University of Groningen

Master Thesis

Classifying Political YouTube Channels With Semi-Supervised Learning

Author:
Anton Laukemper
(s3779572)

Supervisors:
Prof. Dr. Michael Biehl
Prof. Dr. Herbert Jaeger

August 14, 2020

Abstract

Since many investigative journalists accused YouTube's video recommendation system of recommending radical political and conspiratorial content, the inner workings of this algorithm have become the focus of public scientific research. Multiple studies were carried out to analyse the bias of the recommendation engine with conflicting results. An important drawback of all of these studies was that the core feature of the algorithm - personalized recommendations - was not investigated.

The overall aim of this Master project was to design an experiment that would provide a dataset of personalized video recommendations with which a potential bias could be detected. The large time cost of manually classifying such a dataset necessitates algorithmic classification. Due to the easy availability of unlabeled data, *semi-supervised learning* appears to be an attractive approach for this task. The computational research question to be addressed in this thesis was therefore whether unlabeled data would improve the performance of machine learning classifiers if used in a semi-supervised learning algorithm.

To this end, multiple classifiers were trained in a supervised manner for a binary classification task, distinguishing political from nonpolitical YouTube channels, and a multi-label classification task, classifying channels according to a number of political categories. These classifiers were compared to classifiers trained using the semi-supervised *self-training* algorithm.

The results show that self-training did not make effective use of the unlabeled data, as almost all classifiers trained with it performed significantly worse. Nevertheless, the classifier trained only on labeled data was able to distinguish political from nonpolitical channels with an accuracy of 96%, while it achieved a mean precision of 63% and a mean recall of 53% in the multi-label classification task.

Acknowledgement

I would like to express immense appreciation to my supervisors Prof. Michael Biehl and Prof. Herbert Jaeger for allowing me to pursue my own project idea. Especially for all the conversations and discussions in which they sorted my initially quite disorganized thoughts, I am extremely grateful. I would like to stress that without their influence and guidance along the way, this project would have never been realized. Another person, without whom this work would not have been possible, is Mark Ledwich, who supported me from the very beginning with data collection, the experiment design, and who provided the fundamental data on which this project is built. I am glad that we got to cooperate on this project and I am looking forward to continuing it together. I wish to acknowledge the help provided by my dear friends Thorsten, Lukas, Phillip and Leah who read drafts, commented on my work and contributed invaluable feedback. Special thanks I would like to extend to Claudia Limbrock for offering me a place to focus and to be creative during a time when universities and libraries where not accessible. Her co-working space provided me with a unique environment which not only contributed to the quality of this work, but also made the whole writing process an enjoyable experience.

Contents

List of Figures

List of Tables

1 Introduction

Since its launch in 2005, YouTube has become the largest video platform in the world with over 720,000 hours of video material uploaded and 1 billion hours watched every day[1]. A likely reason for its success is the recommendation engine that proposes to viewers a set of videos they might want to watch next. YouTube itself claims that more than 70% of their entire watch time is generated by their recommendation system[2], showing how actively this system influences the users' viewing behaviour.
Youtube is also used as a political platform where political content is uploaded by election campaign organizers, established news channels and media corporations, but also by private or semi-professional content creators (so-called *YouTubers*). Given the large reach of political messages on YouTube, the platform has become a considerable factor in elections and other political decision processes [1, 2].
In 2018, The Guardian published an article in which the first allegations against the political neutrality of YouTube and its recommendation system were made [3]. It was claimed that the company actively helps the spread of misinformation and extremist political content through the way they recommend it in their 'recommended videos' section next to the video player. Furthermore, this recommendation system supposedly has a radicalizing effect on some of its users [4]. Since then, YouTube has been under constant scrutiny by journalists and scholars, pressuring YouTube spokespersons to promise improvements of the algorithm multiple times[3].
Researchers outside of Google have conducted studies to scientifically approach the question of whether the algorithm might be biased towards right-wing content. Ribeiro et al. [5] claimed that there is indeed a pathway from moderate political content to radical right-wing content. Ledwich and Zaitsev [6] found the opposite and discovered that there are more recommendations from radical content to moderate content than the other way around. Faddoul et al. [7] report in their publication about a short term decrease in recommendations to conspiratorial fringe content after YouTube implemented changes to their system, but later a return to the trend of a growing number of conspiracy video recommendations.
These studies all have a common drawback in their methodology: In none of the experiments were the experimenters logged in with a YouTube account,

[1]`www.youtube.com/intl/en-GB/about/press/` (accessed: 24.07.2020)

[2]`www.cnet.com/news/youtube-ces-2018-neal-mohan/` (accessed: 24.07.2020)

[3]`foundation.mozilla.org/en/blog/congratulations-youtube-now-show-your-work/` (accessed: 24.07.2020)

which means that they can only make claims about the non-personalized[4] recommendation system. However, as over three quarters of internet users say that they have a YouTube account and YouTube has over two billion users that log in at least once per month[5], one can assume that a large proportion of users, especially those that use YouTube often, experience the website while being logged in. All of the users who are logged in while watching videos, receive personalized recommendations that are, among other factors, based on their watch history.
Understanding the non-personalized recommendation engine of YouTube is helpful but clearly not sufficient to get full insight into how the mechanisms of the video platform might facilitate political radicalization. With regards to how the personalized recommendation system could contribute to this effect, the main question is whether any bias exists in the system, such that some content is recommended disproportionately often.
This Master thesis project's goal is the creation of a dataset that can offer answers to this question. To this end, an experiment was designed and implemented that gathers information about personalized video recommendations. A challenge for a quantitative analysis of such a dataset is how the recommended videos are coded. If the question is whether the algorithm has (intentionally or unintentionally) learned to serve certain kinds of political videos more often than others, the collected recommendations need to be classified according to political categories. Only then can quantitative statements be made about the data. Manually labeling YouTube videos or channels is an extremely time-consuming process, as hours of video material would need to be sighted. Machine learning based classifiers that automatically label the data are therefore needed to analyse sufficiently large datasets.
In this Master project, two machine learning methods are compared: *supervised learning*, in which a classifier learns patterns from a set of human-labeled examples and applies the gained information to unseen data, and *semi-supervised learning* (SSL), in which additionally unlabeled examples are used in the training process.
Semi-supervised learning is often advantageous in situations where manually labeling examples is very cost-intensive but where unlabeled data is readily available. Analyzing YouTube videos channels fits this description perfectly. Neither supervised nor semi-supervised learning works without a set of la-

[4] Non-personalized means here, that the recommendations are not based on previously watched videos. They are still based on location data and similar features, as explained in Section 2.2.

