EXAMENSARBETE INOM DATATEKNIK,
GRUNDNIVÅ, 15 HP

STOCKHOLM, *SVERIGE 2020*

Automated Intro Detection For TV Series
Automatiserad detektion av intron i TV-serier

JACOB EKEDAHL

TIAGO REDAELLI

KTH
SKOLAN FÖR KEMI, BIOTEKNOLOGI OCH HÄLSA

Automated Intro Detection For TV Series

Automatiserad detektion av intron i TV-serier

Jacob Ekedahl
Tiago Redaelli

Degree Project in Computer Engineering
Basic Level, 15 hp
Supervisor at KTH: Jonas Willén
Examinator: Ibrahim Orhan
TRITA-CBH-GRU-2020:005

KTH
The Institute of Chemistry, Biotechnology and Health
141 52 Huddinge, Sweden

Abstract

Media consumption has shown a tremendous increase in recent years, and with this increase, new audience expectations are put on the features offered by media-streaming services. One of these expectations is the ability to skip redundant content, which most probably is not of interest to the user. In this work, intro sequences which have sufficient length and a high degree of image similarity across all episodes of a show is targeted for detection.

A statistical prediction model for classifying video intros based on these features was proposed. The model tries to identify frame similarities across videos from the same show and then filter out incorrect matches. The performance evaluation of the prediction model shows that the proposed solution for unguided predictions had an accuracy of 90.1%, and precision and recall rate of 93.8% and 95.8% respectively. The mean margin of error for a predicted start and end was 1.4 and 2.0 seconds. The performance was even better if the model had prior knowledge of one or more intro sequences from the same TV series confirmed by a human. However, due to dataset limitations the result is inconclusive.

The prediction model was integrated into an automated system for processing internet videos available on SVT Play, and included administrative capabilities for correcting invalid predictions.

Keywords - intro detection, Hidden Markov model, feature selection, image similarity comparison, average hash, SVT

Sammanfattning

Under de senaste åren så har konsumtionen av TV-serier ökat markant och med det tillkommer nya förväntningar på den funktionalitet som erbjuds av webb-TV tjänster. En av dessa förväntningar är förmågan att kunna hoppa över redundant innehåll, vilket troligen inte är av intresse för användaren. I detta arbete så ligger fokus på att detektera video intron som bedöms som tillräckligt långa och har en hög grad av bildlighet över flera episoder från samma TV-program.

En statistisk modell för att klassificera intron baserat på dessa egenskaper föreslogs. Modellen jämför bilder från samma TV-program för att försöka identifiera matchande sekvenser och filtrera bort inkorrekta matchningar. Den framtagna modellen hade en träffsäkerhet på 90.1%, precision på 93.8% och en återkallelseförmåga på 95.8%. Medelfelmarginalen uppgick till 1.4 sekunder för start och 2.0 sekunder för slut av ett intro. Modellen presterade bättre om den hade tillgång till en eller fler liknande introsekvenser från relaterade videor från samma TV-program bekräftat av en människa. Eftersom datasetet som användes för testning hade vissa brister så ska resultatet endast ses som vägledande.

Modellen integrerades i ett system som automatiskt processar internet videos från SVT-Play. Ett tillhörande administrativt verktyg skapades även för att kunna rätta felaktiga gissningar.

Nyckelord - intro detektion, dold Markovmodell, attributselektion, bildlikhet, average hash, SVT

1 Introduction

This chapter presents the premise of the thesis, explaining the problem definition, followed by its objectives and delimitations.

1.1 Problem Definition

The number of TV-broadcast viewers is steadily decreasing and being replaced by a younger generation who favor digitalization and video on demand streaming services. The changes in the last five years have been significant. In Sweden, nearly one in three internet users watch film and video online on a daily basis, which is almost a doubling since 2015, and more than three times as much since 2014. A majority of people under the age of 36 watch videos online on a daily basis, and more than 94% of all under the age of 46 watch some of the time [1]. This generational shift, with a drastic increase in media consumption and a greater desire for personalized video on demand services, changes what audiences expect from the play services that they use. International media-service providers are also leveraging the digitalization trend to enter the Swedish media market, increasing the competition over audiences.

To illustrate these two trends one can study the case of Netflix who introduced a skip button in 2017, enabling users to skip intro and outro sequences from some of their TV shows. The motivation behind the creation of the skip button was Netflix discovering that their users preferred to watch more than one episode per sitting [2]. This makes the showing of multiple intro sequences during a viewing session something that interrupts the natural progression of the story. The introduction of skip capabilities on one platform as popular as Netflix, puts pressure on other play service providers to introduce similar features to their platforms. Failing to do so increases the risk of them being negatively compared and thus, reduces audience satisfaction and loyalty.

Sveriges Television (SVT) is a Swedish public service broadcasting company. One of their core services is SVT Play, which is a video on demand streaming service hosting both live broadcasts and videos recently shown on their channels. SVT Play does not currently have any intro skipping capabilities. Creating such a feature utilizing manual input from their employees would both be impractical and expensive given the large array of media content under management. There is therefore a need to create an application which is capable of automatically detecting skippable video sequences.

1.2 Objectives

The primary objective of this work is to create a television series intro classification system, capable of identifying when an intro sequence starts and ends, given a set of videos from the same show. The viability and accuracy of the implemented algorithm must also be established. Furthermore, the system must automatically process videos continuously as they are uploaded to SVT Play and save this information in a database.

The created system is also expected to work in conjunction with SVTs content management system.

The final product will be an application divided into a front end and a back end. The back end will perform routine work and store the results from successful intro predictions. The front end will allow an administrator to see the current intro prediction of a video, request a new prediction to be made, or manually annotate a different sequence as an intro.

1.3 Delimitations

For this work the following limitations have been made:

1) The dataset will be created exclusively from TV series which are publicly available on SVT Play.
2) Intro sequences are limited to those that contain a coherent visual and audible thematic style across all episodes of a season for a given show. Furthermore, it must be sufficiently long and consist of a continuous segment. Intro sequences that do not adhere to these limitations are not considered skippable, and therefore not of interest.
3) The system must be implemented without the use of any third party software as a service solution.

2 Background

Background provides the information necessary to get a general understanding of the problem. First skippable film sequences are categorized and defined, followed by an overview of skipping capabilities introduced by other media-streaming services. Lastly, section 2.3 introduces related work, including a description of Netflix's algorithm [5], and a study on automated intro and outro detection for internet television [6].

2.1 Categorization of Skippable Film Sequences

This chapter presents various categories of film sequences that viewers would want to skip are summarized. First the title sequence is introduced and classified, followed by a description of the pre-intro, teaser and outro sequences.

2.1.1 Intro Sequence

A title sequence (also called an opening sequence or intro) is a method in which films or television programs present the title and credit key production and cast members, and helps to establish the tone of the program. Title sequences may take various forms, and may consist of live action, animation, music, still images or motion graphics. In some films and television shows in particular, the title sequence is preceded by a cold open or hook to entice audience members to stay and watch. Television series stand out in that the title sequence typically sound and look the same across all episodes of a season, which makes it a top priority to be able to identify.

Title sequences come in a myriad of different styles, as exemplified in Figure 2.1, and it can be difficult to define what constitutes a title sequence or how to classify them from each other. From a historical context Melis Inceer outlines four types of film title categories: titles superimposed on a black screen, titles accompanied by still images, titles accompanied with a series of moving images and titles built around animation and motion graphics [4]. A new category is also introduced for titles that are accompanied with normal film shots.

Figure 2.1: Illustrates some of the different styles of title frames used by filmmakers.

a) Titles superimposed on a blank screen

The simplest titles are superimposed on a blank screen utilizing different typefaces. Darker backgrounds are preferred over lighter background alternatives, as it is less tiring for the viewer's eyes. For the text to be legible a high contract is needed between the background and the foreground, leading to white text being the preferred option. This type of title design is less common today but is still used due to its low cost and lack of complexity. Filmmakers can also choose this type of title sequence intentionally, as it can give an old-fashioned and authentic or refined look to a film.

b) Titles accompanied by still images

The development of incorporating still images and borders to the title sequence began with a need to make longer title sequences more visually appealing to the audience, as more people began to be credited for their work. This type of style also allows for the introduction of a narrative element, such as introducing important characters or places, as well as establishing a visual style to the film.

c) Titles accompanied with a series of moving images

Titles over a series of moving images can range from simple view of clouds moving in the sky, to a more intricate sequence of images that incorporate camera movement. This style allows for narrative thread to take place while the credits are being displayed. One example that incorporates this style in parts of its introduction is *The Good, The Bad, and the Ugly* (1966), which incorporates animations of a riding horse between title images, and images of the main characters.

d) Titles built around animations and motion graphics

Many contemporary movies rely on animations and motion graphics in the opening credits. The text can be animated and continuously moving or have other effects applied to it, simultaneously as things are taking place around it. *Kiss Kiss Bang Bang* (2005) and *Sin City* (2005) are examples of this style. There are also many films that rely on moving images combined with animated texts, creating a certain degree of ambiguity between if a title sequence is relying on moving images or motion graphics.

e) Titles accompanied by normal film shots

Titles shown simultaneously as the story begins to unfold are the most ambiguous. The credits can remain in the center of the screen or be moved to the side to allow room for movie shots. The movie shots can be simple and atmospheric, where the camera for instance could be following a moving car in the distance, or more complex, with acting taking place. For the purposes of this thesis there is also a question whether these shots

should be considered intro titles at all as important film sequences can take place during such scenes, and there is little to no interest in being able to skip them if that is the case.

2.1.2 Pre-intro Sequence

A pre-intro sequence is a category introduced by the authors of this report. The pre-intro sequence can easily be mistaken for an intro sequence, in that it has similar characteristics in both visual and auditory style. The only distinguishing difference between the two, is that the pre-intro usually starts in the opening sequence of a video and typically presents the production company.

2.1.3 Teaser Sequence

A teaser sequence is a narrative tactic used in film to hook to captivate the audience before the title sequence is shown by introducing parts of the story. Teaser sequences may also be used in TV series to recap events shown in previous episodes. Recapturing sequences are typically characterised by being a quick succession of shots which have high intensity and only last a few seconds each. The repetitive nature of the teaser sequence makes it a noteworthy segment to be able to detect and skip.

2.1.4 End Sequence

The end sequence (also known as outro or credits) is the final part of the film, where the filmmaker lists the people who contributed to the production. End sequences are characterized in a few distinct ways: First, that it appears at the end of a movie or show. Second, that it is almost always comprised of text. Third, the absence of dialogue or speech. Finally, there is very little variation between contiguous frames. Outro detection algorithms would have to utilize these distinct characteristics to try and determine when an outro started.

