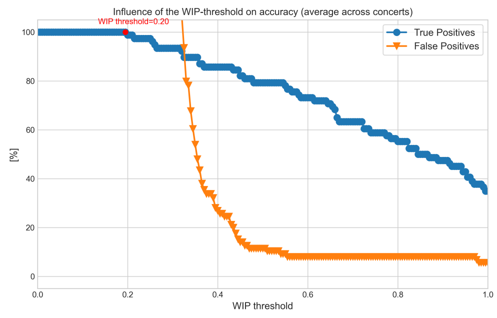
Lastly, the two thresholds WIP and WER were evaluated. Figure 4.1 shows their impact on the potential maxima of TPP and FPP. The optimal WIP was extracted from an experiment where it was initially set to 0, and then evaluated for maximizing TPP and minimizing FPP.

Figures 4.1a and 4.1c, along with 4.1d and 4.1b, essentially exhibit the same data but with varying y-axis scales. They collectively indicate the effect of the WIP threshold on the proportion of TP and FP that would pass the threshold.

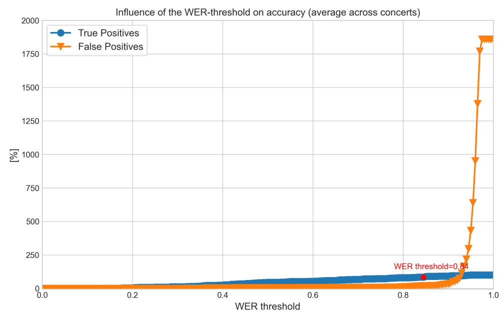
As shown in Figure 4.1a, a WIP threshold of 0.195 ensures 100% of potential TP pass the threshold, qualifying as potential matches for the subsequent WER threshold stage. This identifies 0.195 as the optimal WIP threshold setting.

Moreover, Figure 4.1c reveals that a WIP threshold of 0.195 results in over a 50% reduction of potential FP that pass the WIP threshold, from 4275% down to 2128%, saving a lot of computation resources without losing accuracy.

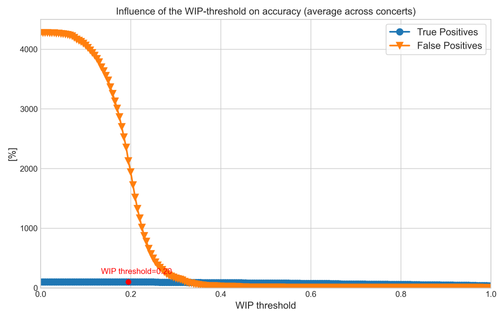
In the process of pinpointing the ideal WER threshold, I referred to figures 4.1d and 4.1b. These figures depict the impact of the WER threshold on the potential maximum of TPP and FPP based on the data after filtering out the songs that didnt pass the WIP stage when WIP threshold is 0.195. It's crucial to remember that the given TPP and FPP values are derived from an experiment where no songs were successfully identified. Consequently, they don't mirror the real scenario but instead provide an estimation of the maximum possible TPP and FPP if all



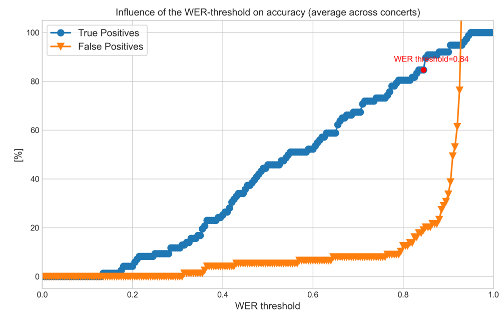
(a) WIP influence (TPP focus)



(b) WER influence after WIP threshold=0.195 (FPP focus)



(c) WIP influence (FPP focus)



(d) WER influence after WIP threshold=0.195 (TPP focus)

Figure 4.1: Percentage of songs that surpass the WIP- or WER threshold for TP and FP

lyric chunks were utilized in fetching potential songs from the Genius API. Let's denote these as TPP_{potmax} and FPP_{potmax} for simplicity. The real TPP and FPP that resulted from my experiments will be denoted as TPP_{real} and FPP_{real}.

Using 4.1d and 4.1b, it was not possible to single out the best WER threshold. However, they did help me determine the range for the WER threshold to conduct further experiments and discover the ideal WER threshold. The stakeholder and I agreed that the real FPP should not surpass 25%, which influenced the range selection for further experiments. Therefore, I selected the maximum WER threshold that would still confine the potential maximum FPP, FPP_{potmax}, to 25% or less. As per Figure 4.1d, this threshold is 0.88. The figure also indicates that the TPP_{potmax} exhibits a nearly linear growth with the WER threshold increase, whereas the FPP_{potmax} begins to rise exponentially beyond a certain point. Figure 4.1b further accentuates this phenomenon.

This implies that raising the WER threshold to boost TPP might result in a considerable surge in FPP, making it a trade-off not worth pursuing. This was another reason to limit FPP to less than 25%. The subsequent experiment and its evaluation will reveal the real FPP and dictate the WER thresholds to be assessed next. Evaluating these experiments by examining the real TPP and FPP revealed that the real FPP is far lower than the potential maximum, as illustrated in Figure 4.2.

From earlier experiments using a baseline WER threshold of 0.82, I was aware that the TPP was higher than at 0.88. This justified the need to experiment with both lower and higher WER threshold values in the upcoming experiments. In the end, I conducted experiments with the settings [0.78, 0.8, 0.82, 0.84, 0.86, 0.88, 0.9, 0.91, 0.92, 0.93, 0.94, 0.95] to identify the best WER threshold, as detailed in Table 4.4.