[5] `www.oberlo.com/blog/youtube-statistics` (accessed: 24.07.2020)

beled data from which inferences can be made. The original labeled dataset that was used in this work stems from a study by Mark Ledwich and Anna Zaitsev from the end of 2019 [6]. In this work, the authors categorize YouTube's own political micro-cosmos and label over 800 channels according to a set of political ideologies and other characteristics. Based on this data, I decided to also classify channels instead of individual videos.

The classification of channels happened in this project in two stages. First, they were categorized as either *political* or *nonpolitical*, where a classification as political means that the channel frequently comments on US-related political or cultural issues [6]. All political channels were then further labeled with 18 different tags, denoting their ideological orientation.

While this project aims to contribute to answering the question of whether the recommendation system might be biased and possibly facilitates radicalization processes, the actual analysis of the experiment's results is not in the scope of this project. Its main contribution is the collection of empirical data and, in particular, an experimental comparison of different classification approaches.

In summary, the two main objectives of this project are (i) the creation of an experiment that generates an unlabelled dataset about personalized video recommendations from YouTube, and (ii) a comparison of two machine learning approaches that results in a trained classifier with which the gathered data can be labelled.

The thesis is structured as follows: The Theoretical Background section of this thesis introduces general information about recommendation systems, in particular the YouTube algorithm, as well as relevant concepts from supervised and semi-supervised learning. In the Related Works section I present the current state of research on the YouTube recommendation system and I shortly review literature on methods used in this project. A description of the experiment design, data collection and how the classifiers were trained and compared follows in the Methods section. Visualizations from the exploration of the data and the outcome of the classifier comparison I present in the Results. In the Discussion I argue why the unlabeled data did not improve the supervised classifiers, as the experiment results show, and the Conclusion gives a summary of the project.

2 Theoretical Background

To put the experiments and methods described in this report in context, I give necessary background knowledge in this section. Different approaches regarding recommendation systems and a description of the specific algorithm employed by YouTube are presented here. The second half of the section offers an overview of important concepts from supervised and semi-supervised learning.

2.1 Recommendation Systems

Recommended items on commercial web services have become a ubiquitous sight. Whether it is related news articles, products other people have bought, movies or videos one might also like - recommendation systems (RSs) are fundamental features of many service providers. The goal of an RS is to predict how positively a user rates an item, and to present them a list of potentially ranked suggestions of items [8]. While there are recommenders that are not personalized, and e.g. just query the most popular items at the moment and present them to everyone, the most frequent version of recommenders are those that present every user different, personalized recommendations [9, 8]. For these recommenders, it is essential to create a user-model that reflects the preferences of every user, either by *explicitly* gathering feedback on presented items through ratings [10] or by *implicitly* inferring the preferences from the user's behaviour, by e.g. monitoring the time an item was viewed [11].
In the literature, different taxonomies of recommender algorithms types have emerged from which the two most important classes are *collaborative filtering* and *content-based* recommenders [12, 8, 9].
The idea to simply suggest items that other users with similar interests liked in the past, was the first implementation of a recommendation system and is called *collaborative filtering* [13]. For this technique, the most similar user-models to the current user are found and the recommender suggests items that these users have liked, but the current user hasn't seen yet. Collaborative filtering has certain disadvantages [14]: Problematic for this approach is the beginning phase in which little or no information about users' history exists yet ("cold start problem"[15]). New or unpopular items can also not be handled well. In large web shops that offer millions of products, some items have never been bought by any customers, so they cannot be recommended by collaborative filtering system. Additionally, these systems are hard to scale if the service has millions of users.

From the fields of information retrieval and information filtering emerged *content-based* RSs as an alternative approach [9]. These recommenders suggest items that are similar to items the users has liked in the past and therefore each item's features and attributes must be modeled in addition to the users' profiles. The advantage of this approach is that also niche content or completely new content can be recommended, even if no other user has ever bought, seen or interacted with it. A limitation of content-based recommenders is that they cannot expand on the users' interests, because they are restricted to preferences shown in the past. An item that is very different from previous liked items, which the user nevertheless might still like, is not recommended by a content-based recommender.

Evaluation Evaluating a recommendation system is not a trivial task, as there is usually not a single goal that the system is optimized for. As Shani and Gunawardana [16] lay it out, "it is now widely agreed that accurate predictions are crucial but insufficient to deploy a good recommendation engine".
They further list a number of characteristics that are considered to be relevant for successful RS [16]: Next to the *accuracy* of the user rating predictions, properties like the *coverage* are important. It measures the number of items that the RS is able to recommend from the catalogue of all available items. For some applications, *serendipity*, *novelty* and *diversity* are major factors, while they are less relevant for others.
Serendipity measures how surprising a successful recommendation is which is operationalized with a distance metric that captures the dissimilarity between items in the catalogue. This dissimilarity is also maximized when the diversity of the recommendations is the goal. Novelty, which describes the unfamiliarity of users with the recommendation, can be operationalized in various ways, but a simple one is to assume that users are familiar with popular items. When novelty is of high importance, the goal is thus to recommend unpopular items more often.
While these are all characteristics of RSs that are often desired by the stakeholders, some effects of recommendation systems can be perceived as detrimental. The following paragraphs summarize a paper by Jiang et al. about so-called "degenerate" recommendation systems [17].