2.2 Skip Features Offered by Other Media-Streaming Services

In 2017 Netflix began testing a button for skipping title sequences on some of their original TV shows, making them one of the first media companies to introduce such a feature to a wider audience [3]. The skip button appears at the beginning of an intro or outro sequence, and allows Netflix's users to jump directly into the story without having to watch the same intro or outro. The skip button arguably improves the viewing experience when watching multiple episodes during one sitting as the intro or outro sequence is a repetitive experience. This feature has since been adopted to more shows and is now available on most of the platforms which Netflix is available on. Amazon Prime is another media-streaming service provider who has since then also introduced a similar feature to their platforms.

The adoption of the skip button also brings with it new applications, such as BingeEasy. BingeEasy is a chrome extension that allows users to skip ads and themed intros on videos from YouTube, Netflix and Amazon Prime. It is one of many applications of its kind that are built on top of the features offered by media-streaming

services. BingeEasy automates the process of pressing the skip button whenever it is available, making it a default setting for people who use it.

2.3 Related Work

There has been some related work within the field of automatically detecting specific content in videos, such as commercials, credits, intros and outros. In this chapter we present some of the previous work done related to identifying intro sequences.

2.3.1 Intro Detection Algorithm by Netflix

Apurva Kansara, a senior software engineer at Netflix provides a brief summary of the intro sequence detection algorithm which Netflix uses [5]: First the algorithm looks for similar frames across multiple video assets. Then the visual fingerprints are extracted from a collection of certain frames along with their subsequent image histograms. These fingerprints are later used as a comparative model, if similar frames appear in another video they are marked as either the beginning or end of the intro sequence.

2.3.2 Automated Intro and Outro Detection on Internet Television

In a paper by Maryam Nematollahi and Xiaoping Zhang [6], a selected group of methods for detecting intro and outro sequences are evaluated. These methods are condensed to extracting computationally efficient features - such as blank screen transitions, histogram of shot boundaries and silence gaps - and develop a framework for identifying intros and outros based on those features. Their best performing algorithm successfully extracts intro and outro transitions with a detection rate of 82% and 76% with a mean margin of margin less than 2.06 seconds.

3 Theory

This chapter presents the theoretical foundation for the proposed solution to the problem defined in section 1.1. Section 3.1 presents a tool used for information retrieval from web pages. Section 3.2 describes feature selection and feature extraction for identifying predictive features from complex data. Section 3.3 presents methods for comparing the similarity between images. Section 3.4 presents the classification and regressive predictive modeling, and how to map a regressive problem to a classification problem. The chapter concludes with 3.5, which presents a tool for solving classification problems of sequential data.

3.1 Web Data Extraction

Web data extraction or web scraping is a technique to automatically load and extract data from web pages. One main component of web scraping is web crawling. The main purpose of web crawling is to fetch web pages for later stages of preprocessing [7]. Once the page has been fetched and the relevant data has been extracted and downloaded, the data can be stored in a local file storage or in a database for further analysis and preprocessing. If the data is complex or the problem one is trying to solve requires it, a common initial step of preprocessing is to extract relevant features from the data using methods of feature extraction and selection.

3.2 Feature Extraction and Feature Selection

Feature extraction is a common way to obtain new informative and hopefully non-redundant attributes from complex data assets, such as video, image or audio. These features facilitate the subsequenting learning and generalization steps for creating models which can be used for classification tasks or making valuable predictions about closely related data.

Feature selection is the task of filtering irrelevant and redundant features. It can be used isolated or in conjunction with feature extraction with the aim to increase the accuracy and comprehensibility of learned knowledge [8].

3.2.1 Categories of Features

Features can be categorized as *relevant*, *redundant* or *irrelevant*. These categorizes are determined based on how each subset of extracted data contributes to improve the predictive accuracy of the resulting model.

3.3 Image Similarity Comparison

The task of comparing images for evaluating their similarities or identifying features which separates them is an area of research consisting of many different approaches.

How well each algorithm performs depends on the features of the images getting compared. Some algorithms can perform well even though the scale, color saturation and rotation of the images are different. The Scale-invariant feature transform or the Oriented FAST and Rotated Brief algorithms are such examples [9]. And other methods might require less computational resources at the cost of accuracy, for example average hashing. Some of the use cases for image similarity comparison are for visual search, feature extraction or detecting copyright infringements [10]

3.3.1 Average hashing

Average hashing is an algorithm which generally takes less computational resources compared to instance oriented fast and rotated brief (ORB). It performs poorly in domains where images have different rotations or color saturations and sometimes outputs false positives.

A simple implementation of this algorithm consists of reducing the size of the images to remove high frequencies and details. The color of each pixel in each image are converted to a grayscale. The average color is calculated from each value of the pixels. The hash is calculated by comparing each pixel brightness compared to this average color. This will output a list of ones and zeroes, called the hash. The similarity is calculated by counting the number of different bits between the hashes, also called the hamming distance [11].

3.3.2 Oriented FAST and Rotated Brief

ORB is a relatively fast algorithm and is rotation invariant and resistant to noise and has been demonstrated to perform two orders of magnitude compared to Scale-invariant feature transform (SIFT) algorithm. It is a combination of the methods called binary robust independent elementary features (BRIEF) and features from accelerated segment test (FAST). FAST is a corner detection method to extract feature points. And BRIEF creates binary feature vectors from the feature points. The same type of data structure used in average hashing, which is then used to compare similarities between images [9].

3.4 Predictive Modeling

Predictive modeling is the process of taking known results and developing a model that can be used to forecast future outcomes [12]. It can be described as the mathematical problem of function approximation; mapping a function from input variables to output variables. In this section two subfields within predictive modeling are presented, regression and classification.

3.4.1 Regression Predictive Modeling

Regression is the problem of predicting a continuous output variable, such as the value of a stock price or a timestamp. It can handle both discrete or continuous input variables. A regression problem where the input variables are ordered by time is called a forecasting problem [13].

3.4.2 Classification in Predictive Modeling

In machine learning and statistics, classification is the problem of identifying which set of categories a new element belongs to. The ability to accurately predict what class an observation belongs to is valuable for various business applications like detecting diseases in patients or predicting if a user will buy a product or not.

Classification of data is an instance of supervised learning, which is a function that maps labeled training data into a set of predetermined categories. Classifying algorithms are commonly separated into multi-labeled and binary classifiers with some algorithms being categorized into both. Depending on the problem definition for this thesis we are focusing on binary classification.

3.4.3 Converting Regression Into Classification

Mapping regression into classification allows the extension of classification systems to regression domains. This is achieved by labeling ranges of a continuous output value into a discrete class, also called discretization [14].

3.4.4 Performance Evaluation for Predictive Modeling

One of the most common methods used to measure performance of a predictive model is to calculate various ratios based on the observed count of true positives, true negatives, false positives and false negatives. The elements that are correctly classified are called true positives and true negatives. A true positive is an outcome where the model correctly predicted the positive class, and a true negative is an outcome where the model correctly predicted the negative class. Similarly, elements that are incorrectly classified are called false negatives and false positives. A false positive is an outcome where the model incorrectly predicts the positive class, and a false negative is an outcome where the model incorrectly predicted a negative class. The observed result can then be organized into a 2x2 contingency matrix, as exemplified in Figure 3.1 [15].

	Condition Positive (CP)	Condition Negate (CN)
Outcome Positive (OP)	True Positive Count (TP) ● Expected: True ● Outcome: True	False Positive Count (FP) ● Expected: False ● Outcome: True
Outcome Negative (ON)	False Negative Count (FN) ● Expected: True ● Outcome: False	True Negative Count (TN) ● Expected: False ● Outcome: False

Figure 3.1: Contingency matrix for validation of a binary classifier.

The condition positive count (P) and condition negative count (N) and are based on the expected outcomes in the dataset. The condition positive count can be calculated by adding the true positive count with the true negative count. Conversely, the condition negative count can be calculated by adding the false positive count with the true negative count.

$$P = TP + FN \tag{3.1}$$

$$N = FP + TN \tag{3.2}$$

The true positive rate (also known as recall rate) of a classifier is estimated by dividing the true positive count by the condition positive count.

$$TPR = \frac{TP}{P} = \frac{TP}{TP + FN} \tag{3.3}$$

The true negative rate of a classifier is estimated by dividing the true negative count by the condition negative count.

$$TNR = \frac{TN}{N} = \frac{TN}{TN+FP} \tag{3.4}$$

The overall accuracy of a classifier is estimated by dividing the total of correctly classified positives and negatives by the total number of samples.

$$ACC = \frac{TP+TN}{P+N} = \frac{TP+TN}{TP+TN+FP+FN} \tag{3.5}$$

Positive predictive value (also known as precision) is determined by the hit rate of predicted true outcomes and is calculated by dividing the true positive count with the total number of predicted true outcomes.

$$PPV = \frac{TP}{TP+FP} \tag{3.6}$$

Of the previously mentioned ratios, the most important for information retrieval are recall (3.3), precision (3.6) and accuracy (3.5). Accuracy combines precision and recall into one number via choice of weighing. F1-score is an alternative to accuracy but based on equal weighing.

3.5 Hidden Markov Model

A hidden markov model (HMM) is a statistical model useful for creating probabilistic models when facing linear sequence labeling problems. It can be used to solve both classification or regression problems. It infers the hidden state we are interested in from a sequence of observed events by augmenting a markov chain [16].

A markov chain is a stochastic model describing a sequence of states in which the probability of the next state occurring is dependent solely on its present state. It embodies the **Markov assumption** $P(q_i = a|q_1 \ldots q_{i-1}) = P(q_i = a|q_{i-1})$ that the past is irrelevant for predicting a future state, only the present matters. Expanding on this assumption, a first-order hidden markov relies on the probability of an output observation (emission probability) to only be dependent on the state which produced the observation. It is often depicted as a state machine, seen in Figure 3. A HMM is defined by the following components [16]:

1. A set of **hidden states**
2. A sequence of T **observations** $O = o_1 o_2 \ldots o_T$
3. A **transition probability matrix** denoted by A where $A = a_{11} \ldots a_{ij} \ldots a_{NN}$ and each element a_{ij} represents the probability of moving from state i to state j. The sum of all probability transitions in matrix A equals one.
4. A sequence of **emission probabilities** denoted by B, where $B = b_i(o_t)$. Each element expresses the probability of an observation arriving from state i.
5. The **initial probability distribution** of what state the markov chain starts in.