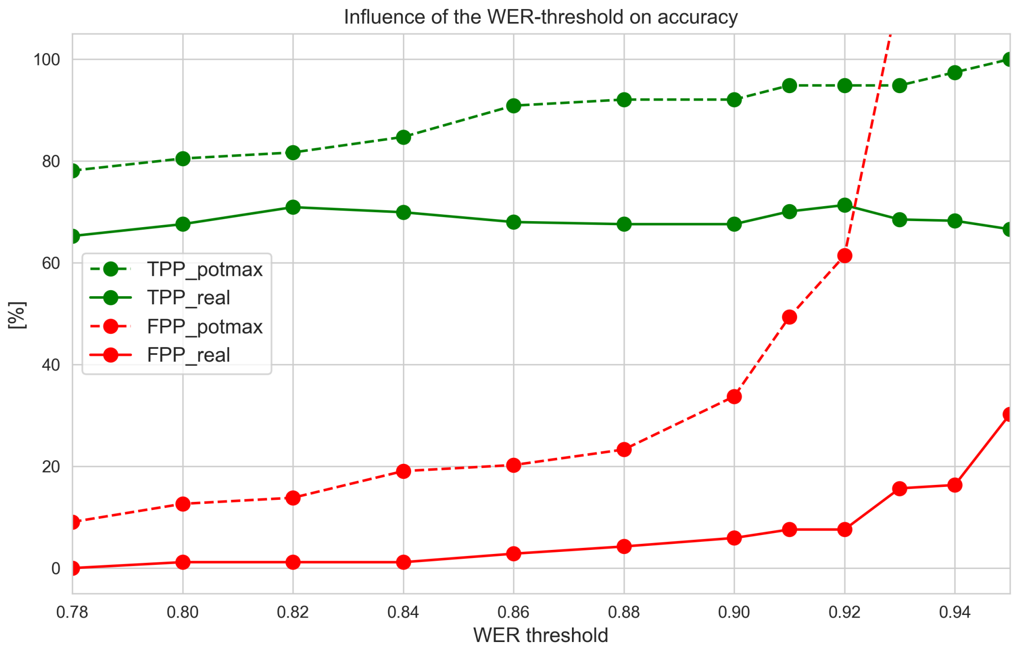


Figure 4.2: WER threshold evaluation regarding the potential maximum of TPP and FPP based on the experiment with parameter setting WIP threshold=WER threshold=0, and regarding the real TPP and FPP

The real TPP and FPP values based on these experiments, along with the values of the potential maximum TPP and FPP obtained from the experiment where WIP and WER were set to zero, are documented in Figure 4.2.

Figure 4.2 presents the real TPP and FPP values based on these experiments, along with the values of the potential maximum TPP and FPP obtained from the experiment where WIP and WER were set to zero. This graph guided me to select a WER threshold of 0.92 as the optimal value. It is clear that the actual accuracy values significantly deviate from the potential maximums and are quite stable. While the potential maximums for TPP and FPP continuously grow, the same happens with the real FPP, although the exponential rise is not visible for these experiment and it appears to be more linear across this range of WER threshold values. Conversely, the real TPP behaves more erratically, featuring small local minima and maxima, but overall being quite stable, fluctuating in a small range between 65.25 and 71.33 % TPP values.

Test dataset

Table 4.5 shows the setlist identification results on the test dataset for Shazam, two different SLI systems by Yesiler [12], and the lyrics-based SLI system I developed in this project. I selected the two SLI systems from Yesiler, namely Re-MOVE(120,30) and Qmax(120,60), for comparison as they achieved the highest TPP and lowest FPP, respectively, among all the systems and configurations evaluated by Yesiler in his paper.

Yesiler's test dataset is also derived from the ASID dataset, but he might have chosen a different split for the dev- and test dataset. Additionally, his test dataset contains 65 concerts, while I had to exclude some concerts, leaving only 41 concerts in my test dataset. Given these differences, using his results as a benchmark could potentially be misleading. However, considering these are the only setlist results available in existing research, this comparison could still serve as an acceptable benchmark for LyricSIS.

In terms of accuracy, my lyrics-based system significantly outperforms Shazam with a TPP of 45.63%, as compared to Shazam's 26.85%. Furthermore, LyricSIS exhibits a FPP that is nearly 2.5 times lower than Shazam's. Depending on the use case of the setlist identification system, this may be critically important. For instance, maintaining low FPP values is crucial as incorrectly identified songs lead to misallocation of royalties, which should be minimized. However, when it comes to speed, Shazam outperforms the lyrics-based SLI system by a factor of 1.7x, requiring only 37% of the audio length for analysis, in contrast to the 64% required by LyricSIS.

Yesiler's measures only included TP, TPP (termed DAP in his thesis), and FP, which led me to calculate the FPP myself using Formula 4.5. Moreover, Yesiler did not provide any runtime information for his systems to complete the SLI task, therefore a direct comparison of the speed of LyricSIS to his was not possible.

As described in Chapter 3.3, Yesiler used his CSI system to identify songs based on overlapping audio chunks of the concert, thereby generating a setlist. Identification was carried out using a distance function to compare the audio features of the chunk to be analyzed with the reference songs in the database. A close enough distance to one of the reference songs was considered a match. He trained a binary classifier (TP/FP) using these distances, which he later used to post-process the generated setlist and reduce the FP, albeit slightly reducing the TP. The results after classifier application are indicated in Table 4.5 as '/w Classification'.

The model with the best TPP, at 80.30%, shows a high number of FPP (92.22%) before classification application. After classification, the TPP is 67.80%, and the FPP is 15.5%, which is comparable to the final FPP of LyricSIS. Therefore, after classification, Yesiler's Re-MOVE system yields nearly 1.5x higher TPP values than LyricSIS, while maintaining similar FPP levels. His Qmax model, with classification, retains these high TPP values (only 3% lower than Re-MOVE with Classification), while reducing the FPP to a value of 6.61%, which is less than half of that achieved by LyricSIS. In contrast to LyricSIS, which relies solely on lyrics, Yesiler's

CSI models are based entirely on the harmonics of a song, making them robust to the presence or absence of lyrics within a song.

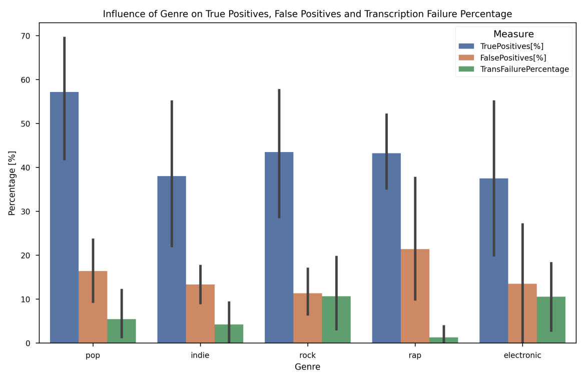
$$FPP = \frac{FP * DAP}{TP} \tag{4.5}$$

Table 4.5: Results on the test dataset for Shazam, Re-MOVE, Qmax, and LyricSIS

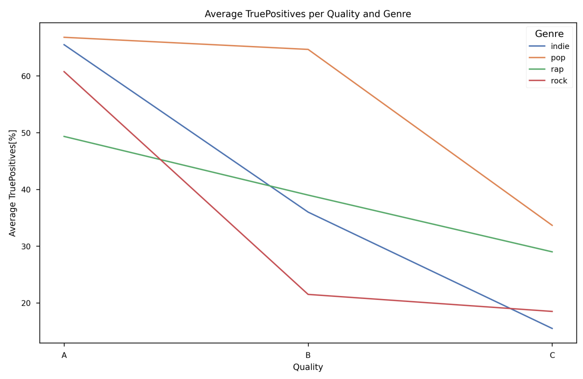
Method	$\mathbf{TPP_{mean}}[\%]$	$\mathbf{FPP_{mean}}[\%]$	$\mathbf{\frac{Runtime}{AudioLength}_{mean}}$
Shazam	26.85	37.29	0.37
Re-MOVE(120,30)	**80.30**	92.22	-
Re-MOVE(120,30) /w Classification	67.80	15.5	-
Qmax(120,60)	78.40	90.00	-
Qmax(120,60) /w Classification	64.90	**6.61**	-
LyricSIS	45.63	14.46	0.64

I must note that comparing LyricSIS with Shazam may seem unfair for various reasons. Shazam, for instance, retains default settings for song identification threshold because its API does not offer an adjustment option. Also, Shazam's primary objective is not to identify every song version in their database but to quickly identify low-quality snippets of commercial music recordings. Despite these considerations, this comparison highlights the limitations of commercial song identification systems based on audio fingerprints when identifying live versions of songs.

