University of Glasgow | School of Computing Science

Honours Individual Project Dissertation

Automatic Illustration of Text via Multimodal Interaction

Stergious Aji (2546916A)

March 24, 2023

Abstract

Automatic videography systems pose a serious challenge for effective performance evaluations. Often, this is limited to qualitative assessments with capricious human participants leading to biased and non-reproducible results. To address this issue, this paper proposes a two-part system. The first part is a new autonomous videography tool that leverages multimodal representations of both images and text to automatically illustrate spoken language in audio sources. The second half involves a novel ground-truth construction interface intended to streamline the creation of test collections for the quantitative evaluations of automatic videography systems. This interface intends to solve the challenge of evaluating videography tools by allowing the swift calculation of standard Information Retrieval metrics such as Average Precision and Recall. Importantly, these metrics permit fair comparisons with alternate systems, which was previously infeasible with subjective user evaluations. The final implemented system successfully produced a user-friendly interface that eases relevance assessment effort. Experiments also demonstrate that the new system enhances many aspects of the preceding videography system.

Education Use Consent

I hereby grant my permission for this project to be stored, distributed and shown to other University of Glasgow students and staff for educational purposes. **Please note that you are under no obligation to sign this declaration, but doing so would help future students.**

Signature: Stergious Aji Date: March 24, 2023

Contents

1 Introduction

This chapter serves to outline the motivations for creating an automatic videography tool, as well as the rationale for a subsequent interface for annotating ground-truth, specifically using multimodal representations of images and text.

1.1 Motivations

1.1.1 Automatic Videography of Audio

Text-to-image retrieval is a topic that has gained increased interest in recent years since the advent of new deep-learning approaches that have outperformed prior work (Gordo et al. 2017). These advancements have illuminated new innovations in a number of areas, particularly automatic content creation (Singer et al. 2022). Furthermore, the rise in video production (Team Pepper 2022) has also demanded a drive for more automated systems, especially in the realm of music. However, on popular platforms such as YouTube (2005), music often lacks any accompanying video content which can be detrimental to its overall prosperity. This can be due to various reasons such as a lack of funding, resources, or song popularity. More commonly, this issue arises from music tracks that were originally recorded in analogue formats which have since been digitised for upload to the platform.

Moreover, the proposed tool has applications beyond music tracks, as it can effectively enhance educational materials and podcasts. According to the study by Yeh (2018), participants who viewed video lectures consisting of both visual and auditory content performed better on the post-exam than those who watched lectures with only one of those elements. As a result, incorporating multiple modes of information such as visual, audio, and textual, can increase the accessibility of the subject and improve user retention. In addition, this could help cater to diverse learning styles such as those identified in Fleming's VARK acronym (Fleming 2006).

One of the main motivations of this project is to build on and enhance previous work on the Automatic Videography of Audio Tracks of Songs by Parker (2022). Parker's thesis achieved a functioning video generation tool for audio tracks sourced from YouTube. However, the system suffered from a number of drawbacks that we seek to address with this project. A more in-depth review of the thesis is conducted in Section 2.3.1 of the Background.

1.1.2 Ground-Truth Annotation Interface

With an evergrowing supply of automatic videography systems, it is important to assess and contrast their respective capabilities. Nevertheless, we have found that there is a lack of objective procedures for evaluating comparative performances between videography systems, particularly in terms of measuring their accuracy at image retrieval from text. Image relevance can and has been assessed with qualitative user surveys but this can often be very subjective and sensitive to personal preferences and present moods (Mohammadi et al. 2014). More importantly, if the objective is to compare against the outcomes of a previously evaluated system, additional user studies must be carried out using the same demographic of individuals to evaluate new systems

(Voorhees et al. 2005). Not only is this approach impractical and time-consuming, but it also leads to results that are unrepeatable and biased.

This can be mitigated if instead, a set of true labels are annotated in advance for provided audio sources which can then be used to evaluate multiple videography systems on those same audio sources. It is vital that a static image collection is used for ground-truth constructing and evaluating in order for the production of repeatable and standardised performance metrics.

This called for a user-friendly interface to assess the relevance of images from the textual content present in given audio sources. Under the hood, this ground-truth constructor interface can theoretically use the same pipeline as the automated videography system that we are already creating.

1.2 Aims

This project aims to produce a piece of software that can take any audio source and automatically generate a video from it. The video will consist of time-aligned imagery corresponding to the spoken words in the audio. In addition, this software will include a novel annotation interface for assessing image relevance from text prompts. This will let the user construct a set of ground-truths for each portion of the supplied audio source. This will be achieved by letting the annotator select the set of images that they perceive as the most fitting to the text or song lyric in a given chunk of audio. This set of images will be picked from a top-k list retrieved by our system.

Since there is a high priority for precise ground-truth construction, the software must be intuitive to use with clear instructions for the assessor to follow. It is imperative, therefore, that we strive to develop a platform with a user-friendly interface that is widely accessible. Additionally, since the aim is for anyone to use this application, appropriate error handling must be in place to accommodate users from a broad range of technical backgrounds. We also aspire to make our system modular in design. This way, more technical users would be able to swap in any image collection with minimal configuring if required.

At the end of this project, we endeavour to provide a set of completed ground-truth for a pre-defined list of audio sources using our annotation interface. This data will be made available in an accessible format so that it can be used to measure the quality of existing and future videography systems using standard Information Retrieval (IR) metrics like Average Precision (AP), F-Measure, and Normalised Discounted Cumulative Gain (nDCG).

Finally, as stated in our motivations, we wish to fix the limitations of the previous video generation system as well as extend its functionality. Our objective is to expand the application of the tool to any audio source, rather than restricting it to the YouTube platform. This also means, our tool must be able to support the transcription of multiple languages and accents. Finally, we intend to tackle the image retrieval process using a different approach that introduces semantics in order to retrieve more relevant images.

2 Background

This chapter serves to lay the foundations in order to contextualise this project within the field of Information Retrieval (IR). Firstly, a look into the origins of the field, specifically image retrieval, will be conducted. This will be followed by a discussion of the recent advancements in computer vision to allow multimodal representations of images with natural language. We will briefly look into the history of relevance assessment to ground our problem of evaluating IR systems and justify our proposed approach. Next, the previous thesis on automatic videography will be critiqued, with the aim of identifying its merits and shortcomings. This will form the basis upon which our project will build. Finally, an outline of what current solutions exist to solve our problem of assessing the relevance of images from text queries will be provided.

2.1 The History of Image Retrieval

Image retrieval is an important branch within the field of IR which has evolved over the past several decades. Recent years have witnessed an exponentially growing collection of images, spanning various domains such as academia, medicine, social media, and the military. This makes it all the more important that efficient mechanisms are in place to scour these databases. The earliest such mechanisms can be traced back to the 1970s. Known plainly as Text-Based Image Retrieval (TBIR) systems, they relied on keyword-based querying of a relational database (Chang and Fu 1979). Consequently, every image in the database had to be labelled with appropriate keywords and a description. This was a popular framework at the time, but with the exponential growth of datasets, manual annotation became infeasible. Moreover, the results were often of low quality due to the limited ability of the systems to accurately interpret the user's intent; not to mention, the subjectivity of human perception during annotation.

Fortunately, researchers at the time also began looking into the application of computer vision to solve the task. The key idea involved extracting visual features from the images like colour, texture, and shape and utilising these to index them into the database. This made it possible to use images as the search criteria in order to find images with similar features (Chang and Kunil 1981). This approach later became known as Content-Based Image Retrieval (CBIR) as better image processing techniques emerged in the 1990s allowing easier extraction of features. Furthermore, as the capability to extract additional visual characteristics improved, images started being represented in increasingly higher dimensions (Zheng et al. 2017). This often led to data sparsity problems in which the movement into higher dimensions caused similar data points to drift apart as the volume of the space grew too quickly for the data to keep up. A well-known phenomenon coined as the *"curse of dimensionality"* (Bellman 1957).

In the 2000s, researchers developed more sophisticated approaches for text-to-image retrieval, incorporating techniques from natural language processing (NLP) and computer vision. This led to the development of more advanced retrieval models that could accurately match textual queries with semantically relevant images. Some notable works in this period include the Language of Pictures system by Lavrenko et al. (2003), the Cross-Media Relevance Models (CMRM) by Rasiwasia et al. (2010), and the Text-Guided Attention Model (TGAM) by Gao et al. (2018).

The field of image retrieval underwent a significant revolution in the late 2010s with the emergence of deep learning-based approaches. These models are trained using large datasets of text and images and can learn complex semantic representations that capture the meaning of both the textual queries and the visual content of the images. Some notable works in this period include the Deep Visual-Semantic Alignment (DVSA) model by Karpathy and Fei-Fei (2015) and the Neural Image Caption (NIC) model by Vinyals et al. (2015). This even spawned new areas of research with the concept of image synthesis using AI as seen in the StackGAN model by Zhang et al. (2017). This brings us to the latest developments with OpenAI and CLIP.

2.1.1 CLIP: Transforming the Future of Text-to-Image Retrieval

The year 2021 marked the release of OpenAI's Contrastive Language-Image Pre-Training (CLIP) model and paper (Radford et al. 2021). This is an effective image-text embedding model based on a transformer architecture. Transformers are a type of deep neural network that was first introduced in a seminal paper by Vaswani et al. (2017). They were originally developed for natural language processing tasks leading to several state-of-the-art, pre-trained systems like BERT (Devlin et al. 2018) and GPT (Radford et al. 2018). However, it has since been adapted to other domains, most notably computer vision (Dosovitskiy et al. 2020). These advancements laid the crucial groundwork for the emergence of CLIP.

The CLIP model uses a transformer architecture that is similar to the one used to create the GPT-2 language model (Radford et al. 2019). The model consists of a stack of transformer encoder layers, each of which consists of a self-attention mechanism followed by a feedforward network. The self-attention mechanism allows the model to attend to different parts of the input sequence and learn relationships between them, while the feedforward network applies non-linear transformations to the input features. The transformer encoder layers are used to process both the text and image inputs, allowing the model to learn a multimodal representation that captures the relationship between them. The text input is tokenized using a byte-pair encoding (BPE) scheme, and the resulting tokens are embedded into a high-dimensional vector space. Similarly, the image input is processed by a series of convolutional layers, followed by a global average pooling layer that produces a feature vector that is in the same vector space as the text.

The joint text-image representation is learned by attending to both the text and image features using the self-attention mechanism in the transformer layers. The model is trained to predict whether a given image and text pair belong together, using a contrastive loss function that encourages similar pairs to be closer in the representation space than dissimilar ones. This training objective allows the model to learn a rich representation of both text and images, which can be used for a wide range of downstream tasks. This flexibility is what makes CLIP more appealing than previous models.

The CLIP model was trained on a dataset of over 400 million images and their associated text descriptions. It achieved state-of-the-art performance on a variety of image classification benchmarks, including ImageNet, COCO, and CIFAR-100. One of the key advantages of the CLIP model is its ability to generalise to new tasks and domains without requiring further training. For example, the authors found that the model was able to perform well on a variety of tasks, such as zero-shot classification, image retrieval, and visual reasoning, without any additional fine-tuning. CLIP achieved an impressive top-1 accuracy of 76.3% on the ImageNet zero-shot benchmark, where the goal is to classify images into novel categories, unseen during training. A significant improvement on the previous state-of-the-art which achieved 11.5% accuracy.

Overall, the results demonstrate that the CLIP model is capable of learning a rich and flexible representation of both text and images, which allows it to perform well on a broad range of tasks, specifically text-to-image retrieval. The model also has strong generalisation capabilities, which makes it well-suited for applications where data is often scarce or noisy. This makes CLIP an attractive choice for a restrictive project such as this one.

2.2 Relevance Assessment

One of the main objectives of this project is to develop an interface to construct a set of known relevant images from text queries. However, this concept of relevance assessment has a deep history involving a variety of viable strategies so this must be investigated in order to find one most suited to achieving our goal.

Relevance assessment is an integral aspect of IR systems that determines the relevance of documents to a user's query. These documents, while most commonly associated with textual data, can encompass various other forms of data including but not limited to images, videos, and music. In IR, relevance refers to the degree to which a document satisfies a user's information need. Relevance assessment is essential in evaluating IR systems as it ensures that the search engine retrieves only the most pertinent documents and presents them to the user in an appropriate order.

2.2.1 Origins

The concept can be traced back to the early days of IR amidst the rapid expansion of computerised catalogues for military, governmental, and educational purposes in the 1950s. With many different classification and indexing strategies emerging, it became increasingly unclear which was the most effective. This was until the Cranfield experiments conducted by Cleverdon (1967) at Cranfield University, England. These experiments pioneered one of the most widely used evaluation frameworks within IR including fundamental techniques such as query formulation and evaluation metrics. One of the key innovations within the Cranfield paradigm was the development of the relevance assessment methodology. This was the idea of using a fixed test collection of queries designed to represent a wide range of information needs, and manually assessed relevant documents for each query. This test collection is then used to evaluate the effectiveness of different IR systems in retrieving those relevant documents. The experiments also introduced the first IR evaluation metrics, including precision and recall, which are still widely used today.

During the early days of IR, it was common to use binary relevance assessments, a simple indication of whether a document is relevant or not to a particular query. In the 1970s, researchers explored the use of graded relevance assessments, which allow assessors to assign scores to documents based on their degree of relevance to a query (Salton 1971; Robertson and Jones 1976). While this method provides a more informative evaluation measure, it greatly increases annotation effort and the subjectivity of judgment.

2.2.2 Techniques

Various techniques have been proposed for performing relevance assessments, including manual assessments, crowdsourcing, and automated techniques:

- Manual assessments involve human assessors who painstakingly look at every document and assign relevance scores based on predefined criteria and queries. This is a very time-consuming and expensive process, often requiring expert assessors with specialised knowledge to be able to judge documents accurately.
- Crowdsourcing, on the other hand, involves an open call for a large group of people to assess documents (Samimi and Ravana 2014). This can be effective, in generating extensive test collections especially when the documents do not belong to a specific domain of expertise. It also captures a wider range of viewpoints as it brings in assessors from diverse backgrounds. Contrastingly, this can equally bring in bias depending on the assessors' incentives and personal preferences. Finally, this technique makes quality control especially difficult, leading to the potential for fraudulent assessments.

- More recently, machine learning approaches are being used to predict the relevance of documents based on their content and other features (Cao et al. 2007). This technique is much faster and more scalable compared to manual annotation, producing more consistent results across multiple evaluations. However, this technique often lacks the flexibility to adapt to changing evaluation criteria or different types of data. Furthermore, such a system still needs human validation to ensure its predictions are accurate, often leading to a more semi-automated approach.

While many more techniques have been developed, manual assessment is still the gold standard in IR evaluation. However, with the exponentially growing size of document collections in the modern day, it is infeasible for humans to judge every document in the corpus for each query. Therefore, strategies such as **pooling** and **active learning** (Rahman et al. 2020), are utilised in order to reduce annotation effort.

2.2.3 Depth-Pooling

Pooling is a popular strategy used for very large-scale collections of documents, in order to optimise the relevance assessment process. It involves creating a pool of the top-k documents, for a particular query, retrieved from a number of different retrieval systems. The key idea here is that this k, which is commonly kept at a constant depth for each query, is much smaller than the size of the corpus. This means that the assessors must only judge the relevance of this modest subset of documents while all the others can be labelled as non-relevant. Trying to further optimise the pooling strategy itself is an evolving area of research. Papers such as Arampatzis et al. (2009) and Lien et al. (2019) propose ways for determining the optimal cut-off point for the number of ranked documents that would satisfy a user's information need by examining the statistics of score distributions.

2.2.4 Active Learning

Active learning (AL) involves a machine learning approach to iteratively optimise its retrievals by posing queries to a human. These queries are typically in the form of unlabelled data which the human can annotate. The learner can then use this knowledge to find the user's specific information need (Burr 2012). While this approach can be effective in identifying relevant documents, we believe it is not suited for our task. AL is more appropriate for sophisticated supervised learning tasks in which acquiring labelled instances is costly, such as specialised information extraction or speech recognition. In our case, a simple binary relevance assessment will suffice.