Degenerate Recommendation Systems Due to the inherent nature of RSs to not recommend a representative view of the full content but only a slice of it that is predicted to be preferred by the user, some people claim that

RSs create ‘echo chambers’ or that users get stuck in ‘filter bubbles’ which is usually implied to be something negative [18]. The goals that platform stakeholder pursue with an RS do not necessarily align with the interests of the users and even if, certain effects of an RS can occur unintentionally.
The authors of [17] wanted to find out what constitutes a so-called ‘degenerate’ recommendation system. Under ‘degeneracy’ they understand the tendency to create echo chambers and filter bubbles. These effects are often mixed together or treated as synonyms but Jiang et al. see them as two separate effects that, nevertheless, often occur at the same time. If a “user’s interest [is] being positively or negatively reinforced by repeated exposure to a certain item or category of items” [17] they are subject to the echo chamber effect, while a filter bubble simply describes “the fact that recommender systems select limited content to serve users online” [17].
The authors model in their study a recommender system that interacts with a user over time. The system serves every time step t a set of items a out of a countably infinite set M. The user has a function $\mu_t : M \to R$ which represents her interests, and is a large positive number if the item set presented matches her interests and small or negative if not. At each time step the user gives feedback to the recommender system by interacting with the item set. In the case of YouTube that interaction could be clicking on a video and watching it for a certain amount of time or engaging with it in other ways (liking, commenting etc.). The recommendation system has an internal model which tries to predict the user’s interest and is updated based on her feedback. However, also the user’s interest function μ_t changes over time and is dependent on the previous interest state μ_{t-1}, the recommended item set a_{t-1} and also the previous interaction with that itemset, which is denoted as c_{t-1}.
The authors claim that a characteristic of every recommender system is that it is ‘degenerate’, i.e. the interest of the user at a later point in time μ_t is arbitrarily different to the initial interest μ_0. From their experiments, however, they conclude that the speed of this interest shift varies strongly among the different recommender types they tested.

2.2 YouTube Recommendation Algorithm

A common way of circumventing the disadvantages of collaborative filtering and content-based filtering are *hybrid* systems that combine the strengths of multiple algorithms [12]. The algorithm YouTube uses to provide their video recommendations is a good example for such a system and is shortly presented here. YouTube has been constantly modifying their recommendation system, and while some publications give insight into its inner workings on the surface level, the details of the system remain hidden to the public. It should be noted that even if YouTube had no financial interest in keeping their algorithm closed-source, releasing the exact functionality of it would likely open the door for adversarial attacks and spammers, abusing the system to push their content.

While the earliest version of their algorithm was based on association rule mining [19], YouTube revealed with the publication of [20] a switch to deep learning for generating video recommendations. The new system consisted of two neural networks, one for generating a candidate set of a few hundred relevant examples from millions of videos, and another one to rank the candidates such that the top dozens most relevant candidates can be presented to the user.

The candidate selection network uses features such as the watch history, search history, geographic features, gender, and age and uses implicit user feedback, e.g. watching a video to the end, to create user embeddings. Based on the similarity of the user embeddings, a collaborative filtering approach selects the set of candidate videos.

The second neural network responsible for ranking the candidates has more features available because it does not need to be scaled to millions of videos. In this phase, the users' previous interactions with a candidate video or the corresponding channel are, among other features, taken into account to determine its ranking. The neural network model predicts the watch time of each candidate and ranks them accordingly. This part of the recommendation system embodies a content-based approach.

In 2019, the latest update[6] on the algorithm was released which addressed certain challenges that were previously left neglected [21].

One problem that many recommendation systems have is a selection bias in the training data. When a user clicks on a recommended video, thus providing positive feedback to the recommender, it might not have happened because she liked it the most, but simply because the recommender ranked it the highest. Recommender systems are thus prone to fall into feedback loops.

A second problem is the different optimization objectives for the algorithm, that are not necessarily aligned. For example, YouTube might want to recommend more videos that the user watches to the end, or they might want to recommend more videos that the user is most likely to share with

[6]YouTube has at the time of writing not yet officially announced that this version is currently used.

friends, etc. These are not necessarily the same videos and an RS that is able to recommend more of one kind potentially recommends less of the other.
To address these challenges, YouTube updated mostly the second phase of the recommendation system - the ranking algorithm. Multiple predictions on different metrics were combined in a so-called “Multi-gate Mixture of Experts architecture”. To circumvent the feedback loop problem, a technique was used that feeds the position of a training example in the ranking as a feature into the neural network. While this method improved the engagement with the recommended videos, the authors admit that this problem remains an open question in research.
The architectures described by YouTube were evaluated offline, i.e. without actual user interaction, by using metrics such as the squared error to determine how well the system predicts the users' preferences. Additionally, the algorithms were evaluated in live A/B testing through user surveys and monitoring of user behaviour [21]. In A/B testing, a subset of users is presented with the new RS, making it possible to compare the new architecture to the previous architecture.

2.3 Supervised Learning

At the core of this project is the classification of a set of YouTube channels with machine learning, since manual classification would be too time-intensive. Because a dataset with manually labeled channels already exists, the problem at hand is a classical case for *Supervised Learning* (SL). This section will explain the fundamentals of SL and introduce the theoretical background necessary to understand the methods used in this project.
Given a set of items and a set of categories according to which the items are supposed to be classified, the goal of SL is to find a function that maps an item to a category. A necessary requirement for this is that some of the items, also called examples, are already assigned to the categories. From this *training data* an SL algorithm ideally generalizes a function that is also able to accurately map unseen data points. How well this function, also called a model, predicts the class of unseen examples can be estimated on a test set. The test set consists of examples for which the classes are known, but which are not included in the training of the model.