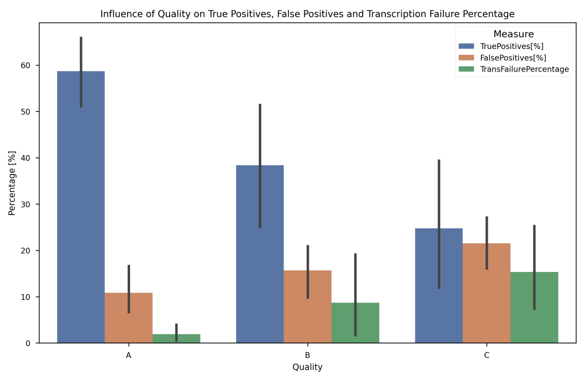
Although LyricSIS demonstrates superior results, a TPP of 45.63% is far from optimal. Numerous factors could be responsible for this, which will be further discussed in this section and in Section 4.3.2. To understand these evaluation results better, I delved deeper than the three overarching accuracy and speed metrics (which are averaged across all concerts in the dataset). I investigated the influence of genre, quality, and audio length on accuracy and speed, which also provided insights into the system's robustness towards these factors. Additionally, I examined how genre and quality impacted the runtime of each pipeline step: source separation, speech enhancement, voice activity detection, transcription, and song search. The results are presented in Figure 4.3.



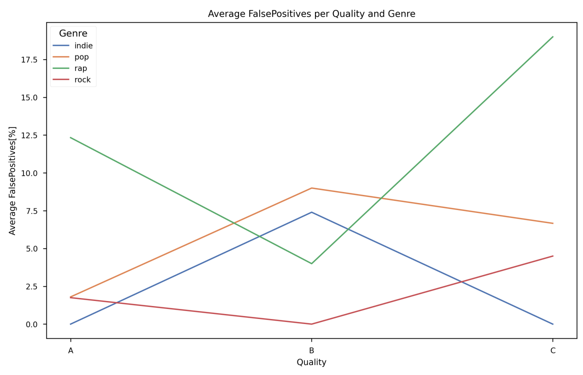
(a)



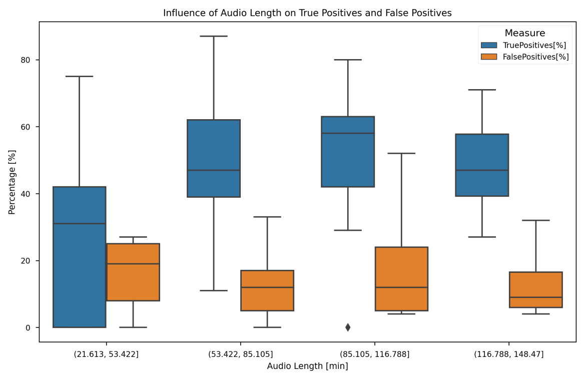
(b)



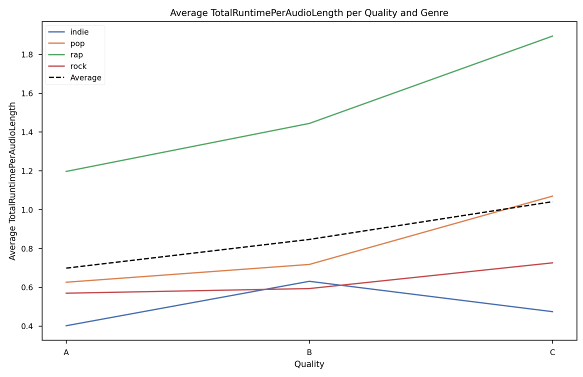
(c)



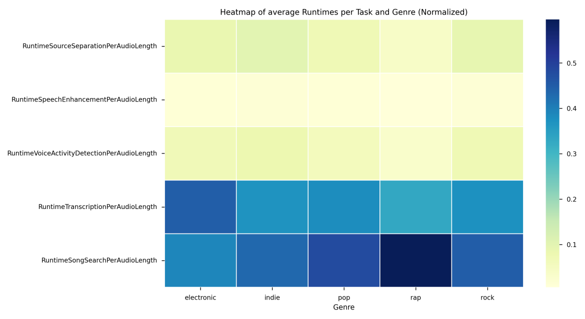
(d)



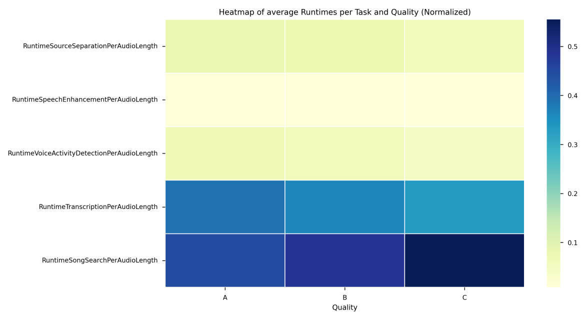
(e)



(f)



(g)



(h)

Figure 4.3: LyricSIS TPP, FPP, and runtimes based on genre and audio quality

Figure 4.3c illustrates the substantial effect of quality on accuracy. Higher quality leads to higher TPP and lower FPP. TPP is more than double for the highest quality A (59%) compared to the lowest quality C (24%). The FPP for quality A is also nearly half that of quality C (11% and 21%, respectively). The green bars represent the percentage of pre-processed audio chunks for which the transcription process failed to produce an output. As the Whisper model transcribes audio by calculating the most probable word prediction, it can fail if the probability threshold is never exceeded. Lower audio quality increases the likelihood of transcription failure, as it becomes more challenging for the model's predictions to surpass the threshold. Moreover, the Whisper internal VAD, which I didn't entirely deactivate, may misclassify vocals as noise for low-quality audio, which is then excluded.

In Figure 4.3a, it's clear that genre significantly influences accuracy. The best results in terms of high TPP are achieved for the pop genre (58% TPP), while the worst are for the electronic and indie genres (38% TPP each). Considering that vocals in pop music are typically clear and prominent, while in electronic music they're more likely distorted, this could partly explain the differences. The high transcription failure rate in rock music may be due to issues with voice quality, as the vocals tend to be more in the background and surrounded by loud instruments, sometimes with growling or screaming. Conversely, rap has the lowest transcription failure rate, probably because its style is the closest to speech, on which the model was trained. However, low TPP values for the indie genre cannot be explained by poor transcription rates, as the transcription success for this genre is similar to pop. Instead, these low TPP values may be due to a limited amount and skewed distribution of indie concerts towards lower qualities, as shown in Table 3.4. Most indie concerts are of quality B, which on average scores below 37% TPP, as shown in Figure 4.3b. The TPP values for the other two genres, rock and rap, are similar, although the distribution of concerts across the qualities for these genres is skewed towards higher qualities. This bias may distort the results, making rock and rap seem better than they would appear with a more evenly distributed dataset.