2.3 Prior Work on Multimodal Assessment

Before starting to build our system, it is important to look at similar existing solutions to our problem. This section will start off with a deep dive into the previous thesis in order to identify which parts of it we want to build on. Following this, we will focus on the novel annotation interface that we plan to develop by looking at two solutions that could solve a similar problem to the one described in this project. By analysing the design and workflow structure that these solutions employ, we can gather ideas and paint a clearer picture of the interface we want to develop. From the research done here, it was evident that most annotation software provide a wide range of different functionality and tools. While this is certainly a positive in giving users as much freedom as possible, it can equally be a drawback. In most scenarios, users want to focus on a single task, and therefore attempting to learn and navigate a large versatile application can be confusing and a waste of time and resources.

The two relevance assessment software we chose to look at are **Labelbox (2018)** and **SuperAnnotate (2017)**. Both pieces of software are similar in aspect to the annotation tool proposed in this project. However, they each have advantages and limitations which further motivate the requirement for our application.

2.3.1 Automatic Videography of Audio Tracks of Songs

Parker (2022) supplied an automatic videography system that generated videos for music tracks sourced from YouTube videos. The project aimed at providing a tool to ease music video creation and investigate the effects of using multimodal information in enriching songs.

Achievements The project achieved a simple web interface, shown in Figure 2.1, where users can supply a YouTube video URL, the artist name and song title as well as the alignment method. On submission, the system will download the video and subsequently isolate the audio and video content. The next steps will depend on the chosen alignment method, so each will be explained one by one. There are three alignment methods available:

- **Forced Alignment:** This method uses a forced alignment model in order to synchronise the retrieved lyrics of the song to the extracted audio. The lyrics are retrieved from the Genius (2009) database using the supplied song title and artist. A word-level dictionary of timings is returned which is used to place relevant imagery at the correct timestamps during video compilation.
- **Lyrical Analysis:** This method only functions for music lyric videos since it works by extracting textual information from the frames of the video using an Optical Character Recognition (OCR) tool. This technique produces phrase-level timings.
- **Caption Extraction:** This method extracts the caption information from the YouTube video directly providing phrase-level alignment information. However, the feasibility of this method hinges on the availability of the captions.

The system scrapes the relevant images from Google Images. However, for this approach to work best, images should be searched by keywords. This requires extracting at most two keywords for each segment within the transcript, which this project accomplishes by leveraging YAKE!: Yet Another Keyword Extractor (Campos et al. 2020). Finally, the alignment information and scraped images are used to compile the final video which is presented to the user.

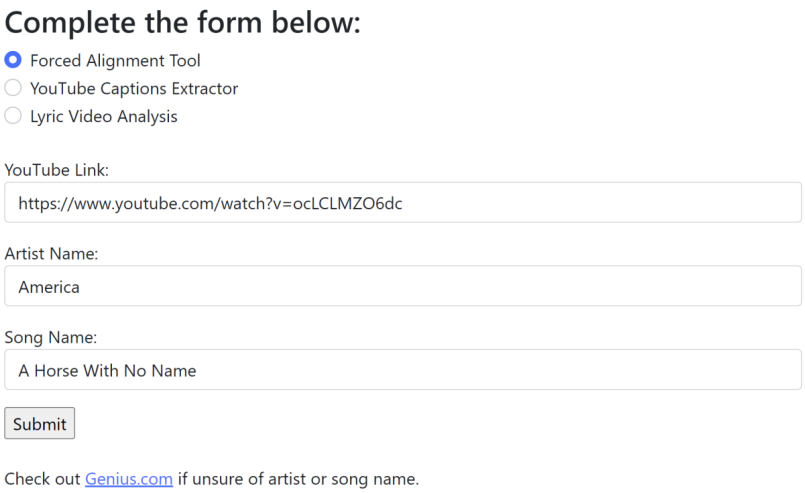


Figure 2.1: *An example of using the user interface from the preceding system*

Problems Parker's system made many achievements in generating complementary videos for audio tracks. However, it was restricted to the YouTube platform and emerged with a number of unforeseen drawbacks.

To begin with, the metadata of YouTube videos was deemed unreliable in procuring the artist's name and the song title. Therefore, the system resorts to obtaining this information from user input. However, this can be problematic since it requires the user to undertake additional research and accurately type the information, which can be prone to errors.

The prior system was also heavily reliant on several Application Programming Interfaces (APIs) such as YAKE! for keyword extraction, pytesseract (Lee 2007) for OCR of frames, and Google Images for scraping images just to name a few. This places a high dependency on these APIs which may not be consistently maintained or may cause version conflicts with other packages. In fact, the AutoLyrixAlign API (Gupta et al. 2020), utilised for the Forced Alignment approach has since been deprecated at the time of writing.

Additionally, the system suffered from recurring instances of irrelevant images being retrieved. This can be attributed to the disregard of the semantic context of the keywords during the search process. Furthermore, keywords with multiple meanings in the English language often yielded incorrect images.

The ultimate problem identified, which directly prompted our idea of ground-truth construction, stemmed from the evaluation phase of the project. The user surveys exhibited a mixed, albeit predominantly favorable assessment of the system. Nevertheless, the significant variance in ratings and responses to the open-ended questions emphasised the arduousness of conducting qualitative evaluations on automated videography systems. Furthermore, such evaluations produced non-repeatable outcomes.

Improvements Now that the main problems have been identified, we must present solutions so that we can gather the appropriate requirements for this project. The first goal is to generalise the system to any type of audio instead of being restricted to English source material. Adding support for multiple languages would further our goal of increased user accessibility to content. Moreover, this would allow easier content creation for many resource-poor languages.

First of all, let us tackle the metadata problem. We want to avoid relying on the user for metadata information for the audio source which means we must acquire this from somewhere else. This data was primarily used to query the Genius (2009) database in order to retrieve the lyrics transcription for songs so this data is not required for non-musical sources. Therefore, in order to identify the song title and artist, we could use the audio itself as the query. Shazam (Wang 2006) is an application that can recognise music and movies by listening to a short sample of the audio. It does this by attempting to match the audio's time-frequency signature to one stored in its large catalogue of audio fingerprints to get the correct track information. While this approach would mean an additional API, its benefits of increased automation outweigh the drawbacks.

In order to retrieve more relevant images, it is clear that semantics must be introduced instead of relying on keyword-based searching. Upon researching the recent advancements with CLIP (2.1.1), we believe this is a viable solution that can mitigate the number of non-relevant images being retrieved. Additionally, the quality and video generation time must be improved when moving forward with this project.

Lastly, we believe our novel ground-truth annotation interface will address the difficulty of evaluating current and future automatic videography systems. The constructed relevance assessments allow for the application of the Cranfield paradigm and computing standard IR metrics. These quantitative measures permit impartial performance comparisons between different systems. More importantly, the experiments can be conducted rapidly and repeatedly without concerns about result consistency.

2.3.2 Labelbox

Labelbox is a cloud-based platform for data labeling and annotation. It is used by many companies to build and improve machine learning models. Founded in 2018, it offers a range of annotation tools including image classification, object detection, and segmentation. Its image classification tool allows users to label images with relevant tags or categories, such as "dog" or "cat". For our specific task, Labelbox allows users to query for specific concepts using natural language, which returns a ranked list of relevant images. The user can then select the most appropriate images by checking a box and saving them for export. An example of this process can be seen in Figure 2.2, where an annotator marks relevant images for the concept, *"painting of cocktails"*.

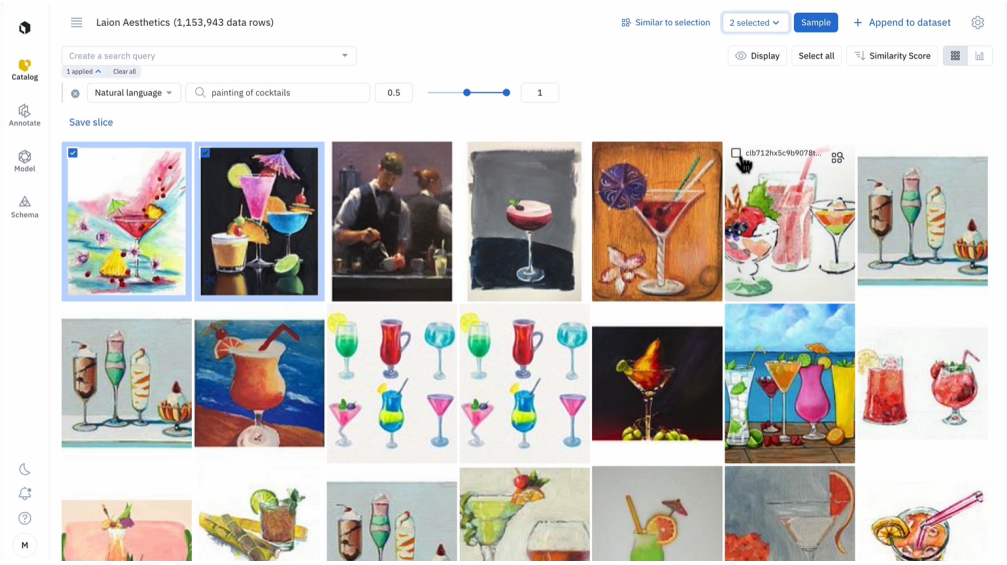


Figure 2.2: *Relevance assessment of the concept "painting of cocktails" using the Labelbox interface*

One of the most appealing features that Labelbox offers is allowing users to customise the embedding model for similarity searching and its aesthetic hyperparameter, as illustrated in Figure 2.3. Currently, Labelbox only supports the CLIP ViT-B/32 model for images. However, the platform is designed to be compatible with any model, which can be configured using their well-documented API.

Figure 2.3: *Labelbox providing the option to customise the image-text embedding model*

The platform also integrates with popular machine learning frameworks and cloud services, allowing companies to streamline their data labeling workflows and accelerate model development. Despite the platform's slick design, getting acquainted with the application takes time, just due to the sheer multitude of available features and tools.

2.3.3 SuperAnnotate

Founded in 2017, SuperAnnotate is another cloud-based platform for computer vision and machine learning, designed to help users create high-quality image datasets for training and

testing models. The platform allows the annotation of images with bounding boxes, polygons, or keypoints, among others, as well as the ability to collaborate with other annotators on the fly. The platform has since grown rapidly and has been used by companies and researchers across diverse industries including autonomous vehicles, medical imaging, and agriculture.

SuperAnnotate is not just restricted to images, it also offers tools for annotating audio, video, LiDAR, and any custom data formats like PDFs or HTML. However, the platform's comprehensive feature set is only accessible via a paid subscription model, which may not be ideal for individuals who require a single tool such as image relevance assessing. Figure 2.4 depicts SuperAnnotate's interface to query images using natural language, similar to Labelbox (2.3.2). The interface also allows placing tags in order to filter the retrieved images by specific attributes such as traffic lights that only show red. However, unlike Labelbox, SuperAnnotate does not provide an off-the-shelf embedding model, necessitating users to feed a model themselves.

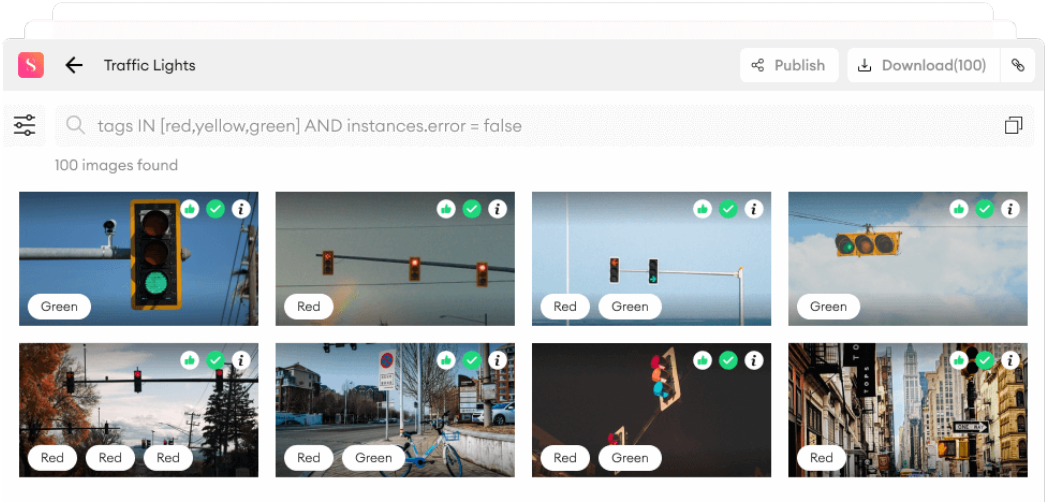


Figure 2.4: *Relevance assessment of the concept "Traffic Lights" using SuperAnnotate's interface*

2.4 Summary

Exploring the background of the project through a review of the history and past accomplishments within the field is essential to establish its relevance and reason for existing. Looking at current solutions to the problem and their contrasting approaches provide valuable insights into the direction we should take our project. This is made explicit by our explorative dissection of the previous system as we propose potential solutions for its limitations. The advent of OpenAI's CLIP and Whisper (Radford et al. 2022) models allow for a unique approach to the videography pipeline that could expand its current capabilities.

Regarding the annotation interfaces, both applications we reviewed share an intuitive interface design including clear checkmarks to indicate images tagged as relevant by the user. Nonetheless, even with the vast feature set that these systems offer, it still requires a lot of manual effort to complete the presented task in this project. To exemplify, the assessor must first listen and transcribe an audio source and subsequently input and annotate every segment within the transcript one by one. We believe our solution could streamline this procedure by automating the transcription process and making it easier to view and assess different audio segments within the interface itself.

3 Analysis/Requirements

This chapter serves to break down the high-level aims of the project in order to produce a list of functional and non-functional requirements, ordered by priority. This will directly determine the design and implementation of the project.

3.1 Requirement Elicitation

In order to elicit the list of requirements, a number of user scenarios are created. These are imagined scenarios of users that would directly benefit from such an application. This type of requirements gathering, most commonly used in agile software development projects, places the focus on the end-user, rather than the developer. Therefore, this prioritises the features that an end-user requires to fulfill their goals rather than designing the development process around what is easy to implement. Another benefit of this method is that it mitigates the risk of potential flaws at a very early stage. It is much easier to accommodate any flaws before any design or implementation is undertaken rather than finding them out during the evaluation stages with real users.

3.1.1 User Scenarios

User Scenario 1: Anna is a young secondary school teacher who has started recording her classes in response to the recent shift in online learning. She regularly posts her recordings to an online platform supported by the school. However, she has had a few complaints from her students that her recordings are not engaging enough and that it is difficult to see what she writes on the board. Anna strives to provide the best for her students and wishes she could make her recordings more engaging by illustrating the concepts she talks about in her audio recordings. However, she does not know anything about video production and is worried about the time required to learn and edit videos daily. She wishes for a tool that automatically illustrates her audio recordings in a swift manner. The created video should include captions to make it as accessible as possible for her students.

User Scenario 2: William has recently built two new automated video generation tools that use differing reranking procedures to retrieve the top images from the text. He has performed some user studies, where he asked a number of users to watch and rate the videos generated by both systems on the same audio tracks. He used A/B testing in order to compare the performances of his two systems. However, the user studies resulted in very mixed results, with some users rating system *A* highly while others rated system *B* as better. Furthermore, upon closer inspection, he found that the user ratings varied wildly depending on the audio tracks present in the videos. This was problematic since the user studies took over 2 hours to complete and William is unwilling to conduct another when it is unclear if the results would even improve with new participants. William wished that he could objectively evaluate his systems against a fixed test collection of ground-truths. He would be able to use this to obtain quantitative metrics for each system, giving a clear answer to which reranking procedure performs better.

User Scenario 3: Lucy is a software developer who is part of a small team building a new videography tool. Her manager has given her the tedious task of assessing the relevance of images for a predetermined set of audio from videos sourced from YouTube. This process requires manually annotating images that are the most fitting for the textual content existing within the audio. This is a very lengthy process since she must listen to the audio track multiple times in order to accurately transcribe the textual content. Following this, she must manually record the locations of the annotated images in a spreadsheet. Her manager has also assigned a colleague to double-check each annotation made. She wishes for a tool that automatically transcribes the audio and provides an easy-to-use interface to annotate relevant images. She requires that this ground-truth could be downloaded in a readable format for presenting to her manager. Additionally, it would be helpful to be able to edit the annotations after the fact, so that her colleague can fix any incorrectly judged images.