Logistic Regression A classical tool for supervised classification tasks is the *logistic regression.* With a standard linear regression [22], a linear relationship between a feature variable and a target variable can be described in the form

$$Y = \alpha + \beta X. \tag{1}$$

Logistic Regression is used when the target variable is categorical. More precisely, for simple logistic regression, the target variable has only two values. Otherwise, when the target variable can have one out of multiple values, *multinomial logistic regression* is used. Such a classification task is also called *multi-class classification.*
A logistic regression model is trained by minimizing the cost function shown in Equation 2 [23].

$$\min_{w,c} \frac{1}{2} w^T w + C \sum_{i=1}^{n} log(e^{-y_i(X_i^T w + c)} + 1) \tag{2}$$

The term $\frac{1}{2}w^T w + C$ is a regularization term that prevents the model from overfitting, i.e. failing to generalize from the training set to new data. It is controlled by the parameter C.

Multi-Label Classification While each instance in logistic regression has one out of two target variables (binary) and in multinomial logistic regression one out of many (multi-class), multi-label classification describes tasks in which an instance can belong to up to n classes. The output of such a classifier is thus a binary vector $V \in \{0,1\}^n$ such that

$$V_i = \begin{cases} 1, & \text{if the input is assigned to label } i \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Multi-label classifiers can also be realized with algorithms intended for binary decision problems such as logistic regression by training n classifiers, one for each class. Each classifier learns to discriminate between the instances that belong to that class and all other instances. This way, the i-th classifier outputs the probability that the input belongs to class i.
In algorithms such as multinomial logistic regression the output probability for each class can be directly calculated through the *softmax* function shown in Equation 4 [24].

$$P(y = i|x) = \frac{e^x}{\sum_{k=1}^{K} e^x} \tag{4}$$

Since this function normalizes the output vector such that the sum of the probabilities equals 1, it is typically used for multi-class problems. In some applications, it is also used for multi-label classification tasks by using a threshold value [25].

Evaluation To evaluate a binary classifier, different metrics can be used. If the goal is e.g. to determine whether a tumor is malignant or not, one is typically interested in how many of the cases that the classifier identified as malignant were truly malignant (*precision*), and how many of the malignant cases were identified as malignant by the classifier (*recall*). Usually, there is a trade-off involved between these two metrics. Equation 5 shows the *F1-score*, a metric that combines both precision and recall in one metric, by calculating their harmonic mean.

$$F1 = \frac{true\,positives}{true\,positives + \frac{1}{2}(false\,positives + false\,negatives)} \tag{5}$$

A metric like the F1-score can only evaluate the model for a specific classification threshold. If this threshold is e.g. 0.6, all examples for which the model predicts that they belong to class A with a probability of 0.6 or higher are classified as A. To make a statement about the general quality of a model, independent of this threshold, one can plot the *Receiver Operating Characteristic (ROC) curve* and compute the *area under the curve (AUC)* [26]. The ROC curve shows the recall and *false positive rate*, i.e. the probability of false alarms, for every possible threshold. The AUC then summarizes the information from this curve into one metric, so that different models can be easily compared.
In multi-label classification tasks, even more metrics exist because there are many more ways in which a classifier's prediction can be considered correct. Consider an image recognition task where the goal is to identify all items on an image of a table with a glass of water and a plate. The choice of performance metric determines whether a classifier that recognizes only a glass of water is considered more correct than a classifier that recognizes a glass, plate and additionally a banana with high confidence.
One metric that is employed in determining the quality of recommender systems but also in multi-label classification tasks related to the one in this project [27, 28] is the *Mean Average Precision* (MAP). For a given prediction, the predicted labels are ordered by probability and the precision and recall at each position are calculated. The average of these values (average precision (AP)) is then calculated according to Equation 6 [23].

$$AP = \sum_n (R_n - R_{n-1}) P_n \tag{6}$$

Calculating the mean of the APs for a whole set of predictions results in the MAP.

Text Classification A specific domain for supervised classification is the classification of natural language texts. The entities that are to be categorized can be short snippets of text or whole articles, but each entity is called a 'document'. To make a machine learning algorithm able to process text it needs to be represented in vector form in some way. Two common approaches for this vectorization exist, namely *bag-of-words* representation and *word embeddings*.

In the bag-of-words approach, the order of words is disregarded and texts are viewed as a collection of words. Which words appear in the text with which frequency are the only features used by the algorithm to discern between the documents. For this purpose, the vocabulary of the whole set of relevant documents is determined and each document is represented as a vector with the length of this vocabulary. The element i in the vector signifies whether or how often the word at the i-th position in the vocabulary appears in the document.

An alternative approach to document vectorization is to represent each word as a vector in an arbitrarily sized feature space and combine these vectors, so-called word embeddings, to represent a document. The idea behind embeddings is that words with similar meaning also correspond to vectors close to each other in the feature space [29]. To generate these embeddings, neural networks are trained on large sets of documents to learn which words often occur in the same contexts [29].

Relevant for both methods is a proper preprocessing of the text. Natural language contains a lot of signals that do not contain any discriminative value. Punctuation, special symbols and the most common words, so called *stop words* are usually removed from the text before it is vectorized. The goal of such preprocessing is to keep the dimensionality of the vectors low. For this purpose, words are often also *lemmatized* which means that different grammatical inflections of one word are mapped to the same word [30]. After a text is processed with such methods, it is split into different *tokens* that each represent one term, that can then be vectorized easily. Terms do not necessarily have to be single words but could also refer to proper names like New York.

2.4 Semi-Supervised Learning

While supervised learning is used in situations where labeled training data is available, and unsupervised learning covers scenarios where this is *not* the case, semi-supervised learning can be placed somewhere in between. In many situations, it is costly to assign classes to some examples but cheap to gather many unlabeled examples. The goal of semi-supervised learning is to extend supervised learning methods by extracting additional information from the unlabeled data.

The information one can gain from the unlabeled data comes from its distribution. Given $P(x)$ as the probability of encountering a certain example x and the class conditional probability $P(x|y)$ which expresses the probability to encounter an example x given that we know it belongs to class y, semi-supervised learning helps in situations where the assumption about the link between $P(x)$ and $P(x|y)$ is correct [31].

Every semi-supervised method makes its own assumptions about this connection. In a binary classification task, a Gaussian Mixture Model, for example, would assume that the data comes from two Gaussian distributions. More training data, even if not labeled, makes the estimations of the distribution parameters more accurate. Not necessarily, however, if the data points actually stem from a totally different distribution.