Figure 4.3b and Figure 4.3d demonstrate how different genre-quality pairs affect the TPP and FPP accuracy values. The electronic genre is not shown, as unfortunately, the dataset lacks electronic concerts for all qualities. Figure 4.3b reveals that for pop, the difference between quality A and B has a greater impact on accuracy than between B and C. This effect is reversed for the rock and indie genres, while for rap there's a linear decrease. Moreover, the TPP for rock and indie at quality C is below 20%, while rap and pop maintain TPP values above 30%. At high audio quality, all genres exceed 60% TPP, except for rap, which falls below 50% TPP, making it the most challenging genre for the SLI task of this system at high audio quality.

Figure 4.3d shows unexpected results for the influence of genre and quality on FPP values. All genres for quality B display local minima or maxima. This oddity, coupled with the fact that two genres show local minima while the other two present local maxima at quality B, might stem from uneven concert distribution across quality and genre in the dataset, or even misclassification. Larger datasets could clarify these anomalies, but with the current dataset, further analysis may not be fruitful. As shown in Figure 4.3e, audio length has no significant or clear effect on system accuracy, except for short concerts which tend to score slightly worse. There are no identifiable patterns across different audio lengths, which are binned for this plot. However, it's worth noting that the system remains robust, as lower-scoring short concerts can still reach extreme values of nearly 80% TPP.

Figure 4.3f illustrates the system's overall speed, measured as total relative runtime (total runtime per audio length), shedding light on the effects of quality and genre. Figures 4.3g and 4.3h further delve into the impacts of genre and quality on the relative runtime of each pipeline stage. As demonstrated in Figure 4.3f, the average total relative runtime within genres progressively increases as the quality deteriorates. For instance, while the total relative runtime is just 0.7 for quality A, it substantially lengthens for quality C, exceeding 1.0.

Moreover, it's evident that the setlist identification (SLI) of rap concerts requires approximately twice the time as other genres—pop, rock, and indie—with indie taking the least amount of time. All genres, except indie, exhibit a linear increase in runtime as quality decreases. For indie, similar to the previously mentioned graph, the maximum runtime occurs at the median quality. The most plausible explanation for this, I believe, is the statistical insufficiency concerning the number and distribution of samples across different quality levels within this genre.

Figures 4.3g and 4.3h reveal that the three pre-processing steps remain quite stable, regardless of the quality or genre. Also, compared to transcription and song search, pre-processing has a less significant impact on the total relative runtime. Figure 4.3h shows that the relative runtime for transcriptions tends to decrease slightly with lower quality. This trend is logical, as poor quality often results in voices being misinterpreted as noise during pre-processing and consequently filtered out, thus reducing the volume of audio to be transcribed.

Contrarily, the relative runtime for song searches increases with lower quality, likely due to inaccurate or incorrect transcriptions. This leads to fewer songs being found—as also evidenced in Figure 4.3c when examining the two bars per quality—and thereby prompts the algorithm to use more lyric chunks from the AIGL to request potential songs from Genius. Fewer lyric chunks are skipped due to fewer songs being found. Although all other runtimes slightly decrease with lower quality, the rising relative runtime for the search algorithm appears to be dominant, explaining the overall increase in total relative runtime with worsening quality.

Additional insights

In this section, I dive deeper into the system's evaluation results by manually analyzing them at the concert level. To understand better why the system behaves as described earlier, I've complemented the dataset-level analysis with a genre and quality-specific one. As noted, the system performs optimally for quality A and shows the highest accuracy for the pop genre.

Language Figure 4.4 shows the beginning of transcription of an ABBA concert, which opens with a song in Swedish, but continues in English. An example of a transcribed announcement is visible between these songs.

```
‘‘Du står där din telefon står där nästan som ett hål Inte ringer du och
säger älskling du sa att du såg emot den Du är så tyst mot var Ingen
knackar på min dörr Du som plakat mig för att snar i ett gummes sa att
du såg emot den Ingen provar mig just nu Ingen annan var du och din
Bara du står en signal Vi Tystnade med så totalt Vi Ringade då jag sa
Om jag fick en signal tog jag ett sprang Hjärtat gjorde en bra Ding
dong ding dong Om du ringer Ring det ända stök av Om du ringer Ring
det Ska.  Lika tyst.  Varenda dag.  Om det vore så.  Det var något med
domens värde.  Om den bara sade tyst.  Om den inte blott var tyst.  Om
jag fick nån löp.  För att vänta.  Här är.  Inget.  Broar mig just nu.
Ingen annan.  Nu.  Åh.  Vi.  Vi.  Varenda dag.  En signal.  Vi.  Vi.
Just nu är det så.  Så.  Vi.  Vi.  Vindrade då.  Låt så.  Var.  Om jag
fick en signal.  Så jag.  Har.  Läts att gjorde en val.  Vindom.  Vindom.
Nu.  Vi.  Vi.  Vind är det.  Vänster.  Vann.  Åh.  Vi.  Vi.  Vindrade
då.  If you’re happy, you’re not a bad person If you’re happy, you’re
not a bad person Thank you!  Good evening, Zurich!  Welcome to the work
show!  Are you well?  Ah, super!  Well, it’s been one year since we were
here last time, and it’s good to be back, I can tell you that.  If you
know the songs tonight, please join in and sing and dance with us.  We’re
gonna continue with one of our very oldest.  This one’s called Bang!  A
Boomerang Thank you somebody, happy it’s a question I’ve given to you You
can learn how to show it, so come on, give yourself a break Every smile
and every little touch Don’t you know that they mean so much?  Sweet,
sweet, you say so tender Always will return to standard Like the bang!  A
boom-ba-boom-ba-rag Bang!  A boom-ba-boom-ba-rag Love is always around
and you can look for it anywhere.  When you feel that you’ve found it, my
advice is to take it care.  Never use it as a selfish tool.  Never ever
be such a fool.  Every feeling you’re showing is a boomerang.  You know
it’s a bang, a boomerang.  Dummy dog talking, dummy dog talking.  Bang, a
boomerang.  Love is a tube, you hum, we hum.’’
```

Figure 4.4: Example of multiple languages within one concert (ABBA here) and a transcribed announcement

As seen, the system, due to its design of segmenting audio into smaller portions and implementing a modified Whisper language identification process, holds up against language changes within a concert. Also, the test dataset includes three concerts in non-English languages: Spanish, Finnish, and Italian, sung by J Balvin, J Karjalainen, and Max Pezzali respectively. Their TPP scores range from 42% to 75%, hovering around or surpassing the overall average, signifying the system's robustness to different languages, in accordance with those covered by the Whisper model. Figure 4.5 displays these languages along with their overall WER score, which provides a hint at the transcription quality for these languages.