3.1.2 User Stories

- *As a user* of the application, *I want to* upload audio files, *so that* I can generate a video for it.
- *As a user* of the application, *I want to* specify YouTube video URLs, *so that* its extracted audio can be used as the source.
- *As a user* of the application, *I want to* build and export a test collections for audio sources, *so that* I can use them to evaluate my system.
- *As an user* of the application, *I want to* edit previously made annotations, *so that* I can fix any mistakes made.
- *As a user* of the application, *I want* to view the full audio transcription, *so that* I can annotate specific parts of the audio.
- *As a user* of the application, *I want to* view and download the generated video, *so that* I can share it and upload it to any platform.

3.2 Requirements

In order to formalise and prioritise the main requirements identified, the MoSCoW method (Clegg and Barker 1994) was chosen. While it would be desirable to develop every feature that could help achieve the project's goal, the project's limitations make it impractical. Therefore, it is crucial to assess each feature's importance by evaluating how much it would contribute to achieving the intended goal. Each feature will fall into one of four categories:

- **Must Have:** These are essential features that must be implemented to fulfill the project's aim.
- **Should Have:** These features are highly desirable, but missing them will not directly lead to a project failure.
- **Could Have:** These are features that are beneficial but are viewed as extra perks.
- **Won't Have:** These features are not worth pursuing.

The list is divided into two sections: Functional Requirements, which describe interactive features, and Non-Functional Requirements, which describe program properties that users cannot interact with. Requirements marked by an * were added at later stages in the project development as new feature ideas emerged.

3.2.1 Functional Requirements

1. **Must Have:** The ability to upload an audio file as an input audio source.
2. **Must Have:** The ability to specify a YouTube video URL as an audio source.

3. **Must Have:** The ability to view the full audio transcription.
4. **Must Have:** The ability to generate an illustrated video for any audio source.
5. **Must Have:** The ability to view the top-k retrieved images for a given audio chunk.
6. **Must Have:** The ability to annotate the most-fitting images for a given audio chunk.
7. **Must Have:** The ability to view the text present in a given audio chunk.
8. **Must Have:** The ability to view the constructed ground-truth data.
9. **Must Have:** The ability to download the constructed ground-truth data in a portable, human-readable format.
10. **Should Have:** The ability to view previously processed audio sources, generated videos, and constructed ground-truth.
11. **Should Have:** The ability to edit previously made ground-truths.
12. **Should Have:** The ability to upload audio sources of any language.
13. **Could Have:** The ability to load ground-truths for editing.
14. **Could Have:** The ability to reconfigure the static image collection.
15. ***Could Have:** The ability to view the percentage complete for ground-truth construction of processed audio sources.
16. ***Could Have:** The ability to save or skip chunks with repeating textual content when ground-truth constructing.
17. ***Could Have:** The ability to view the timestamp information of audio chunks.

3.2.2 Non-Functional Requirements

18. **Must Have:** The system using a static collection of images for repeatably consistent ground-truth construction.
19. **Should Have:** The application being accessible from any Operating System platform.
20. **Should Have:** The application having a presentable user interface that is easy for any user to understand and learn quickly.
21. **Should Have:** The application with appropriate error handling.
22. **Should Have:** The application being fast and responsive.
23. **Should Have:** The system being able to recognise music tracks to automatically retrieve the artist name and song title.
24. **Should Have:** The application being able to transcribe any audio source providing phrase-level timestamps.
25. **Should Have:** The system retrieving more relevant images by leveraging semantics of natural language as well as visual features.
26. **Should Have:** The videos being generated in a more reasonable amount of time than the prior system.
27. **Should Have:** The generated video having a higher resolution than the prior system.
28. **Should Have:** The application displaying progress bars for long loading times.
29. **Could Have:** The application being designed with extensibility in mind so that the image collection can be reconfigured.
30. **Could Have:** The videos including text as well as images.
31. **Could Have:** The application visually differentiating between previously processed audio sources.
32. **Could Have:** The system performing tempo or beat analysis to appropriately time image changes.

3.3 Summary

This chapter presented a detailed outline of the essential features for the project's development. The approach involved crafting user scenarios and stories to gain insight into potential user needs, thereby deriving a MoSCoW prioritised list of requirements. This process provides greater clarity for the ensuing stage of the project - Design.

4 Design

This chapter serves to provide the initial design choices including a high-level view of the system workflow and the wireframes of the intended user interface. All decisions will be explained with a general aim to optimise the user experience and fulfill the identified system requirements. Next, a deeper look into the design choices of the internal workings of components. This allows our system to be implemented in any language while maintaining clean and agile software design principles.

4.1 High-Level Plan of System

In order to start planning out the workflow of our system, it is necessary to simplify and abstract the key components that make up the backbone of the system. Our system is comprised of two sub-systems, an automatic videography tool, and a ground-truth annotation tool. However, both sub-systems use the same underlying pipeline. The simplified workflow diagram displayed as Figure 4.1 shows the basic, black-boxed components that make up our pipeline: a **transcription extractor**, a **video generator**, and an **annotation interface**.

The user inputs an audio source which is processed to extract a synced transcription of the audio content. Next, the user has the option to either choose to generate a video or start constructing the ground-truth for it. Each option produces an output that the user can subsequently view and export. In line with the requirements, these outputs are saved to the database for future editing.

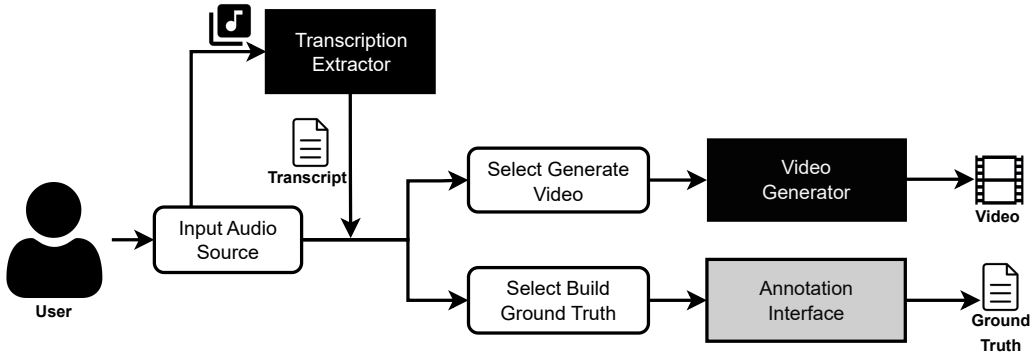


Figure 4.1: *Simplified workflow diagram showing the basic high-level design of the system*

4.1.1 Transcription Extractor

Firstly, the user inputs an audio source. As stated in the requirements, this can either be in the form of a YouTube video URL or a direct upload of an audio file. This input is passed along to the Transcription Extractor. This component produces a transcription of the audio with phrase-level timing information. It is vital that this transcription is synced accurately with the audio since this information will be used to place relevant images during video compilation.

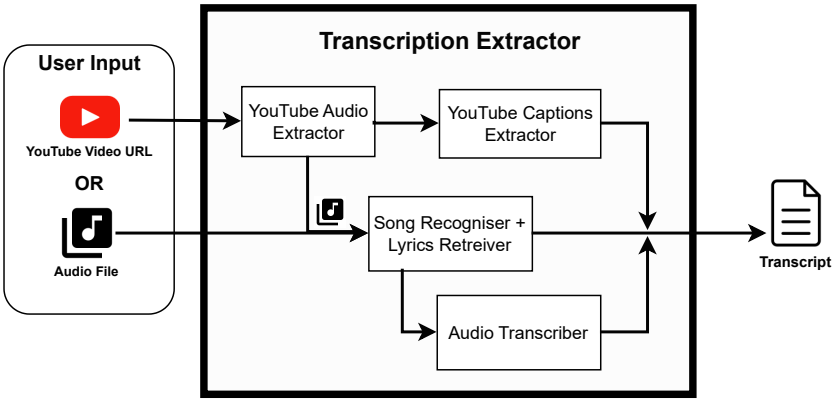


Figure 4.2: *Internal view of the Transcription Extractor component*

There are a number of ways to retrieve the synced transcription, firstly, we can leverage YouTube's captioning system if the user chooses to input a YouTube video URL. However, we cannot always rely on this since captions are not guaranteed to be present. Secondly, for music, it is possible to retrieve synced lyrics from open-source databases, which requires identifying the artist name and track title. In order to avoid the YouTube metadata problem from the previous thesis (2.3.1) and allow this to function for uploaded audio files, we must process the audio to recognise and retrieve the track information. Finally, we require a backup audio-to-text transcription tool for non-music audio sources or YouTube videos containing no captions. This workflow is depicted in Figure 4.2.

4.1.2 Video Generator

This component should activate when the user chooses to generate a video. It takes in the audio and the synced transcription which is divided into phrase-level chunks. Next, the videography tool should retrieve the top-1 relevant image for each chunk and sequence them into a video according to the timestamp information present. The audio should then be added to the rendered video which will subsequently be outputted for the user to view and export. Since, it is common for some phrases to be repeated, especially in music, we may want to retrieve different images for each repeated phrase. Our main goal is to produce a video that is engaging to the user. In these cases, we can simply retrieve the next relevant image in our collection.

4.1.3 Annotation Interface

The annotation interface consists of a system workflow of its own since the user will be actively interacting with the interface in order to incrementally construct the ground-truth for the audio. Figure 4.3 presents the workflow that an annotator would follow.

Since one of our requirements specifies a static image collection (Requirement **#18**), this gives the benefit of being able to pre-process the entire collection since we know that the images are not going to change. Additionally, as we want to include the semantics of text and images, we must embed all the images into a multimodal space into which text can also be embedded into.

Firstly, the annotator would select an audio chunk to assess. This freedom to choose which part of the audio to annotate is essential in order to facilitate the requirement of editing annotations (**#11**). However, it can be assumed that the annotator will construct the ground-truth in the order of the audio. Once a chunk is selected, the top-k images are retrieved and shown to the assessor by embedding the chunk text and performing a search in the multimodal vector space. The annotator is then expected to identify and label the images that they judge as the most fitting

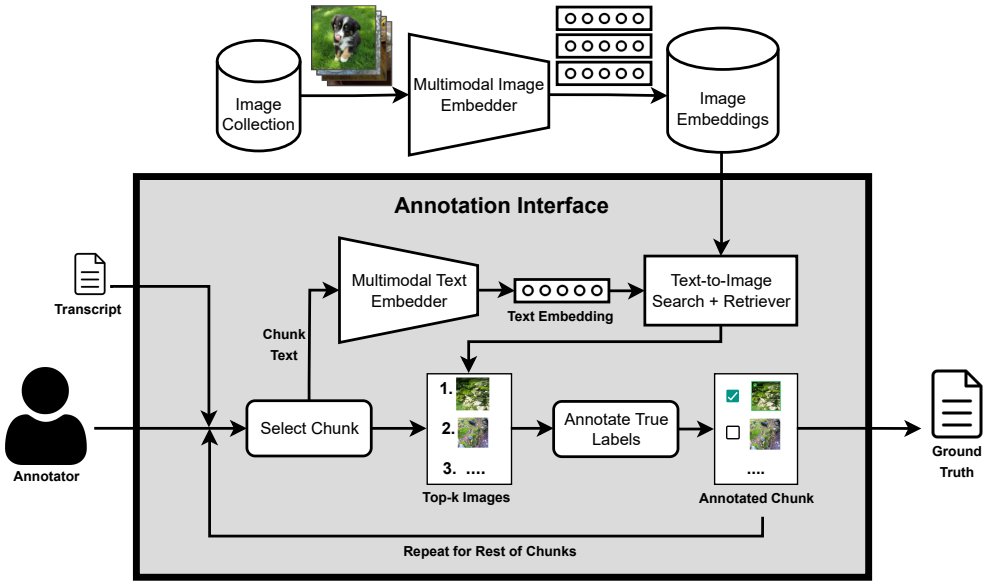


Figure 4.3: *Internal view of the Annotation Interface architecture*

to the chunk text. This procedure will be repeated for every chunk of the audio until a completed ground-truth set is constructed. This is subsequently outputted to the annotator in an accessible format.

4.2 User Interface Design

The user interface (UI) is a critical part of our application due to its importance within our task specification. Hence, ensuring our design was intuitive even for untrained users, to enable the freedom to crowdsource proved to be a challenge. Competing in the field with major annotation platforms only mounted more incentive to design an interface that is unique yet familiar. Furthermore, it was important to ensure pages within the UI remained uncluttered. For this reason, before any development commenced, we started prototyping interface designs using Figma (2016).

4.2.1 Home Page

The simplified workflow outlined in Figure 4.1 is used as the basis to form the various pages within the application. Firstly, we require a page for the user to enter the audio source to initiate the process. Hence, this 'Home' page requires a way for the user to upload an audio file from their local machine (**#1**), as well as a text field to take in a YouTube video URL (**#2**). Appropriate validation is necessary to handle erroneous inputs such as incompatible file formats or URLs not from the YouTube domain. More importantly, these errors should be clearly displayed to the user so that they can respond accordingly. An initial design of the Home page can be seen in Figure 4.4.

In addition to taking inputs from the user, we also include a navigation bar with intended links to an 'About' page and a 'Collections' page. Since we want our application to be open to anyone wanting to generate videos or develop ground-truths, we believe an About page is necessary to provide information about the interface and how to use it. The idea of a Collections page

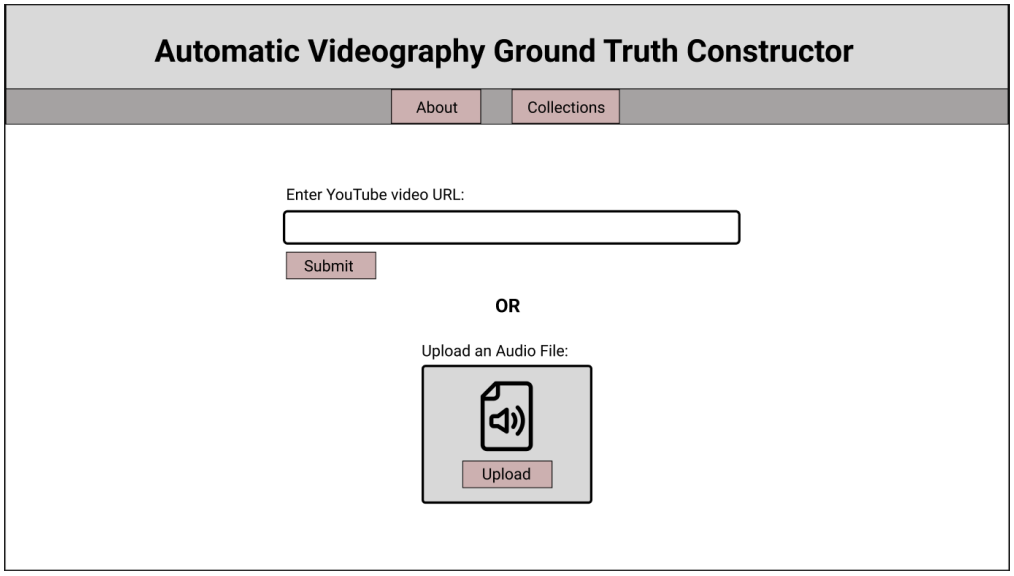


Figure 4.4: *An initial prototype for the Home page, illustrating how the user can input an audio source.*

was spawned to satisfy the potential to save processed audio sources and generated videos (**#10**). Furthermore, since these links are required to be easily accessible, we include this navigation bar on subsequent pages.

4.2.2 Audio Page

Once the audio source is submitted, the Transcription Extracter component should produce a full transcription of the audio. This should lead to the next page displaying the results of the audio processing with the two different options of where to proceed next. More specifically, the user can choose to generate the video with the 'Generate Video' button, or start constructing a ground-truth with the 'Build Ground Truth' button. The full transcription should also be shown to fulfill requirement **#3**.

Aside from this, to save the audio processing result in a database, it must be uniquely identifiable. This led to the idea of associating the audio source with a Title and an optional Author. In the case where the audio is sourced from a file uploaded by the user, the title can simply be the filename. Whenever the audio is extracted from a non-music YouTube video, the title and author can be taken from the YouTube metadata. Finally, for music tracks, the title and author fields can be filled with the recognised track information such as the song title and artist name respectively. Our design, shown in Figure 4.5, also includes an associated image since we believe this could be visually appealing and satisfy the requirement to be able to differentiate collected audio sources (**#31**). This could simply be the video thumbnail for YouTube sources, while for music tracks, the image field can be satisfied by their corresponding cover art.