There are three assumptions of which at least one is used by every SSL algorithm [32]. The *semi-supervised smoothness assumption* states that two points that are close together in a high-density region, belong to the same class. A more specific version of this assumption is the *cluster assumption* which postulates that two points in the same cluster are likely belonging to the same class. Lastly, the *manifold assumption* states that the high-dimensional data is likely to lie on a subspace (manifold) with a lower dimension.

Two different settings of SSL with different goals exist [31]. In *inductive learning*, the goal is, similar to standard supervised learning, to train a classifier on training data, and use it to predict unseen examples, hoping that it generalizes well enough. The difference is then simply that the training data also includes unlabeled examples. *Transductive learning*, is the easier task, because the goal is to use the classifier only on the unlabeled data in the training sample.

Further, some SSL methods can be described as *wrapper* methods because they can internally use any classifier, provided it can output the probability $P(y|x)$ as a prediction for a data point or some other kind of confidence score [31]. Examples for such wrapper methods are *self-training* and *co-*

training. Self-Training was chosen as the SSL technique in focus for this project because it is the most simple SSL method and its compatibility with other classifiers allows for an intuitive comparison with supervised learning. Details about this approach are therefore explained here.
The basic idea of Self-Training is that a given classifier teaches themself, by iteratively using its own predictions for further training epochs [31].
Algorithm 1 shows the pseudo-code of the algorithm as seen in [31]:

Algorithm 1 Self Training Algorithm

```
1: $Input$ : labeled data $\{(x_i, y_i)\}_{i=1}^{l}$, unlabeled data $\{x_j\}_{j=l+1}^{l+u}$
2: initially, let $L = \{(x_i, y_i\}_{i=1}^{l}$ and $U = \{x_j\}_{j=l+1}^{l+u}$
3: repeat
4:     Train $f$ from $L$ using supervised learning.
5:     Apply $f$ to the unlabeled instances in $U$.
6:     Remove a subset $S$ from $U$; add $\{(x, f(x)|x \in S\}$ to $L$
```

The subset S contains the predictions with the highest confidence. The assumption that underlies the use of self-training is that these high-confidence predictions tend to be correct. Depending on which classifier is used inside of the self-training wrapper, the semi-supervised smoothness assumption hides implicitly in that assumption.
Possible problems of the self-training algorithm are initial mistakes in the prediction that are amplified by further iterations [31]. Since no convergence criterion exists, the training of the wrapper algorithm ends with a custom stopping criterion. It can e.g. run for a fixed number of iterations, until all unlabeled examples are part of the training set, or until the confidence score of the predictions falls below a certain threshold.

3 Related Works

To provide an overview of the current state of research related to YouTube, classification of content on the platform, and how self-training performs in practical tests, this section summarizes a selection of relevant literature.

3.1 Studies on Radicalization on YouTube

After the initial accusations [3] by a former employee of Youtube in the beginning of 2018, numerous articles emerged in newspapers and media outlets that claimed the YouTube recommendation system facilitates radicalization [4], spreads conspiracy theories [33] and extremist content [34].
Various researchers also investigated YouTube and the recommendation algorithm in scientific studies regarding similar claims. Rebecca Lewis analyzed from January 2017 to April 2018 connections between channels of the so-called "Alternative Influence Network" [35]. Through a snowball method, in which connections from a set of seed channels were followed, 88 influential channels were discovered that had a "general opposition to feminism, social justice or left-wing politics" [35] in common. In this study, a connection is present between two channels if the host of channel A has a guest appearances in a video of channel B. The study concludes that there are two main radicalization effects. First, the viewers of a moderate channel are exposed to more extreme positions through e.g. interviews with more radical figures of the scene. Secondly, the influencers themselves radicalize as well through interactions with their audience and their demand for more extreme content. Lewis claims that this radicalization is a social and not a technical problem and that even without any recommendation algorithm these two effects would still exist.
Ribeiro et al, conducted in 2019 a study in which the user intersection and user migration between politically moderate and extremist groups were analyzed [5]. Additionally, they examined non-personalized YouTube recommendations by following the recommendations on a set of seed videos from each category and checking whether this path led to more or less radical videos. Based on their results, they assert that, even if not very high, there is a flow from mild to extreme content.
In October 2019, Munger and Phillips published a new theory on radicalization on YouTube in which the recommendation algorithm does not play a central role [36]. According to their "Supply and Demand" theory, videos on YouTube did not radicalize the audience, but the radical content on YouTube was created to satisfy the demands of an already extremist audience that

was previously not met by classical media. They underline this theory with quantitative data that shows how search queries regarding niche topics show much more results from radical channels than mainstream media.
Ledwich and Zaitsev also disagree with the claim that the YouTube algorithm leads users from moderate to fringe content, based on the results from a study that gathers non-personalized recommendations from and to multiple political channels [6]. When comparing (i) the number of recommendations that point to videos with extreme content and (ii) the number of recommendations that lead away from them, they found that, on average, the YouTube algorithm tends to rather divert traffic towards videos from mainstream media than towards radical fringe content.
Lastly, a study co-authored by the aforementioned former YouTube employee was published in March 2020 and examined how many videos spreading conspiracy theories and misinformation were recommended by YouTube's algorithm [7]. They tracked the proportion of these videos among all recommendations they gathered over time and found out that, indeed, YouTube's efforts to reduce misinformation were partly successful but 2% of the recommendations still promoted conspiracy theories and misleading information.
In summary, there is neither consensus yet on how the recommendation algorithm contributes to the radicalization of YouTube consumers, nor is there consensus on whether such an effect exists in the first place. While the goal of this project is not to prove the presence of any kind of radicalization mechanism, it aims to contribute to this research by examining whether the potential for such a mechanism lies in the functioning of the algorithm.

