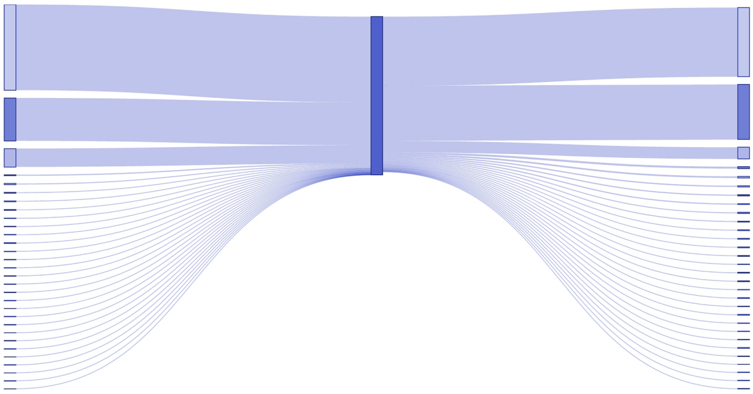
University of Zurich UZH

A Flexible Viewership Analytics System for Online TV



Master Thesis Sept. 23, 2015

Basil Philipp
of Meilen ZH, Switzerland

Student-ID: 09-721-101
basil.philipp@uzh.ch

Advisor:

Prof. Abraham Bernstein, PhD
Institut für Informatik
Universität Zürich
http://www.ifi.uzh.ch/ddis

Acknowledgements

I would like to express my gratitude to Prof. Abraham Bernstein, PhD for giving me the opportunity to write my master thesis at the DDIS group, and also for his valuable input, time and support.

I would also like to thank my parents for allowing me to realise my own potential; without their support I wouldn't be where I am today.

Last but not least, I am grateful to Sofia and my friends for all the advice and encouragement that they gave me.

Zusammenfassung

Anbieter von online TV verwenden Technologien, die es ermöglichen das Zuschauerverhalten in grösserem Umfang aufzuzeichnen als dies mit bisherigen, stichprobenbasierten Methoden möglich ist. Der fragmentierte Markt und die grosse Datenmenge setzen neue Ansätze voraus, um diese Daten in Wissen umzuwandeln. Wir entwickeln ein System, welches mit mehreren Datenquellen umgehen kann, und den Benutzern ermöglicht komplexe Analysen durchzuführen. Wir demonstrieren die Fähigkeiten des Systems, indem wir eine beispielhafte Marktanalyse und eine Zuschauerflussanalyse durchführen. Des Weiteren präsentieren wir ein Vorhersagemodel für die Anzahl Zuschauer einer Sendung.

Abstract

The technologies used by online television providers make it possible to collect significantly more information on viewer behaviour than is possible with traditional, panel-based measurements. The fragmented market and the large data size call for novel approaches to handle this data and turn it into valuable insights. We propose a system that can deal with multiple data sources and offers advanced analyses of the data. We demonstrate the capabilities by showing an exemplary market analysis, an audience flow analysis and a viewership prediction.

Table of Contents

1 Introduction **1**

2 Related Work **3**

3 Market **5**

3.1 Data Providers 5

3.2 Data Consumers 6

4 Requirements **7**

5 Data **9**

5.1 Session Data 9

5.2 User Data 9

5.3 EPG Data 9

5.4 Data Format 11

6 System Architecture **13**

6.1 Data Importer 13

6.1.1 Importing Users, Sessions and EPG 14

6.1.2 Transformations 14

6.1.3 Import Performance 16

6.2 Database 17

6.3 Analytics Layer 17

7 Market Analysis **19**

7.1 Data 19

7.2 Analyses 19

7.2.1 Market Share 20

7.2.2 Ratings 20

7.3 Demographics 22

7.4 Differences to Panel-based System 25

8 Audience Flow Analysis **29**

8.1 Computation 29

8.1.1 Algorithm 31

8.2 Stability Analysis . . . 32
8.2.1 Data . . . 32
8.2.2 Results . . . 33

9 Viewership Prediction 35
9.1 Data . . . 35
9.2 Features . . . 36
9.3 Metrics . . . 37
9.4 Models . . . 38
9.4.1 One vs. Multiple Models . . . 38
9.5 Results . . . 39

10 Conclusions and Future Work 41

1

Introduction

Live video content is increasingly consumed over the Internet. The television set in the living room is being replaced by multiple, internet capable devices like smartphones, tablets and computers. Contrary to a television, those devices are often tied to one individual person and the content is not delivered over a cable network, but over the Internet. Those changes in viewing behaviour and technology make it possible to discover what people are watching to a much greater detail than was possible before.

Traditionally TV shares are calculated by extrapolating the viewing behaviour of a small representative sample. By its very nature this leads to one person standing in for thousands of other viewers, which represents a huge loss in data granularity. Since in online TV (OTV) each device sends a request for getting the desired content, evaluating these request logs makes it possible to analyse the viewing behaviour of each viewer. Having a data point for every interaction between a viewer and her OTV content offers the possibility to provide highly accurate market research information about viewing behaviour. This information can be used for a variety of analyses of high interest to all participants in the TV-industry and is of a density that is not attainable with panel-based systems.

This influx of data which is at least three magnitudes larger than panel-based data requires different technical solutions. Whereas panel-based data is usually sparse enough that it can be processed on a standard notebook computer, OTV data belongs to the world of "Big Data" and requires more sophisticated approaches. Due to its higher resolution the volume of the data is much larger and arrives at a much higher velocity than panel-based data.

Not only the amount of data is larger, but also the number of sources. Usually in panel-based settings data from only one panel is used. This ensures that the data format is consistent amongst data points and the quality of all the data points is the same. Since OTV is streamed by multiple providers in Switzerland, using different technologies, the data generated by the request logs is much more heterogeneous. The proposed system thus needs to be able to deal with multiple sources.

This master thesis lays the groundwork for a TV analysis platform for the digital world and shows the results of the first analyses run on the platform. Our hope is that such a platform will allow us to extract insights from the data that lead to better targeted programs and advertisements, which will in turn provide an improved viewing experience for the viewers.

This master thesis is structured as follows. Chapter 2 shows how this topic fits in with traditional TV analysis and gives an overview over the related technological literature. In chapter 3 we provide an overview of the Swiss OTV market and in chapter 4 we show what the requirements for an analytics system in this market are. In chapter 5 we explain the input format of the data used. The technological make up of the system is detailed in chapter 6. In chapter 7 we present the results of an exemplary market analysis of OTV data. In chapter 8 we look at how audience flows can be computed and how stable they are over time. Our final analysis of the data is presented in chapter 9, where we show how our system can be used to forecast viewership numbers. We finish by presenting our conclusions in chapter 10 and by giving an outlook on future work.

2

Related Work

Trying to predict the audience flow and the number of viewers for a program is not a problem specific to OTV. The first research in this topic was already conducted in the 1980s on first screen TV. Rust and Alpert (1984) present a model for predicting audience flow and the total number of viewers for a show. Their model works by segmenting the viewers by age, gender and education and assigning one of nine categories to the programs. Their model then iteratively adapts the utility of a viewer segment for a certain program type. They then predicted the audience flow by assigning each viewer of a program to the most probable next program by maximising the utility. Their best model managed to predict the correct next show in 76% of the cases. While the basic idea is still valid today, it is important to note that they built their model on the data of only 5'434 viewers that sent in their viewing diaries and predicted programs for the only four available channels at the time.

Predicting viewership numbers seems to be an activity that is largely done in advertisement agencies in order to produce forecasts for their clients. The forecasts of the market shares of a show are important since advertisement agencies buy up to 80% of their advertising slots for the autumn season during spring and summer (Mandese, 1995) based on those predictions. Because of that, Napoli (2001) investigate what factors influence the forecasting errors. They come to the conclusion that recurring lead-in and lead-out programs greatly reduce the forecasting error for a show. They also state that forecasting error have been increasing over time, as a result of the TV landscape becoming more diverse and complex. They also point out that accurate prediction might be easier for certain demographics than others. Additionally, it is interesting to see that competing programs on other channels seem to have little effect on the viewing choices. By comparing the forecasts made by advertisement agencies for 140 programs on four big channels in the United States to the actual program ratings they calculate that the average mean percentage error of those predictions is 21.35%.

The most recent and also most comprehensive work on the forecasting of TV ratings is done by Danaher et al. (2011). They give an overview of all the academic work that was concerned with creating a forecasting model for TV viewership. The quality of the models is very heterogeneous, also due to the fact that the studies make very different assumptions. They also always simplify the problem by using a short prediction horizon and datasets with few channels. The authors create their own model, which delivers average performance but within real world constraints, using four years of panel data. They predict ratings up to six months in advance, a time frame often needed by practitioners but never used in academics before: The next longest prediction horizon was 12 weeks. They also show that directly predicting the ratings of a show is better than the often used approach of first predicting the total number of viewers for a timeslot, and then splitting this total number amongst the shows. Another important insight is that while they agree that the lead-in and lead-out program are predictive for the rating of a show, it can't be used in practice: When using real-world time-horizons those features are often not known. Having analysed possible features under those real-world constraints they come to the conclusion that their result of a mean average percentage error of 36% poses a limit of how accurate predictions can be.

Building and maintaining models on large datasets that are a combination of multiple sources is no trivial task. The system needs to be robust enough to handle failure of one or multiple sources and has to deal with different levels of data quality. Raeder et al. (2012) look at such a system that they have built. To keep such a system going, the goal must be to minimize human and computer effort. They show how simple tests can greatly minimize such effort.

Stonebraker et al. (2010) show the relative strengths of parallel database management systems (DBMS) and MapReduce. They come to the conclusion that MapReduce is better suited for semi-structured data, quick analysis and data transformations on data that is only read once. Parallel DMBS excel at efficiently querying large datasets. For complex tasks often a combination of different paradigms is necessary, which highlights the need for "smart software" that ties them together.

(Lamb et al., 2012) explain how the Vertica Analytic Database achieves the high performance needed for a modern "web scale" analytics system. The main difference to traditional relational databases is that Vertica is column oriented and not row oriented and supports multicore execution. This is an optimization for analytical workloads where a significant fraction of the rows in the database are accessed. This stands in contrast to transactional workloads where mostly single rows in the database are accessed. Due to the optimization for this kind of workloads, it is possible to exceed the performance of traditional one-size fits all relational databases by multiple orders of magnitude. This not only applies to querying the data, but also to loading the data into the database in the first place.

3

Market

To understand what an OTV analytics system has to accomplish, it is important to know what the relevant market looks like. We will use the Swiss OTV market as an example. The main stakeholders for such a system are the OTV providers, the TV channels and the advertisement agencies. While other countries will most likely have their own special arrangements for legal and historical reasons, we think that the overall structure will be similar.