4.2.3 Video Page

The Video page is rendered when the user clicks on the 'Generate Video' button from the Audio page. This page provides an embedded video so that the user could view the video on the interface itself instead of forcing them to download it to view it. This page will be minimalistic in design, as illustrated in Figure 4.6, and the 'Author - Title' divider will be utilised as the link back to the Audio result page. Once the video is generated, this will be saved in the backend for easy access. Prior to the rendering of this page, we intend to provide a descriptive loading screen

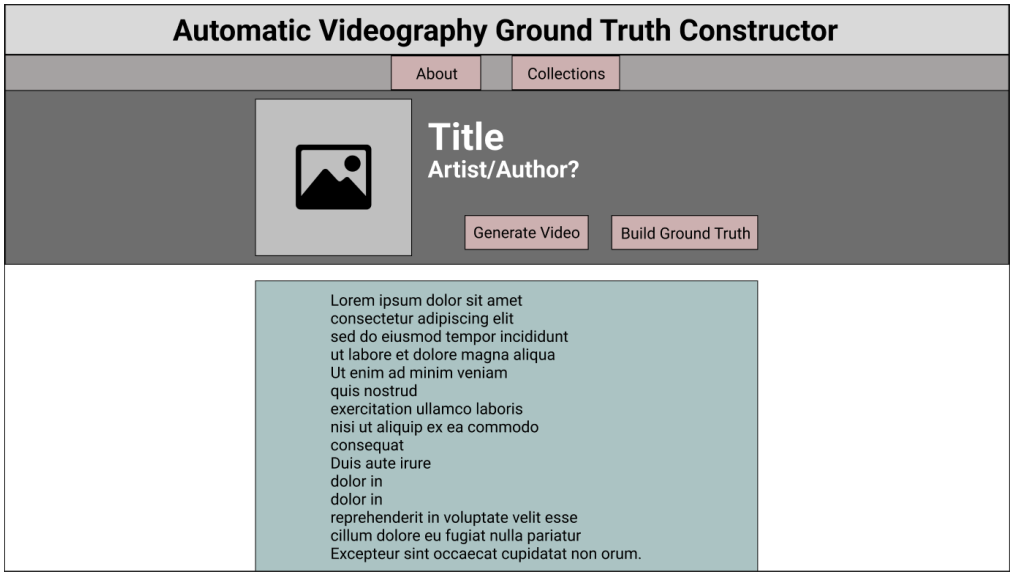


Figure 4.5: *An initial prototype for the Audio page showing the resulting transcript after processing the audio.*

since it is expected that the video compilation will take several minutes as made evident by the previous system. This intermediate loading page adheres to our listed requirements (**#28**) giving an estimated time of completion while simultaneously preventing the user from interacting with other parts of the interface during video generation.

4.2.4 Chunk Page

Alternatively, the user can choose to construct the ground-truth for the audio source by clicking on the 'Build Ground Truth' button on the resulting Audio page. This should subsequently navigate the user to the first audio chunk page. This should show the text present within that chunk and the retrieved top-k images using the system (**#5**, **#7**). As seen in Figure 4.7, we picked a depth $k = 10$ since it is small enough for the assessor to efficiently comb through for the most relevant images. This is especially true when this task must be repeated for every chunk in the audio source. We want to avoid a burdensome experience that could affect the quality of relevance assessments. However, we found that this cut-off point is highly dependent on the effectiveness of the retrieval system as well as the textual query, further discussed in 5.3.5.

We chose to use a binary relevance assessment strategy where the assessor must only mark relevant images meaning unmarked images are considered irrelevant (**#6**). This is faster than a graded relevance system which requires significant thought and may discourage accurate assessments. This is also backed by approaches employed by similar technologies (2.3.2 & 2.3.3). In addition to these interactive images, the page includes 'Previous' and 'Next' buttons to easily navigate to bordering chunk pages. We also aspire to save the user's assessments when these buttons are pressed in order to reduce annotation effort and allow editing of previous chunks (**#11**). Not to mention to allow the possibility of skipping chunks with repeating textual contents (**#16**).

4.2.5 Ground-Truth Page

Once all audio chunks have been annotated, the system should render a resulting Ground-Truth page similar to Figure 4.8. This consists of a small preview of the ground-truth data and some functionality to download the constructed data (**#8**, **#9**). While the ground-truth data may

Figure 4.6: *An initial prototype for the Video page providing the generated video.*

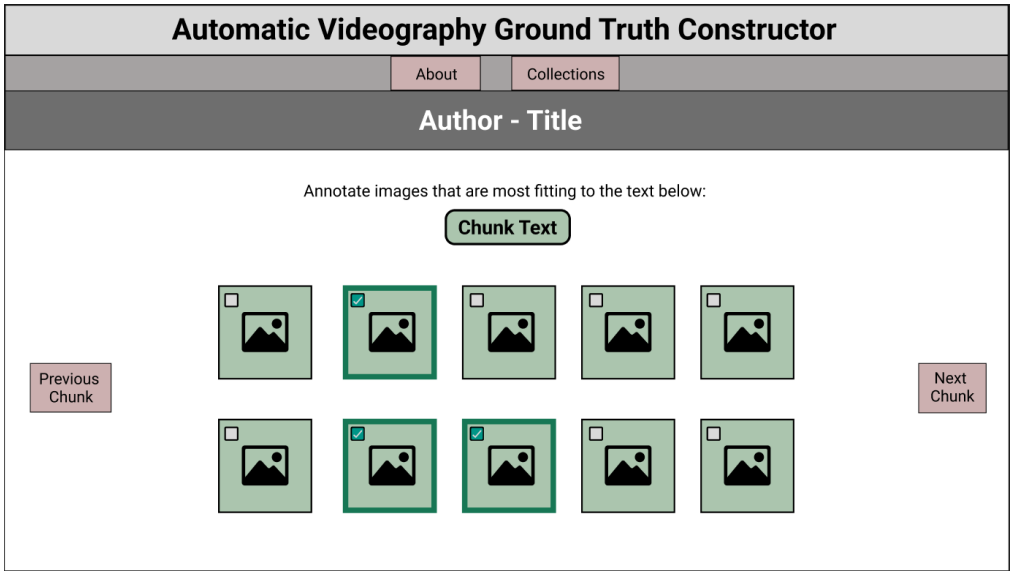


Figure 4.7: *An initial prototype for the Chunk pages that provide the ability to assess the relevance of retrieved images for specific audio chunks.*

be technical, we intend to deliver it in a human-readable format so that assessors can perform last-minute checks if required. The preview feature also enables retrieval system designers to peek at the format of the ground-truths to aid in the evaluation process.

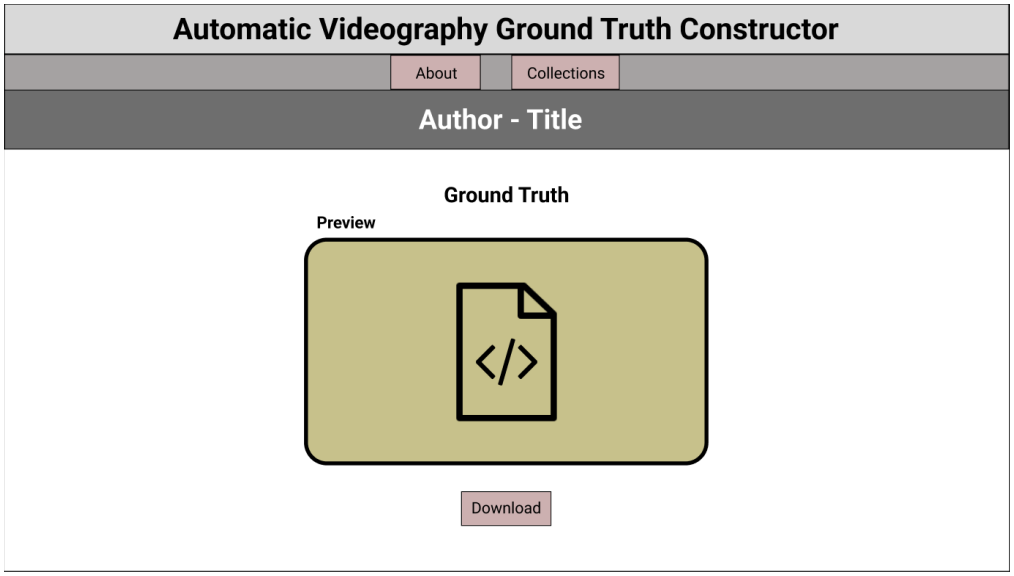


Figure 4.8: *An initial prototype for the Ground-Truth Page providing a preview of the constructed ground-truth available for download.*

4.3 Summary

This chapter provided a comprehensive overview of the design choices made across all areas within the project. From the system architecture to the high-level decisions of the user interface following best Human-Computer Interaction (HCI) design practices. Now that a detailed plan is in place, an intuitive entryway into actually implementing the project is uncovered, in a way that aligns with the requirements specification made in Chapter 3.

5 Implementation

This chapter will provide a detailed look into the implementation of the project. The rationale behind all technology choices and their involvement to achieve our desired product will be explained here.

5.1 Software Development Process

5.1.1 Version Control

GitHub (2008) was the chosen technology to host the remote copy of the repository and version control throughout the implementation. This ensured all code was regularly backed up, providing a layer of security against local data loss or corruption. In addition, it provided a detailed tracking of the project's development, allowing the freedom to go back to previous versions if necessary. Since this was an individual project every feature was developed sequentially making branching unnecessary due to the lack of potential merging conflicts. However, branching was utilised at the beginning of the project when it was unclear which direction the project should be taken. For instance, several options for the image collection were considered, most notably taking them from the Wikimedia Commons (2004) or ImageNet (2006), each stored in separate branches.

5.1.2 Application Framework and Language Choice

It was decided the project should be developed as a web application due to the system requirement **#19**. A web application can be easily deployed to the internet making it accessible to everyone, irrespective of the device or operating system used. Additionally, it offers the advantage of eliminating the need to download and execute large files.

Python (Van Rossum and Drake 1991) was the main programming language chosen due to its flexibility and a huge selection of libraries allowing rapid development in such a time-constrained project. Popular libraries such as NumPy (Harris et al. 2020) offer a wide range of optimised functions for data processing and manipulation. Furthermore, this decision led to the idea of developing the web application using Django (2005), a popular Python web framework.

Django follows a Model-View-Template (MVT) design architecture illustrated in Figure 5.1. It consists of three fundamental components:

- The **Model** manages the data stored in a relational database.
- The **View** communicates with both the model and template in order to complete and send back an HTTP[1] response to the client, corresponding to their HTTP request.
- The **Template** acts as the frontend of the application providing dynamic HTML webpages to the client.

Django has an intermediary component that catches the client's HTTP requests to pattern-match requested URLs to the correct view. The view then fetches data from the models stored in the backend, and renders a template back to the client.

[1] HTTP: Hyper-Text Transfer Protocol

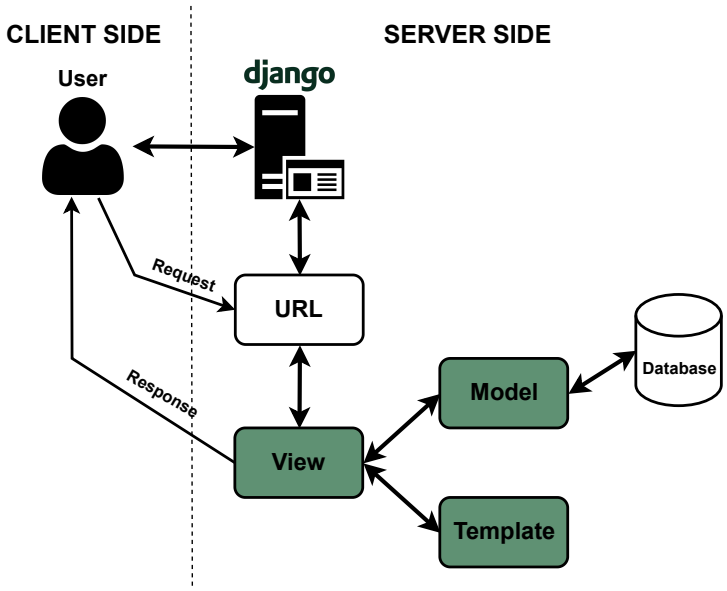


Figure 5.1: *An architecture diagram showing the MVT design pattern of Django web applications.*

5.2 Backend Structure

5.2.1 Static Image Collection

In line with requirement **#18**, several sources for image collections were considered. Out of those, there were two notable sources: the Wikimedia Commons and ImageNet. Wikimedia Commons hosts a range of images that are, more importantly, captioned with accurate textual descriptions. In addition to the latest multimodal, multilingual image-text dataset released by Srinivasan et al. (2021), a research group within Google. It appeared to be an appealing option as it allowed the flexibility to train our own image-text model using the provided textual descriptions. However, complications arose in downloading the dataset. We found that in order to save space, the dataset only included image URLs meaning that the images must be programmatically downloaded one by one. After weeks of testing, even going as far as parallelising the download process, we deemed this effort infeasible due to its slow pace.

We ultimately decided on procuring the collection from ImageNet, an open-source image database containing over 14 million images, available in zipped formats. Moreover, it provided many smaller subsets of the dataset that allowed the chance to familiarise ourselves with CLIP's pre-trained model architecture. The ImageNet-1k subset was the chosen collection for our project which spanned over 1.1 million images allowing reasonable pre-processing times while also being large enough for satisfactory image retrieval results.

5.2.2 Models

Our application consists of only two models: Audio and Chunks. As depicted in the Entity Relation (ER) diagram in Figure 5.2, each Audio object consists of 0 or more Chunk objects. This was to accommodate purely instrumental audio tracks that would contain no spoken language. While this is not useful for our current application domain, we believed this feature could be expanded upon in future iterations where the audio content itself could be used in the image retrieval process using sentiment analysis.

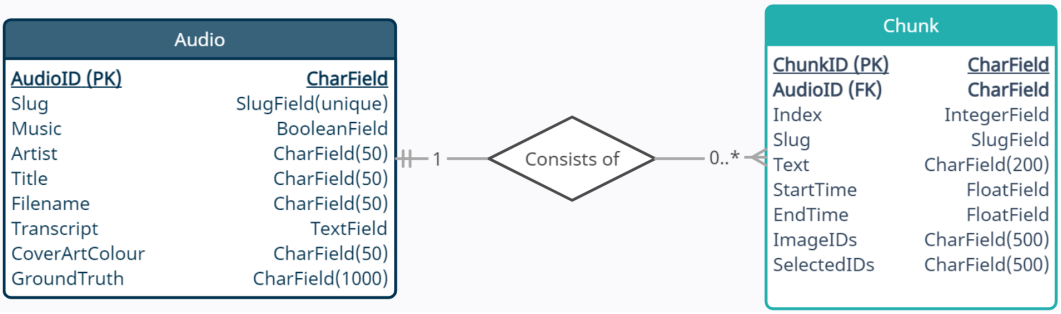


Figure 5.2: *An Entity Relation diagram showing the database schema made through Django models.*

Audio The Audio model holds all information associated with inputted audio sources after processing is complete. This includes the artist/author's name, title, transcript, and the ground-truth data. A boolean flag for whether the audio is a music track or not is used to course the appropriate processing steps. The file name field is used to find corresponding files stored in the static directory, such as the audio, cover art image, and transcript files.

Chunk The Chunk model consists of specific information about the segment in the audio such as the text present, start and end times, as well as, the top-10 and annotated images. It also holds a foreign key that references the appropriate Audio object.

5.3 Program Structure

All the main functionality is performed in the View component, represented by the file, **views.py**. However, each individual component outlined in the system architecture is abstracted into its own Python file in order to adhere to best software engineering practices. Each file, found in the `utilities` folder, provide the required functions to accomplish a single task. This makes locating and adding functionality much easier compared to a disorganised program structure.