Transcription problems influencing the song search As previously discussed, the AIGL can contain multiple errors leading to unidentifiable songs. The following are the most frequent or crucial issues:

- hallucinations
- repeat loops
- transcriptions unrelated to song lyrics, such as announcements
- complete failure of audio chunk transcriptions, meaning no transcription at all for a lyric chunk
- homophones, words pronounced the same but have different meanings, origins, or spelling, like knight and night, meat and meet

Figures 4.3a and 4.3c demonstrate that the genre and quality significantly influence the amount of audio snippets that fail to transcribe. According to the Whisper paper, hallucinations and repeat loops most likely occur with lengthy audio files or those with silent parts.

While one can overcome these issues by using high-quality audio, splitting long files into smaller chunks, fine-tuning the model, and/or employing a voice activity detection to eliminate silence parts, the remaining problems are trickier to tackle. However, based on my manual inspection and Whisper's paper, which suggests large models tend to transcribe homophones correctly, I concur that the Whisper model is rather resilient towards homophones. This resilience especially stands out when the 'condition on previous text' feature is active, as it considers previous predictions while transcribing next, reducing the chance of misidentifying homophones. Still, one particular reason for erroneous transcription of homophones in this project is the fact that the audio is sung, not spoken. Given the model isn't trained on singing data, it struggles to identify the correct transcription. As the Whisper paper suggests, fine-tuning might improve the handling of mis-transcribed homophones, especially in this project. Figure 4.6 provides an example from the ABBA concert.

Non-lyrical transcriptions like announcement lyrics (see Figure 4.4) might inflate the FPP rate and deflate the TPP. If these announcement lyrics serve as a lyric chunk to search for potential songs, the likely result is identifying songs not part of the concert, thus boosting the FPP. Moreover, such chunks might be skipped during the search, potentially lowering the TPP. If these lyrics are within the chunk a potential song is compared with to find a match, the similarity values might decrease, potentially causing failure to identify performed songs. This could lower TPP and increase potential FPP.

While announcements can be seen as additional lyrics which harm the song search process, the opposite of this is missing lyrics, which might be a result of the above mentioned problems during transcription, but could also be a result of accurate transcription in the case that a song is not fully performed. The system does not exhibit robustness in relation to the length of a song performed during a concert. E.g. if only half of a song is performed and still the song is identified, many lyric chunks unrelated or not belonging to the identified song get skipped due to the algorithms implementation. This could potentially lead to lower TPP (and FPP) values.

Genius Database While I attempted to verify if the Genius database contains the songs in each concert by checking the artist and song info via the Genius API, this approach can fail. Therefore, it's uncertain whether the test dataset songs are even discoverable in the Genius database. A case in point is the concert of TheHum, which scored 0% TPP in the final evaluation on the test dataset, as the Artist isn't listed in their database.

(state that they sometimes give entire book texts or others, which slows down the process aswell and is not really stopable even though their API has the song only option. Probably some mis-leading metadata within their database.)

MusicBrainz Database In the process of fetching ISWC for the identified songs, an intermediate step involves finding the MB_ID and ISRC, using the artist name and song name of the identified songs. In total, I manage to find the MB_ID and ISRC for approximately half of the identified songs. However, not for every MB_ID or ISRC the information of the related ISWC exists and the success rate for finding ISWCs for the identified songs is significantly lower at 27%. This indicates a need for access to a rich, extensive, and reliable ISWC database.

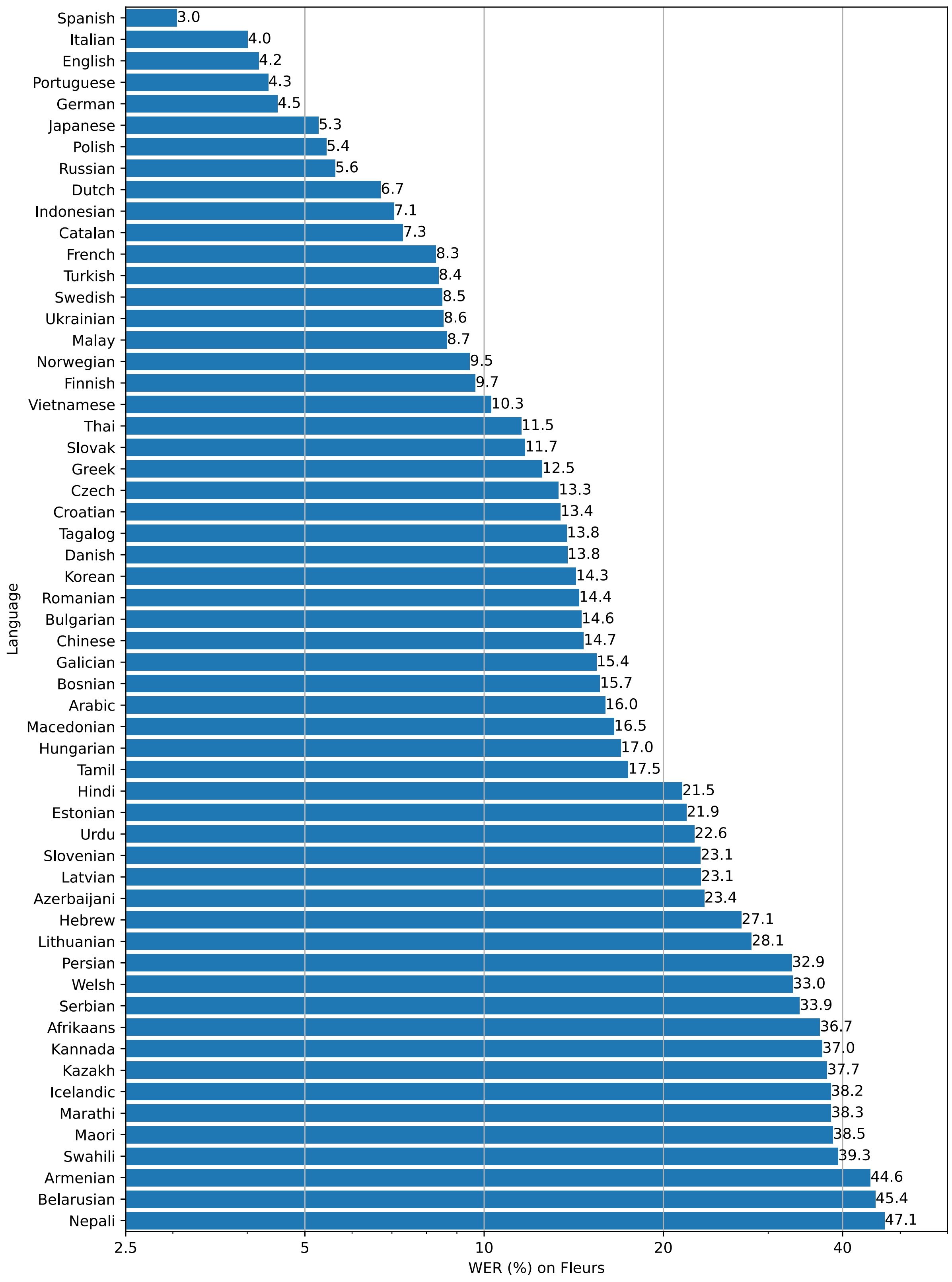


Figure 4.5: Whisper accuracy across languages

Making somebody happy is a question of give and take. (original lyrics)
Thank you somebody, happy it's a question I've given to you. (transcribed lyrics)

Figure 4.6: Example of wrong transcription due to homophones, taken from ABBA concert transcription

5 Conclusion

This chapter concludes the thesis by summarizing key evaluation results from chapter 4.3, assessing achievements in relation to the project's goals, discusses the system limitations and evaluates the potential use-cases defined in Section 2.4.