3.1 Data Providers

We will define OTV providers as companies that make it possible to watch the content of TV channels live over the Internet. With the introduction of Smarpthones and Smart TVs this no longer means that the content must be watched on an Internet browser, it can also be consumed over a dedicated application on a portable device.

Those OTV providers are either start-ups that specialise in OTV like Teleboy[1], Wilmaa[2] and Zattoo[3] or established telecommunication companies like Swisscom[4] or UPC-Cablecom[5] that offer OTV as an additional service to their customers.

The content they stream is created by broadcasting companies that produce and buy content for one or multiple channels. Examples are the Swiss public broadcasting organisation SRG SSR[6] that produces the TV channels SRF-1, SRF-Zwei and SRF-Info or the 3 Plus Group AG which produces the channels 3+, 4+ and 5+. Some of those broadcasting companies stream their own channels over the Internet on their own websites or applications and are therefore OTV providers as well. We will call them OTV broadcasters to differentiate them from pure OTV providers.

Within the groups of OTV providers and OTV broadcasters are companies with market shares and budgets of very different sizes.

The OTV providers are the main data providers for the OTV analytics system, as they offer the widest range of channels. The OTV broadcasters can only offer data for the

[1] *www.teleboy.ch*

[2] *www.wilmaa.ch*

[3] *www.zattoo.ch*

[4] *www.swisscom.ch*

[5] *www.upc-cablecom*

[6] *www.srg.ch*

channels that they produce themselves. Nevertheless this data can be valuable as OTV broadcasters usually offer their content for free online, which might attract different viewers than the paid-for or ad-supported offers from OTV providers.

Trading off the costs and benefits of adding redundant data providers is important (Raeder et al., 2012). They make the whole system more robust to failures of data providers, as having viewer information for the same channel from multiple sources increases redundancy and might increase the size and quality of the dataset. On the other hand, integrating a new source does not happen automatically but costs developer time. And if the quality of the new data source is below average it might reduce the overall quality of the data set.

3.2 Data Consumers

The data providers are themselves interested in the data. Be it because they can gain additional insights into their own data, or by having access to market data they would otherwise not have. Since all data providers run independent systems, they might only see a small fraction of the complete OTV market, depending on their size. For them providing data to the dataset can be seen as getting access to the overall OTV data.

The OTV providers are probably more interested in aggregate statistics regarding the engagement numbers, because they cannot control the content. OTV broadcasters are additionally interested in what type of content viewers consume in order to optimize their line-up.

The third players in the market are the advertisement agencies. They buy advertisement slots on TV channels and sell those to their clients to place advertisements. Just like OTV providers, they are less interested in the type of content that is watched, but in what slots are viewed by what demographic of viewers.

4

Requirements

The basic requirements for the OTV analytics system can be deduced by analysing its proposed role and by looking at similar systems. If this system was used in production, additional and more detailed requirements would surely surface from the actual users who are domain experts.

1. **Independence of source failures**
   The viewing data should be available for analysis as quickly as possible. Delivering the data on a daily basis seems to be industry standard. When combining the data from multiple sources, the system must be able to deal with errors in one or multiple sources independently on a daily basis. If one faulty source could stop the system for the day, each new source would increase the likelihood of a system failure which is not acceptable.

2. **Atomicity**
   The data transformations and the data imports should be atomic, i.e. they should either be completed correctly or not at all. Inconsistent states are not allowed to happen, as this could corrupt the data already stored in the system.

3. **Performance**
   a) The import of new data should be reasonably quick. If the data of the previous day is delivered over night it should be imported and transformed by morning when the working day starts.
   b) Similarly the analyses should run in a reasonable timeframe to make ad-hoc analyses and explorative data analysis possible.

4. **Automation**
   a) The import of new data should happen automatically. New data should arrive overnight. It is therefore unrealistic to assume that a developer will be present to start the import.
   b) Quality control also needs to run automatically as a human might forget to start it.

5. **Traceability**

    a) The results of the analyses run on the data can have a real-world impact. An extreme example would be shows being cancelled for having bad ratings. It is therefore important to be able to clearly show how the data was treated as this might influence the results. It should also be possible to audit the transformations. If due to different data formats or heterogeneous data quality different transformations are run for different providers, it should be clear on which entries what transformations were run.

    b) It must also be clear for each record from what data provider it was imported. This is necessary if for instance only a subset of the data providers should be included in an analysis, or if a data provider has to be removed from the system.

    c) On the code level, version control should make it possible to run atomic deployments of new versions and to be able to know which code version was used in production for what time. This is for example necessary to troubleshoot bugs in the code and find out what time period of data was affected.

    d) It should be clear which data files were imported and which not.

6. **Reproducability**
It must be possible to reproduce data as it was when all transformations where run on it. This is for example important when a bug on the production system needs to be reproduced on a development machine. This goes hand in hand with the requirement of traceability, because only when it is visible what happened to a data set, can it be reproduced.

7. **Security**

    a) Even though the data is assumed to be anonymised, the data the system holds is still sensitive. Combining the viewing patterns with outside information might make it possible to identify individuals. Narayanan and Shmatikov (2006) have shown that this was possible with the data of the Netflix prize. Additionally, having their data made public in un-aggregated form might have a negative impact on data providers.

    b) To protect against data loss stemming from hardware failure regular off-site backups are necessary.

5

Data

The data the system imports and works on can be split into three distinct parts. The first part are the server logs that contain the actual viewing sessions. The second part is the additional information about the users for the demographic analyses. The last needed piece of data is the electronic program guide (EPG) information. The EPG data contains the information on what program was running on what channel at what time. Separating those sets of data before the import reduces the data size, as the data is denormalized. It also makes it possible to combine different EPG data for a set of sessions. This could be of advantage when a data provider is known to have inaccurate EPG data which can then be replaced by better EPG data of another data provider.

We will now look at each kind of data in turn.

5.1 Session Data

A session is defined as the time between channel switches. Therefore a session is always mapped to exactly one channel and one user. The attributes of the session data can be found in table 5.1.

5.2 User Data

The user data is needed to infer the demographics of the viewership and to tie sessions by the same person together. The attributes of the user data can be found in table 5.2.

5.3 EPG Data

Since the session data only reflects how much time a user spent on a certain channel, we need additional information to infer which programs were actually viewed and for how long. For this we need the EPG information whose attributes are defined in table 5.3.

Field	Description
UserID	Anonymous user ID. This ID has to stay constant for the same user over multiple imports, but should not make it possible to identify a person. A possible candidate would be to hash the username of the user using a secure one-way hash function.
`Channel`	Channel the viewer was watching.
`TStart`	Time at which the session was started (Wall-clock time). Granularity is seconds.
`Duration`	Duration in seconds for which the session lasted.
`TPvr`	Wall-clock time at which the program watched was broadcasted. If the viewer was watching live TV, `TPvr` is equal to `TStart`. If the viewing is time-shifted then `TStart` > `TPvr`.
`Device`	Device used for viewing.
`Country`	Country from which the session was started.
`Location`	Name of the location from which the session was started.
`Zip`	Zip code of the location from which the session was started.
`Asset type`	Indicates if the session watched is live content or time-shifted content.

Table 5.1: Attributes of the session data.

Field	Description
`UserID`	Anonymous user ID. Must be the same in the sessions data, so that the user and session information can be joined.
`Age`	Age of the user in years.
`Gender`	Gender of the user.
`Language`	Default language set by the user.

Table 5.2: Attributes of the user data.

Field	**Description**
`Title`	Title of the show.
`Categories`	Type of content categories the program belongs to. Examples are ”news” or ”documentary”.
`Channel`	Channel the program is broadcasted on.
`TStart`	Time at which the program started broadcasting (Wall-clock time). Granularity is seconds.
`Duration`	Duration of the program in seconds.

Table 5.3: Attributes of the EPG data.

5.4 Data Format

We expect the data to be delivered as three separate files. One for the sessions, one for the EPG data and one for the user data. The records should be stored as comma separated values (CSV), one per line. CSV is an appropriate format, as it is very simple and can therefore easily be generated by a data provider. It can also be manipulated with a lot of the common UNIX tools like `head` and `wc` to make quick sanity checks on the server without having to open the file in a GUI program. Nevertheless, the system could also deal with data in multiple different formats, as long as they can be converted to a format that can be imported into a relational database.

6

System Architecture

The OTV analytics system is made up of multiple, interconnected components that are loosely coupled. This has the advantage over one large monolithic system that a component can quickly be exchanged, even only for one particular task or analysis. We will now look at each component in turn. The data from the OTV providers is downloaded to the server and then loaded into the database. This data importer is the first component. The second component is the database which stores the complete data. On top of the database sits an analytics layer which computes different analyses and stores and displays the results. This is the third and final component.

In this chapter we will show that the system currently fulfils five out of the seven requirements from chapter 4. Requirement 4 (automation) and 7 (security) were not fully covered since this system was not used in a production environment. The system is not completely automated because the data we received for this thesis was delivered as one complete set and not in daily increments. The import was therefore started manually, from which point on the complete process would finish automatically. For use in a production environment the start of the process should be triggered with a Cron job or similar automation tool. The system was of course secured with appropriate, state of the art measures, but we did not perform automated, off-site backups of the data.

We will now look at each component in turn and explain how they interact with each other.

6.1 Data Importer

The data importers are designed using the template pattern. One base importer runs the basic steps needed to import a generic dataset. Each importer for a concrete data source extends this base model and makes it possible to extend a step or skip a step if necessary. It is therefore easily possible to handle data sources of different qualities and adapt to individualities of specific data providers. The basic steps are the following:

1. Import user data
2. Import session data
3. Import EPG data

4. Run transformations

Some providers might for instance not provide EPG information, in which case the step of importing EPG data can be skipped. For another data provider it might be important to rename the channels before importing, because they have yet to adapt internally to the renaming of a channel (for example the Swiss channels ”SF-1” and ”SF-Zwei” were renamed to ”SRF-1” and ”SRF-Zwei”in September 2011 (Schweiz am Sonntag, 2015)). This can easily be done by extending the session import step.

Since each data provider has its own, separate import process one erroneous data provider does not halt the imports for the remaining data providers. The data importer therefore fulfils requirement 1, independence of source failures.

6.1.1 Importing Users, Sessions and EPG

As described in section 5.4, we expect the data to be delivered as three separate CSV files. The files are put into a folder for each type of data. The data importer knows what type of data lies in which folder and runs the appropriate import step, which basically consists of a SQL `COPY` statement. Once the file is imported into the database it is moved into a separate folder. This ensures requirement 2, atomicity, as running the data importer a second time will not import the same file again. It also ensures requirement 5 traceability, as it makes it easy to see which files are already imported and which aren't. If multiple data providers are used, the folders are separated into a directory tree for each data provider. The data importers for each data source then know which directory tree is theirs and only run their imports on the files present in their folders.