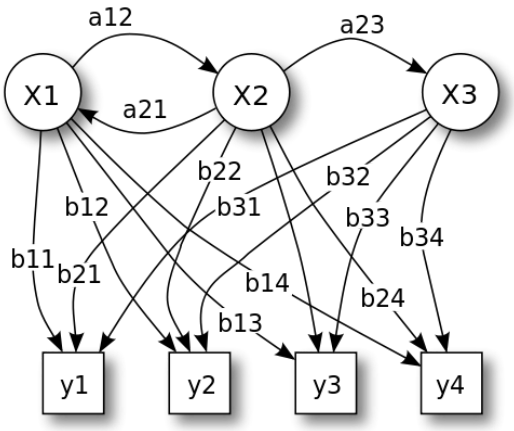

Figure 3: Depiction of a hidden markov model. X are the hidden states, y are observations, a is the state transition probabilities and b is the emission probabilities [17].

3.5.1 Application of a Hidden Markov Model

A hidden markov model is characterized to solve three fundamental problems, which are:

1. Determine the likelihood of a hidden state being present given a sequence of observed events.
2. Given a sequence of observed events determine the best sequence of hidden states.
3. Given an observation sequence and a set of hidden states in the hidden markov model, learn the transition probabilities and the emission probabilities.

In this thesis we focus on identifying the best sequence of hidden states using the viterbi algorithm and learning the transition and emission probabilities using supervised learning [16].

3.5.2 The Viterbi Algorithm

The viterbi algorithm is a recursive optimal solution for finding the most likely sequence of hidden states given a HMM and a sequence of observations. It is optimal from a probabilistic perspective, which means that the algorithm tries to identify the sequences of hidden states that maximizes the joint probability of observations given the HMM [18].

3.5.3 Learning the Parameters of the Hidden Markov Model

Given a dataset with sequences of observations and labels representing each observation we can derive the transition probabilities using a maximum likelihood estimation (MLE). The algorithm is based on the assumption that each event is independent of the others. Using this simplification one can make the conclusion that the product of observing each event is equal to observing all events. The probability density of observing event x is thus given by: $P(x; \mu, \sigma) = 1 \div \sigma\sqrt{2\pi}\ exp(-(x-\mu)^2 \div 2\sigma^2$. For example, if we pass in N number of labels (hidden states) denoted by S, where $S = s_1 s_2 \ldots s_N$ and s_2 is followed by s_3 10% of the time and s_1 90% of the time the transition probabilities of s_2 will be 10% and 90% respectively [19].

Using the MLE we can derive the emission probabilities by calculating for each observation the probability of being derived from each state. More exactly, for a given observed event , this would be the number of occurrences event is derived from a given state divided by the total amount of occurences in the dataset. The emission probabilities can be calculated by counting the occurrences of each state at the start of each observation sequence divided by the total amount of sequences in the dataset.

4 Methodology

This chapter presents the chosen methods and tools used to realize the objectives defined in section 1.2.

The method used to develop the solution consisted of the following steps:

1. Analyze the tv series to be evaluated from svtplay and identify which methods to use and features to select for creating a predictive model.
2. Develop and evaluate a predictive model

4.1 General choice of Approach

To estimate the start and the end of an introduction for a given video, two main strategies on which to base the predictive model on was considered. The first strategy relies solely on the features extracted within each individual video. Conversely, the second strategy is based on identifying similarities across related videos. A combination of both strategies would also be possible.

In the case of TV series, it was observed that intro sequences belonging to videos from the same show or season had a high degree of visual, auditory similarity, as perceived by a human observer. The observation was further substantiated by the technical summary of the Netflix Algorithm provided by Apurva Kansara [5], and the finding that a majority of intro sequences from the same season had similar lengths. Additionally, because of the various styles that a filmmaker can choose when making an intro, a strategy based solely on features of a single video would be much more complex problem to solve, in terms of identifying features with a high predictive value, capable of separating intro sequences from other video sequences. Because of these factors, the choice was made to focus primarily on identifying similarities across related videos.

4.2 Video Dataset

The dataset was derived entirely from videos which at the time were publicly available on SVT Play. The chosen limitation was to only pick videos that were present under the TV series category. In total 243 videos belonging to 32 different shows were used, which was only a small subset of the 1300 videos and 100 shows available under the same category. The created dataset was then used to evaluate the performance of the tested prototypes, including the predictive model described in section 4.2.

The process of annotating a video intro consisted of writing into a shell script the start and end points, and the url of the video which should be included in the dataset. The shell script in turn executed a python script which validated the timestamps before saving the data.

The criteria for being included was initially that the video had to have an intro which was a continuous sequence longer than 4 seconds. This means that intro sequences

which were shorter, or consisted of multiple sequences split between shots were excluded. Videos that did not have an intro sequence were still included if it was part of a season which mostly consisted of accepted intro sequences. In the later stages of the development, a few videos that did not have any intro sequences were also included, making up a total of 4% of the dataset. This was to some extent test the algorithm capability of correctly detecting the absence of an intro sequence.

4.2.1 Statistical Analysis

A statistical analysis of the dataset was conducted in order to investigate if there were any features which could be utilized when constructing the predictive model. From the analysis it was revealed that 35% intros started at the opening scenes of the episode. The average length of an intro sequence was 35.6 seconds, with a standard deviation of 15.9 seconds. On average intro sequences started after 39.4 seconds, with a standard deviation of 10.8 seconds. The one minute mark was also a common starting point for intros, which can be seen in figures 4.1 and 4.2 respectively.

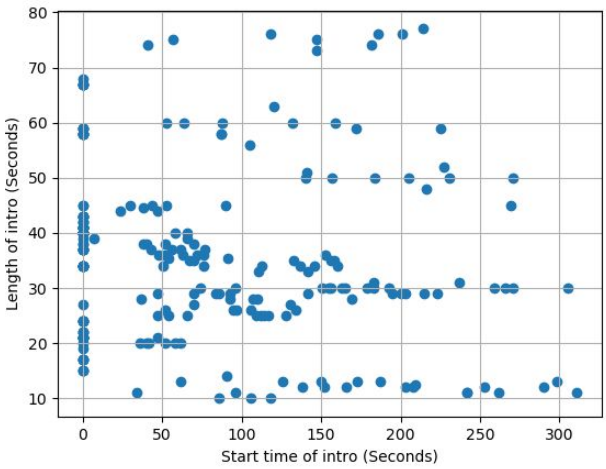

Figure 4.1: The figure illustrates the distribution of when intro sequences start, correlated with the length of the sequence.

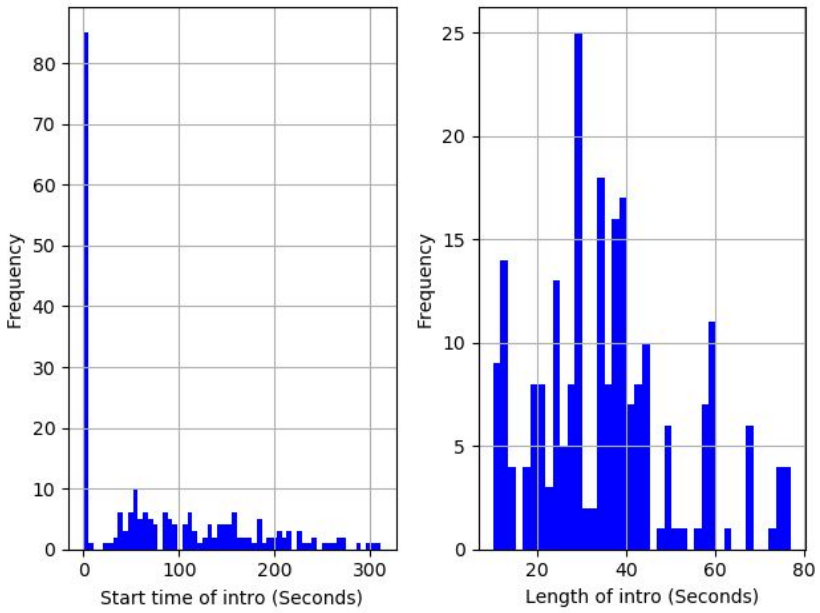

Figure 4.2: Illustrates the distribution of intro sequence start time and the intro lengths respectively.

Another observation made was that the variance of intro lengths within the same show was relatively low for 86% of seasons represented in the dataset, as illustrated in Figure 4.3. A variance of zero implies that the intro sequence is the same length across all episodes of a season. This means that if one episode has a known intro sequence length, other episodes from the same season have a high probability of having a similar length.

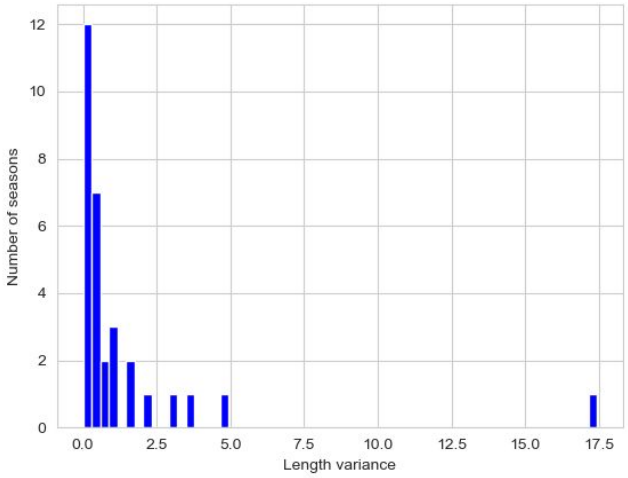

Figure 4.3: Presents the intro length variance of episodes within a season. Note that 4/35 seasons are not presented in the diagram due to having a variance greater than 20.

4.3 Creation of the Predictive model

This section presents the methods used for creating and evaluating the model which would be able to detect an introduction given a video from Svtplay. Section 4.3.1 describes how we collect videos and organize them within our system. In section 4.3.2, 4.3.3 and 4.3.4 the idea of extracting relevant features for introduction is described. These features were extracted for all the videos in the dataset 4.2. The chapter concludes with 4.3.5, which presents the method used for creating a classifier and evaluating it, based on the data extracted described in the previous sections.