3.2 Classification of Videos and YouTube Channels

This subsection gives an overview of different approaches to the classification of YouTube videos and channels in the literature. Starting with the classification into more general categories, it finishes with multiple studies that use political categories.
YouTube itself is annotating videos and channels with semantic topics [37, 38, 39]. Their approach for channel classification as described in [37] does not make use of the image or audio content, but rather uses meta-data such as titles, description and keywords. Furthermore, features from the user behaviour, such as search queries and which videos were watched in the same session, are processed.
With a similar approach, they created in 2016 the *YouTube-8M* dataset consisting of over 8 million YouTube videos, annotated with 4800 classes [38]. The original version of this dataset has since then been deprecated and

updated[7]. This dataset has led researches to combine features from different modalities [28]. With deep learning models, *MAP* scores of over 0.85 were achieved [27, 28].
Regarding the explicit political classification of videos, a very salient research focus is the analysis of political advertisement videos. In [40], different features extracted from the image content of video ads shown on YouTube are used to classify videos as political or nonpolitical. Different machine learning classifiers were used by [41] to perform automatic coding of political ads according to variables such as the mood of background music, whether opponents appeared in the video, or which political issues were addressed.
With a background in media bias research, Dinkov et al. classify the YouTube channels of news media according to their left-center-right bias [42]. From existing labels for different media companies and their corresponding YouTube channels, they created a dataset of 421 channels and 3345 videos. Multiple features were extracted from the video captions, video snippets (title, description, tags), audio information and video metadata (views, likes, etc.) to be combined in a deep learning model. With the three classes *left*, *center* and *right* their model achieved an accuracy of 73.42%.
Recently, [7] aimed to identify conspiracy theories and falsehoods in YouTube videos. For their approach, they combined multiple classifiers each trained on a separate feature set into one classifier. The feature sets consisted of the video caption, video snippet, the 200 top comments and the perceived impact of comments ('toxic', 'spam', 'unsubstantial', etc.) classified by Google's Perspective API[8]. The resulting ensemble was able to identify conspiratorial videos with a precision of 78% and recall of 86%.
It can be seen that various methods and features were successfully used to classify YouTube videos and channels. Even though deep learning models performed excellently in previous studies, this technique is not well suited for the goal of this project due to the small size of the training data. Using features from audio or image data was also disregarded, because of the cost to collect this feature data and because I did not expect these features to carry a large amount of additional information with regards to the task at hand.

3.3 Self-Training - Empirical Evaluations

The self-training algorithm has been used in various practical investigations and comparison studies. Two relevant studies are highlighted in this sub-

[7] `research.google.com/youtube8m/download.html`

[8] `perspectiveapi.com` (accessed: 24.07.2020)

section to show in which situations self-training has worked well and when other methods are superior.

A survey and comparative study on multiple SSL algorithms was published by Da Silva et al. [43]. The domain on which the classifiers were compared was the sentiment analysis of tweets. The sentiment analysis was treated as a classification problem of *positive*, *neutral* and *negative* sentiments.

As a stopping criterion for their self-training algorithm, the authors selected a confidence threshold so that the algorithm terminated if no predictions on the unlabeled examples exceeded this threshold. In their experiment, they varied the number of initially labeled examples and compared the F1-score on the testing set. Multiple datasets were used and additionally multiple threshold values were evaluated for the self-trainer algorithm.

Their results show that when only little labeled data is available, the classifiers trained with a lower threshold performed better on some datasets. Overall, however, a higher threshold led to better results. Compared with the other SSL wrapper method *co-training*, self-training performed better when more labeled data was available

Researchers Didaci and Roli compared how self-training and co-training perform in the context of ensemble classifiers [44]. For this purpose, they used different classifier algorithms inside each wrapper method and checked how well the ensemble operates on different datasets.

Two stopping criteria were used for the wrapper algorithm. In both cases, the number of examples added to the training set in each iteration was fixed. For large datasets, they used a number of iterations set to 2/3 of the maximally possible iterations. For small datasets, they ran the algorithm until all of the unlabeled data was added to the training set. After each iteration of the self-training algorithm, the error of the classifier on the test set was calculated.

For all classifiers the authors compared, a non-monotonic decrease in the error could be observed. This means that adding unlabeled data to the training set first increased the performance of the ensemble classifier, but later decreased it again. Nevertheless, most classifiers performed better in the end than before the first iteration, when it was only trained on labeled data.

From these two studies, some limits to self-training but also its strengths can be seen. The study by [43] proved that self-training can work in the context of text classification, while [44] showed that ensemble classifiers can benefit from unlabeled data.

4 Methods

This section describes the data that was available at the beginning of the project, the methods used to gather more data, and how the data was processed for classification. The section ends with an explanation of the classification algorithms and the methodology of how they were compared.

4.1 Dataset Description

The dataset by [6] contains 803 political channels with so-called 'hard tags' and 'soft tags'. Hard tags are labels that were given to the channel if they appeared in studies such as [5], if they were recognized as official news outlet or if the channel belonged to an existing news network. In general, hard tags are labels given from external sources. In contrast, the soft tags are labels given by the authors and their co-labeller. They were created to capture the specific context of YouTube because "traditional ways of dividing politics are not natural categories that would accurately describe the politics of YouTube channels" [6].
Table 1 lists the 18 soft tag labels that were used for the multi-label classification and how many channels in the dataset were tagged with each label:

Label	Number of Channels
Anti-SJW	276
Partisan-Right	232
Partisan-Left	120
SocialJustice	106
Conspiracy	80
ReligiousConservative	54
AntiTheist	47
Socialist	46
WhiteIdentitarian	44
Libertarian	37
MissingLinkMedia	37
Educational	34
StateFunded	32
Provocateur	26
MRA	18
Revolutionary	10
LateNightTalkShow	7
AntiWhiteness	3

Table 1: The number of channels per label. Out of all channels, 365 were tagged with one label, 260 with two, 84 with three and eight channels had four labels.