6.1.2 Transformations

The transformations are steps necessary to ensure consistent data over multiple data sources. Transformations make for example sure that known inaccuracies in specific data sources are smoothed over. They also compute additional attributes, like for example assign sessions to language regions.

The transformations are all written as SQL statements on the already imported data. They are written as numbered SQL files and read and executed by numerical order by the data importers. Having one transformation per SQL statement makes it possible to fulfil requirement 5 traceability, requirement 6 reproducibility and requirement 2 atomicity, as the database keeps track of the statements executed and atomicity is an inherent property of SQL transactions. While doing those tasks would also be possible with MapReduce we believe that the database is better suited for this task, as the data is well structured and should be read multiple times (Stonebraker et al., 2010).

To make sure that the transformations are scoped to only new entries, each transformation method requires the ID of the first record in the database that needs to be included. Since the IDs are generated sequentially, this makes sure that only new entries are affected by a transformation. As was explained in the previous subsection, multiple data providers can be supported by splitting the directory tree into separate subtrees for each data provider. This principle also holds true for the transformation files. Transformations common to all data providers are put in a common folder known to all data providers. Transformations specific to a data provider are put in a folder specific to the corresponding data importer.

Language regions The first transformation we want to look at is the one that assigns language regions to sessions. This is necessary, since in Switzerland there are four distinct language regions. There is a German, French, Italian and Romansh language region. As explained by Mediapulse (Mediapulse, 2015a), the official supplier of the statistics for TV usage in Switzerland, the reports are always done by language region for the German, French and Italian regions. As we will show in chapter 7, the assumption that the viewing behaviours differ by language region is a valid one and the sessions should therefore be assigned to their respective language region.

In the database the correspondence of a session to a language region is modelled with the attributes `region_de`, `region_fr` and `region_it`. Those attributes are set to 1 if the session is from the corresponding language region and to 0 if this is not the case. The sum of those attributes is therefore at most 1.

The Swiss department of statistics publishes a list of all the Swiss communes, which indicates to which language region a commune belongs (Bundesamt für Statistik, 2015). Since a commune can have multiple ZIP codes which are assigned to different language regions, we found that assigning sessions by their ZIP code of origin was the most robust solution. We therefore compiled a list of all 4930 ZIP codes with their respective language region. Sessions are joined by ZIP code with this list to assign their language region.

This method works flawlessly, when the ZIP code of a session is known. But since the ZIP code is obtained by geolocating the IP-address, the ZIP code is not always known. Geolocation can't locate all IP addresses. MaxMind, a large service that offers IP georesolution, can only resolve 81% of the IP addresses to a 25 KM radius [1]. Not being able to assign almost a fifth of the sessions to a language region, would mean a large loss of data for analyses that are language region specific. By analysing exemplary data, we have additionally found that sessions from devices using mobile Internet are less likely to have a ZIP code than from sessions using a cable connection. Since mobile devices are unique to OTV it is very important that they are correctly represented in the data. To mitigate this, we assign sessions without a ZIP code by looking at the language set by the user. For each language a user can set, we learned from the data how likely that session is to originate from a language region. For example when the user language is French, the session might be from the French-speaking language region with a probability of 86%, from the German-speaking language region with a probability of 10% and from the Italian-speaking language region with a probability of 4%. These probabilities can be directly assigned to the attributes `region_de`, `region_fr` and `region_it`. In total those three attributes again sum up to 1 and can be seen as weights that indicate to what extent a session belongs to a certain region.

[1] *https://www.maxmind.com/en/geoip2-city-database-accuracy?country=Switzerland&resolution=25*

Those rules need to be updated periodically of course to reflect changing distributions. Another, easier method would be to assign the sessions to a region with probabilities weighted by the relative sizes of the language regions. We believe however that our method is superior because it takes into account additional information about the users.

This transformation assumes that if a ZIP code is present, it is correct. This is, however, not always the case,. MaxMind indicates that up to 15% of its ZIP codes are not correct. A wrong ZIP code does however not always mean that the language region will be deduced incorrectly. This will only be an issue if the correct ZIP codes are close to a language frontier.

Program Sessions As explained in chapter 5, we only expect data providers to give us session level information which means that we know what channel a user watched but not what programs. To be able to run analyses at the program level we need to cut those sessions into program sessions. This is done by joining the session data with the EPG data. The algorithm works by looking at a session as a timeline that stretches from `TPvr` to `TPvr` + `Duration`. The programs are modeled as program timelines that last from `TStart` to `TStart` + `Duration`. For each session timeline we look at the program timelines from the same channel that overlap with it. Each overlap is turned into a program session and saved in the database. What makes this non-trivial, is that the start and end time of the new program session have to be computed. Additionally, we need to take time shifted viewing into account as well.

Denormalize User Information When running the first versions of some of the analyses shown in chapter 7, we realized that the performance penalty of joining the user table with the sessions table was very large. For queries that computed an aggregation over the complete program sessions table, the cost of joining could easily lead to a 20 times slower performance when compared to analyses without joins. Having to wait 10 minutes compared to 30 seconds makes a large difference in the capability of the users to run ad-hoc analysis and run explorative data analysis. Since this was an explicit requirement for the system, we decided to denormalize most of the user data. This means that we add the attributes of the user of a session to the session entry in the database. Usually, one tries to avoid denormalizing a database schema since it makes updating an entry more complicated. This is because if a denormalized attribute is updated, it needs to be changed at multiple places. But since the data in our database is not supposed to be updated but is read very often, denormalizing makes sense.

6.1.3 Import Performance

As described in chapter 4 our goal is to be able to run a complete import between the time it arrives in the very early morning hours and the start of a working day. If we assume that the data arrives by 04:00 AM in the morning this leaves the system 4 hours until 08:00 AM. Testing has shown that the current system is well within this range. Copying a day of data takes two minutes on average on a standard server with a standard hard-disk. Using a server with SSDs might improve the speed. Running the transformations takes approximately 8 minutes in total per day and data provider. This means that even with multiple data providers the import performance should not be an issue. The data importer therefore covers requirement 3 (performance).

6.2 Database

The database is at the center of the OTV analytics system. It stores all the data and is also used to compute the majority of the analyses. We decided to use Vertica[2], a column-oriented database that was created out of the C-Store research project(Lamb et al., 2012). Most of the large technology companies like IBM, Oracle and SAP offer some sort of big-data, column oriented database. Amazon offers such databases as a service with Amazon Redshift, but since it can only be hosted on Amazon Servers we decided against evaluating it, as we didn't want to store such sensitive data on cloud servers abroad. We finally chose to use HP Vertica as it fulfilled our requirements and one data provider we spoke to uses Vertica in-house and had made good experiences. Additionally, HP offers a free Community Edition of Vertica. It offers the same functionality as the full version but is limited to 1TB of data and a cluster size of three. None of those were limits that concerned us for the purpose of this thesis.

6.3 Analytics Layer

The analytics layer of the OTV analytics system is the interface between the data and the users. As we have seen in chapter 3, prospective users are business intelligence analysts at a data provider, TV show producers or advertisement agents. This is a very broad set of potential users who will most likely all require specific features. Building a specialized tool for all of those stakeholders was out of scope because it would not only require a very large effort in collection the requirements, but also in developing those solutions. We therefore decided to use an existing Online Analytical Processing (OLAP) tool and connect it to the database. We settled on using Tableau[3] as we felt that it would provide us with enough flexibility to cover different types of analyses while being powerful enough to deliver results quickly.

We were able to do most of our desired analyses with Tableau. For some very specific analyses and predictions, which are described in detail in chapters 8 and 9, we had to write complex queries, save the results and use additional software libraries. This is not possible with Tableau. For those very specific analyses we therefore chose to build our own tools that were custom made for answering those very specific questions. We feel that this approach of using off-the-shelf software for all but the most specific analyses has served us well in keeping time writing boiler plate code to a minimum.

[2] *https://www.vertica.com/*

[3] *http://www.tableau.com/*

7

Market Analysis

In Switzerland the measurement of TV and radio was mandated by the Swiss Confedaration to the Mediapulse Foundation (Mediapulse, 2015d). Mediapulse makes the data available to its customers daily over a custom software called InfoSys+, which makes it possible to compute market shares, show ratings and reach and also offers many additional analyses (Mediapulse, 2015b). Providing the same breadth of functionality in the OTV analytics system was out of scope. However, Mediapulse publishes a yearly report (Mediapulse, 2015a) intended for the public, showing the most important and relevant summary statistics (Rating, share and reach) for the German-speaking, French-speaking and Italian-speaking parts of Switzerland. This report can be seen as a minimal requirement that has to be fulfilled by the OTV analytics system. In this chapter we will show what data we used and what kinds of analyses we computed with the OTV analytics system. Additionally we point out how the analyses of a source-based system differ from analyses of a panel-based system.

7.1 Data

We received data from one Swiss OTV provider. For proprietary reasons we cannot divulge which OTV provider it is. The data was received in the format described in chapter 5. For the analyses in this chapter we used data for the months January, February and March.

The data was imported into the OTV analytics system and transformed using the data importer described in section 6.1.

7.2 Analyses

The main measures used in the yearly Mediapulse report (Mediapulse, 2015a) are market share and ratings. We will now look at each market analysis of the OTV data in turn. Those analyses were all computed using the Tableau visualization tool described in section 6.3 that was connected directly to the database described in section 6.2.

Session ID	Channel	Duration	Region DE	Region FR	Region IT
1	C1	3600	0	1	0
2	C2	1800	0.8	0.15	0.05

Table 7.1: Example sessions for calculating market share

7.2.1 Market Share

The market share for a channel is defined as the percentage of the total viewing time spent watching programs over a certain time period. In our case the time period was set to the three months that we had the data for.

In order to calculate the market share for the different language regions we multiplied the duration of each session by the values of the different region attributes (see 6.1.2). Example 7.2.1 illustrates this.

Example 7.2.1. Taking the two simplified sessions in table 7.1 we see that the first session has been completely assigned to the French-speaking region. Therefore, the duration of 3600 seconds is added to the total viewing duration of the French-speaking region, and channel C1 is assigned 3600 seconds viewing duration in the French-speaking region, and 0 seconds in the German- and Italian-speaking regions. The second session adds 1440 seconds ($1800 \cdot 0.8$) to the total viewing duration of the German-speaking region, 270 seconds to the French-speaking region and 90 seconds to the Italian-speaking region. Channel C2 is awarded 1440, 270 and 90 seconds in the German-speaking, French-speaking and Italian-speaking region respectively. If those two were the only two sessions for our time period, channel C1 would have a market share of 93% ($\frac{3600}{3600+270} = 0.93$) in the French-speaking part of Switzerland.

Figure 7.1 shows the result of an analysis of the market shares of channels in the German-speaking part of Switzerland. The same analyses were also done for the other language regions of Switzerland. The time range for such an evaluation can be adjusted in the tool. This allows the user to look at hourly, daily and weekly market shares.