4.3.1 Web scraping and download of videos

For constructing a model to be used for predicting the introduction of videos we had to create an internal overview within our system of the videos available at Svtplay. This system required the urls of all accessible videos within the subsection of TV series, as well as their episode number and season number. We used a library called beautifulsoup4, for crawling the website from the series section of Svtplay. Using the beautifulsoup4 library we could make queries to fetch the grid layout containing all their tv series with information such as the title of all the shows and each show base url. For each show we could continue crawling the website using the fetched url from the previous step to retrieve all the episodes with their specific urls, episode number and season number. All episode specific information would be stored within a document database called MongoDb along with flags to determine if the video has been downloaded or preprocessed.

For downloading the videos we used an open source command-line tool called svtplay-dl. The tool downloads the subtitles, audio, and video separately which required other tools for preprocessing the downloaded video.

4.3.2 Preprocessing of video

Two steps were conducted for converting the files downloaded from Svtplay into a manageable size and format for feature extraction. The first step was to combine the audiofile with the video file into one video, we chose mp4 as the format of the video. The next step was to remove all parts of the video after 8 minutes as we had identified it as a low possibility for an introduction to happen later based on the analysis discussed in section 4.1.1. This would save resources in terms of memory for storing the videos and the time to do further feature extraction and selection. We used a tool called Ffmpeg for this purpose.

The next step of preprocessing was to calculate and store the average hash value from every frame with an interval of 0.3 seconds in the video. The value of 0.3 seconds was chosen as it was a desirable compromise between time to compare the hashes in later stages of feature extraction in relation to predictive performance of the model.

The last step of preprocessing was to prepare a json file with a representation of the video splitted up in segments of 0.1 seconds. This json file would be used for loading and

storing relevant features and already known start and end for introductions which could at a later stage be extracted and used to construct a predictive model. The size of the segments was chosen as it divides the interval for the frames evenly while providing the framework needed to be able to store additional features with an error of margin below 0.1 seconds.

4.3.3 Feature extraction

In order for us to identify introductions from video we would have to reduce the dimensionality of the data we are able to retrieve, which apart from subtitles is the video itself. Similar to the works presented in section 2.3.1 [6], we identified matched sequences from video from the same series as features with high predictive qualities. We would also extract those sequences which would correlate with an introduction sequence, if such was present.

Other features such as sequences of black frames or sequences with the absence of subtitles or audio were also tried in different stages of development. They were not included in the final predictive model as they would lower the accuracy and increase the number of predictions given by the model. This was probably due to the predictive quality of matched frames being so high and the amount of data in our dataset. For example, sequences of black frames would lower the accuracy from 90,14% to 89.98% and output more than one prediction 50% of the time compared to 100% previously.

The algorithm devised for how we determine which frames is regarded as similar is defined as: Given two videos, represented by two sets of extracted frames A and B. For any given hash in A, compare it against all hashes in B until the hamming distance is below a chosen threshold for similarity. If a similar frame has been detected, mark that frame in A as a match. If the frame in B happens to be within an intro sequence in B save it as a match correlating with an intro. Once the comparison has been completed against all relevant videos, the list of matches are summarized into two lists. One for matches and one for matches correlating with intros. Each element in the list is identified by a time and the frequency of matches. This step is vital for the future process of feature selection.

The algorithm chosen for comparing frames between videos was the average hash algorithm as we would not expect distortion in rotation or color degradation between identical introductions from different videos. Average Hash is the hashing algorithm which performs best in regards to time complexity [20]. Since we would have to perform a maximum of 2,560,000 comparisons between two videos, the time to complete a comparison also would have been taken into account. We arrive at number 2,560,000 as the maximum total comparison $T = (L/i)^2$ where L is the length of a video in seconds and i is the interval for the frames to extract hashes from. This would happen in the instance where no matched frames would be detected.

For choosing which videos a given video should be compared against we identified that videos within the same season and within close proximity would yield the best matches. This was due to some series changing their introductions to a higher degree if they were further apart, in regards to episode and season number. The algorithm

devised for this part of the problem was to expand the search from the video itself to find episodes within the same season until the required amount of videos had been found. If the required amount could not be reached we would select episodes by the order of season and season number, lowest first. The restriction of number of videos to compare against is due to the increased risk of false matches, as well as increased time taken to complete this step. To determine the required amount of videos to compare against we would conduct a test seen in Figure 4.4 for 5-10 different videos.

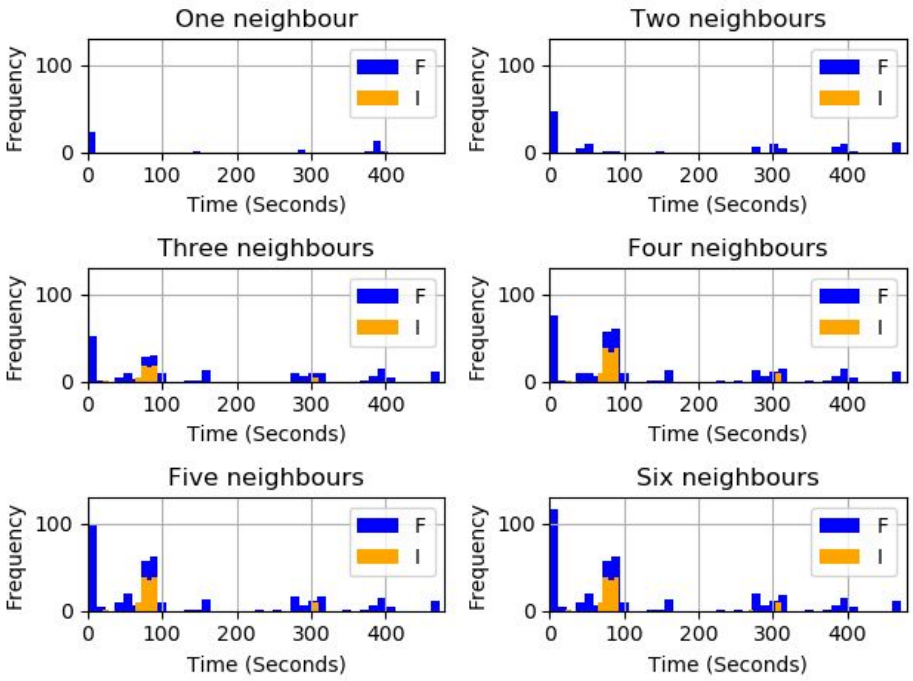

Figure 4.4: Example of how the amount of videos compared against affect the number of matches and how our final system of filtering out non likely sequences of matches being an introduction. F is denoted by the frames matched and I by the ones correlating with an introduction.

To detect similar images we had to choose a cut off value for what the maximum hamming distance score should be. A visualisation of such a test can be seen in figure 4.5, where the objective is to get as few false matches without losing accuracy.

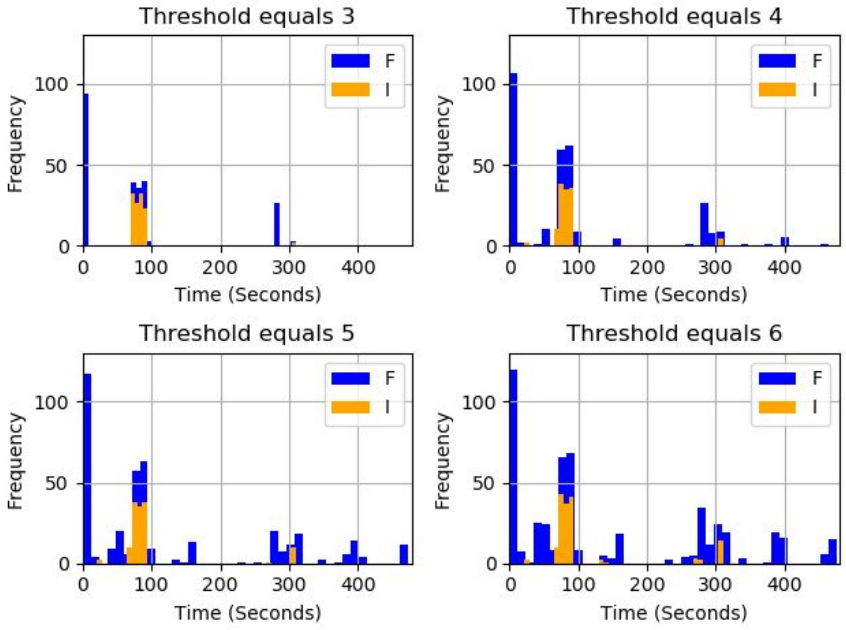

Figure 4.5: Different cutoff values for what is deemed as a match or no match. The number of videos compared against are 6 and F is denoted by the frames matched and I by the ones correlating with an introduction.

4.3.4 Feature selection

Since the number of input variables from the previous step was quite large and hard to visually determine which part would be a probable introduction we would need another step to remove false matches.

We propose a two step method for constructing a single sequence from a sequence of matched frames and matched frames correlating with introductions, seen in Figure 4.6, Figure 4.7, Figure 4.8.

The first step is to concatenate matched frames which are close enough to each other to be identified as being part of the same sequence. This was due to some matches not being registered due to the introductions not being synchronized between each other. If the total length of the sequence is below a specific threshold value it is removed. The result of this step can be seen in Figure 4.7. Each sequence is also labeled with a ranking, which is calculated by retrieving the mean value for the number of matches for all the frames in a given sequence multiplied by its total length.

The last step consists of choosing the sequence with the highest rank unless that sequence is determined to be a pre-intro, seen in Figure 4.8. A set of rules for when to not pick the highest ranked sequence is defined as:

1. The second ranked sequence is larger than 35% of the total score between itself and the highest scored sequence.
2. The second ranked sequence occurs after the highest ranked sequence.
3. The highest ranked sequence starts before 30 seconds of the video has elapsed.

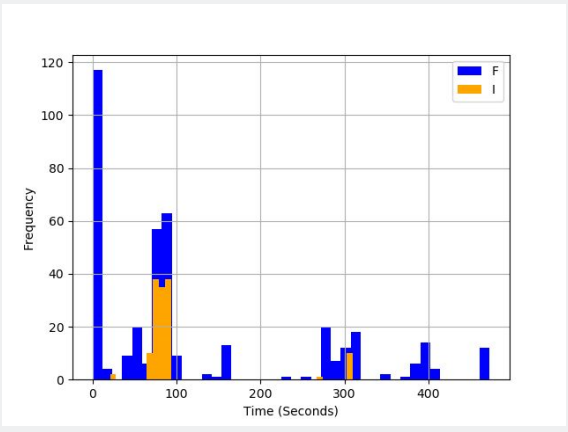

Figure 4.6: Example from a video and its frequency distribution for matched frames denoted by F and matched frames correlating with introduction denoted by I. The number of videos compared against are six and the cut off value for the hamming distance is less than five.