5.3.1 Audio Retriever

Represented in the **audio_retriever.py** file, this component lends functions that will be used to extract the audio file from the user's submission on the Home page. Firstly, the **`home`** view will clear the `audio` and `transcript` directories in order to save space. Next, the client's HTTP POST request will be analysed to extract the YouTube video URL or uploaded audio file. In the case of the YouTube video URL, the **`download_yt`** function is called in order to isolate and download the audio from the video using the library pytube (Ronnie Ghose 2022). This function also retrieves the English or Automated English captions, if present, as well as saving the thumbnail picture as the cover art. These can be pulled from the resulting **`pytube.Youtube`** object. Alternatively, in the case of a file upload, the file will be validated if it's of an audio format and then saved into the `audio` folder for further processing.

5.3.2 Audio Recogniser

Now that the audio source is saved within the `audio` folder, it can be processed in order to identify its track title and artist name. It is initially assumed that it comprises of a music track. The **`recognise_audio`** function attempts to match the audio source to a known music track using ShazamAPI (Numenorean 2021). This API also provides the cover art URL for successful recognitions. This cover art is subsequently downloaded and saved within the `coverart` folder using the HTTP Python library, Requests (Reitz 2011). In addition to this, at later iterations,

it was decided to use the primary colour of the cover art to further visually differentiate audio sources (Requirement **#31**). For this, the package, ColorThief (Feng 2019), was chosen to use image processing to identify the primary colour of the cover art which is then used to theme pages and links associated with the audio source.

5.3.3 Synced Transcription Retriever

This component holds multiple strategies to obtain a synced transcription of the audio in the LyRiCs (LRC) file format (Shiang-shiang 2023). The first method is retrieving the synced lyrics from the Musixmatch (2010) platform which holds an extensive database of song lyrics that are aligned with the song itself. The database is queried using the identified song title and artist name. If successful, the lyrics are saved within the `transcript` folder, as well as the external database within the Audio model. Evidently, this method only works with music tracks, hence, this component also provides the capability of robust audio-to-text transcription, leveraging OpenAI's Whisper model (Radford et al. 2022). Due to memory limitations, only the *'small'* model is loaded which was found to frequently misidentify words, hence its last resort status. The resulting transcription is then formatted into the required LRC format.

This is the last processing step within the **`home`** view which will conclude by instantiating a new Audio object using the obtained information. For non-music sources, the title field is filled in with the YouTube video title or uploaded file name. While the artist field is the video creator or simply left as empty for uploaded files. Finally, the user is redirected to the Audio page handled by the **`audio`** view.

The **`audio`** view divides the transcript using the phrase-level timings and initialises new Chunk objects for each segment denoted by new lines in the rendered Audio page. Each line is a clickable link to the corresponding Chunk page. For audio sources with no spoken language, no transcription is made and instead the user is appropriately notified in the rendered page.

5.3.4 Image Retriever

The **image_retriever.py** file is responsible for image pre-processing and retrieval. In order to facilitate the freedom to configure the type of pre-trained CLIP model to use, a custom **`CLIP`** class was made. This class holds properties such as the loaded model, tokenizer, and processor, as well as the file paths for all images and their corresponding embedding vectors. We chose the CLIP ViT-B/32 model for its moderate size for reasonable loading times and high performance. However, this model was trained with only English text and so to accommodate our requirement to support multiple languages (**#12**), we use a compatible Multilingual CLIP model that supports 100 languages: XLM-Roberta Large ViT-B/32 (Frallan 2022). Due to limited memory capacity, only one model can be loaded at one time so a Boolean flag is used to decide which model is loaded. Both models encode images and text into 512 dimensions of latent space and utilise the same vision transformer, meaning they can be interchanged according to the use case.

All images in the collection were pre-processed and embedded in batches using the pre-trained CLIP model and processor, to create a normalised NumPy matrix. This matrix was subsequently saved in a file so that it can be loaded into memory on application start-up.

The top 10 images are dynamically retrieved when a Chunk page is requested, handled by the **`chunk`** view. The **`query_prompt`** function takes in a text prompt, tokenizes it, and embeds it into the 512-dimensional feature space using the pre-loaded model and tokenizer. Following this, an exhaustive similarity search is performed, this involves calculating the cosine similarity between the embedded text vector with every image vector stored in the matrix. Since all vectors are normalised beforehand by their Euclidean (L2) norm as demonstrated in Equation 5.1, the cosine similarity score is simply the dot product between the vectors, accomplished with

```
class CLIP:
...
   def query_prompt(self, prompt, k=10):
      # Tokenize and embed the text prompt using the CLIP model
      inputs = self.tokeniser(prompt, return_tensors="pt").to(self.device)
      text_embedding =
          self.model.get_text_features(**inputs).cpu().detach().numpy()
      # Normalise embedding with the Euclidean (L2) Norm
      text_embedding = text_embedding / np.linalg.norm(text_embedding)

      # Compute cosine similarity scores between prompt text and images
      scores = np.dot(text_embedding, self.image_vectors.T)
      # Sort in descending order and cut-off top-k
      top_k = np.argsort(-scores[0])[:k]

      return top_k
...
```

Listing 5.1: *An algorithm implementing the* ***`query_prompt`*** *function that performs an exhaustive inner product search and retrieval of the top-k images for the passed in text prompt. Defined as an internal function within the custom* ***`CLIP`*** *class.*

`numpy.dot(text_emb, image_embs.T)`. Listing 5.1 provides a code snippet implementing this process.

$$\begin{aligned} \text{normalise}(\vec{x}) &= \frac{\vec{x}}{\|\vec{x}\|_2} \\ \|\vec{x}\|_2 = \sqrt{\sum_{i=1}^{N} |x_i|^2} &= \sqrt{x_1^2 + x_2^2 + \ldots + x_N^2} \end{aligned} \tag{5.1}$$

These similarity scores are subsequently sorted in descending order to get the 10 images that scored the highest with the text query which is then listed to the user on the Chunk page. In order to save unnecessary computation, these image indices are saved in the database associated with the Chunk object.

5.3.5 Ground-Truth Builder

Following the wireframes laid out during the design, the annotator can click on the 'Build Ground Truth' button on the Audio page and is, thereafter, transported to the first Chunk page to begin the relevance assessment exercise. As explained in 5.3.4, the **`query_prompt`** function will be called on an ad-hoc basis for each chunk in order to display the highest-scoring images. The annotator can then label the most fitting images by placing a check mark to denote relevance. This application uses the jQuery plugin, imgCheckbox (Cuénod 2015), to dynamically inject checkboxes onto HTML **`<img>`** tags on the webpage.

All annotated images are saved within the Chunk object when the user submits an HTTP POST request which can occur when the 'Previous' or 'Next' buttons are pressed. This allows previously annotated Chunks to be viewed and amended fulfilling requirement **#11**. Additionally, throughout testing it was found to be burdensome to annotate chunks with repeating textual

content, leading to the addition of requirement **#16**. This feature greatly reduced annotation effort and efficiency since all duplicate chunks can be updated at once and skipped during relevance assessment.

The full ground-truth can only be viewed when the user clicks the 'Finish' button on the last chunk of the transcript. The ground-truth is built chunk by chunk, appending to a Python dictionary storing relevant information of the audio source and chunks. The indices of the assessed images are stored here as well as their corresponding file paths in order to ease exporting. In order to release the data in a human-readable and easily accessible format (**#9**), the resulting Python dictionary is serialised into JavaScript Object Notation (JSON) (Pezoa et al. 2016). This is an open standard format that is language-independent. A preview of the data is subsequently rendered in the Ground-Truth page with a clear button to download it.

Determining Pooling Depth The final pooling depth of $k = 10$ was determined through trial and error since it was difficult to estimate the effectiveness of CLIP models on multi-conceptual text prompts that are so common in spoken language, especially song lyrics. Retrieving the top-10 images was determined to be the ideal depth that is able to fit in one page without excessive rescaling as well as not increasing relevance assessment effort. However, it was noticed that in rare occasions of extreme ambiguity of text, more images were required to be assessed to find any relevance. This was also picked up during user evaluations in 6.2.1 where it may be beneficial to increase k for very abstract ideas.

5.3.6 Video Compiler

The video compiler component implemented in the **videography.py** file, takes care of generating the video primarily using the library MoviePy (Zulko 2017). The video is built incrementally, much like the ground-truth construction, by selecting the top-1 image for each chunk and concatenating **`ImageClips`** in sequence. Their durations are simply the difference between the chunk start and end times. **`TextClips`** showing the chunk text are placed below corresponding images. Black **`ColorClips`** are used to fill parts of the audio lacking spoken words. Lastly, the audio source is added to the final **`VideoClip`** which is then written into an MP4 file in the `video` folder. This file is then embedded into the HTML of the Video page for the user to subsequently play and download.

5.4 Summary

This chapter described the steps taken in manifesting the blueprint designs into a concrete product. Key implementation details were explained as well as the challenges faced along the development process. The rationale for the chosen language and web framework was given, in addition to how the various libraries are integrated into the project in order to reduce the complexity of the project's development. A full account of the program's structure and technical accomplishments were also provided. A walkthrough of the implemented application is provided in Appendix A. Now that the application has been fully implemented, it presents an opportunity to assess its efficacy in achieving the project's original goals.

6 Evaluation

This chapter serves to evaluate the fully implemented application by looking at how many of the requirements specified in Chapter 3 have been fulfilled. This chapter is split into two sections, the first part evaluates the implemented Automatic Videography tool by looking at ways to improve its efficiency, as well as comparing it against the system that came before. The latter half entails evaluating the usability of the relevance assessment interface.

Since this product's purpose is to aid users in their creative pursuits and the assembly of test collections, rather than addressing a quantifiable problem, some requirements pose a greater challenge for evaluation. Non-functional requirements are particularly challenging because, unlike functional requirements which can be either implemented or not, non-functional requirements are more subjective in nature. A number of experiments must be conducted in order to gather empirical measures that can help answer this question. Nevertheless, it is inevitable that some level of interpretation will be necessary, but efforts will be made to approach the highest degree of objectivity.

6.1 Automatic Videography

6.1.1 Improving Retrieval Efficiency with FAISS

The retrieval system involves meticulous similarity search and sort operations which pose serious scalability issues. Expanding to larger image collections or running multiple videography pipelines at once is currently impractical. Therefore more efficient strategies for retrieving the images were analysed including Facebook's AI Similarity Search (FAISS) (Johnson et al. 2019). This library offers a range of efficient indexing strategies for fast similarity searching, with the key idea of clustering similar documents into identical cells. In our case, it was necessary to utilise the inner product indexing technique, **`faiss.IndexFlatIP`**, to match CLIP's training procedure. This path ultimately lead to a dead end due to a huge decline in retrieval accuracy, even with the remarkable 98% decrease in the mean search time compared to an exhaustive search.

Experiments were conducted to evaluate different parameter tunings for the inverted file index with a Product Quantizer encoding, **`faiss.IndexIVFPQ`**. Each configuration was then compared with the original flat indexing strategy, i.e. exhaustive searching. The indexed collections were tested on a set of 10 audio sources which provided a total of 399 phrase-level text prompts. Three sets of values were used for the **`nlist`** parameter namely: 2048, 4096 and 16 384. These values determine the number of Voronoi cells to partition the index of image vectors. It is common to pick a value that is a power of 2 that is close to $4\sqrt{M}$, where M is the total number of document vectors (Milvus 2020). Our filtered static collection consists of 1 281 121 images resulting in a recommended value that is close to the second test value.

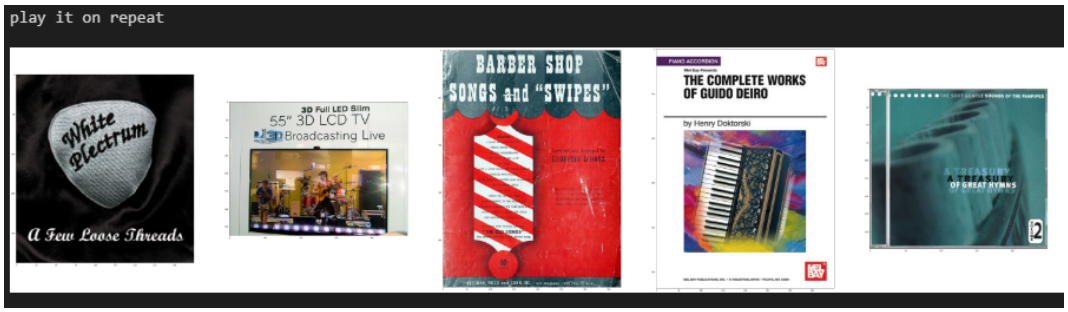
Table 6.1 outlines the results of the experiment including the mean retrieval times to obtain the top 10 images for all 399 text queries. The mean Jaccard similarities shown was used to determine the accuracy of the retrievals by measuring the overlaps against the original exhaustive approach. From the results, it is evident that while the configurations are able to retrieve images over 30 times faster, they achieved a 0% accuracy within the top ten images. This elicited inspecting the

Indexing Method	Mean Retrieval Time (s)	Mean Jaccard Similarity
Flat (Exhaustive)	0.2833	-
IndexIVFPQ (`nlist=2048`)	0.0024	0.0
IndexIVFPQ (`nlist=4096`)	0.0035	0.0
IndexIVFPQ (`nlist=16384`)	0.0094	0.0

Table 6.1: *The results of the FAISS experiment to find the best indexing configuration to expedite the search and retrieval process. The experiment was tested on* 399 *text prompts extracted from* 10 *audio sources. The mean time to retrieve the top ten images from a text prompt is shown for each strategy as well as the mean Jaccard similarity of the retrievals against the original Flat index method. All retrievals were performed using an* `nprobe` = 20.

quality of images retrieved by the different FAISS systems revealing some degree of relevance. Although, in many instances, the retrieved images did not pertain to the given prompt. Figure 6.1 illustrates a particular comparison for the text query, *"play it on repeat"*. We believe more work is required to enhance the scalability of this project's videography system, which can be achieved by experimenting with alternate FAISS indexing strategies.

(a) *Original Flat Index (i.e. exhaustive search)*



(b) *FAISS IndexIVFPQ (`nlist=16384`) configuration*

Figure 6.1: *The top five images retrieved by both configurations, ordered by their similarity score to the text query: "play it on repeat".*

6.1.2 Video Generation Time Testing

Despite the unsuccessful efforts in improving retrieval speeds with FAISS, one of the objectives of this project was to improve the slow video generation times of the previous system. This led

to non-functional requirement **#26**:

- **Should Have:** The videos being generated in a more reasonable amount of time than the prior system.

In order to check if this requirement has been satisfied, experiments must be conducted in order to find and compare the average time taken to generate a video using both systems. This experiment was run with 20 audio sources each of varying lengths, averaging around 3 minutes and 20 seconds (Standard Deviation: $\sigma = 0.84$ min). The mean number of phrase-level chunks present per source was 47.9 ($\sigma = 18.4$ chunks) with an overall total of 958 chunks and images being retrieved during the experiment. Before conducting the experiment, it was postulated that the time taken to compile a video is determined on the number of MoviePy video clips needed. As made evident by Figure 6.2, this is generally the case as the video compilation time is dependant on **both** the number of phrase-level chunks present and the audio length.

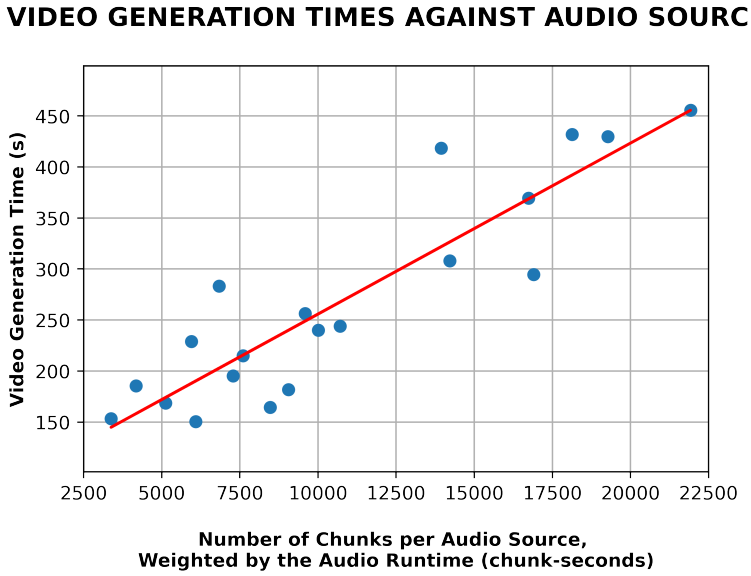


Figure 6.2: *Graph showing the general relationship between the video generation time, and the number of chunks present in audio sources, weighted by the audio runtime. The data was plotted from the video generation times of the current system on 20 distinct audio sources.*

This is because the clips can comprise **`ImageClips`** and **`TextClips`**, which is determined by the number of chunks present. In addition, **`ColorClips`** are used to fill gaps between chunks meaning this is influenced by the audio duration. Therefore, in order to minimise the effect of both the number of chunks and the audio runtime, a diverse range of sources were picked that are representative of the tool's use case.