A description of each of these labels can be found in appendix D of [6].
A label was given to a channel if more than half of the raters agreed on that label. For 63 channel no soft tag was given because there was no agreement on any label. For the multi-label classification, these channels were useless and were thus removed from the training set. For the binary classification task of labelling channels as either political or nonpolitical, channels with only hard-tags were kept in the dataset. Still, 45 channels had neither soft, nor hard tags, so they were removed completely, leaving 758 labeled channels for the binary classification task and 740 for the multi-label classification task.

4.2 Data Collection

Data collection was a large part of this project as on the one hand, data about personalized YouTube video recommendations was gathered, and on the other hand, an algorithmic classifier for such data was trained for which

also additional data was needed. To make distinctions between political and nonpolitical channels, the classifier required examples of channels that do not primarily produce content on political or cultural issues. Furthermore, a machine learning algorithm needs a set of features, on which it can distinguish the channels from each other.
This section first proposes an experiment to collect meaningful data on personalized video recommendations, then it describes how a selection of nonpolitical channels was sampled, and lastly, the collection of feature data is outlined.

4.2.1 Personalized Recommendations

Video Recommendations that are personally tailored to the taste of a user can only be presented by YouTube if the user logs into their YouTube account with which their watch history can be recorded. Necessary steps to see personalized recommendations are therefore the creation of a YouTube account, logging in, and watching videos with that account.
Gathering data about personalized recommendations on a large scale is difficult because YouTube does not offer this feature in their *YouTube Data API*[9] and traditional scraping methods are not equipped to deal with YouTube's *two-factor authentication* (2FA) during log-in. Two-factor authentication is an authentication technique that requires, in addition to username and password, further proof of identity, often in form of an sms code.
In order to automatically log into YouTube, watch videos and record which recommendations appear on the website, I implemented a data scraper based on *Selenium*. Selenium is a framework that enables the automated control of web browsers and is usually employed to test the interfaces of web applications [45]. The implemented scraper is able to receive 2FA validation codes via the messaging platform Discord[10] and can watch multiple YouTube videos in parallel in different browser tabs.
With this scraper, it is possible to launch studies that explore the nature of YouTube's recommendation algorithm by recording and analyzing recommendations on a large scale. One such study is proposed here.

Experiment Background The experiment design is based on the model by Jiang et al. [17] explained in section 3.1. According to their theory, every recommendation system eventually shifts the users' interests from their original position to a (potentially) arbitrarily different position. I propose

[9] `developers.google.com/youtube/v3` (accessed: 24.07.2020)
[10] `www.discord.com` (accessed: 03.08.2020)

a study that would investigate whether any bias with regards to this shift exists in YouTube's recommendation system. Such a bias could be the tendency to select some items more often, independent from the interests of the user, or the tendency to shift the user's interest more strongly in one direction than in others.
With regards to 'degenerate' behaviour of the YouTube algorithm, there are multiple hypotheses that were stated by journalists and researchers. Two of the most common are:

- Hypothesis 1: Filter bubble effect
  The composition of recommendations in the feed and in the 'watch next' recommendations is less diverse for users logged in to an account with a viewing history compared to users who are not logged in or have no watch history (and thus receive non-personalized recommendations) [6].

- Hypothesis 2: Right-wing advantage
  Watching a right-wing video has a higher impact on later recommendations than watching a neutral or left-wing video [6].

To either confirm or dismiss these hypotheses I designed an experiment which was integrated into the *Recfluence* project[11] in collaboration with its author Mark Ledwich. The Recfluence project was started in 2018 to collect non-personalized recommendations on YouTube. Under the name 'UserScrape' a pilot version of the proposed experiment was launched to also gather information about personalized recommendations. Code and further information can be found in Appendix A.

Experiment description For a comparison of how the YouTube algorithm reacts to different initial interests and watching behaviours, 17 new accounts were created. To each of these accounts, a specific 'viewer profile' is assigned that describes what kind of interest the test-account has. The viewer profiles are based on ideology categories created by Ledwich and Zaitsev in [6]. From the 18 labels that were described in section 4.1 they aggregated 13 ideology classes according to rules that can be found in appendix D of [6]. These ideology classes represent 13 of the 17 viewer profiles. The remaining four viewer profiles are control-accounts, leading to the following list of viewer profiles:

[11]`Recfluence.net` (accessed: 04.08.2020)

1. White Identitarian
2. MRA
3. Conspiracy
4. Libertarian
5. Provocative Anti-SJW
6. Anti-SJW
7. Socialist
8. Religious Conservative
9. Social Justice
10. Center/Left MSM
11. Partisan Left
12. Partisan Right
13. Anti-Theist
14. An equal number of videos from each class (uniform condition)
15. A number of videos watched from each class which is proportional to that classes' view count
16. Nonpolitical
17. A fresh account without viewing history

The viewer profile defines the set of videos that an account initially watches, on which the recommendation system bases its recommendations. Account 1 will e.g. automatically watch videos solely from channels classified as 'White Identitarian', account 2 watches videos from channels classified as 'MRA' and so on. The videos to be watched are sampled from all videos that channels with the specific label in the dataset uploaded. The videos are sampled weighted by popularity, to better represent the watching behaviour of a typical user. Each account watches 13 videos from their viewer profile per experiment trial. Since some videos can be rather long, this number was restricted to a relatively low number. The videos need to be watched in full length so that every video contributes equally to the recommendation system.
To measure the impact of the different watch histories on the personalization of the recommendations, a set of videos is watched by every account and the recommendations are collected and compared. This set of videos is created by sampling one video from the top five channels of each category, again weighted by popularity. Every account thus watches per trial the same five videos from every ideological category - in total 65 videos. The number of videos was restricted to five to limit the execution time of the experiment but could also easily be increased.
Since it is also an important question how a political watch history influ-

ences the recommendation on nonpolitical videos, each account additionally watches 50 videos from the top *trending* videos[12] in the US on the day that the trial is run. These videos are algorithmically selected by YouTube and contain mostly nonpolitical content. In total, the ‘test set’ of videos that each account watches consists of 115 videos, 65 sampled political videos from different political categories and 50 trending videos. To not influence the recommendations of subsequent test videos, these videos are directly deleted from the watch history after the corresponding video recommendations are saved.