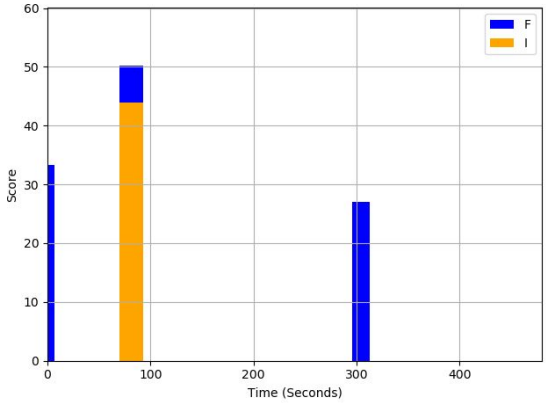

Figure 4.7: Example of how the filtering of frames is conducted from a set of matched frames denoted by F and matched frames correlating with introduction denoted by I. The sequence is valid if the distance between the frames is less than 4 seconds and the total length of the sequence is more than 5 seconds.

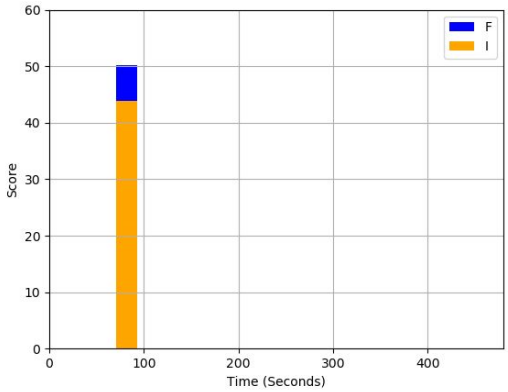

Figure 4.8: The final step of the feature selection process. The sequence with the highest score is chosen for both the matched frames denoted by F and matched frames correlating with an intro denoted by I. Potential pre-intro is disregarded.

4.3.5 Classification Predictive Model

Since the data used for making predictions is sequential and the state from any given part of a video can be classified as either being an intro sequence or not, we have chosen the Hidden Markov Model for classification. This idea was reinforced by analyzing the dataset and identifying a strong correlation between any given hidden state of a video and the next. Other viable options for this problem was also taken into consideration such as Support Vector Machines (SVM) or neural networks discussed in [21] . They were later disregarded due to the inherent time constraints of this thesis project.

We created the model using the pomegranate library. The parameters for the model were learned in two steps. First step was to create a dataset for training and evaluating the performance of the model. It was created by extracting all the sequences of the selected features from the previous step with a label representing hidden state, the presence of an introduction, for all the features in each video. Next we divided the dataset into 70% for training and 30% for evaluation.

The last step was to learn the transition probability distribution, emission probabilities and the initial probability distribution using the maximum likelihood estimation algorithm.

For making a prediction given a sequence of features or observations we used the viterbi algorithm which would convert the sequence of observations to a sequence of hidden states. Since this prediction could result in multiple sequences of being a potential intro, we would remove sequences below 3 seconds. The final step would be to choose the sequence with a length closest to the average length of an intro for that TV series or the average length of an intro for the whole dataset if the former was not present.

4.3.6 Method for Validating the Predictive Model

The performance evaluation of the predictive model was done following the method outlined in section 3.4.4. The choice was made to convert the regresionall problem of identifying float values for the start and end times of an intro sequence, into a classification type of problem. In this way a clearer visualization can be made on how well the model is able to correctly detect an intro sequence, without being overly dependent on correct annotations for start and end sequences in the dataset. The result was discretized by categorizing true positives as predictions with a margin of error of less than 20 seconds, when combining the start and end error of a prediction. True negatives were categorized as the model making a correct assumption of the absence of an intro sequence. False positives were categorized as false alarms, that is, the model found an intro sequence where none existed. False negatives occured when the model failed to find an existing intro. To make up for the inability of classification problems to

determine the accuracy of the model in terms of time, the start and end errors of true positive outcomes was used as a substitute.

Testing of the predictive model was done by first splitting the dataset into two parts, 70% for training and 30% for evaluating the trained model. This means that of 243 videos, 73 were used for testing purposes. Evaluation was then done by comparing the trained models predicted intro sequences, with the real sequences from the dataset. In order to reduce the variance of the result, tests were repeated 10 times, each time randomizing the elements in the dataset used for training and testing.

5 Results

This chapter provides a general description of the created product and evaluates the performance of the predictive model.

5.1 Product Specification

The final product was an automatic intro sequence detection system for TV series publicly available on SVT Play. The purpose of the system was to facilitate the possible future integration of an intro skipping button for SVT's streaming services. The core of the system was implemented as a python program that routinely scrapes *svtplay.se* for newly uploaded videos to process. Once a video has been processed, the intro prediction and other extracted features are saved inside a database.

Through a command line interface (CLI), users (or other processes) can perform actions such as; rebuilding the prediction model; starting the video processing schedule; requesting an intro prediction to be made; or setting a manual intro sequence, in case an incorrect prediction was made.

In addition to the command line interface, clients can perform video queries and basic system requests through a RESTful web service. By providing a web based interface to available system resources, the system is more easily integrated with other web services, such as SVT's content management system. The documentation for the REST-API is presented in Appendix A.

Interaction with the web service can be done through the client application which was created in python. The application has a graphical user interface similar to that of a media player and is made to allow users to analyze the difference between annotated and predicted intro sequences. In addition, it can be used manually annotate intro sequences or request a new prediction to be made. The graphical user interface of the media player is presented in Appendix B.

5.2 Performance Evaluation of the Prediction Model

The created model was tested using the method outlined in section 4.2.6. Two scenarios were tested. In the first scenario, the model performance was tested for unguided predictions, and in the second scenario, the model performance was tested for guided predictions. *Unguided predictions* are defined as a set of videos from the same show where no video has any previously manually annotated intro sequences. In such cases the prediction strategy is to choose the sequence which is closest to the average length of the dataset. *Guided predictions* on the other hand, utilize previously annotated intro sequences from the same show and picks the sequences which has the closest fit to the average of videos from the same season. The result for unguided and guided predictions are presented in section 5.3.1 and 5.3.2 respectively.

5.2.1 Unguided Predictions

As mentioned in 5.3, unguided predictions occur when the model has no prior knowledge of verified intro sequences that are related to the video being processed. The test results for unguided predictions are presented in Table 5.1

Table 5.2 presents the final results after analyzing the contingency table. The results show that the model could accurately predict the presence or absence of an intro sequence in 90.14% of cases for unguided predictions. Furthermore, the precision and recall rate for the model was 93.83% and 95.75% respectively, indicating that the model performs well in terms of its ability to correctly classify intros which are present. The true negative rate of 12.12% indicates that the model performs poorly when trying to identify the absence of an intro.

The error margin for predicted start and end times for cases of true positives are presented in Table 5.3. The table shows that the model was slightly better at predicting the start of an intro, than predicting the end of the intro. Start predictions had a mean margin of error of 1.4 seconds, with a standard deviation of 2.5 seconds. End predictions had a mean margin of error of 2.0 seconds, with a standard deviation of 3.1 seconds. The normal distribution of the prediction error is presented in Figure 5.1.

Table 5.1: Contingency matrix for observed unguided predictions.

	Condition Positive (CP)	Condition Negate (CN)
Test Outcome Positive (OP)	654	29
Test Outcome Negative (ON)	43	4

Table 5.2: Performance evaluation of the model when making unguided predictions

Metric	**Score**
True Positive Rate (Recall)	93.83%
Positive Predictive Value (Precision)	95.75%
True Negative Rate	12.12%
Accuracy	90.14%

Table 5.3: Margin of error of unguided predictions for all true positives (in seconds).

Metric	**Start**	**End**
Mean margin of error	1.4	2.0
Median margin of error	0.7	1.0
Standard deviation	2.5	3.1

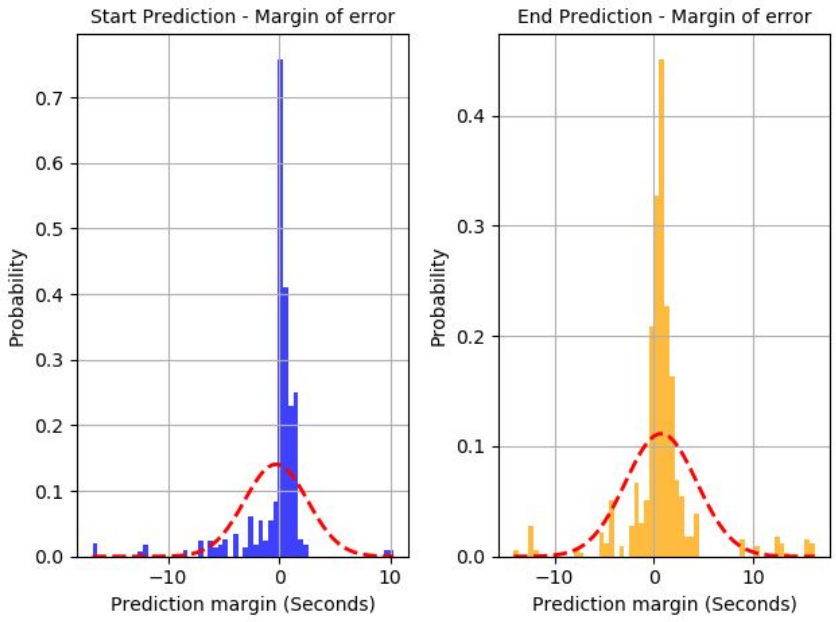

Figure 5.1: The figure illustrates the margin of error for start and end predictions for all true positives for all unguided predictions. The red line illustrates the normal distribution of the margin of error.

5.2.2 Guided Predictions

As mentioned in 5.3, guided predictions occur when the model has prior knowledge of one or more real intro sequences which are related to the video being processed. For this to occur, the system administrator has to define an intro for one or more episodes from the same show. The test results for guided predictions are presented in Table 5.4.

Table 5.5 presents the final results after analyzing the contingency table. The results show that the model could accurately predict the presence or absence of an intro sequence in 93.70% of cases, for guided predictions. Furthermore, the precision and recall rate for the model was 97.56% and 95.91% respectively, indicating that the model performs well in terms of its ability to correctly classify intros which are present. The models ability to classify the absence of an intro is indicated to be poor with a true negative rate of 19.05%.