Table B.2 in the Appendix lists the resulting video generation times for all 20 sources on both the previous and current systems. A summary is shown in the form of box plots in Figure 6.3. As illustrated, the current system generates videos 42% faster than the previous system on average. The current system achieved a mean video generation time of 4 minutes 29 seconds ($\sigma = 1.65$ min), while the latter, 7 minutes 44 seconds ($\sigma = 2.63$ min). It is interesting to note that the prior system's times vary a lot more. This is most likely due to the fact that it necessitated downloading images from Google's image catalogue, resulting in its retrieval time being subject to unstable internet speeds. In contrast, this system has the benefit of retrieving from a locally stored collection.

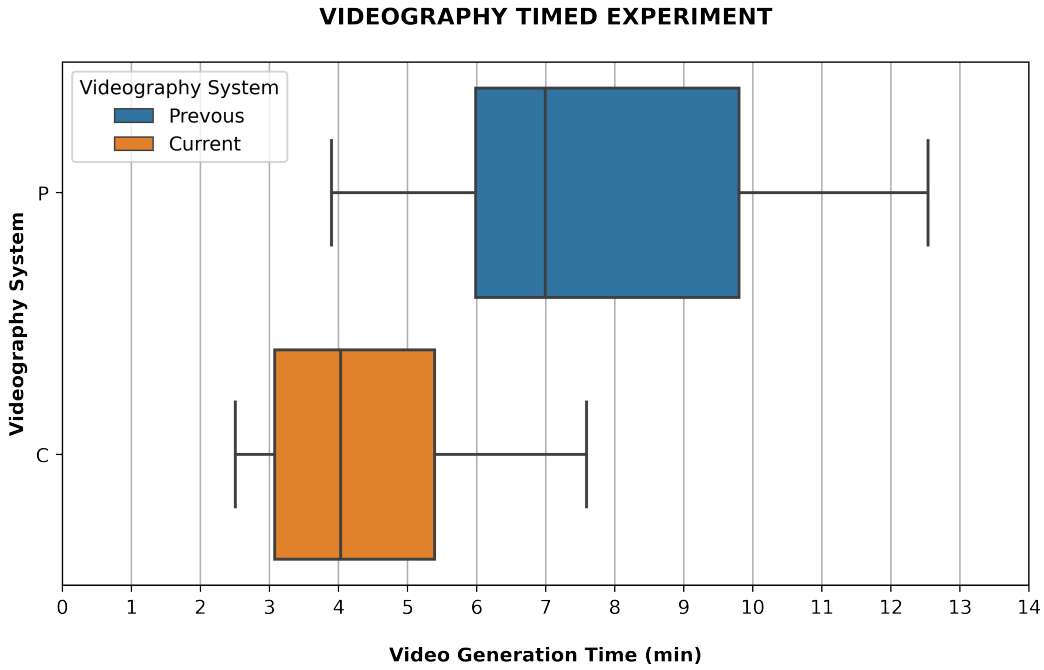


Figure 6.3: *Box plot showing a comparative summary of the video generation times for both the preceding and current videography systems, tested on 20 distinct audio sources. The Caption Extraction method used was throughout the experiment for the prior system.*

In summary, the experiment results empirically show that the non-functional requirement **#26** has been successfully fulfilled. Considering that this system elevates video compilation complexity by including **`TextClips`**, something that was absent from the preceding system, the results are promising. However, this can be further improved by using a more intelligent indexing strategy using resources such as FAISS, which offers faster similarity search times than the current exhaustive approach.

6.1.3 Video Quality Testing

Another requirement of the project was **#27**:

- **Should Have:** The generated video having a higher resolution than the prior system.

The previous system restricted the resolution of images to 200 by 100 pixels due to the high disparity of image sizes downloaded from Google. This was found to cause frequent incompatibility issues with MoviePy's video compilation algorithm. This resulted in numerous complaints about breaking immersion during its user evaluation phase. To address this issue, a potential solution is to minimise this disparity by filtering images with outlier resolutions from the dataset. Fortunately, the ImageNet-1k dataset, which this project employs, already addresses this concern by providing images with consistent resolutions to streamline machine learning tasks. Experiments conducted identified that the images stored in the collection had a mean resolution of 473.0×405.5 with a standard deviation of 208.1 pixels for width and 178.7 pixels for height. Therefore, the quality of videos did not become a burden during this project. Figure 6.4 shows a comparison of the quality of compiled videos for *America*'s song, *"A Horse With No Name"*, demonstrating that this requirement has been fulfilled.

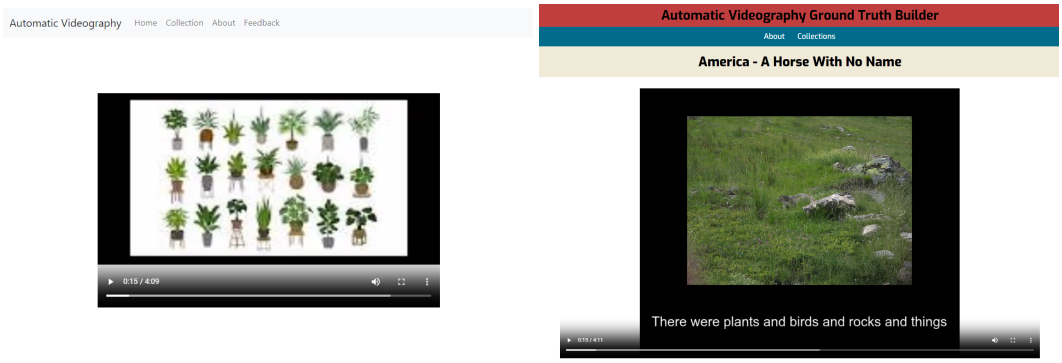


Figure 6.4: *Comparison of the quality of videos generated by the preceding (left) and new (right) videography systems. The videos were screenshotted at the same timestamp in order to show a fair comparison of the same segment within the song "A Horse With No Name" by America.*

6.2 Ground-Truth Annotation Interface

6.2.1 Usability Testing

This project aimed to produce an interface that eases the relevance assessment of images from text for videography tasks. Hence, one of the conditions set forth was that the interface needed to be user-friendly enough for any user to carry out a relevance assessment on a specified audio source. To test this non-functional requirement **#20**:

- **Should Have:** The application having a presentable user interface that is easy for any user to understand and learn quickly.

A usability survey must be conducted with a diverse set of users that is representative of the target demographic. 16 participants took part in the user study, where they were each given a single task to complete using the developed web application. The study complied with the Ethics Checklist outlined by the School of Computing Science as all data collected was anonymised and no personal data was collected. A signed copy of this checklist can be found in Appendix C.

A template of the briefing form given to each participant can be found in Appendix D. In summary, each participant was given an audio source picked from a predefined list of music tracks and educational videos on YouTube. The participant was then expected to find and input the required information in order to start constructing a ground-truth using the application. As the study was focused on evaluating the interface's usability, participants had to rely solely on the information presented to navigate the website, as no explicit instructions were given on how to complete the task. All participants were given a maximum of 10 minutes, however, it was not expected for them to complete the ground-truth for the full audio. This was due to the varying lengths and the number of chunks present in the sources. Nonetheless, from prior testing, it was expected that most participants could complete 85-90% of the ground-truth for audio sources averaging around 50-55 chunks.

Usability Rating Once the participants finished their allotted tasks, they were asked to complete a survey asking questions about their experience with the interface. A copy of the survey form is provided in Appendix E. The first question asked participants to rate the interface design's overall intuitiveness on a scale of 1 to 5. The higher the rating, the higher they found the application easy to navigate through. On average, the usability of the site was rated 4.4 out of 5 by the participants, with a standard deviation of 0.5. This is because all participants chose either 4 or 5, as illustrated in Figure 6.5. While this score shows promise, it is not sufficient to completely check off our criterion. Therefore, the remainder of the survey comprised of open-ended questions.

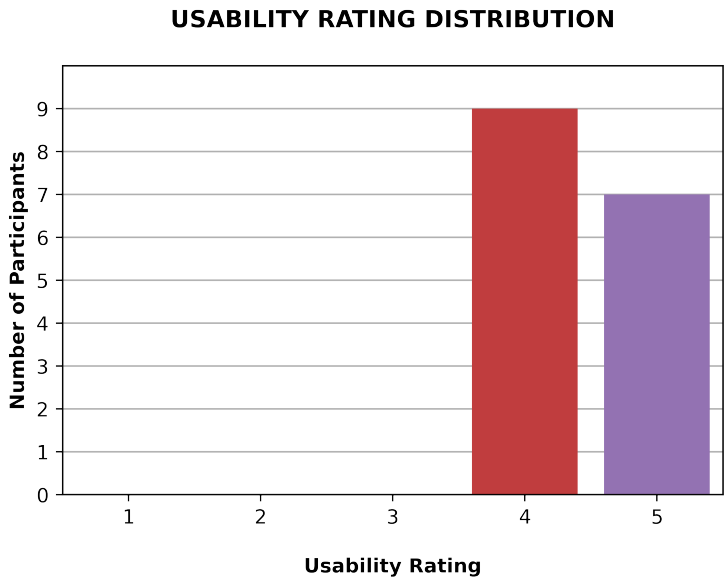


Figure 6.5: *Bar graph showing the distribution of participant ratings regarding the intuitiveness of the interface.*

These aimed to obtain precise details about the participants' preferences and dislikes regarding the application. Moreover, the open-ended questions enabled respondents to suggest additional features that could potentially enhance the application in future iterations.

Issues and Improvements The next two questions intended to uncover any issues and ways to improve the existing system. Untrained users are more susceptible to encountering bugs because they must engage with the system more extensively to learn how to use it. 7 out of the 16 participants were able to complete their tasks without any problems, however, the remaining experienced one of four distinct issues. The most common issue experienced was a failure after the user interacted with the application during the execution of a background process. This was commonly found to occur during time-intensive routines such as the audio processing after a source is submitted, and the initial retrieval of images. This was a problem that was foreseen during the design phase and intended to be circumvented by displaying a separate loading screen. Its purpose would be two-fold, it would block further user interactions with the frontend and exhibit the estimated time of completion. However, due to time constraints, fully implementing this feature was not realistic when there were several higher priority features that required attention. In hindsight, this can be viewed as the biggest limitation to the current implementation which should be fixed in the next iteration.

Three participants highlighted that they experienced occurrences where none of the images retrieved were relevant to the concepts described in the chunk text. They noted that this made it difficult to ascertain how to proceed, as one had to decide between selecting no images or selecting at least one that exhibits some notion of the concept. From observation, this most frequently occurred when the textual cue itself was ambiguous and described abstract ideas that are difficult to illustrate. For instance, text prompts such as *"Let it go"*, or non-lexical expressions like *"oh oh/la-la"*, which are so common in song lyrics, are particularly difficult to delineate. It is worth noting that the ImageNet-1k dataset, despite its considerable size of more than 1.1 million images, is limited to only one thousand *"synsets"*, as its name implies. Therefore, the system is unable to portray niche ideas solely because it lacks an adequate number of images to represents them. This suggests that a more extensive image dataset could improve retrieval results, not to

mention, utilising a larger pre-trained CLIP model.

Some minor issues mentioned in the survey included the uncertainty about which URL of a YouTube video to input. One participant was observed to attain the URL by *"right-clicking on the video"* rather than from the browser address bar. This resulted in a validation error on form submission since the URL pattern (`https://youtu.be/...`) was slightly different to the expected pattern (`https://www.youtube.com/watch?v=...`). This can be easily resolved either by providing clearer instructions regarding the type of URL required or by modifying the validation check to accommodate both types of URLs. Another participant was unable to navigate back to the Audio page once within a specific Chunk. This could be attributed to the fact that the Audio link is concealed behind the *'Author - Title'* banner, which is only perceived as such on hover. Adding a more explicit 'Back' button to the page can fix this confusion. Lastly, in one of the surveys, a participant expressed doubt about whether to annotate images that effectively conveyed the semantics of the text prompt but were slightly blurred in appearance. This underscores the subjective nature of relevance assessment tasks in general, as ultimately, it is the end viewer's perception that matters which can not be precisely defined. A graded relevance assessment or the removal of blurry images from the dataset could help alleviate this ambiguity, however, more studies must be done to find the best solution.

Future Feature Ideas The final part of the survey asked the participants for suggestions for future features that they would like to see. This provided the opportunity for the application's target users to suggest new feature ideas that would ease relevance annotation effort. There were many good ideas discussed such as adding a sound bite on Chunk pages so that the assessor could listen to the chunk in addition to reading. Identifying the mood from the audio sample could aid in judging abstract text prompts. A different participant proposed the inclusion of a *"Share"* feature that would enable users to directly share the produced video on various social media platforms like YouTube, Facebook, and TikTok. On the topic of integrating the application with other platforms, one respondent recommended enabling the search for YouTube video sources within the site to eliminate the need for the user to manually fetch the right URL.

One participant was assigned the job of creating the ground-truth for the German song, *"99 Luftballons"* by *Nena*, in order to test the effectiveness of the multilingual CLIP model. However, the fact that the participant was not fluent in German made the relevance assessment exceptionally difficult since they needed to translate each chunk phrase into English to understand and identify the relevant images. They went on to suggest incorporating a language translation API that would simplify the assessment process for annotators assessing audio sources that are not in their native language.

Other notable ideas included adding an option to expand the retrieval pool for obscure chunk phrases, adding an option to generate videos from text files, and adding a link to navigate directly to the last edited chunk page. These are discussed in more detail in the Future Work reflection in 7.3.

6.3 Requirement Satisfaction Analysis

Now that the idiosyncratic requirements have been fully evaluated, the next step is to examine each requirement listed in Chapter 3 to form a conclusive assessment of how effectively the project has accomplished its goals.

6.3.1 Functional Requirements

1. ✓ **Must Have:** The ability to upload an audio file as an input audio source.
2. ✓ **Must Have:** The ability to specify a YouTube video URL as an audio source.
3. ✓ **Must Have:** The ability to view the full audio transcription.

4. ✓ **Must Have:** The ability to generate an illustrated video for any audio source.
5. ✓ **Must Have:** The ability to view the top-k retrieved images for a given audio chunk.
6. ✓ **Must Have:** The ability to annotate the most-fitting images for a given audio chunk.
7. ✓ **Must Have:** The ability to view the text present in a given audio chunk.
8. ✓ **Must Have:** The ability to view the constructed ground-truth data.
9. ✓ **Must Have:** The ability to download the constructed ground-truth data in a portable format.
10. ✓ **Should Have:** The ability to view previously processed audio sources, generated videos, and constructed ground-truth.
11. ✓ **Should Have:** The ability to edit previously made ground-truths.
12. ✓ **Should Have:** The ability to upload audio sources of any language.
13. × **Could Have:** The ability to load ground-truths for editing.
14. ~ **Could Have:** The ability to reconfigure the static image collection location.
15. × ***Could Have:** The ability to view the percentage complete for ground-truth construction of processed audio sources.
16. ✓ ***Could Have:** The ability to save or skip chunks with repeating textual content when ground-truth constructing.
17. ✓ ***Could Have:** The ability to view the timestamp information of audio chunks.

6.3.2 Non-Functional Requirements

18. ✓ **Must Have:** The system using a static collection of images for repeatably consistent ground-truth construction.
19. ✓ **Should Have:** The application being accessible from any Operating System platform.
20. ✓ **Should Have:** The application having a presentable user interface that is easy for any user to understand and learn quickly.
21. ✓ **Should Have:** The application with appropriate error handling.
22. ✓ **Should Have:** The application being fast and responsive.
23. ✓ **Should Have:** The system being able to recognise music tracks to automatically retrieve the artist name and song title.
24. ✓ **Should Have:** The application being able to transcribe any audio source providing phrase-level timestamps.
25. ✓ **Should Have:** The system retrieving more relevant images by leveraging semantics of natural language as well as visual features.
26. ✓ **Should Have:** The videos being generated in a more reasonable amount of time than the prior system.
27. ✓ **Should Have:** The generated video having a higher resolution than the prior system.
28. × **Should Have:** The application displaying progress bars for long loading times.
29. ✓ **Could Have:** The application being designed with extensibility in mind so that the image collection can be reconfigured.
30. ✓ **Could Have:** The videos including text as well as images.
31. ✓ **Could Have:** The application visually differentiating between previously processed audio sources.
32. × **Could Have:** The system performing tempo or beat analysis to appropriately time image changes.