7.2.2 Ratings

While the market share is usually used at the level of channels, ratings are often used at the level of individual programs. The rating of a show is the number of viewers weighted by the fraction of the show they watched. Example 7.2.2 shows how the rating is calculated.

Example 7.2.2. Let us assume that a program lasts for 1800 seconds (30 minutes). In table 7.2 we see two sessions for that hypothetical program. Session 1 watched the program for its entirety and session 2 only watched half of it. If those two were the only two sessions for that program, its rating would be 1.5.

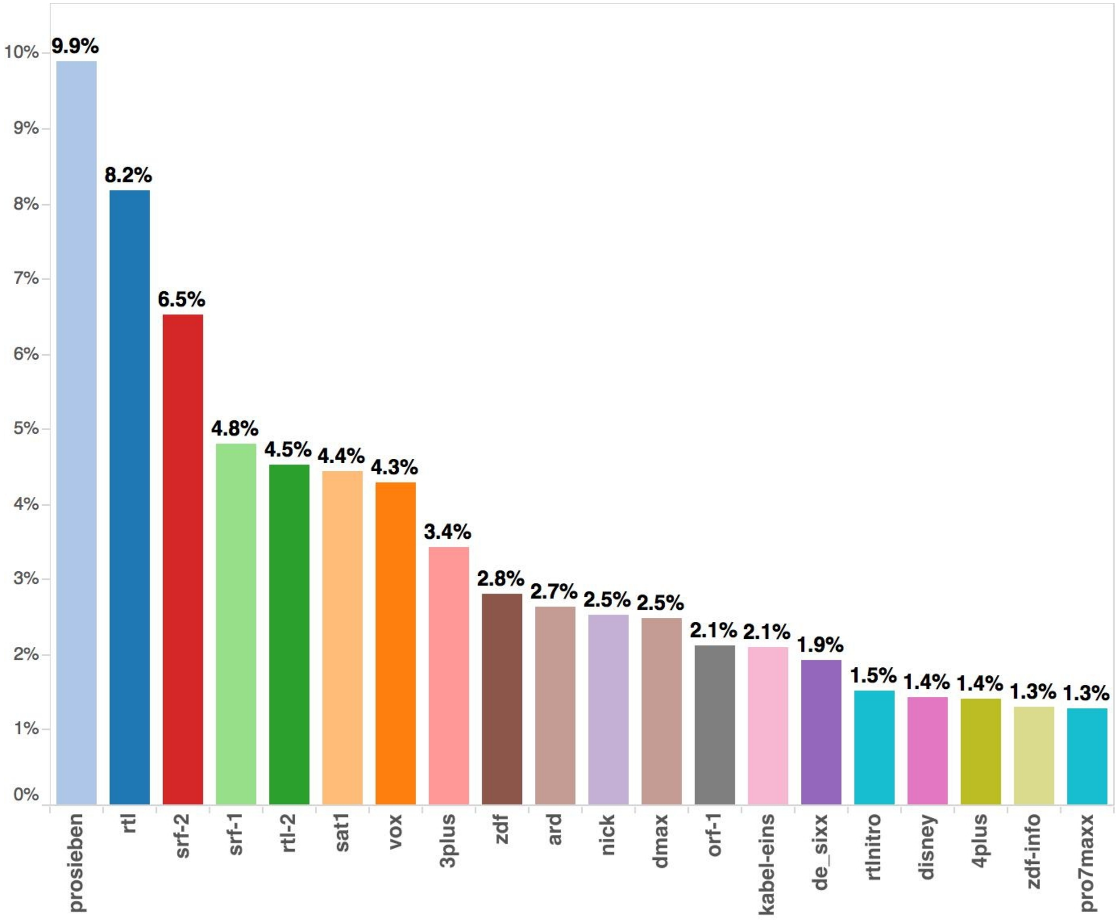


Figure 7.1: Market shares for top-20 channels in German-speaking part of Switzerland.

Session ID	**Duration**
C1	1800
C2	900

Table 7.2: Example sessions for calculating rating

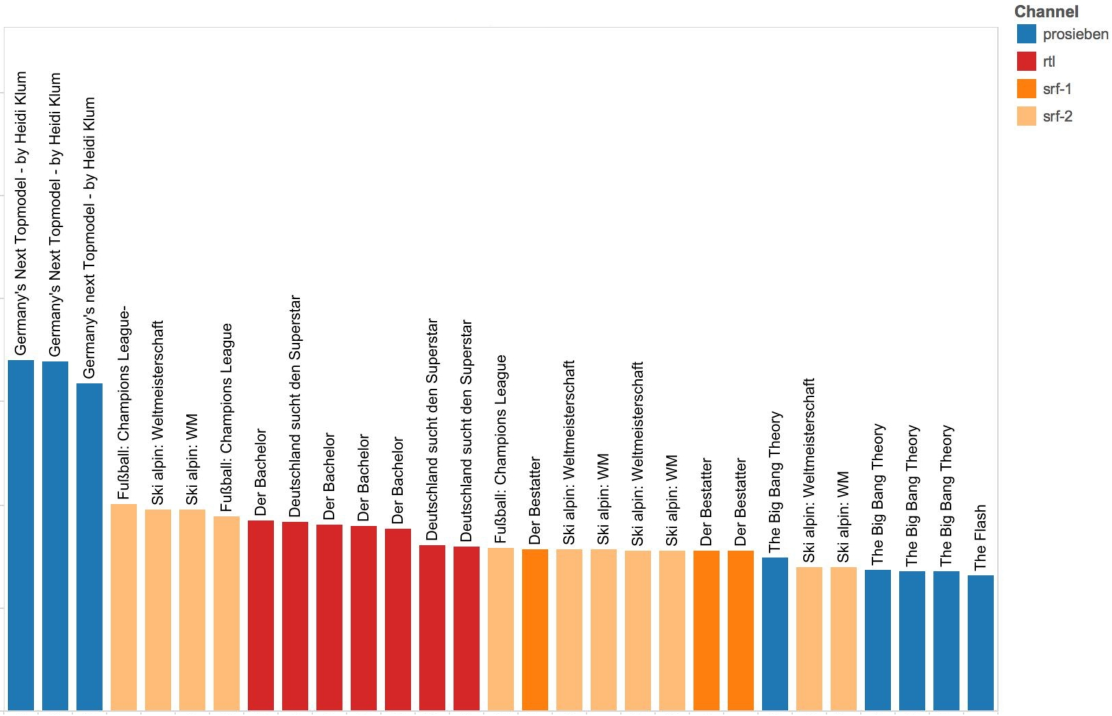


Figure 7.2: Top 30 programs by rating for German-speaking Switzerland in February 2015

The rating for a program is calculated by the OTV analytics system using the program sessions that were computed by joining the EPG information and the session data (see subsection 6.1.2). Each program session s for a show p contributes a rating point $r_{s,p} \in [0, 1]$. The duration of the program p is denoted as $d_p \in \mathbb{Z}$ and the duration of the session s is denoted as $d_s \in \mathbb{Z}$. The rating point for each program session can therefore be calculated as $r_{s,p} = \frac{d_p}{d_s}$. To show the rating of a program p we sum up all the rating points $r_{\_,p}$. Multiplying $r_{s,p}$ with the region weights, makes it also possible to calculate the ratings by region.

Figure 7.2 shows the results of an analysis computing the ratings in the German-speaking region of Switzerland. For proprietary reasons we cannot show the absolute ratings for the programs.

7.3 Demographics

The data we received for this market analysis also contained the age and gender of the users. This makes it possible to look at the age and gender distribution of channels.

Figure 7.3 shows the percentage of the total viewing duration for each channel split by gender. When comparing the gender distributions to the target demographics of the channels those distributions seem plausible. Sports oriented channels like TSR-2 and

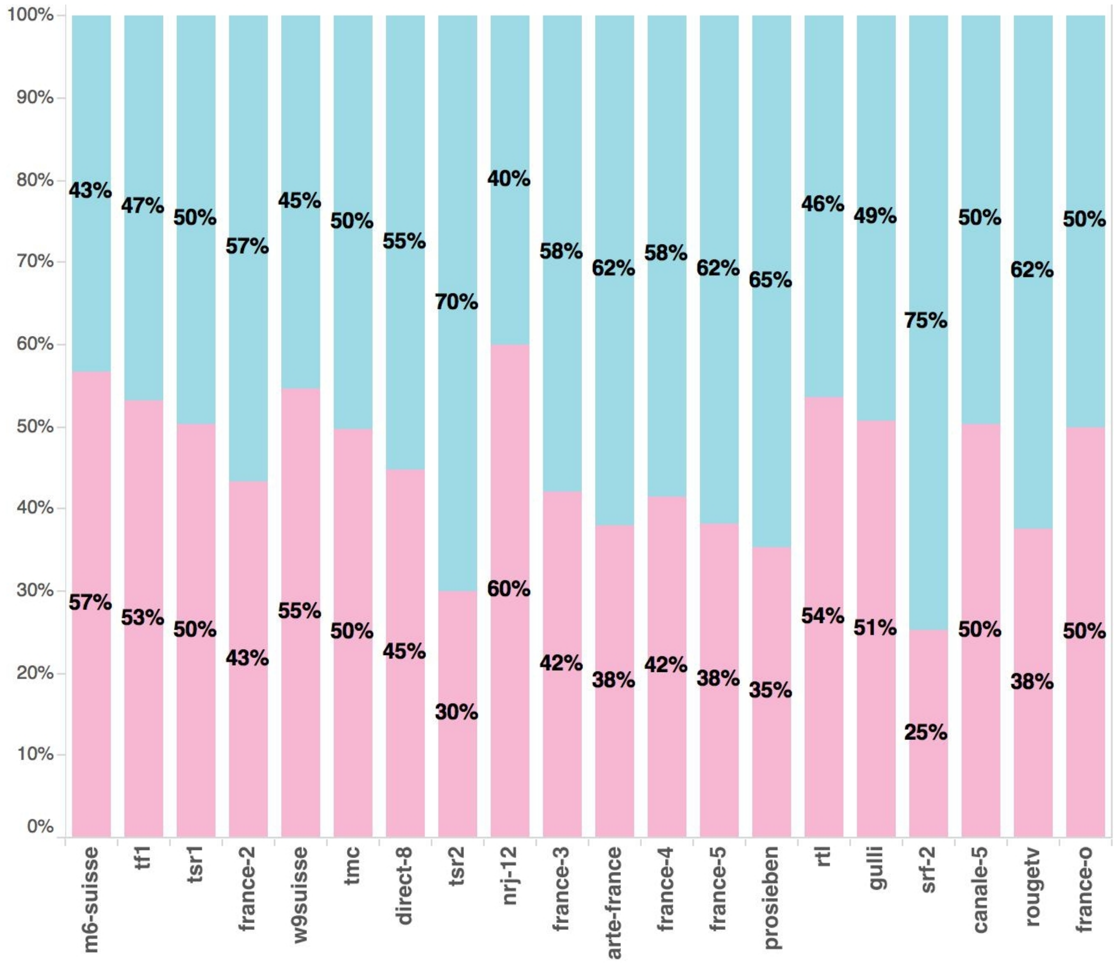


Figure 7.3: Gender distribution for top 20 channels in the French-speaking part. Pink: women, blue: men.