The error margin for predicted start and end times for cases of true positives are presented in Table 5.6. The table shows that the model was slightly better at predicting the start of an intro, than predicting the end of the intro. Start predictions had a mean margin of error of 1.3 seconds, with a standard deviation of 2.5 seconds. End predictions had a mean margin of error of 1.9 seconds, with a standard deviation of 3.1 seconds. The normal distribution of the prediction error is presented in Figure 5.1.

Table 5.4: Contingency matrix for observed guided predictions.

	Condition Positive (CP)	Condition Negate (CN)
Test Outcome Positive (OP)	680	17
Test Outcome Negative (ON)	29	4

Table 5.5: Performance evaluation of the model when making guided predictions.

Metric	**Score**
True Positive Rate (Recall)	95.91%
Positive Predictive Value (Precision)	97.56%
True Negative Rate	19.05%
Accuracy	93.70%

Table 5.6: Margin of error of guided predictions for all true positives (in seconds).

Metric	**Start**	**End**
Mean margin of error	1.3	1.9
Median margin of error	0.6	0.9
Standard deviation	2.5	3.1

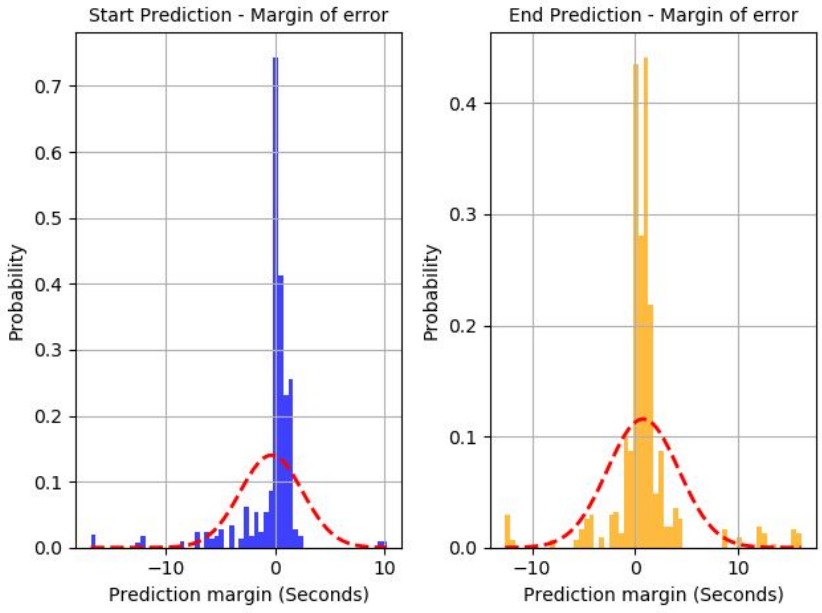

Figure 5.2: The figure illustrates the margin of error for start and end predictions for all true positives for guided predictions. The red line is the normal distribution of the margin of error.

6 Analysis and Discussion

This chapter analyzes and discusses the results presented in chapter 5 in regards to the objectives defined in section 1.2 and the methodology presented in chapter 4.

6.1 Interpreting the Dataset

The dataset which was created for generating a predictive model was small and favored towards having videos from shows which mostly had intro sequences. This required us to create the same type of models multiple times to reduce the variance of the result. Which in itself led to the same data being evaluated multiple times. Therefore some key metrics from the result such as true negatives and false negatives are hard to evaluate with a statistical confidence since they are sparse in our initial dataset. It is probable that the cause for this is due to our biases for selecting intro sequences with a clear, identifiable intro sequence as outlined in the initial delimitations. But it is also likely that most TV series have an intro sequence. Therefore it is probable that our dataset which is 18.7% the size of all available videos from SVT Play is a good representation of reality.

6.2 Analysing the Model Performance

Since true negatives and false negatives doesn't yield any statistical significance due to the small amount of negatives in our dataset, the most interesting metrics to analyze for the predictive performance of the model is thus the mean error of margin and precision.

Section 2.3.2 presents a paper which discusses a predictive model based on blank screen transitions, shot boundaries, silence gaps and histograms for identifying introductions of a given video. They managed to achieve precision of 82% and a mean error of margin of 1.42 seconds for both start and end predictions. By comparing this result to our best model for prediction with a precision of 97.5% and a mean error of margin of 1.3 seconds and 1.9 seconds for start and end we can conclude that the positive predictive performance of the model is quite good.

In Figure 5.1 and 5.2, it is shown that most of the errors are likely to happen close to the true value of start and end. This can also be confirmed from the median value of 0.7 seconds and 1.0 seconds seen in Table 5.2 as well as 0.6 seconds and 0.9 seconds seen in table 5.4. The error of margin is most likely due to imprecise labels for when an introduction starts and ends. A simple adjustment for improving this result would be to have more precise labels present of introductions in our dataset. But we made a choice early in the project that the quantity of data was more important compared to precise annotations due to the manual labor required and the time restrictions we had.

A misprediction in terms a prediction estimating the end to be later than the true value or the start occuring sooner than the true value for start would happen when falsely matched frames would be appended to the final sequence of matches later to be used by the markov model for getting a prediction. This problem could be negated by

fine tuning the threshold values for concatenating matched frames into sequences, number of videos to compare against and the cut off value for what is viewed as a similar frame when comparing the average hashes. Other algorithms for image comparison such as Oriented Fast and Rotated Brief or perceptual hashing might also perform better than average hashing in regards to false matches. The reason why they were not implemented is due to the increased time it would take for completing the prediction of a video after download.

A general misprediction or not a prediction at all would happen for introductions which were not identical between multiple videos in the same tv series. This could potentially be solved by identifying similar tracks of audio amongst videos in the same tv series or by combining this model with other predictive models based on other features.

Both the unguided and guided predictions will not perform well in instances where there are no other videos to compare against. We determined that this was an acceptable exception due to SVT requiring a tool for manual confirmation and adjustment of a predicted introduction. These exceptions would arise when the first episode of a season with a new introduction would be uploaded or when there only existed one episode for one tv series and thus not very likely to happen often.

6.3 Implications of False Predictions

The implications of a high margin of error or a falsely classified intro would be a less enjoyable experience for the viewer. It would result in the viewer skipping over parts of the video which is related to the story and not the introduction. This would reduce the usability of the predictive model and require more manual labor to verify that each predicted introduction is valid. Thus, the system would not be serving its purpose and can is unviable for partial or complete automation usage.

There is an indication of the model falsely classifying introductions for videos lacking one with a true negative rate of 12.12% and 19.05% for both unguided and guided predictions. It would seem likely that this would be the case for videos which has identical sequences classified as pre intro or teaser with a length of minimum 5 seconds. But due to only 4% of the dataset containing videos without an introduction it is undetermined how often this would happen and therefore it is likely given the precision of the model, it would be suitable for automation.

6.4 Extent of Work Completed

The developed system meets all the outlined objectives as specified in section 1.2, and can be considered feature complete. The system is capable of automatically processing and making predictions on new videos as they are uploaded to SVT Play. A graphical user interface was implemented so that administrators can query the system to perform new intro prediction, or annotate the real intro sequence, in case of invalid predictions. Additionally, a REST-API was created to provide access points to system resources, and facilitating possible future integration with other web services, such as SVT's content management system.

6.4.1 Partially Complete Tasks

There are some aspects of the system that were not fully realized and would need to be addressed before the system can be considered ready for release. These issues are primarily related to improving the reliability of the system and adding additional features to the user interface.

System Reliability

Database integration was not fully realized, which means that after the video has been processed, extracted features are saved as files. This poses problems both in terms of scalability and concurrency. The scalability problem is caused by the fact that the system keeps accumulating files for each video being processed. This would be acceptable if the files were removed once a video was no longer publicly available on SVT Play. The concurrency problem can cause prediction requests to fail, or lead to the scheduled batch process to crash. This happens when the system tries to simultaneously compare two or more videos from the same show. This problem is caused by one process attempting to open a file which is already being used by another file.

User Interface

The current implementation of the graphical user interface, although complete, is very cumbersome to use. This stems from the fact that videos must first be downloaded before they can be analyzed. Additionally, there is no way to analyze videos present in the filesystem. Another problem which limits the administrative efficiency, is that there is no user interface for searching or manipulating multiple videos simultaneously, despite these features being publicly available through the REST-API.

6.4.2 Incomplete Task

The task to integrate the created system with SVT's content management system was never realized. To achieve this objective, a web form would need to be created so that video files could be processed before they are released on SVT Play. Without such a feature, there can be a significant delay between the release date and the time that a prediction is available or an annotation can be made.

6.5 Alternative Approaches

This section presents the alternative approaches which were considered.

6.5.1 Leveraging Wisdom of the Crowd

Wisdom of the crowd is based on the assumption that the collective intelligence derived from the responses of multiple humans can be used to improve decision making and improve prediction accuracy. This assumption can also be integrated into machine learning programs [22].

The current prediction model is not capable of accurately predicting the intro sequence for all cases. To correct prediction flaws in the current implementation, an administrator would need to manually annotate a different intro sequence from the predicted sequence, which negatively impacts the automation of the system.

By leveraging the gathered intelligence from audiences, machine learning could be used to make the system self-correct. Crowd intelligence could be gathered from viewers either directly, by flagging invalid predictions or letting viewers publish correct sequences, or indirectly, by logging video skipping behaviour.

6.5.2 Audio Similarity Comparison

Since most intro sequences from the same show or seasons share the same soundtrack across all episodes, a solution based on audio similarity across related videos would likely work well. For finding these sections of audio correlating with introduction one could implement or use existing tools for identifying music from sequences of audio. By using a sliding window technique to analyse parts of the video separately one could identify the specific timestamps for an introduction.

6.6 Non-Technical Considerations

This section discusses the economical, social and environmental considerations relevant to the proposed product.

6.6.1 Economical Considerations

The cost of introducing skipping capabilities to a media-service platform, would come with an initial development cost, and operating costs, in terms of deployment and maintenance. Additionally, because the system is only semi-automatic, in the sense that it is not capable of eliminating all prediction errors, some sort of administration is also required for verification. The administrative cost of verification is dependent on the accuracy of the predictive model, and to what extent the user interface used by administrators is capable of providing an overview of the prediction results. One way to reduce the administrative cost of the system, would be to manually annotate the intro sequence for one episode of each show, both improving the accuracy of the prediction, but also allowing a check to be made on the predicted intro length of subsequent

predictions. Administrators would only need to check on predictions whose intro length deviates from the annotated intro length. The need for administrative control could also be reduced if a crowd intelligence solution was used to self-correct the system, as discussed in 6.5.1.