The project was able to satisfy the majority of the requirement specification, with over 85% (27/32) of the requirements being fulfilled. More importantly, all of the highest priority **Must Haves** were satisfied. The status of requirement **#14** is amber because the feature has not been incorporated into the application's frontend. Nevertheless, the image collection can be substituted in the source code owing to the program's extendable design. Overall, it can be safely asserted that the project has achieved its aim, with the final product functioning largely in accordance with its original intent.

6.4 Summary

This chapter provided an extensive evaluation of the final product in order to check if it fulfilled the fundamental goals of the project. This was done by analysing the original system requirements specified within Chapter 3. In order to evaluate the more abstract requirements, several experiments were conducted to gather empirical evidence of its effectiveness. Overall, it is safe to conclude that the project delivered a functional product that met most of the primary objectives.

7 Conclusion

7.1 Summary

This project aimed to create two new technologies, namely a new Automated Videography system that improves on the previous product built by Parker (2022) and a ground-truth annotation interface. In the end, the project resulted in a successful system that can generate a video composed of relevant imagery to accompany any audio source. Furthermore, a user-friendly interface was devised, that aims to simplify the process of assessing the relevance of images derived from textual queries. This interface effectively streamlines this task of ground-truth construction intended for the quantitative evaluation of automatic videography systems. To the best of our knowledge, this is something that is missing from existing annotation software.

The project followed an agile software development framework by constructing user scenarios in order to facilitate clear system requirements and design. The application was implemented with extensibility in mind in order to ease the addition of new features. The system and its accompanying frontend interface were subsequently evaluated in order to determine whether all the pre-conceived goals have been fulfilled. This involved running three main experiments.

Firstly, the implementations of more efficient image indexing and retrieving methods were explored using FAISS which were unsuccessful. This did not cause a huge hindrance to the project since the system's exhaustive search was found to be sufficiently fast with the ImageNet-1k collection. Secondly, a comparison of the generation times of the new and preceding videography systems was done. The new system resulted in a mean video generation time of 4 minutes and 29 seconds, reducing waiting times by 42%. The final experiment entailed testing the usability of the ground-truth annotation interface with sixteen participants. On average, the respondents rated the intuitiveness of the interface 4.4 out of 5. In addition, every participant was able to complete their ground-truth construction task. This data is made available in the project GitHub repository[1]. The purpose of this data is to enable reliable and repeatable performance evaluations of automatic videography systems while also alleviating the need for subjective user evaluations.

7.2 Reflection

Looking back, many valuable insights were gained throughout the development process of the project. It underlined the importance of careful organisation in order to manage such a large-scale individual project. Breaking down substantial tasks into smaller, practicable workloads, that could be tackled iteratively, proved vital to the successful completion of the project within the specified timeframe. In addition, it was learned early on that planning out the next steps within the project lifecycle was especially important. This softened the impact of major obstacles that would have necessitated retracing and re-evaluation of the problem. One of the biggest flaws made during the project was the mis-prioritisation of the **Should Have** requirement **#28**. The lack of a loading page became the cause of the leading fault encountered during usability testing. In hindsight, it is apparent that this feature should have been ranked higher as a **Must Have** in order to enforce its implementation sooner.

[1] `https://tinyurl.com/2p8he6sr`

7.3 Future Work

This project offers many potential future prospects both in terms of fixing issues identified in the final evaluation and the addition of new features. First and foremost, the missed requirement for an intermediary loading page should be put in effect for time-consuming processing tasks. Secondly, this project has a clear motivation to move toward using larger image collections for better relevance assessment and videography results. However, the current implementation employs exhaustive search and retrievals which can not be scaled to datasets spanning hundreds of millions of images. Further work with FAISS should be conducted in order to improve efficiency while also maintaining high retrieval accuracy. Possible routes could be reducing the dimension size of feature vectors from 512 to 128 using Principle Component Analysis (PCA) to identify salient dimensions. Smaller feature vectors could aid the indexing operation providing better clusters.

There were several potential features that were suggested by the participants in the usability survey. To name a few, adding an audio sample to each Chunk page so that assessors can judge relevance better based on both audio and text. Adding an option to directly share generated videos to social media platforms. A possible integration of the YouTube search API for further user-friendliness, bypassing the requirement of inputting a URL. Adding a language translation layer to pages so that it is not required for assessors to be native speakers to be effective in labelling images. In some cases showing the top ten images is not enough to uncover relevance, adding a button to expand the retrieval pool depth could help in this aspect. Instead of only allowing video generation from audio sources, users may want to create a video from a written transcription. Text-to-speech tools can then be used to accompany the final video. As demonstrated, the end product provides ample scope for expansion to further our goal of easing content creation and relevance assessment.

On a final note, the work carried out by Parker (2022) has been approved for inclusion into the 2023 European Conference on Information Retrieval (ECIR) proceedings upon revision of the original demonstration paper. The paper is available for viewing at Ganguly et al. (2023) and the final product of this project will be subsequently showcased at the conference. Similarly, we intend to submit this project's work to the 2023 Conference on Information and Knowledge Management (CKIM).

A Walkthrough of Final Application

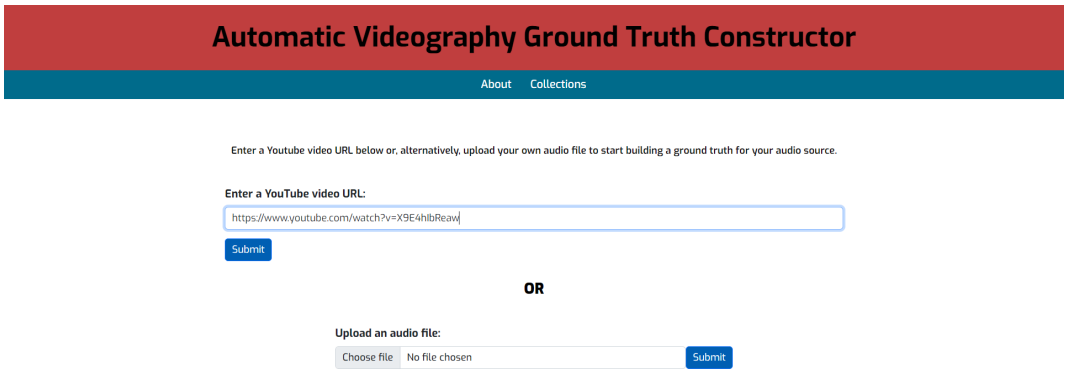


Figure A.1: *Example of using the Home page by inputting a YouTube video URL for the song "Jigsaw Falling Into Place" by Radiohead*

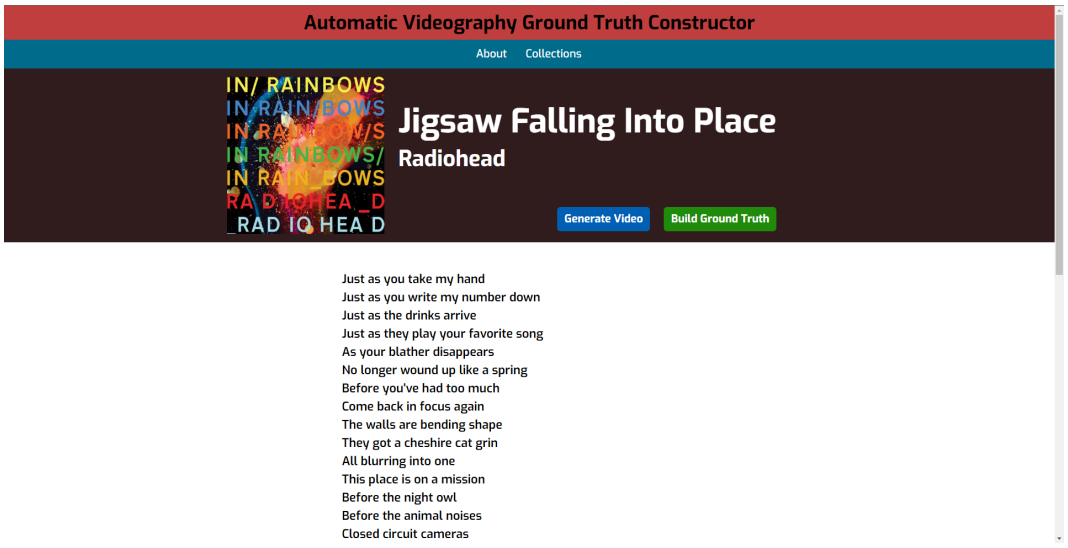


Figure A.2: *Resulting Audio page for the song "Jigsaw Falling Into Place" by Radiohead*

Figure A.3: Resulting Video page for the song "Jigsaw Falling Into Place" by Radiohead

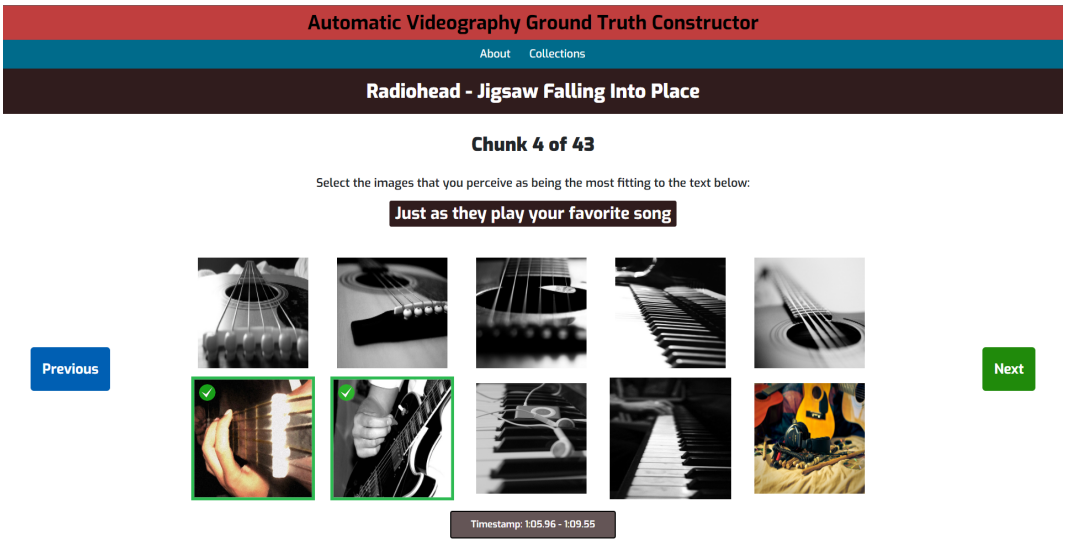


Figure A.4: Example of annotating images for Chunk 4 for the song "Jigsaw Falling Into Place" by Radiohead

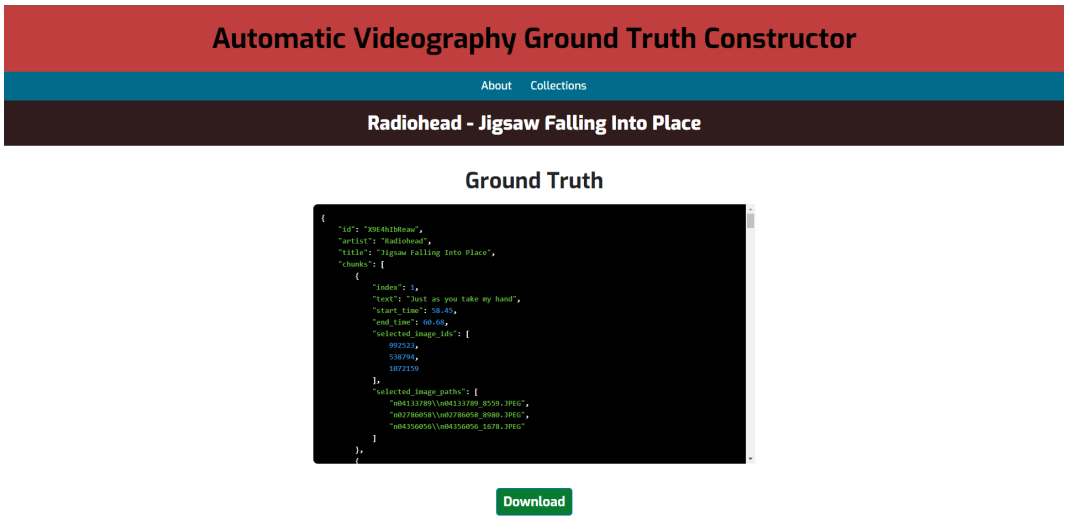


Figure A.5: *Resulting Ground-Truth page after completion for the song "Jigsaw Falling Into Place" by Radiohead*

B Video Generation Time Experiment

Audio Source	Runtime (mm:ss)	Number of Chunks
1	04:37	61
2	02:25	42
3	02:26	62
4	03:58	81
5	04:34	80
6	04:30	62
7	02:03	34
8	03:10	36
9	04:23	53
10	03:26	69
11	04:05	74
12	03:08	51
13	04:38	36
14	04:04	21
15	03:49	37
16	03:20	38
17	02:10	26
18	04:09	43
19	03:49	26
20	04:40	26

Table B.1: *Table listing runtime and chunk count information for all 20 audio sources used in the video generation time experiment in 6.1.2.*

Audio Source	Video Generation Time (mm:ss)	
	Previous System	New System
1	08:47	04:55
2	06:55	02:30
3	09:48	03:02
4	12:32	07:10
5	12:24	07:36
6	09:48	06:09
7	05:46	03:05
8	06:03	04:43
9	08:30	06:58
10	10:49	05:08
11	11:32	07:12
12	08:13	04:16
13	06:03	03:60
14	03:54	02:48
15	06:12	02:44
16	06:21	03:35
17	04:37	02:33
18	07:04	04:04
19	04:37	03:49
20	04:37	03:15

Table B.2: *Table listing the time from initialising the video generation process and the interface displaying the compiled video for both the preceding and new videography systems. The experiment was run with 20 distinct audio sources which are listed in Table B.1.*

C Ethics Checklist Form

School of Computing Science
University of Glasgow

Ethics checklist form for 3rd/4th/5th year, and taught MSc projects

This form is only applicable for projects that use other people ('participants') for the collection of information, typically in getting comments about a system or a system design, getting information about how a system could be used, or evaluating a working system.

If no other people have been involved in the collection of information, then you do not need to complete this form.

If your evaluation does not comply with any one or more of the points below, please contact the Chair of the School of Computing Science Ethics Committee (matthew.chalmers@glasgow.ac.uk) for advice.

If your evaluation does comply with all the points below, please sign this form and submit it with your project.

---

1. Participants were not exposed to any risks greater than those encountered in their normal working life.

   Investigators have a responsibility to protect participants from physical and mental harm during the investigation. The risk of harm must be no greater than in ordinary life. Areas of potential risk that require ethical approval include, but are not limited to, investigations that occur outside usual laboratory areas, or that require participant mobility (e.g. walking, running, use of public transport), unusual or repetitive activity or movement, that use sensory deprivation (e.g. ear plugs or blindfolds), bright or flashing lights, loud or disorienting noises, smell, taste, vibration, or force feedback

2. The experimental materials were paper-based, or comprised software running on standard hardware.

   Participants should not be exposed to any risks associated with the use of non-standard equipment: anything other than pen-and-paper, standard PCs, laptops, iPads, mobile phones and common hand-held devices is considered non-standard.

3. All participants explicitly stated that they agreed to take part, and that their data could be used in the project.

   If the results of the evaluation are likely to be used beyond the term of the project (for example, the software is to be deployed, or the data is to be published), then signed consent is necessary. A separate consent form should be signed by each participant.

   Otherwise, verbal consent is sufficient, and should be explicitly requested in the introductory script.

4. No incentives were offered to the participants.

   The payment of participants must not be used to induce them to risk harm beyond that which they risk without payment in their normal lifestyle.