One important limitation to mention from the onset is the relatively small size and uneven distribution of concerts per genre and quality in the test dataset. This imbalance makes it challenging to draw definitive conclusions with respect to these factors. Nevertheless, within the confines of this limitation, it is crucial to highlight the system's restrictions concerning genre and quality based on the test dataset. These limitations might potentially generalize, but a more extensive and evenly distributed dataset of concerts would be necessary for further investigation.

The primary goal of this project was to design and develop a lyrics-based setlist identification system capable of generating setlists from live concert audio data. This involved crafting a comprehensive multi-stage pipeline that could pre-process raw audio and yield a setlist of identified songs including their ISWC code, emphasizing accuracy, robustness, and cost-effectiveness. The underlying aim was to verify the feasibility of such a system, specifically whether the lyrics-based song identification can be used for the setlist identification task and to see if lyrics of songs can indeed be used to eventually identify the ISWCs for these songs.

I successfully developed an ALT module that converts the raw audio of concert into AI-Generated Lyrics, demonstrating the potential of using the OpenAI's speech-to-text model Whisper in the ALT domain with monophonic input audio. The ALT module shows some limitations regarding the transcription accuracy, reaching from wrong transcriptions due to hallucinations, repeat loops and homophones to entirely failed transcriptions for some audio chunks, which could be seen especially for bad quality (B, and C). Despite the limitations, it could be seen in the evaluation of the ALT module on two benchmark datasets that it outperformed existing state-of-the-art ALT methods for the Jamendo dataset, and reached second place for the Hansen dataset, being very close to the state-of-the-art. Additional, the ALT module exhibited impressive robustness by being able to manage a wide array of languages, accents, and audio lengths. Overall the ALT module is a success.

Furthermore, the system successfully incorporated a lyrics-based song search module, which includes the automatic generation of the final setlist, that leveraged the comprehensive lyrics database from Genius.com and the open music encyclopedia MusicBrainz to identify songs and obtain related metadata, including ISWC codes. This accomplishment aligned with the project's objective of showing that it is possible to find the corresponding ISWCs for songs based on their lyrics.

Additionally, an evaluation module was developed, which classifies identified songs as True Positives and False Positives and lists besides other parameters the performance measures of the system, providing insights into results at the concert level and in aggregated form for datasets with multiple concerts.

The system was evaluated using a test dataset of 41 concerts, where it demonstrated effectiveness in identifying live versions of songs, achieving a True Positive rate of 46% and a False Positive rate of 15%.

While the system achieved many of the goals and objectives by fulfilling the functional requirements and many non-functional requirements, and proved the feasibility of the concept, several limitations have been revealed by the findings from the evaluation results. The following are the key limitations:

- **Accuracy**:
  The overall accuracy of the system falls below 50%. This limitation significantly curtails its standalone usability for tasks such as copyright infringement detection and setlist creation, as it as a standalone system might not provide reliable results in its current state.
- **Speed**:
  The current processing speed of the system is relatively slow compared to audio fingerprint-based identification systems such as Shazam. This issue could potentially limit its applicability in scenarios where timely results are paramount.
- **Audio Quality Dependancy**:
  The system is currently not robust enough to handle low (C) or medium (B) audio quality inputs effectively. This limitation results in diminished accuracy when the system is used on concerts of lower audio quality, thereby restricting its usage to only high-quality audio concerts.
- **Genre Dependency**:
  The system shows varying levels of performance across different music genres. This inconsistent performance indicates a lack of genre independence, thereby limiting the system's flexibility and its ability to cater to a broader range of musical styles.
- **Requirement of Lyrics**:
  Even though the system was designed only for songs with lyrics and is therefore entirely based on lyrics, which means it can only identify songs that contain lyrics, this poses a significant limitation to the system's applicability, versatility, and flexibility, if it ever shall be applied in a more broader or generalistic use case, where songs maybe dont have lyrics.

These limitations are linked to the system's requirements and underline areas where further improvement and development are beneficial. Addressing these limitations will not only enhance the system's performance but also extend its range of applications, making it a more versatile and practical tool for automatic setlist generation and song identification.

The observed successes and limitations provide a valuable perspective on the feasibility of the various potential use cases elaborated in Chapter 2.4, based on the performance of both the ALT module and the overall system. For instance, social media and streaming platforms could deploy the ALT module to transcribe lyrics from songs and performances, thereby enhancing user experience. In these scenarios, minor transcription errors might not be overly detrimental, as having 80% correct lyrics could still be preferable to none at all. However, lyrics database providers, who derive value from having precise lyrics, may find the output less satisfactory. While the system could serve as a decent starting point, it's likely that human post-processing would be necessary to ensure accuracy. Collection societies could potentially use the transcribed lyrics to cross-verify their catalogues, rectifying inaccuracies and resolving conflicts within song entities, including corresponding ISWCs. This could be feasible even with minor transcription errors; the process of comparing song lyrics could be adjusted to account for these discrepancies by lowering the threshold for identifying lyrics as identical. Nonetheless, the feasibility of all other potential use cases involving the system for automatic setlist identification should be evaluated based on the risk of missing songs matches and FP. For instance, using the system as an upload filter for user-generated content might provide helpful indications, but should not be solely relied upon. The use case of employing the system for setlist identification of live performances or concerts might still be viable. Despite the system not identifying every song correctly and occasionally falsely identifying some, the potential benefits could still outweigh the drawbacks. The justification for this is similar to the case of using the ALT module to display song lyrics on platforms for enhanced user experience.

In summary, the project has made significant strides towards achieving its objectives, proving the feasibility of a lyrics-based setlist identification system and demonstrating potential for further refinement, which will be elaborated in Chapter 6, and application of in the music industry. These findings and developments align well with the company's broader goals of enhancing transparency and fairness in the music industry. Although more work remains, the project's successful development and execution provide a promising foundation and a blueprint for further improvements, in line with the company's ultimate goal of benefiting all stakeholders in the music industry.