Determining the exact business value of implementing a new feature which only aims to improve audience satisfaction is hard, if not impossible, even if surveys show that the feature is seen as a positive addition to the user experience [2]. However, it can be argued that the risk of losing audience members to competitors who offer similar features outweigh the cost associated with deploying such a system.

6.6.2 Environmental Considerations

According to projections made by Cisco [25]: 60% of the world's population will be online by the year 2022, with IP video traffic making up 80% of all internet traffic. This means that a significant portion of the energy consumption needed to run the internet, is allocated to the distribution of online videos.

The possibility of detecting redundant video content in general, and skippable film sequences in particular, could be utilized by media-streaming service providers to reduce the amount of video traffic transmitted to the end user. Thus, reducing the amount of energy necessary to watch streamed videos. As an example, the strictest application of this philosophy would be to always skip intro and outro sequences, completely eliminating the need to transmit IP video packets for those parts of the video. Another possible implementation would be to only transmit part of the intro or outro, and wait to determine if the viewer skipped the part in question, before transmitting the remaining IP video packets.

6.6.3 Ethical Considerations

Watching multiple episodes of a show in a single sitting is associated with a range of negative health effects, as is the case with any sedentary behaviour [23]. These negative health effects include increased fatigue, poor sleep quality, insomnia symptoms and poor dietary choices [24].

One can argue that by improving the service of watching TV series by giving people the option to skip parts of the video which are uninteresting is an encouragement to a more sedentary behaviour. You can also argue that these additional features might cause people to watch less television by only letting them watch parts they are interested in. Regardless of the position taken, one must question who has the end responsibility over the negative personal and societal effects associated with sedentary behaviour, the consumer or the producer? We believe that the consumer holds this responsibility as it is the one which controls its own actions. By providing the consumer more control over what to spend their time on, by providing the ability to skip parts unrelated to the story and giving the user more options should have a positive effect on society.

7 Conclusions

The project resulted in an semi-automated intro detection system for videos available at SVT Play. The front-end is an administrative tool that enables users to view, analyze and modify predicted intro sequences. The tool is also capable of requesting new intro predictions, in case the video was not yet processed. The back-end consists of a database; command line interface, used to process videos; and a web service which allows for the system to be integrated by other web services. The created system is fully functional but would require additional updates to address fault tolerance and performance concerns. The intro prediction system could help media-streaming providers to add intro skipping capabilities to their services, which in turn may improve audience satisfaction and loyalty.

The proposed solution, based on statistical classification model, which analyzes similar frames across related videos, proved to have a high degree of accuracy for most tested shows. However, it is possible that the model will perform poorly when trying to identify the absence of an intro, or other edge cases such as if the intro has a high degree of variability compared to other intros from the same show. The extent of these errors cannot be determined due to limitations with the dataset.

7.1 Proposal for Future Improvements

Beyond addressing reliability, performance and user interface issues presented in 6.3 the next logical step would be to do a trial run, where the system is integrated with services facing end users.

System Improvements

1. Implement audio similarity extraction to try and improve the performance of the model.
2. Improve the performance by implementing threads when processing videos. Expand the capabilities of the system to include other skippable video sequences, such as the pre-intro, previous and outro sequences.
3. Make it possible for users to report invalid intro predictions and implement some sort of correction mechanism based on crowd intelligence.
4. Expand the dataset and try different classifiers or a combination of classifiers to improve the performance of the model.

References

[1] P. Davidsson, M. Palm, Å. M. Mandre. *“Svenskarna och internet 2018”* [internet]. Sweden: Internetstiftelsen. [Accessed: January]. URL: https://svenskarnaochinternet.se/rapporter/svenskarna-och-internet-2018/

[2] *“Unsurprising: Netflix Survey Indicates People Like To Binge-Watch TV”* [Internet]. 2013, 13 december. [Accessed: January 2020]. URL: https://www.cinemablend.com/television/Unsurprising-Netflix-Survey-Indicates-People-Like-Binge-Watch-TV-61045.html

[3] Newton. C. *“Netflix is Testing a Button for Skipping the Opening Credits”*. New York: Vox Media; 2017. [Accessed January 2020]. URL: https://www.theverge.com/2017/3/17/14959650/netflix-skip-intro-button

[4] Inceer, M., *"An Analysis of the Opening Credit Sequence in Film"*. 30 May 2007: College Undergraduate Research, University of Pennsylvania

[5] A. Kansara, *“Extracting contextual information from video assets”* [internet blog]. Netflix Technology Blog; April 6, 2015. [Accessed January] URL: https://netflixtechblog.com/extracting-contextual-information-from-video-assets-ee9da25b6008

[6] M. Nematollahi, X. Zhang, *“Automatic Video Intro and Outro Detection on Internet Television”* [Internet] 2014. [Accessed January 2020] URL: “Automated intro and outro detection on internet television”

[7] N. V. Kamanwar and S. G. Kale, "Web data extraction techniques: A review," *2016 World Conference on Futuristic Trends in Research and Innovation for Social Welfare (Startup Conclave)*, Coimbatore, 2016, pp. 1-5. [Accessed January 2020] URL: https://ieeexplore-ieee-org.focus.lib.kth.se/document/7583910

[8] S. Khalid, T. Khalil and S. Nasreen, "A survey of feature selection and feature extraction techniques in machine learning," 2014 Science and Information Conference, London, 2014, pp. 372-378. [Accessed January 2020] URL: https://ieeexplore-ieee-org.focus.lib.kth.se/document/6918213

[9] Karami E, Prasad S, Shehata M. *"Image Matching Using SIFT, SURF, BRIEF and ORB: Performance Comparison for Distorted Images"* Canada: Faculty of Engineering and Applied Sciences, Memorial University.

[10] Zingade A. *"Image Similarity Using Deep Ranking"*. 2017, 8 December. [Accessed 5 January 2020]. [Blog on the Internet]; 27 oktober 2017. URL: https://medium.com/@akarshzingade/image-similarity-using-deep-ranking-c1bd83855978

[11] Krawetz N. "Looks like it" 2011, 26 May. In: Hacker Factor [Blog on the internet]. Internet: Hacker factor. [Accessed 5 January 2020] URL: http://www.hackerfactor.com/blog/index.php?/archives/432-Looks-Like-It.html

[12] Halton C. *"Predictive Modeling"* [internet]. New York: Investopedia; 2020. [Accessed 5 January 2020]. URL: https://www.investopedia.com/terms/p/predictive-modeling.asp

[13] Malato G. "How to measure the goodness of a regression model" . 2020, 16 April. In: Towards data science [Internet]. Towards data science; 2016 [Accessed 5 January] URL:

https://towardsdatascience.com/how-to-measure-the-goodness-of-a-regression-model-60c7f87614ce

[14] Torgo L, Gama J. Regression by Classification. Porto: LIACC - University of Porto.

[15] Olson D.L., Delen D. (2008) Performance Evaluation for Predictive Modeling. In: Advanced Data Mining Techniques. Online ISBN: 978-3-540-76917-0, Springer, Berlin, Heidelberg; 2008. Chapter 9 - Performance Evaluation for Predictive Modeling.

[16] Jurafsky D, Martin J. H, Hidden Markov Models [internet] Stanford University, 2019. [Accessed 5 January] URL: https://web.stanford.edu/~jurafsky/slp3/A.pdf

[17] T. dunning. Hidden Markov Model [picture]. 2012 [Accessed 5 January] URL:https://commons.wikimedia.org/w/index.php?curid=18125206

[18] Giannakopoulos T., Pikrakis A, Introduction to Audio Analysis: A MATLAB Approach. Academic Press. 2014. Chapter 7 - Audio Alignment and Temporal Modeling; p. 185 - 207.

[19] Ebert T, Hidden Markov Models, California State University. [Accessed: January]. URL: http://web.csulb.edu/~tebert/teaching/lectures/552/hmm/hmm.pdf

[20] Content Blockchain. “Testing Different Image Hash Functions”. [Internet].[Accessed 5 january]. URL: https://content-blockchain.org/research/testing-different-image-hash-functions/

[21] Xing, Zheng zheng & Pei, Jian & Keogh, Eamonn. (2010). A Brief Survey on Sequence Classification. SIGKDD Explorations. 12. 40-48. 10.1145/1882471.1882478.

[22] Wang L., Michoel T. Wisdom of the crowd from unsupervised dimension reduction. The University of Edinburgh, The Roslin Institute [Cited January]. URL: https://arxiv.org/abs/1711.11034

[23] Walton-Pattison E, Dombrowski SU, Presseau J. “‘Just one more episode’: Frequency and theoretical correlates of television binge watching”. J Health Psychol. 2018; 23(1):17-24

[24] Birch J. “How binge-watching is hazardous to your health”. The Washington Post [Internet]. 2019, 3 June. [Accessed January]. URL: https://www.washingtonpost.com/lifestyle/wellness/how-binge-watching-is-hazardous-to-your-health/2019/05/31/03b0d70a-8220-11e9-bce7-40b4105f7ca0_story.html?fbclid=IwAR3EjHUkmJ-KPIu6bHMplYiA6lYdbFdYz7xItTZKhOd7Vqb4OzbcPw8QsYg

[25] Cisco. *“Cisco Visual Networking Index: Forecast and Trends, 2017–2022”* [Internet], 27 February 2019, [Accessed: January]. URL: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html

A Appendix - REST-API

This section presents the REST-API and how to access and query the available endpoints. The services included are: Querying the video repository (Table C.1), setting intro annotations (Table C.2), requesting a new intro sequence prediction (Table C.3) and querying the current status of a previously requested service process (Table C.4).

Querying The Video Repository

Returns a collection of videos and related information such as show, title, season, episode, and any annotated or predicted intro sequences.

Table B.1: Illustrates the http-request, available query parameter and an example response for accessing the video repository.

HTTP Request	GET: */videos*
Query Parameters	&url=<string> # Queries by video URL. &show=<string> # Queries by show name. &show_id=<string> # Queries by show id. &title=<string> # Queries by video title. &season=<int> # Queries by season. &episode=<int> # Queries by episode. &prediction=<bool> # Queries for existence of prediction. &annotation=<bool> # Queries for Existence of annotation. &limit=<int> # Operator: limits query result. &page=<int> # Operator: return a different result page.