SRF-2 have an above average percentage of male viewing time.

Figure 7.4 shows the percentage of the total viewing duration grouped by age class. Comparing the age-distribution with the target audience of the channels again leads us to believe that those results are plausible. The channel with 60% of the viewing time consumed by users aged 15-29 is MTV Italia, which runs contemporary music videos and reality shows.

Since the data also tells us what kind of device was used for each session, we are able to show for each program how much of it was watched on what device. Figure 7.5 shows the distribution of the devices types for selected shows. The shows were chosen out of the top programs for the German-speaking part with the goal to show a mix of different types of programs. It is interesting to see that overall there is not a large difference in device usage amongst shows. We also saw this when looking at other shows

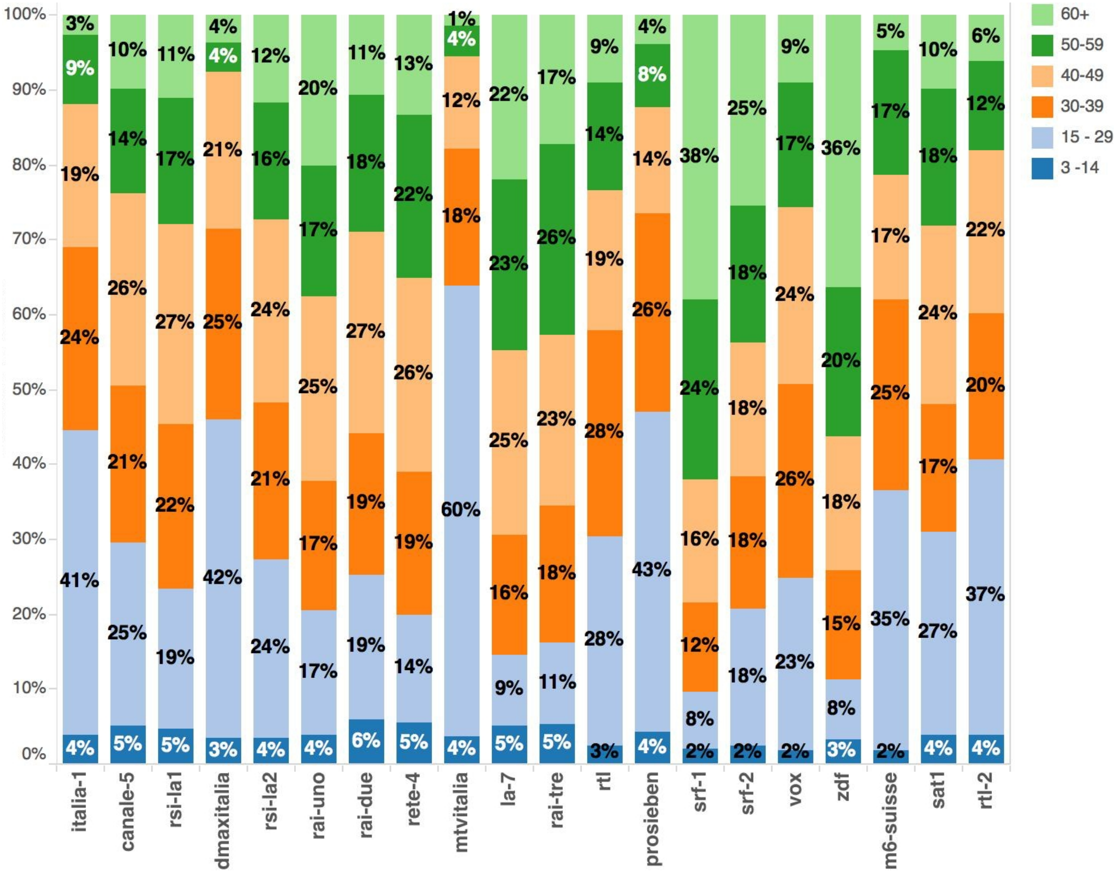


Figure 7.4: Age distribution for top 20 channels in Italian-speaking part of Switzerland.

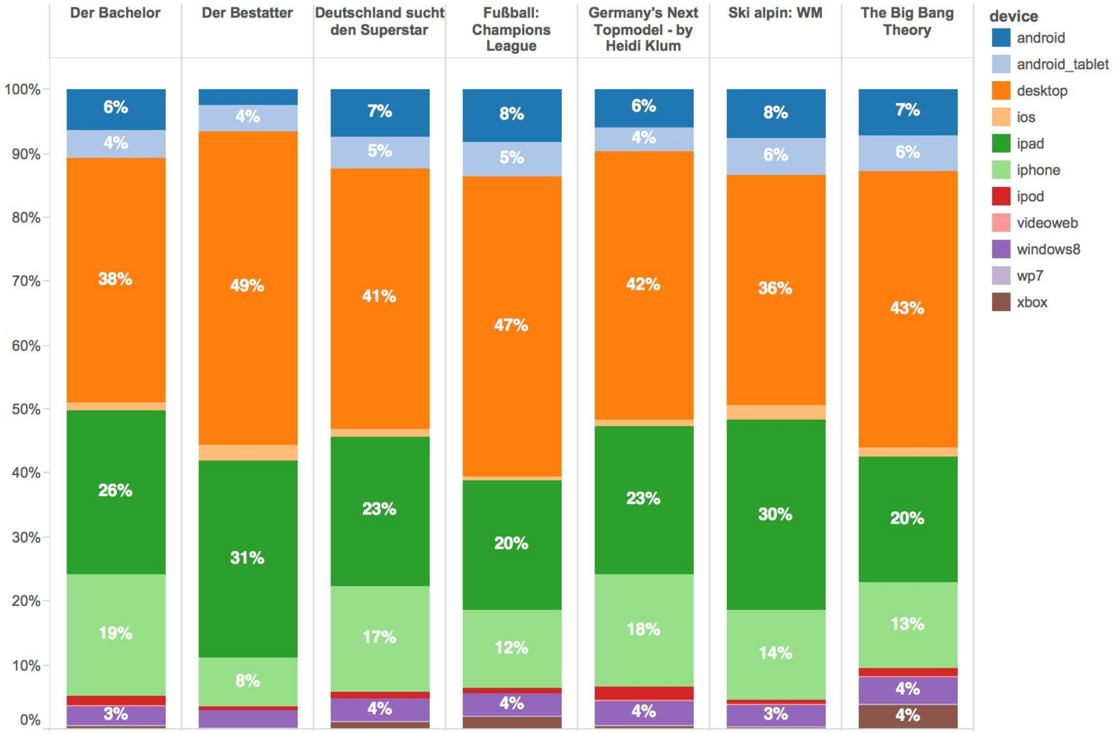


Figure 7.5: Device distribution for selected shows

not depicted here. The largest fraction is always made up of desktop computers, which includes devices like notebooks and desktop computers that access the content over a web browser. iOS devices like the iPhone and the iPad have a much larger share than android devices.

An additional analysis that we can perform is look at how often content was viewed live or time-shifted. Figure 7.6 shows the distribution between live viewing and time-shifted viewing. That sports are mainly consumed live makes intuitive sense. Within non-sports content there are large differences to be seen. Some shows are viewed time-shifted more than 50% of the time.

7.4 Differences to Panel-based System

The main difference between a panel-based system and source based system is the number and the quality of the data points. Mediapulse is responsible for the panel-based system in Switzerland. According to their 2014 yearly report (Mediapulse, 2015c), they have measurement devices in 1870 homes and the sample size of viewers used for their yearly report was 2342. Those viewers are then used to extrapolate to a universe of 5'062'000 viewers, which means that one person in the panel stands for roughly 2000 viewers.

In a source based system, no extrapolation is needed, provided that all the different

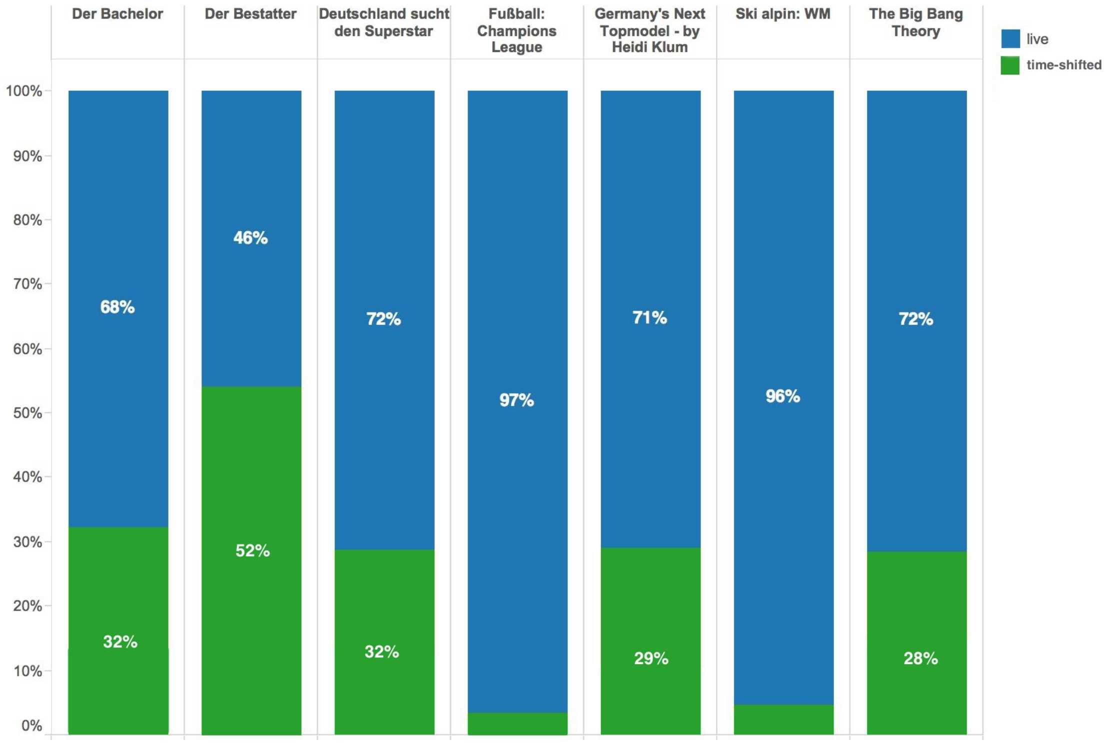


Figure 7.6: Distribution of live vs. time-shifted viewing for selected shows

sources that make up the universe can be integrated. But even with only one source the number of data-points is much larger. One channel with a market share of 0.3% in the German-speaking part of Switzerland had tens of thousands of users over the three month period we had data for, which is significantly more than the complete panel. Assuming that all the viewers in the panel watch about the same amount of television, a market share of 0.3% would be based on about 7 viewers ($2342 \cdot 0.003$). One user changing the channel for whatever reason leads to a 14% decline in market share for that channel. This can lead to very unstable results for smaller channels, which makes it more difficult for them to attract advertisement money. A source-based system is therefore especially useful for smaller channels.