5. No information about the evaluation or materials was intentionally withheld from the participants.
   Withholding information or misleading participants is unacceptable if participants are likely to object or show unease when debriefed.

6. No participant was under the age of 16.
   Parental consent is required for participants under the age of 16.

7. No participant has an impairment that may limit their understanding or communication.
   Additional consent is required for participants with impairments.

8. Neither I nor my supervisor is in a position of authority or influence over any of the participants.
   A position of authority or influence over any participant must not be allowed to pressurise participants to take part in, or remain in, any experiment.

9. All participants were informed that they could withdraw at any time.
   All participants have the right to withdraw at any time during the investigation. They should be told this in the introductory script.

10. All participants have been informed of my contact details.
    All participants must be able to contact the investigator after the investigation. They should be given the details of both student and module co-ordinator or supervisor as part of the debriefing.

11. The evaluation was discussed with all the participants at the end of the session, and all participants had the opportunity to ask questions.
    The student must provide the participants with sufficient information in the debriefing to enable them to understand the nature of the investigation. In cases where remote participants may withdraw from the experiment early and it is not possible to debrief them, the fact that doing so will result in their not being debriefed should be mentioned in the introductory text.

12. All the data collected from the participants is stored in an anonymous form.
    All participant data (hard-copy and soft-copy) should be stored securely, and in anonymous form.

Project title **Automatic Illustration of Text via Multimodal Interaction**

Student's Name **Stergious Aji**

Student Number **2546916**

Student's Signature 

Supervisor's Signature 

Date 02/03/23

D Usability Evaluation Brief

Automatic Illustration of Text

Text-to-Image Relevance Assessment Task

This document explains the experiment that you are about take part in. You will be asked to use a prototype application that provides the capability to construct ground truths for automatic videography generation tasks. The experiment aims to measure the usability of the interface and so you will not be given detailed instructions as to how to complete the task. You must use what is available on the web pages to navigate through the application and complete your objectives.

The task will involve constructing a ground truth using the application for the provided audio source below. You are free to browse YouTube or the local device for the audio source. Once the appropriate source is found, you should input this into the application and can subsequently start annotating the true labels for each textual chunk within the audio. You will be given 10 minutes but you are not expected to finish annotating the full ground truth.

Your chosen audio source is:

[AUDIO SOURCE INFORMATION]

Once, the experiment has concluded, you will be asked to complete a short usability survey of your experience interacting with the web application. The data gathered will be anonymised and no personal details will be collected.

E Usability Survey Form

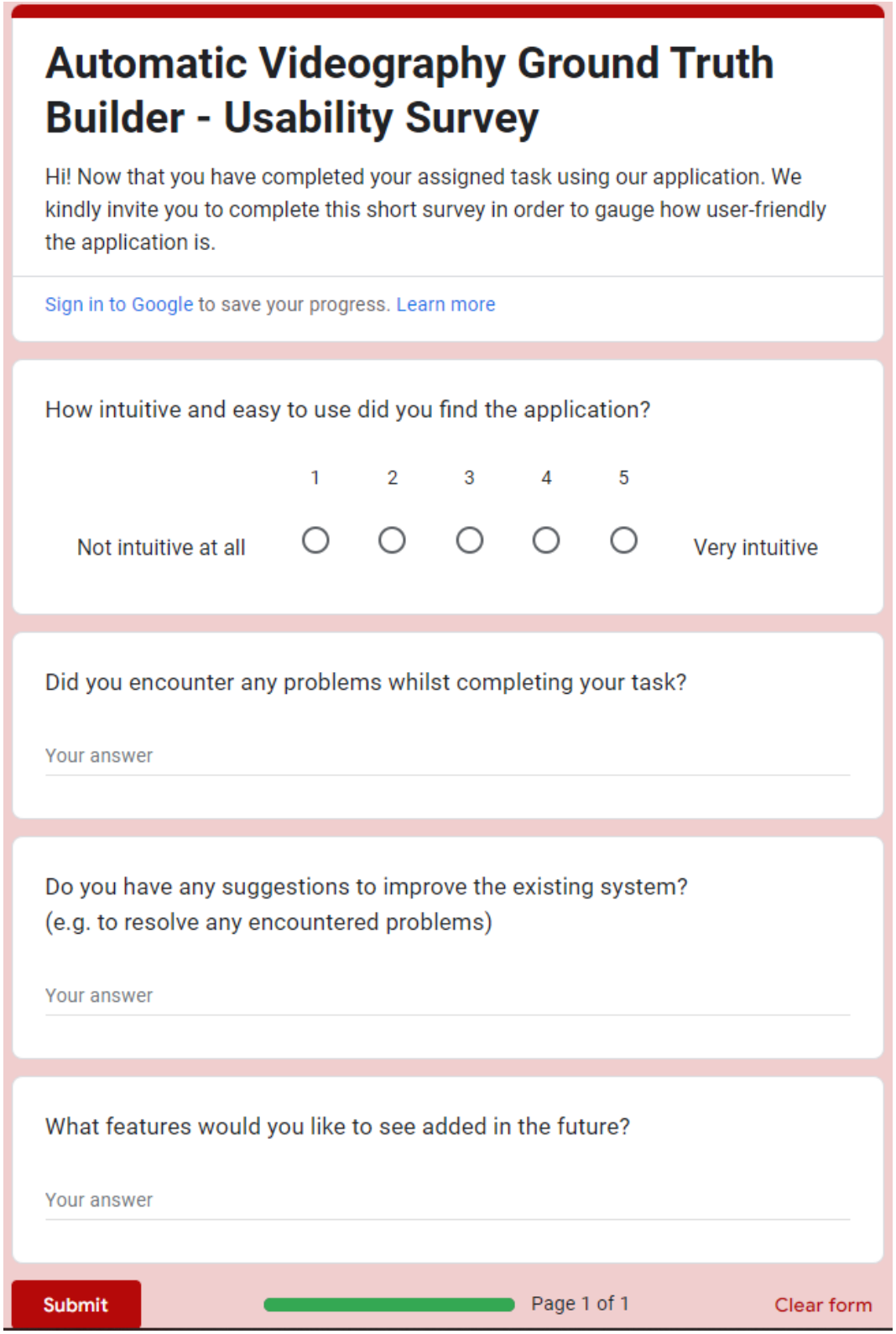


Figure E.1: *The usability survey form that was given to all sixteen participants after completing their assigned task.*

Bibliography

Arampatzis, A., Kamps, J. and Robertson, S. (2009), Where to stop reading a ranked list? threshold optimization using truncated score distributions, *in* 'Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval', pp. 524–531.

Bellman, R. (1957), *Dynamic Programming*, Dover Publications.

Burr, S. (2012), 'Active learning.', *Synthesis Lectures on Artificial Intelligence and Machine Learning* **6**(1), 1.

Campos, R., Mangaravite, V., Pasquali, A., Jorge, A., Nunes, C. and Jatowt, A. (2020), 'Yake! keyword extraction from single documents using multiple local features', *Information Sciences* **509**, 257–289.

Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F. and Li, H. (2007), Learning to rank: from pairwise approach to listwise approach, *in* 'Proceedings of the 24th international conference on Machine learning', pp. 129–136.

Chang, N. and Fu, K. (1979), 'A relational database system for images', *Pictorial Information Systems* pp. 288–321.

Chang, S.-K. and Kunil, T. (1981), 'Pictorial data-base systems', *Computer* **14**(11), 13–21.

Clegg, D. and Barker, R. (1994), *Case method fast-track: a RAD approach*, Addison-Wesley Longman Publishing Co., Inc.

Cleverdon, C. (1967), The cranfield tests on index language devices, *in* 'Aslib proceedings', Vol. 19, MCB UP Ltd, pp. 173–194.

Cuénod, J. (2015), 'imgcheckbox', `https://jcuenod.github.io/imgCheckbox/`. Accessed: 2023-03-12.

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

Django (2005), 'The web framework for perfectionists with deadlines', `https://www.djangoproject.com/`. Accessed: 2023-02-14.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S. et al. (2020), 'An image is worth 16x16 words: Transformers for image recognition at scale', *arXiv preprint arXiv:2010.11929* .

Feng, S. (2019), 'Color Thief'. Accessed: 2023-03-11.
URL: *`https://github.com/fengsp/color-thief-py/`*

Figma (2016), 'Figma'. Accessed: 2023-03-07.
URL: *`https://figma.com/`*

Fleming, N. D. (2006), 'Vark visual, aural/auditory, read/write, kinesthetic', *New Zealand: Bonwell Green Mountain Falls* .

Frallan, F. (2022), 'Multilingual-clip', `https://github.com/FreddeFrallan/Multilingual-CLIP`. Accessed: 2023-02-14.

Ganguly, D., Parker, A. and Aji, S. (2023), Automatic videography generation from audio tracks, *in* 'Advances in Information Retrieval: 45th European Conference on Information Retrieval, ECIR 2023, Dublin, Ireland, April 2–6, 2023, Proceedings, Part III', Springer, pp. 281–287.

Gao, H., Zhang, J., Mao, J., Zhu, J., Song, W. and Huang, H. (2018), Text-guided attention model for image captioning and visual question answering, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition', pp. 19–28.

Genius (2009), 'Genius'. Accessed: 2023-03-06.
URL: *https://genius.com/*

GitHub (2008), 'Github'. Accessed: 2023-03-10.
URL: *https://github.com/*

Gordo, A., Almazan, J., Revaud, J. and Larlus, D. (2017), 'End-to-end learning of deep visual representations for image retrieval', *International Journal of Computer Vision* **124**(2), 237–254.

Gupta, C., Yılmaz, E. and Li, H. (2020), 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, pp. 496–500.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C. and Oliphant, T. E. (2020), 'Array programming with NumPy', *Nature* **585**(7825), 357–362.
URL: *https://doi.org/10.1038/s41586-020-2649-2*

ImageNet (2006), 'Imagenet'. Accessed: 2023-03-10.
URL: *https://www.image-net.org/*

Johnson, J., Douze, M. and Jégou, H. (2019), 'Billion-scale similarity search with GPUs', *IEEE Transactions on Big Data* **7**(3), 535–547.

Karpathy, A. and Fei-Fei, L. (2015), Deep visual-semantic alignments for generating image descriptions, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition', pp. 3128–3137.

Labelbox (2018), 'Labelbox: The leading training data platform for data labeling.'. Accessed: 2023-03-06.
URL: *https://labelbox.com/*

Lavrenko, V., Manmatha, R. and Jeon, J. (2003), A model for learning the semantics of pictures, *in* 'Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval', ACM, pp. 187–194.

Lee, M. (2007), 'pytesseract: Python-tesseract is a python wrapper for google's tesseract-ocr', `https://pypi.org/project/pytesseract/`. Accessed: 2023-03-06.

Lien, Y.-C., Cohen, D. and Croft, W. B. (2019), An assumption-free approach to the dynamic truncation of ranked lists, *in* 'Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval', pp. 79–82.

Milvus (2020), 'How to Select Index Parameters for IVF Index', https://milvus.io/blog/select-index-parameters-ivf-index.md. Accessed: 2023-03-20.

Mohammadi, P., Ebrahimi-Moghadam, A. and Shirani, S. (2014), 'Subjective and objective quality assessment of image: A survey', *arXiv preprint arXiv:1406.7799* .

Musixmatch (2010), 'Musixmatch developer', https://developer.musixmatch.com/. Accessed: 2023-02-14.

Numenorean (2021), 'Shazamapi'. Accessed: 2023-02-14.
URL: *https://github.com/Numenorean/ShazamAPI/*

Parker, A. (2022), 'Automatic Videography of Audio Tracks of Songs', https://github.com/AndrewParker770/Automatic-Videography-of-Audio-Tracks-of-Songs. Accessed: 2023-02-20.

Pezoa, F., Reutter, J. L., Suarez, F., Ugarte, M. and Vrgoč, D. (2016), Foundations of json schema, *in* 'Proceedings of the 25th International Conference on World Wide Web', International World Wide Web Conferences Steering Committee, pp. 263–273.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Gohil, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J. et al. (2021), 'Learning transferable visual models from natural language supervision'.

Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C. and Sutskever, I. (2022), 'Robust speech recognition via large-scale weak supervision'.
URL: *https://arxiv.org/abs/2212.04356*

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I. et al. (2018), 'Improving language understanding by generative pre-training'.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I. et al. (2019), 'Language models are unsupervised multitask learners', *OpenAI blog* **1**(8), 9.

Rahman, M. M., Kutlu, M., Elsayed, T. and Lease, M. (2020), Efficient test collection construction via active learning, *in* 'Proceedings of the 2020 ACM SIGIR on International Conference on Theory of Information Retrieval', pp. 177–184.

Rasiwasia, N., Mahajan, D., Chen, V., Takacs, G. and Ahuja, N. (2010), A new approach to cross-modal multimedia retrieval, *in* 'Proceedings of the 18th ACM international conference on Multimedia', ACM, pp. 251–260.

Reitz, K. (2011), 'Requests'. Accessed: 2023-03-12.
URL: *https://pypi.org/project/requests/*

Robertson, S. E. and Jones, K. S. (1976), 'Relevance weighting of search terms', *Journal of the American Society for Information science* **27**(3), 129–146.

Ronnie Ghose, Taylor Fox Dahlin, N. F. (2022), 'pytube'. Accessed: 2023-02-14.
URL: *https://github.com/pytube/pytube/*

Salton, G. (1971), *The SMART retrieval system—experiments in automatic document processing*, Prentice-Hall, Inc.

Samimi, P. and Ravana, S. D. (2014), 'Creation of reliable relevance judgments in information retrieval systems evaluation experimentation through crowdsourcing: a review', *The Scientific World Journal* **2014**.

Shiang-shiang, K. D. (2023), 'Lyrics (lrc) file format'. Accessed: 2023-02-14.
URL: *https://docs.fileformat.com/misc/lrc/*

Singer, U., Polyak, A., Hayes, T., Yin, X., An, J., Zhang, S., Hu, Q., Yang, H., Ashual, O., Gafni, O. et al. (2022), 'Make-a-video: Text-to-video generation without text-video data', *arXiv preprint arXiv:2209.14792* .

Srinivasan, K., Raman, K., Chen, J., Bendersky, M. and Najork, M. (2021), 'Wit: Wikipedia-based image text dataset for multimodal multilingual machine learning', *arXiv preprint arXiv:2103.01913* .

SuperAnnotate (2017), 'Superannotate: The ultimate training data platform for ai.'. Accessed: 2023-03-07.
URL: *https://www.superannotate.com/*

Team Pepper (2022), 'The rise of video content: A brief explainer'. Accessed: 2023-02-15.
URL: *https://www.peppercontent.io/blog/rise-of-video-content/*

Van Rossum, G. and Drake, F. L. (1991), 'Python'. Accessed: 2023-03-10.
URL: *https://www.python.org/*

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. and Polosukhin, I. (2017), Attention is all you need, *in* 'Advances in neural information processing systems', pp. 5998–6008.

Vinyals, O., Toshev, A., Bengio, S. and Erhan, D. (2015), Show and tell: A neural image caption generator, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition', pp. 3156–3164.

Voorhees, E. M., Harman, D. K. et al. (2005), *TREC: Experiment and evaluation in information retrieval*, Vol. 63, Citeseer.

Wang, A. (2006), 'The shazam music recognition service', *Communications of the ACM* **49**(8), 44–48.

Wikimedia Commons (2004), 'Wikimedia commons'. Accessed: 2023-03-10.
URL: *https://commons.wikimedia.org/*

Yeh, H.-C. (2018), 'Exploring the perceived benefits of the process of multimodal video making in developing multiliteracies'.
URL: *https://scholarspace.manoa.hawaii.edu/handle/10125/44642*

YouTube (2005), 'Youtube', YouTube, YouTube, LLC. Accessed: 2023-03-03.
URL: *https://www.youtube.com/*

Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X. and Wei, D. (2017), Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks, *in* 'Proceedings of the IEEE International Conference on Computer Vision', pp. 5907–5915.

Zheng, L., Yang, Y. and Tian, Q. (2017), 'Sift meets cnn: A decade survey of instance retrieval', *IEEE transactions on pattern analysis and machine intelligence* **40**(5), 1224–1244.

Zulko (2017), 'Moviepy', `https://github.com/Zulko/moviepy`. Accessed: 2023-03-11.