HTTP Response JSON	[{
	"downloaded": true,
	"introAnnotation": {
	"end": 222.0,
	"start": 209.5
	},
	"introPrediction":{
	"end": 220.0,
	"start": 211.0
	},
	"preprocessed": true,
	"season": 1,
	"episode": 2,
	"show": "exit",
	"showId": "exit",
	"title": "Avsnitt 2",
	"url": "https://www.svtplay.se/video/24136954/exit/exit-avsnitt-2"
	}]

Set Intro Annotation

Allows for manual annotations of intro sequences denoted in seconds. In order to work, either the video-url or a show identifier must be specified in the query arguments. Be aware that querying by show will set the intro sequence for all videos related to the specified show.

HTTP Request	POST*: /videos/set/intro*
Query Parameters	&url=<string> # Queries by video URL. &show=<string> # Queries by show name. &show_id=<string> # Queries by show id. &title=<string> # Queries by video title. &season=<int> # Queries by season. &episode=<int> # Queries by episode.

Body	{
	"start": <float>,
	"end": <float>
	}

HTTP Response JSON	[{
	"downloaded": true,
	"introAnnotation": {
	"end": 222.0,
	"start": 209.5
	},
	"introPrediction":{
	"end": 220.0,
	"start": 211.0
	},
	"preprocessed": true,
	"season": 1,
	"episode": 2,
	"show": "exit",
	"showId": "exit",
	"title": "Avsnitt 2",
	"url":
	"https://www.svtplay.se/video/24136954/exit/exit-avsnitt-2"
	}]

Table B.2: Illustrates the http-request, available query parameter and request arguments and an example response for requesting an intro annotation change

Request Intro Sequence Prediction

Sends a request to the server to perform an intro sequence prediction on the specified video and returns a service process instance which can be polled regularly to check the status of the process (See Table B.3).

Table B.3: Illustrates the http-request, available query parameter and an example response for requesting to start a new server service process.

HTTP Request	GET: */services/request/predict-intro*
Query Parameters	&url=<string> # Queries by video URL.

HTTP Response JSON	{
	"_id": "5e1a8d0dd1b9dc1008af6731",
	"process": "predict-intro",
	"args": [
	"https://www.svtplay.se/video/23536930/allt-jag-inte-minns/allt-jag-inte-minns-sasong-1-avsnitt-1"
	],
	"started": null,
	"result": null,
	"status": "pending",
	"executionTime": 0,
	"startDelay": 0
	}

Query Service Processes

Returns a collection of service processes matching the query argument. There are four different states: pending, working, finished and halted. Calculating the intro sequence may take a few seconds or a couple of minutes depending on how many related videos have already been preprocessed.

HTTP Request	GET: */services*
Query Parameters	&id=<string> # Queries by request id

HTTP Response JSON	[{
	"_id": "5e1a8c9b9e7286458c704010",
	"process": "predict-intro",
	"args": [
	"https://www.svtplay.se/video/23536930/allt-jag-inte-minns/allt-jag-inte-minns-sasong-1-avsnitt-1"
	],
	"started": "2020-01-12T03:03:55.694Z",
	"result": {
	"prediction": {
	"start": 40.2,
	"end": 116.7
	}
	},
	"status": "finished",
	"executionTime": 6.333,
	"startDelay": 0.033
	}]

Table B.4: Illustrates the http-request, available query parameter and an example response for requesting the current status of server service processes.

B Appendix - Graphical User Interface

This section presents the graphical user interface. It is an editorial tool for retrieving predictions of introductions in a video from SVT Play as well as providing the user with the ability to manually adjust and save a new introduction for a video. It stores and retrieves data from the database and the predictive model via asynchronous calls to the REST-API. When starting the application, a window shown in Figure C.1 is displayed to the user.

Start:

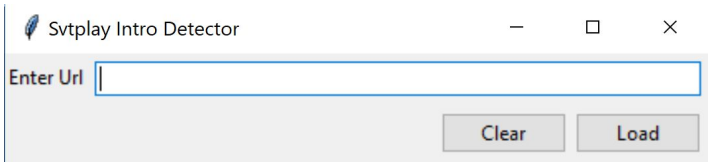

Figure C.1: The initial view of the application upon start. A textfield for submitting a url for download and prediction.

After the user submits a URL for a video present in Svtplay, a call to the REST-API service is made for retrieving a prediction and an introduction if such is present as well as downloading the video using the svtplay-dl library. To keep the app responsive and communicating to the user that an action has been triggered two fields for loading is displayed, shown in Figure C.2. One field represents the response from the REST-API and the other the process of loading the video later to be displayed. After the download has completed, the video and audio file is combined and cut down to 8 minutes using the ffmpeg tool.

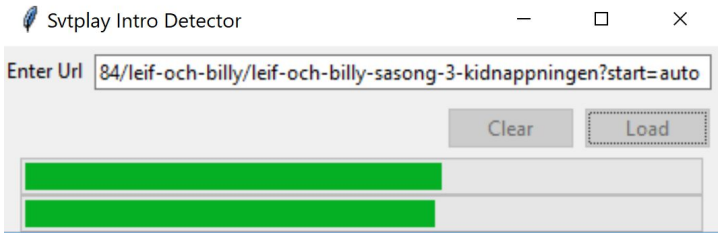

Figure C.2: The process of downloading and retrieving the prediction from a url submitted previously.

Once the prediction, potential introduction and the video has been downloaded the user gets presented with the editorial tool shown in Figure C.3. It displays the predicted intro as two thin vertical lines with a number above representing the estimated start and end in seconds. Two movable markers, green for start and red for end of the chosen intro gets moved to the already confirmed intro if such is available or the predicted intro stated previously. When a marker is manually moved, either by using the mouse or by

entering a floating value in seconds in the related textfield, the screen showing the video updates the frame shown to match the selected time for the adjusted marker. There is built in validation which prohibits the user from moving the markers over each other. The markers can also be moved from its current position by either pressing the buttons Play Start or Play End which would be changed to Pause Start or Start End dependent on which has been chosen. Only one marker can be moved and played from at any given time. Once a play button has been pressed the audio extracted previously is played from the starting position of the marker, the frames shown gets updated and synchronized in real time, 24 frames per second which matches the framerate of the video at svtplay. Once the user is pleased with the start and end time for the introduction, the user can press save. This triggers a REST-API call for saving the introduction for a video with a url submitted previously.

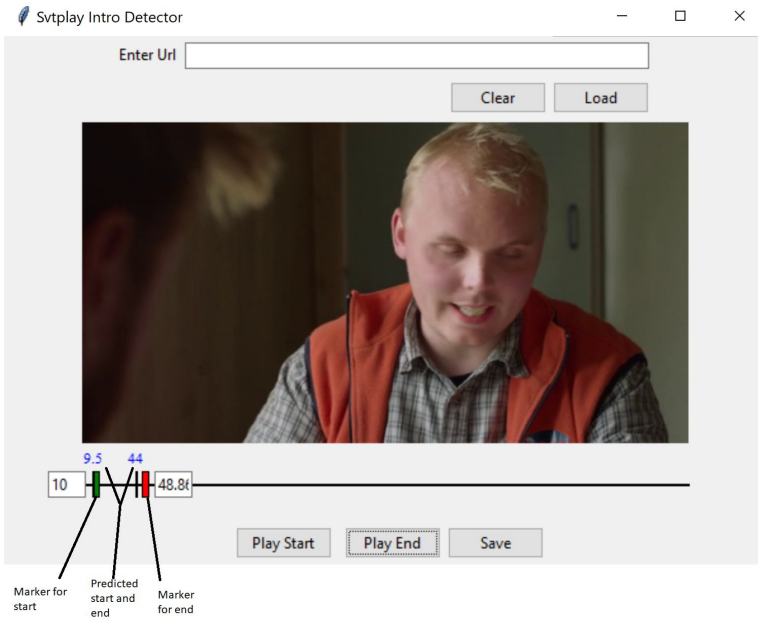

Figure C.3: Editorial view of the application. Displays a player which can show video in real time as well as adjustable markers for selecting and confirming the introduction of a chosen video.

C Appendix - Project Dependencies

This section provides an overview of the dependencies used in the final project and states their corresponding licensing agreement.

Core Application Dependencies

Name	License	Source
numpy	BSD	https://pypi.org/project/numpy
imageio-ffmpeg	BSD	https://pypi.org/project/imageio-ffmpeg/
beautifulsoup4	MIT	https://pypi.org/project/BeautifulSoup/
lxml	BSD	https://pypi.org/project/lxml/
scendetect	MIT / BSD 3-Clause	https://pypi.org/project/scenedetect/
opencv-python	MIT	https://pypi.org/project/opencv-python/
Click	BSD	https://pypi.org/project/click/
tqdm	MIT / MPL 2.0	https://pypi.org/project/tqdm/
pathlib	MIT	https://pypi.org/project/pathlib/
scikit-image	BSD	https://pypi.org/project/scikit-image/
ImageHash	MIT	https://pypi.org/project/imagehash3/
praat-parselmouth	GPLv3+ / GPLv3	https://pypi.org/project/praat-parselmouth/
matplotlib	Python Software Foundation	https://pypi.org/project/matplotlib/
moviepy	MIT	https://pypi.org/project/moviepy/
pymongo	Apache-2.0	https://pypi.org/project/pymongo/
pomegranate	none	https://pypi.org/project/pomegranate/
seaborn	BSD 3-Clause	https://pypi.org/project/seaborn/
schedule	MIT	https://pypi.org/project/schedule/

Table D.1: List of python packages used in the core application and their corresponding licenses.

GUI Application Dependencies

.

Name	License	Source
pygame	LGPL	https://pypi.org/project/pygame/
requests	Apache-2.0	https://pypi.org/project/requests/

Table D.2: List of python packages used in the GUI application (which were not listed in Table B.1) and their corresponding licenses

Node Application Dependencies

Name	License	Source
cookie-parser	MIT	https://www.npmjs.com/package/cookie-parser
debug	MIT	https://www.npmjs.com/package/debug
express	MIT	https://www.npmjs.com/package/express
http-errors	MIT	https://www.npmjs.com/package/http-errors
mongodb	Apache-2.0	https://www.npmjs.com/package/mongodb
morgan	MIT	https://www.npmjs.com/package/morgan
pug	MIT	https://www.npmjs.com/package/pug
python-shell	MIT	https://www.npmjs.com/package/python-shell
semaphore	none	https://www.npmjs.com/package/semaphore

Table D.3: List of node.js packages and the corresponding licenses for the created web service.

D Appendix - Source Code

The source code, along with additional documentation for this thesis project can be found in the following Git-Repository:

https://github.com/JacobEkedahl/detect-intros-from-video

TRITA CBH-GRU-2020:005