For the advantage of having more data points a source based system has the disadvantage of not having complete demographic information of all the viewers. When a person is recruited for a panel, all her or his demographic information is collected and verified. This is not the case with the proposed OTV source based system, since the demographic information in the system depends on what information the data providers ask from their users. From the perspective of a data provider there is a fine balance to strike between losing potential users by asking them too much information about themselves, and providing valuable market data. Additionally, it is also possible for a user to give false information.

By adding elements of a panel-based method to a source based system, the drawback of the partially missing demographics could be mitigated to a certain degree. Asking selected users if they are willing to complete their demographic information could help fill missing data. This questionnaire could for instance only be showed to viewers that fulfil certain criteria, for instance viewers that watch a channel or show that is watched by an audience that has a high percentage of missing demographics. As an incentive, users could be for example given a free month of service by the OTV provider. This method of asking selected users to fill out a questionnaire could also be used to gain relevant information that goes further than age and gender: Education level and income is also information that could be of interest to the broadcasters, the advertisement agencies and the data providers. Asking every user for such sensitive information is intrusive and unnecessary. A well thought out sampling solution could make it possible to gain more information on the audience and offer a good trade-off between the privacy of the users and the desire of the the stakeholders to gain more information about the viewers.

The panel used in Switzerland is constructed in such a way as to represent the demographics of the whole country. This might not necessarily be the same population as the people that watch TV. For example older people might watch a lot more TV than their representation in the panel might lead one to believe. By its very nature this problem cannot be resolved with a panel because in order to know how often the Swiss population watches TV a panel is needed in the first place. With an ideal source based measurement we could get closer to the true value.

8

Audience Flow Analysis

Audience flow is the flow of viewers switching from program to program. Audience flow can be used for different purposes. It was used by (Rust and Alpert, 1984) to predict individual viewer choice. This model could then be used to predict the impact of competitive programs by looking at how the viewer behaviour would change if a certain type of program was introduced in a competing time slot. Audience flows can also be used without any modelling as the audience flow of a program can give a broadcaster important information to which competing program viewers are lost.

We chose to add audience flow analysis to the OTV analytics system because it is a good example of a non-trivial analysis that is specific to TV data. In order to get a baseline for a model that predicts the audience flows of a program, we were also interested in seeing how stable audience flows are over time. Therefore, in this chapter we will first explain how an audience flow is computed and then present the results of our stability analysis.

8.1 Computation

Figure 8.1 shows an audience flow for a particular program. The left hand side displays the incoming audience flow i.e. from what channels the viewers come. The right hand side displays the outgoing audience flow i.e. to which channel the viewers go. In the system the animation of the audience flow is interactive, hovering over a channel highlights it and displays the percentage of the audience that this channel contributed to the total audience of the program.

Before we compute the audience flow, we must first decide what our definition of ”watching” a show is. This is important, because we want to separate viewers that quickly zap through the channels from viewers that consciously watch the contents of the program. An obvious solution is to take a minimal duration in seconds that one needs to have viewed the show for. The second value we must define is the duration before and after a show which we still take into consideration.

Example 8.1.1. Let us assume a viewer v_1 watches a show p on channel c_1. The show started at time t_p. Viewer v_1 was watching TV on channel c_1 and turned off her TV 10 minutes before p started ($t_p - 10$ minutes) to make a phone call or have a cup of coffee.

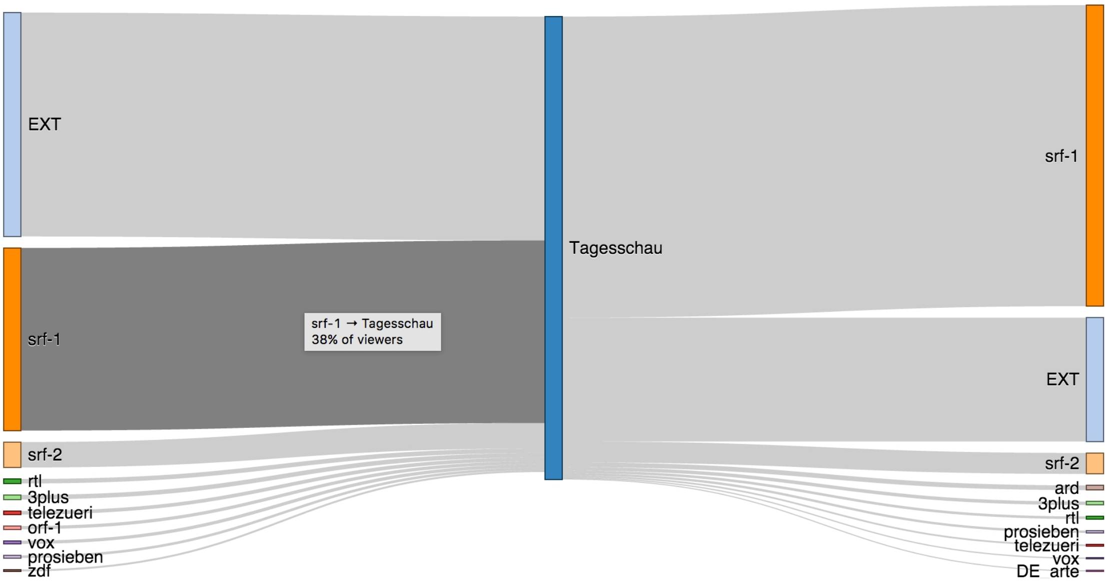


Figure 8.1: Audience flow for a broadcast of the ”Tagesschau” displaying the top 10 channels.

She turned her TV back on at time t and started watching program p. Do we assign her as coming from c_1 where she was 10 minutes ago or put her in the group of viewers that came from the outside, i.e. turned on their TV? Should the 10 minutes be counted as a break in the flow, or do we just assume that in viewers v_1's mind this is still the same session? Depending on how we choose this duration, we will classify v_1 as having stayed on the channel c_1 or having turned on her TV to watch program p.

Now let us assume that after having watched the show viewer v_1 is zapping around channels and only stays on a channel c_3 for more than a couple of seconds after 5 minutes. For the outgoing audience flow we must ask ourselves again if we should classify her as having switched from channel c_1 to c_3 or having turned off her TV.

Example 8.1.1 shows that choosing a suitable value for the buffer time is important as it will change the results. We define this buffer time as a number of seconds b. Note that b must be the same value for incoming and outgoing channel flows because the outgoing channel flow of one program is part of the incoming channel flow of another program. Therefore, the time interval before a program, where we take sessions into consideration, is $[t_p - b, t_p]$. If the end time of the program is denoted by f_p then the interval we take into consideration after the program finishes is $[f_p, f_p + b]$. Keep in mind, that sessions that started during the program itself are always taken into consideration: If one viewer switches channels after watching 10 minutes of a half hour show he or she can start watching another show before the original show has finished. Therefore the total time frame we take into consideration for incoming channel flows is $[t_p - b, f_p]$ and for outgoing channel flows it is $[t_p, f_p + b]$.

The minimal duration a person has to watch to be counted as a viewer of a program is d_{min} and is also denoted in seconds.

There are no industry standards we know of that state to what b and d_{min} should be set. After some experimentation and talking to some domain experts, we chose to set $b = 180$ and $d_{min} = 60$. The OTV analytics system is built in a way that these values can easily be changed, should the need arise.

The last definition we need to introduce is the one of a channel flow. When looking at figure 8.1 we see that an incoming or outgoing flow is made up of multiple flows, one for each channel. The incoming channel flow for the channel "SRF-1" is highlighted in the figure. We call such a key-value pair, where the key is the name of the channel and the value is the number of viewers, a channel flow. An audience flow can therefore be seen as a set of channel flows. The channel flow labeled 'EXT' is special in the sense that all the viewers that can't be assigned to a channel are bundled in there. Those are the viewers that turned on their TV just to watch the relevant program, or more generally viewers that didn't have a session of at least length d_{min} overlapping the relevant time interval before or after a slot. We will now look at the algorithm in detail.

8.1.1 Algorithm

We will show the algorithm for the incoming audience flows. It is almost identical for outgoing audience flows.

If we want to find the incoming audience flow for the program p we first need to find all the viewers that watched the program p for at least d_{min} seconds. Let us call the set of those viewers V_p. Now for each viewer we look at the time interval $[t_p - b, t_p]$ (the time interval with which a session before program p needs to overlap [1]). For each viewer we find the last session which overlaps that interval, lasted at least d_{min} seconds and started before the session for the program p . If such a session exists we take the channel of this session and increase the value of the corresponding channel flow by one. If no such session exists we increase the value of the external channel flow by one. After we have gone through all the viewers in V_p the set of the channel flows makes up the incoming audience flow. A formal version of this algorithm is shown in algorithm 1.

The algorithm for the outgoing channel flow is virtually identical, it is only necessary to change the time interval to include the buffer after the program and not before. One must also take the first session of a viewer in that interval and not the latest.

Looking at the algorithm, it is also clear that the size of the incoming audience flow (i.e. the sum of the values of the incoming channel flows) must be equal to the size of the outgoing audience flow. Since we look at exactly the same set of viewers in both cases, and assign each viewer to exactly one channel flow in both cases, the size of the incoming and outgoing channel flow must be the same for a program.

It is important to note, that this algorithm only takes into account sessions that were live. The concept of an audienceflow is not clearly defined for time shifted viewing.

[1] Strictly speaking only the interval $[t_p - b, t_p - d_{min}]$ is of importance. For reasons of simplicity we will not use this optimization when explaining the idea behind the algorithm.

```
Data: All program sessions $S$, relevant program $p$, minimal duration $d_{min}$,
      buffer time $b$
Result: A set of channel flows that make up the incoming audience flow
$C$ = create_empty_set_of_channel_flows();
$V_p$ = select_viewers($p, d_{min}$);
for $v \in V_p$ do
    $S_v$ = select_viewer_sessions_in_interval($[t_p - b, t_p]$, $v$);
    $S_v$ = filter_by_min_duration($S_v$, $d_{min}$);
    if $S_v \neq \emptyset$ then
        $s_v$ = latest_before_program($S_v$, $p$);
        $c_v$ = get_channel($s_v$);
        $C[c_v]$ += 1;
    else
        $C[EXT]$ += 1;
    end
end
```

Algorithm 1: Algorithm for the incoming audience flow.