6 Future Work

The development of the lyric-based setlist identification system was a complex process with numerous challenges. Despite the limitations, several opportunities exist to improve and enhance the system's capabilities. This chapter provides suggestions for future research and development to overcome the limitations stated in Chapter 5 and better meet the system requirements:

- **Fine-tuning for Speed and Accuracy Enhancements**:
  The system's accuracy could be improved by fine-tuning the Whisper model on a singing dataset to adapt it from the speech-to-text to the singing-to-text domain. Using a smaller model as a baseline for fine-tuning may maintain or even increase accuracy while significantly decreasing the transcription runtime, hence enhancing the total system speed and accuracy. Given that the system's transcription performs better for rap, which is closer to the original training data format, speech, fine-tuning the system with a singing-to-text dataset could improve the transcription accuracy for other genres, potentially achieving the system's requirement of genre independence.

- **Fingerprint-based Approach**:
  Transform the overall approach from a method based solely on lyrics to a fingerprint-based approach, as soon as a music database is available. While retaining the use of lyrics, the proposed approach would transform them e.g. into a vector that represents the lyrics as done in [14], acting the same way as an audio fingerprint. This lyric-representing fingerprint could then be complemented with additional audio fingerprints, which could represent melody, harmony, and rhythm, among other factors. Such an enhancement could potentially increase the speed, accuracy, and robustness to audio quality of the setlist identification process. For instance, even in the case of poor quality audio, it may still be possible to identify a song based on rhythm, melody, or other acoustic features, even when the vocals become indistinguishable (e.g., see concert video of Sobs here: https://www.youtube.com/watch?v=qtJ7wjtipkY). Importantly, this fingerprint-based approach could also enable the identification of songs that do not contain any lyrics, breaking this limitation.

- **Lyrics Database Access**:
  Securing direct access to a lyrics database could significantly reduce the time taken to search for potential songs. The search algorithm could be optimized to provide a performance boost using more scalable text similarity methods such as TFIDF and k-nearest

neighbors, as did Vaglio in his research [5]. The system could benefit from faster, more accurate search results.

- **New Transcription Models Evaluation**:
  The evaluation of novel transcription models could potentially enhance the system's accuracy. For instance, the recently launched Whisper large-v2 model is a fine-tuned variant of the larger model and could be conveniently evaluated as it requires no additional implementations. Alternatively, an investigation into Meta's recently open-sourced transcription model MMS [42], which surpasses the former state-of-the-art Whisper model by 50% on benchmark datasets consisting of 56 languages, could be worthwhile, especially if there's interest in analyzing concerts in less commonly used languages. A detailed examination of MMS's results, however, reveals that the Whisper model surpasses the MMS model for more widely spoken languages such as English or Spanish. Both suggested models could potentially improve accuracy, though they might extend the runtime.

- **Increase Dataset Size**:
  To enhance the statistical significance of the results, especially concerning genre and quality subsets, acquiring a larger dataset is crucial. This would make subsequent evaluations more reliable and trustworthy.

- **Performance Testing**:
  The system could benefit from rigorous software testing to validate its functionality and robustness at the implementation level, thus improving its error handling and code maintenance.

- **Pre-processing Enhancement**:
  The introduction of additional pre-processing methods might improve transcription accuracy, such as de-reverberation, which means the process of reducing the reverb that often is applied to the vocals in music production.

- **Keyword Emphasis**:
  Improvements can be made by emphasizing keywords over the most common words within the lyric-based song search. This adjustment could potentially lead to more accurate song identification.

- **Robustness to Partly Performed Songs**:
  Using dynamic programming, the search algorithm could be enhanced to identify the part of lyrics that has actually been performed, thereby improving the accuracy of the identification process.

- **Audio Segmentation**:
  The introduction of a pre-analysis phase, which carries out an audio-based song segmentation to correctly split the audio into distinct songs, could be beneficial. This could be a promising area of exploration. Consequently, the need for lyric-based song segmentation

would be eliminated. A comparison between both song segmentation techniques could then be conducted to determine the most effective solution.

- **Stem-level Identification**:
  By incorporating non-vocal stems of the audio into the analysis, stem-level audio fingerprint-based song and setlist identification could be possible. This approach could be especially useful for the cover song identification task.

- **Audio Quality Resilience**:
  Resilience to poor audio quality could be enhanced by exploring different pre-processing steps to improve audio quality or incorporating low-quality audio samples into the fine-tuning dataset.

- **Live Performance Analysis**:
  For faster performance during live concerts, the analysis of the performance could occur in a streaming manner, allowing the setlist to be created as the performance is ongoing.

These future directions would enable the system to overcome its current limitations and evolve into a more reliable, accurate, and efficient tool for the automatic setlist identification as well as the automatic lyric transcription task.

7 Recommendations

This section provides recommendations for the utilization of LyricSIS, considering its limitations, the results of the evaluation, and the earlier defined potential business cases. These suggestions will guide the practical application of the system and provide insights into the system's usability or potential shortcomings for the envisaged business use cases.

- **Consideration of system limitations for setlist identification**:
  Given the system's limitations, particularly the achieved True Positive Percentage (TPP) and False Positive Percentage (FPP), it should not be used as a standalone system for setlist identification tasks. If the TPP is less than 100%, some songs might be missed, leading to incomplete setlists. Similarly, an FPP greater than 0% could indicate the presence of incorrectly identified songs. These inaccuracies can have severe consequences, particularly in applications involving copyright infringement detection or royalty distribution. Users of the system must be aware of these limitations and consider the potential risks or consequences accordingly.
- **Augment system with audio-based fingerprints and apply fine-tuning**:
  To improve the system's accuracy, robustness, and speed, future work should prioritize applying fine-tuning to the Whisper model with a singing dataset and augmenting the system with audio-based fingerprints, as soon as a music database is accessible. This approach can enhance the overall song identification process, making it more accurate, faster, and more robust against varying audio qualities, and enable identification of songs without lyrics.
- **Utilize system for general audio pre-processing**:
  Based on the design choices I made, additional use cases can be exploited. The system's modular step-by-step design allows it to be used for general audio pre-processing tasks. For instance, it can generate song stems, enhance speech, or detect voices in the audio. With the evolving trend towards stem licensing in song copyright, the source separation capability of the system can be a valuable asset for the company to gain a competitive edge in the music industry.
- **Prioritize future work based on the use case**:
  The system's future development should be guided by the specific use case at hand. If the primary use case involves setlist creation for live performances, the focus should be on improving the system's robustness to audio quality. For applications like user-generated content (UGC) upload filters to enhance copyright infringement detection, improving the

system's speed should be prioritized. If the use case demands high overall accuracy, a balanced approach focusing on both speed and accuracy, such as fine-tuning a smaller model, should be considered. If the use case relys only on generating lyrics with the ALT module, the improvement of this module should be focused on.

These recommendations aim to guide the effective use and future development of the system, ensuring that it meets the needs of the business use cases while taking into consideration the current limitations. By following these recommendations, the company can maximize the benefits derived from the system, reduce potential risks, and contribute to creating a more equitable and transparent music landscape.

Bibliography

[1] Cooke, C.: *Dissecting the digital dollar.* London, 2018.