8.2 Stability Analysis

While some research has already been done in trying to predict audience flows (Rust and Alpert (1984) and Rust and Eechambadi (1989)) this research is from a time where only a handful of channels were available compared to the more than 100 that the OTV provider we have data from offers. It was also only possible to watch TV on an actual television set, whereas today we can use a multitude of devices to access live TV content. Before trying to predict the individual viewing behaviour of viewers, we wanted to know how stable audience flows are over time. Do for example the daily news on a Monday have a vastly different audience flow than the same program on a Tuesday? We will first explain what data we used before presenting our results.

8.2.1 Data

The data was provided by the same Swiss OTV provider from which we already used the data mentioned in chapter 7. We limited ourselves to the top 10 channels across language regions for that time period.

To compare the stability of audience flows we created pairs of programs. We created pairs by taking each program and selecting the next program in the series. To be in the same series programs had to run on the same hour of the day on the same channel and have the same title. An example for such a pair would be the program "Geo 360" that ran on SRF-2 at 15:10 on the 12th of June 2015 and at 15:05 on the 22nd of June 2015. In this example the programs are 10 days apart, but there are also pairs where the distance is only one day. This of course depends on the schedule the broadcaster chooses. We found 14'508 of such pairs in the available data. We call this set of pairs the "consecutive pairs".

Additionally, we created a second group of pairs, where the programs had to be exactly one week apart (i.e. on the next equal weekday). An example for such a pair would be the program "Die Simpsons" that ran on ProSieben at 16:40 on the 8th of June 2015 and at 16:40 on the 15th of June 2015. We found 10'374 of such pairs in the data. We call this set of pairs the "weekday pairs".

8.2.2 Results

To compare the stability of the audience flows we computed the Spearman rank correlation coefficient for each pair. We computed the Spearman rank coefficient for the top 10 channel flows between the incoming and outgoing audience flows of each pair. Using more than the top 10 channel flows tends to lead to very noisy data for programs with small audiences. We also think that expanding the number of channels would be of little use. For the programs we selected, on average 95.4% of the viewers are in one of the top 10 incoming channel flows and 95.0% are in one of the top 10 outgoing channel flows.

For each pair we also calculated the p-value of the Spearman rank correlation coefficient with the null hypothesis being that there is no correlation between the Spearman rank correlation coefficients of the audience flows. This was again done for the incoming and outgoing audience flows.

We will first look at the consecutive pairs of audience flows. Table 8.1 summarises how many pairs of consecutive audience flows had correlated Spearman rank correlation coefficients at significance level $\alpha = 0.05$. We see that more than half of the pairs of consecutive audienceflows had significantly correlated Spearman rank correlation coefficients. The difference in significant correlation between incoming and outgoing audience flows is very small.

Table 8.2 summarises how many weekday pairs of audience flows had Spearman rank correlation coefficients whose correlation was statistically significant at $\alpha = 0.05$. More pairs are significantly correlated when building pairs over weekdays compared to pairs with no restrictions. The difference between the incoming and outgoing audience flows is again negligible.

Those results show that the majority of the Spearman rank correlation coefficients is significantly correlated. In practice, this means that the viewers for a program series tend to come from a predictable set of channels. This is valuable information, for it could be used to not only predict the viewership of a channel (see chapter 9), but also the channel switching behaviour of the viewers. Such predicted audience flows could for example be used to place advertisements in such a way that viewers see an advertisement the optimal number of times.

Type	**Significant**	**Total**
Incoming	54.39% (7912)	14’508
Outgoing	53.60% (7777)	14’508

Table 8.1: Significance levels for consecutive audience flow pairs at $\alpha = 0.05$.

Type	**Significant**	**Total**
Incoming	60.30% (6256)	10’374
Outgoing	60.25% (6251)	10’374

Table 8.2: Significance levels for weekday audience flow pairs at $\alpha = 0.05$.

9

Viewership Prediction

The last feature of the OTV analytics system we present is viewership prediction which makes it possible to predict the number of viewers a program will have.

To our knowledge Danaher et al. (2011) offer the most recent work in viewership prediction and also present an overview of previous results in the field. They explain that viewership prediction is important because advertisement agencies buy the advertising slots up to six months in advance. The clients of the advertisement agencies are billed for the predicted number of contacts. If the predicted number of viewers is not reached, the broadcasters give free advertising slots to the advertisement agencies to compensate. On the other hand, if more viewers watch the show than predicted, the advertisement agency could have billed the client more or sold advertising slots to additional clients. In both cases money is lost.

To our knowledge no viewership prediction with OTV data has been published as of yet.

We will first show what data we used, and then explain what features we selected to use for the prediction. We will then explain the metrics used and present the different prediction models before showing the results.

9.1 Data

We were again given data by a Swiss OTV provider, the same as for all the previous analyses. We used data from June until mid September. Earlier data would have been available, but couldn't be used, since the way programs were categorised was significantly improved by this OTV provider in the summer of 2015.

We limited ourselves to only use sessions that were live because we wanted our results to be comparable to previous results in the literature, which were all done with live data. Additionally, the advertisement industry (still) bases their pricing on the number of live viewers a show has. Time shifted views are not considered, therefore predictions including both live and time-shifted viewership are of limited use.

Like Danaher et al. (2011) we limited ourselves to a number of channels. We selected the top 10 channels based on market share over all language regions in Switzerland for the period we had data for. Together those channels represent 41.8% of the total viewing duration on that OTV provider for the relevant time period. We think the

Channel	**Total Programs**	**Training Programs**	**Test Programs**
SRF-1	4475	2685	1790
SRF-2	3383	2029	1354
Pro-7	4341	2604	1737
M6	6018	3610	2408
TF1	7916	4749	3167
RTL	3042	1825	1217
RTL-2	2589	1553	1036
Sat-1	2960	1776	1184
Vox	3009	1805	1204
3+	2583	1549	1034

Table 9.1: Date set size per channel

selected channels offer a good mix of public and private channels. The channels are also diverse in content, ranging from news and movie channels to sports channels.

We used the same definition for what constitutes a viewer of a program as in chapter 8: To be counted as a viewer one must watch the program for more than 60 seconds.

The data for each channel was split into 60% training data on which the models were trained and 40% test data. The split was made chronologically, which means that all the programs in the training data were broadcasted before the programs in the test data. Table 9.1 shows the selected channels and the number of programs for each.

In total, we have used 40'316 programs. 24'185 in the training set and 16'131 in the test set. This is a large number of programs. Danaher et al. (2011) predicted the ratings for 5212 programs, which is already a large set compared to previous research.

9.2 Features

We wanted to predict how much viewers a certain program will have. Danaher et al. (2011) write that day of the week, time, month, program type and lead-in consistently appear as features. From the information we have on the programs (seen table 5.3) we could construct all those features, of which only lead-in needs additional explanation: A lead-in for program p is the program that runs directly before program p. Scheduling the correct lead-in for a program can increase the size of an audience of a program (Rust and Alpert, 1984). Using lead-in means using the number of viewers the previous show had as a feature.

Trying different sets of features we found the features in table 9.2 to perform the best. We will now explain why we added some features and omitted others that were used in the literature previously. We added title and episode because they improved our results.

Feature	Description
`Title`	Title of the show.
`Episode`	Episode of a show.
`Categories`	Type of content categories the program belongs to. Examples are "news" or "documentary".
`Start Hour`	Hour of the day at which the program started.
`Duration`	Duration of the program in seconds.
`Is series`	If the program is in a series. For example the evening news or a show like "Desperate Housewifes" are in a series.
`Day of the week`	The day of the week the program was broadcasted on.
`Month`	The month of the year the program was broadcasted on.
`Average viewers`	If the program is in a series, this is the number of viewers all the previous programs in this series got on average. If the program is not in a series, this is the average number of viewers that watch TV on this channel during the same hour and weekday the program is broadcasted. For each program, this feature is computed using only past data.

Table 9.2: Features used for prediction.

The feature "program type" is encoded as categories that are set by the data provider. The start hour, the day of the week and the month make it possible to deal with daily, weekly and seasonal viewing patterns. Duration was another feature that showed a high correlation with the label to be predicted, as was the "Is series" feature. Average viewers performed well as a feature also. We did not use the lead-in feature because, as Danaher et al. (2011) note as well, the lead-in for a show that lies a couple of months in the future might not be known. This means that this feature needs to be predicted as well, which adds a source of error. Danaher et al. (2011) also did not use it for this reason. Average viewers on the other hand can be computed for any show: One day of data from the same weekday a week prior is enough to calculate it. Since we assume our training set to be significantly larger than one week of data, this will always be the case for the large majority of programs. For the programs in the first week of the training set the number of average viewers is set to 0 for lack of a better value.

9.3 Metrics

To be able to compare the quality of a prediction model one needs to define an error metric. A well known metric in the field of machine learning for example is the root mean squared error. While this metric might be useful to tune a model, it is difficult to use it to compare different models in our use case. We are not interested in the absolute error, but in the error relative to the actual viewership. In the literature one metric that is often used is the mean absolute percentage error (MAPE) (Napoli, 2001). It is calculated as $\frac{|y_{true}-y_{pred}|}{y_{true}}$. Example 9.3.1 shows the results it produces. Its important properties are that it does work for over- and under-prediction and that it takes the true size of the program into account.

Example 9.3.1. Let us assume program p_1 has a viewership of 3000. If we predict it to have an audience of 2850 we can set $y_{true} = 3000$ and $y_{pred} = 2850$. This results in a MAPE of 0.05 or 5%. Now we assume another program p_2 has a viewership of 200. We predict it to to have 220 viewers. Therefore $y_{true} = 200$ and $y_{pred} = 220$ and the resulting MAPE is 0.1 or 10%. The prediction error for p_2 might be smaller in absolute terms than for p_1, but relative to the audience it is much larger. This is correctly reflected by the MAPE.

9.4 Models

In the literature different models are used and Danaher et al. (2011) give a complete overview. Most often linear regressions are used. To our knowledge, together with Danaher et al. (2011) this is the only study that evaluated multiple models on the same data set. We believe that the reason for this is that the bulk of the previous work has been done in the 80s and 90s, a time where it was much more difficult and time consuming to try different models in parallel than today, where faster hardware and programming libraries (e.g. Scikit-learn (Pedregosa et al., 2011)) make it more convenient.

The models we evaluated were Support Vector Regression, Linear Regression, Bayesian Ridge Regression and Decision Trees. Finding the best parameters for the models was done using cross-validation on the test set and with a scoring function that represented the MAPE.