[2] Phonographic Industry(IFPI), I. F. of the: *Global Music Report 2023.* 2023. url: `https://globalmusicreport.ifpi.org/`.

[3] Inc., P.: *Unclaimed Royalties.* 2023. url: `https://pex.com/blog/`.

[4] Simard, S.: *Interspecific carbon transfer in ectomycorrhizal tree species mixtures.* In: (1995).

[5] Vaglio, A. et al.: *The Words Remain the Same: Cover Detection with Lyrics Transcription.* In: *22nd International Society for Music Information Retrieval Conference ISMIR 2021.* 2021.

[6] Gu, X. et al.: *Mm-alt: A multimodal automatic lyric transcription system.* In: *Proceedings of the 30th ACM International Conference on Multimedia.* 2022, pp. 3328–3337.

[7] Gao, X. et al.: *Automatic lyrics transcription of polyphonic music with lyrics-chord multi-task learning.* In: IEEE/ACM Transactions on Audio, Speech, and Language Processing 30 (2022), pp. 2280–2294.

[8] Ou, L. et al.: *Transfer learning of wav2vec 2.0 for automatic lyric transcription.* In: *Proc. ISMIR.* 2022.

[9] Ltd, S. E. 2023. url: `https://www.shazam.com/home`.

[10] Inc., S. 2023. url: `https://www.soundhound.com/`.

[11] BV, D. M. 2023. url: `https://djmonitor.com/`.

[12] Yesiler, F. et al.: *Investigating the efficacy of music version retrieval systems for setlist identification.* In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE. 2021, pp. 541–545.

[13] Wang, J.-C. et al.: *Automatic Set List Identification and Song Segmentation for Full-Length Concert Videos.* In: *ISMIR.* 2014, pp. 239–244.

[14] Abrassart, M. et al.: *And what if two musical versions don't share melody, harmony, rhythm, or lyrics?* In: arXiv preprint arXiv:2210.01256 (2022).

[15] Radford, A. et al.: *Robust Speech Recognition via Large-Scale Weak Supervision.* 2022. doi: `10.48550/ARXIV.2212.04356`. url: `https://arxiv.org/abs/2212.04356`.

[16] Livingstone, S. R. et al.: *Acoustic differences in the speaking and singing voice.* In: *Proceedings of Meetings on Acoustics ICA2013.* Vol. 19. 1. Acoustical Society of America. 2013, p. 035080.

[17] Defossez, A. et al.: *Real time speech enhancement in the waveform domain.* In: arXiv preprint arXiv:2006.12847 (2020).

[18] Kim, J.: *Discussion: VAD #96.* 2023. url: `https://github.com/openai/whisper/discussions/96`.

[19] Takahashi, N. et al.: *D3net: Densely connected multidilated densenet for music source separation.* In: arXiv preprint arXiv:2010.01733 (2020).

[20] Kong, Q. et al.: *Decoupling magnitude and phase estimation with deep resunet for music source separation.* In: arXiv preprint arXiv:2109.05418 (2021).

[21] Hennequin, R. et al.: *Spleeter: a fast and efficient music source separation tool with pre-trained models.* In: Journal of Open Source Software 5.50 (2020), p. 2154.

[22] Luo, Y. et al.: *Music Source Separation with Band-split RNN.* In: arXiv preprint arXiv:2209.15174 (2022).

[23] Stoller, D. et al.: *Wave-u-net: A multi-scale neural network for end-to-end audio source separation.* In: arXiv preprint arXiv:1806.03185 (2018).

[24] Défossez, A. et al.: *Demucs: Deep extractor for music sources with extra unlabeled data remixed.* In: arXiv preprint arXiv:1909.01174 (2019).

[25] Défossez, A.: *Hybrid spectrogram and waveform source separation.* In: arXiv preprint arXiv:2111.03600 (2021).

[26] Rouard, S. et al.: *Hybrid transformers for music source separation.* In: *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE. 2023, pp. 1–5.

[27] Ephraim, Y. et al.: *Speech enhancement using a minimum-mean square error short-time spectral amplitude estimator.* In: IEEE Transactions on acoustics, speech, and signal processing 32.6 (1984), pp. 1109–1121.

[28] Ephraim, Y. et al.: *Speech enhancement using a minimum mean-square error log-spectral amplitude estimator.* In: IEEE transactions on acoustics, speech, and signal processing 33.2 (1985), pp. 443–445.

[29] Kim, M. et al.: *iDeepMMSE: An improved deep learning approach to MMSE speech and noise power spectrum estimation for speech enhancement.* In: *Proc. Interspeech.* Vol. 2022. 2022, pp. 181–185.

[30] Krawczyk-Becker, M. et al.: *On speech enhancement under PSD uncertainty.* In: IEEE/ACM Transactions on Audio, Speech, and Language Processing 26.6 (2018), pp. 1144–1153.

[31] Nicolson, A. et al.: *Deep Xi as a front-end for robust automatic speech recognition.* In: *2020 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE).* IEEE. 2020, pp. 1–6.

[32] Park, T. J. et al.: *A review of speaker diarization: Recent advances with deep learning.* In: Computer Speech & Language 72 (2022), p. 101317.

[33] al., L.-A. et: *SILERO VOICE ACTIVITY DETECTOR.* 2023. url: `https://pytorch.org/hub/snakers4_silero-vad_vad/`.

[34] Babu, A. et al.: *XLS-R: Self-supervised cross-lingual speech representation learning at scale.* In: arXiv preprint arXiv:2111.09296 (2021).

[35] Bapna, A. et al.: *mslam: Massively multilingual joint pre-training for speech and text.* In: arXiv preprint arXiv:2202.01374 (2022).

[36] Morris, A. C. et al.: *From WER and RIL to MER and WIL: improved evaluation measures for connected speech recognition.* In: *Eighth International Conference on Spoken Language Processing.* 2004.

[37] Gupta, C. et al.: *Automatic lyrics alignment and transcription in polyphonic music: Does background music help?* In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE. 2020, pp. 496–500.

[38] Stoller, D. et al.: *End-to-end lyrics alignment for polyphonic music using an audio-to-character recognition model.* In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE. 2019, pp. 181–185.

[39] Hansen, J. K. et al.: *Recognition of phonemes in a-cappella recordings using temporal patterns and mel frequency cepstral coefficients.* In: *9th Sound and Music Computing Conference (SMC).* 2012, pp. 494–499.

[40] Mauch, M. et al.: *Integrating additional chord information into HMM-based lyrics-to-audio alignment*. In: IEEE Transactions on Audio, Speech, and Language Processing 20.1 (2011), pp. 200–210.

[41] Demirel, E. et al.: *MSTRE-Net: Multistreaming acoustic modeling for automatic lyrics transcription*. In: arXiv preprint arXiv:2108.02625 (2021).

[42] Pratap, V. et al.: *Scaling Speech Technology to 1,000+ Languages*. In: arXiv preprint arXiv:2305.13516 (2023).