9.4.1 One vs. Multiple Models

When predicting the viewership for programs on multiple channels, one has two different options to work with the training data. The first is to build one model and add the channel as a dummy variable. The predictions can then be done with that one global model. The second option is to split the training data by channel and train one model for each channel. For each program one needs to use the appropriate model to predict the viewership. Just as Danaher et al. (2011) we tried both and came to the same conclusion that using one model per channel beats using one global model. In our case using separate models decreased the MAPE by a quarter for our best model.

We believe the main advantage to be that the models can be tuned individually to each channel. For example we could use different maximal tree depths for each channel when using a Decision Tree. This is important, because as we will see in the results section, the prediction accuracy varies a lot by channel.

Model	**MAPE**
Support Vector Machine	0.4127
Linear Regression	0.5444
Bayesian Ridge Regression	0.6290
Decision Trees	0.2718

Table 9.3: MAPE for evaluated models prime time

Model	**MAPE**
Support Vector Machine	0.4216
Linear Regression	0.7756
Bayesian Ridge Regression	0.8038
Decision Trees	0.2863

Table 9.4: MAPE for evaluated models all-day

9.5 Results

We will now present our results for the different models. In keeping with previous studies, we limit us to predicting the viewership of programs in the prime time (18:00 to 23:00). The obtained results are presented in table 9.3.

We compare our results to Danaher et al. (2011) because their study is the most similar to ours. They also try to stay as close to the real world problem by keeping the challenging programs in their dataset (previous studies often removed programs that were broadcasted irregularly for example). Their best model has a MAPE of 0.3672, while our Decision Tree model reaches a lower MAPE of 0.2718. One must keep in mind though that those results are not directly comparable. For example their prediction horizon of up to six months was significantly longer than ours. For lack of more data our longest prediction horizon is 43 days. The prediction accuracy also depends on the quality of the data. For example, it might be that the categorisation of the programs is more accurate in one dataset than the other. But the fact that predictions with similar accuracy are possible with OTV and cable TV data is very interesting.

Table 9.4 shows the mean absolute percentage errors for the evaluated models over the whole day. We see that predicting programs that are not in the prime time is harder because the prediction accuracy decreases for all models, except for the Decision Tree which even slightly improves.

Another interesting insight is that the prediction accuracy is highly dependent on the channel. Table 9.5 shows the average MAPE by model and channel. SRF-2 has a higher error in all models compared to the other channels. This pattern can also be seen in Danaher et al. (2011), where the sports heavy channel has also a significantly higher error

	3+	M6	Pro-7	RTL	RTL-2	SAT.1	SRF-1	SRF-2	TF1	VOX
SVR	0.28	0.31	0.44	0.37	0.46	0.42	0.60	0.67	0.32	0.31
Lin.	0.39	0.28	0.30	0.25	0.29	0.67	0.64	2.70	0.25	0.33
Bay.	0.31	0.29	0.28	0.26	0.42	0.65	0.68	3.93	0.26	0.32
Dec.	0.25	0.22	0.23	0.17	0.28	0.27	0.27	0.90	0.18	0.18

Table 9.5: MAPE in prime time by channel and mode

Model	MAPE w/o outliers	MAPE
Support Vector Regression	0.66	0.67
Linear Regression	1.28	2.70
Bayesian Ridge Regression	3.03	3.93
Decision Trees	0.44	0.90

Table 9.6: Prime time MAPE SRF-2 with absolute percentage errors in the 95th percentile removed.

than the other channels. In our case the reason for the high errors on SRF-2 were for example the Tennis Wimbledon Finals, whose audience were up to 10 times larger than the average audience for that time of day. This leads to a very high absolute percentage error for a small number of shows that greatly influences the MAPE. If we define outliers as being predictions with an absolute percentage error in the 95th percentile and remove them, the error for SRF-2 is significantly smaller. For illustration purposes table 9.6 shows the MAPE for SRF-2 with outliers removed, compared to the MAPE without any correction. For the Linear Regression and the Decision Tree the MAPE is more than halved by removing the 95th percentile. The Support Vector Regression seems to offer a more consistent prediction accuracy over channels, as removing the outliers has very little effect. The MAPE of the Bayesian Ridge Regression model also profits from the removal of the outliers, but stays at a very high error level.

The selection of channels and percentile does however not change which model is best. The Decision Tree is consistently the best models independent of channels or outliers.

Napoli (2001) have calculated the MAPE of the predictions that advertising agencies make to be 21.35%. While this is a lower MAPE than our best model can achieve, one must again take into account the different circumstances. The predictions made by the advertising agencies were for only 140 programs on four channels compared to our 16'131 programs on 10 channels. But since the viewership prediction problem is of commercial interest, it is entirely possible that there are unpublished, proprietary models that could achieve a lower MAPE than ours using the same data.

10

Conclusions and Future Work

OTV makes it possible to gain valuable insights into what viewers like to watch. Treating this information like classic, panel-based TV data is not possible due to sheer volume and would also not be useful to unlock the additional knowledge in it. We have built a flexible OTV analytics system and have shown its capabilities by importing real-life data and running three different types of analyses on it.

The market analysis showed that the data generated by the system can be used to extract relevant, domain specific statistics over the imported data. We have shown that the system can take full advantage of the large set of data points that a source-based approach delivers.

The audience flow feature demonstrated the capabilities of the system to run non-trivial, highly specific analyses and display them using a custom web-interface. The audience flows themselves can be used on their own for broadcasters or advertising agencies to better understand the viewing patterns of viewers. Additionally, we showed that the majority of audience flows in a series have a stable ranking of the channels that make up the incoming and outgoing audience flows. The fact that the patterns are stable to a certain degree could be exploited in future work to build tools that optimize the placement of ads or programs.

The most complex feature of the OTV analytics system is the viewership prediction. We have shown that the accuracy of our results are comparable to previous academic work in viewership prediction on data from traditional TV. Comparison to this previous work is limited by our relative lack of data. While our model performs well compared to other academic works, we cannot say how it fares in comparison to unpublished, proprietary models used by advertising agencies. Nevertheless, we believe the fact that viewership prediction for OTV is possible with reasonable accuracy to be further proof that our proposed system is capable of delivering valuable insights into OTV data.

Advanced automatic anomaly detection is a feature that should be added to the OTV analytics system. While simple error detection already exists, more complex errors like systematic over- or under-reporting of a data provider need a dedicated anomaly detection functionality. By cross-validating traffic patterns amongst data providers one could profit from having multiple data-sources.

Time shifted viewing was only included in chapter 7. Since in the future companies might also pay for viewers that watched their ads time shifted, extending audienceflows and the viewership prediction with time shifted sessions might make sense. Even if this is of no commercial interest it would be interesting to see to what degree time-shifted viewing can be predicted.

References

Bundesamt für Statistik (2015). Raumgliederung. *http://www.bfs.admin.ch/bfs/portal/de/index/infothek/nomenklaturen/blank/blank/raum_glied/01.html* accessed on 04.09.2015.

Danaher, P. J., Dagger, T. S., and Smith, M. S. (2011). Forecasting television ratings. *International Journal of Forecasting*, 27(4):1215–1240.

Lamb, A., Fuller, M., Varadarajan, R., Tran, N., Vandier, B., Doshi, L., and Bear, C. (2012). The Vertica Analytic Database: C-Store 7 Years Later. *Proceedings of the VLDB Endowment,*, 5(12):1790–1801.

Mandese, J. (1995). THE BUYING & SELLING SUPPLY AND DEMAND, 'UPFRONT,' RATINGS POINTS, MEDIA FRAGMENTION: OH, HOW TV HAS CHANGED. *http://adage.com/article/news/buying-selling-supply-demand-upfront-ratings-points-media-fragmention-tv-changed/83412/* accessed on 01.09.2015.

Mediapulse (2015a). Annual reports. *http://www.mediapulse.ch/de/tv/publikationen/jahresbericht.html* accessed on 04.09.2015.

Mediapulse (2015b). Evaluation Software. *http://www.mediapulse.ch/en/tv/what-we-offer/evaluation-software.html* accessed on 08.09.2015.

Mediapulse (2015c). Jahresbericht Deutsche Schweiz 2014.

Mediapulse (2015d). Portrait Mediapulse. *http://www.mediapulse.ch/en/about-us/portrait.html* accessed on 08.09.2015.

Napoli, P. M. (2001). The Unpredictable Audience: An Exploratory Analysis of Forecasting Error for New Prime-Time Network Television Programs. *Journal of Advertising*, 30(2):53–60.

Narayanan, A. and Shmatikov, V. (2006). How To Break Anonymity of the Netflix Prize Dataset.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn : Machine Learning in Python. *Journal of Machine Learning Research*, 12(Oct):2825–2830.

Raeder, T., Stitelman, O., Dalessandro, B., Perlich, C., and Provost, F. (2012). Design principles of massive, robust prediction systems. *Proceedings of the 18th . . .*, pages 1357–1365.

Rust, R. T. and Alpert, M. I. (1984). An Audience Flow Model of Television Viewing Choice. *Marketing Science*, 3(2):113–124.

Rust, R. T. and Eechambadi, N. V. (1989). Scheduling network television programs: A heuristic audience flow approach to maximizing audience share. *Journal of Advertising*, 18(2):11–18.

Schweiz am Sonntag (2015). Neue Namen für SRG-Sender. *http://www.schweizamsonntag.ch/ressort/aktuell/1827/* accessed on 04.09.2015.

Stonebraker, M., Abadi, D., DeWitt, D. J., Madden, S., Paulson, E., Pavlo, A., and Rasin, A. (2010). MapReduce and parallel DBMSs. *Communications of the ACM*, 53(1):64.

List of Figures

7.1 Market shares for top-20 channels in German-speaking part of Switzerland. 21
7.2 Top 30 programs by rating for German-speaking Switzerland in February 2015 22
7.3 Gender distribution for top 20 channels in the French-speaking part. Pink: women, blue: men. 23
7.4 Age distribution for top 20 channels in Italian-speaking part of Switzerland. 24
7.5 Device distribution for selected shows 25
7.6 Distribution of live vs. time-shifted viewing for selected shows 26

8.1 Audience flow for a broadcast of the "Tagesschau" displaying the top 10 channels. 30

List of Tables

5.1 Attributes of the session data. 10
5.2 Attributes of the user data. 10
5.3 Attributes of the EPG data. 11

7.1 Example sessions for calculating market share 20
7.2 Example sessions for calculating rating 21

8.1 Significance levels for consecutive audience flow pairs at $\alpha = 0.05$. 34
8.2 Significance levels for weekday audience flow pairs at $\alpha = 0.05$. 34

9.1 Date set size per channel 36
9.2 Features used for prediction. 37
9.3 MAPE for evaluated models prime time 39
9.4 MAPE for evaluated models all-day 39
9.5 MAPE in prime time by channel and mode 40
9.6 Prime time MAPE SRF-2 with absolute percentage errors in the 95th percentile removed. 40