Additionally to the recommendations gathered from the test videos, the feed i.e. the personal homepage for each user, which also contains video recommendations, is stored and compared. The feed is essentially a representation of the user’s interests from the recommendation engine’s point of view because it is not based on any specific seed video unlike 'watch next' recommendations that appear next to a video. The feed thus gives valuable information about the internal model of the recommendation algorithm. The feed is refreshed and stored multiple times to have more data points along which the different conditions can be compared.

The full procedure of an experiment run is presented in Figure 1. The experiment is run every day to collect longitudinal information about the recommendation system in case that YouTube announces changes. This way, the effects of the changes can be validated with a change in the measured recommendations. In summary, 17 different accounts each add 13 randomly sampled videos from their viewer profile to their watch history. Their feed is recorded 100 times and also the personalized recommendations from 115 test videos that are sampled every day anew are recorded.

With this experiment the aforementioned hypotheses can be addressed in the following way:

- Hypothesis 1: Filter bubble effect
  To measure whether personalization decreases the diversity of the recommendations’ content, the number of distinct political categories that appear in the recommendations for the account without a watch history will be compared with accounts that have already watched videos according to their viewer profile. An additional comparison could be drawn between test videos watched from the category of each account’s viewer profile and test videos from the other categories. This way it can be inspected whether users are ‘isolated’ in their own bubbles.

[12] See `www.youtube.com/feed/trending` (accessed: 10.08.2020)

- Hypothesis 2: Right-wing advantage
  For this hypothesis, the recommendations of the uniform condition (account 14) would be examined and compared with those of account 17. If content from right-wing categories appears proportionally more in the uniform condition than for the fresh account, it can be concluded that watching a video from such channels has a higher impact on the recommendations than watching other channels.

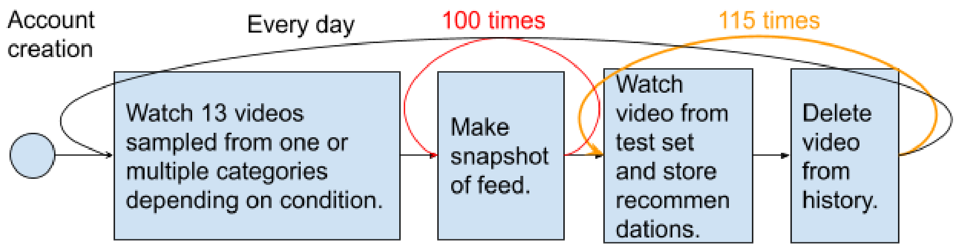

Account creation
Every day
100 times
115 times
Watch 13 videos sampled from one or multiple categories depending on condition.
Make snapshot of feed.
Watch video from test set and store recommen dations.
Delete video from history.

Figure 1: The experiment procedure.

4.2.2 Collection of Nonpolitical Channels

To train a classifier to discern between political and nonpolitical channels, examples for nonpolitical channels are needed. For creating a balanced dataset, attention had to be paid not to introduce any unwanted biases. The popularity of a channel, for example, should not have much of an influence on its classification - if done by a human. However, for the machine learning approach, small differences in the training data could accumulate and introduce a bias. More popular channels might show a different distribution in the word frequency because of a selection bias. In other words, channels that use certain words in their titles, descriptions and videos are possibly less likely to be successful. If only very popular channels are picked for the negative examples, one might miss out on these potentially different vocabularies, that could still be valuable for the classification task.
Therefore I selected for the nonpolitical examples only channels that have approximately the same number of subscribers as the political examples, which have a mean subscriber count of 443.391 and a median of 93.350. The distribution of the number of subscribers per channel is depicted in Figure 2. The mean count of total channel views is 142.196.397 and the median 11.896.201.

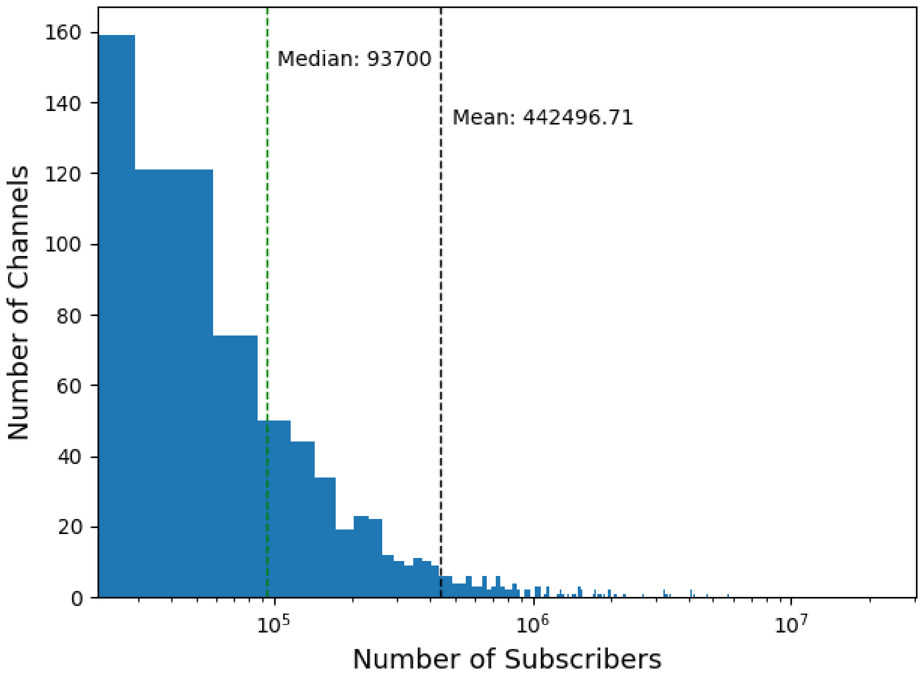


Figure 2: Subscribers Histogram - Political Channels.
The channel with the most subscribers has 21.7 million subscribers while some channels have 0 subscribers.

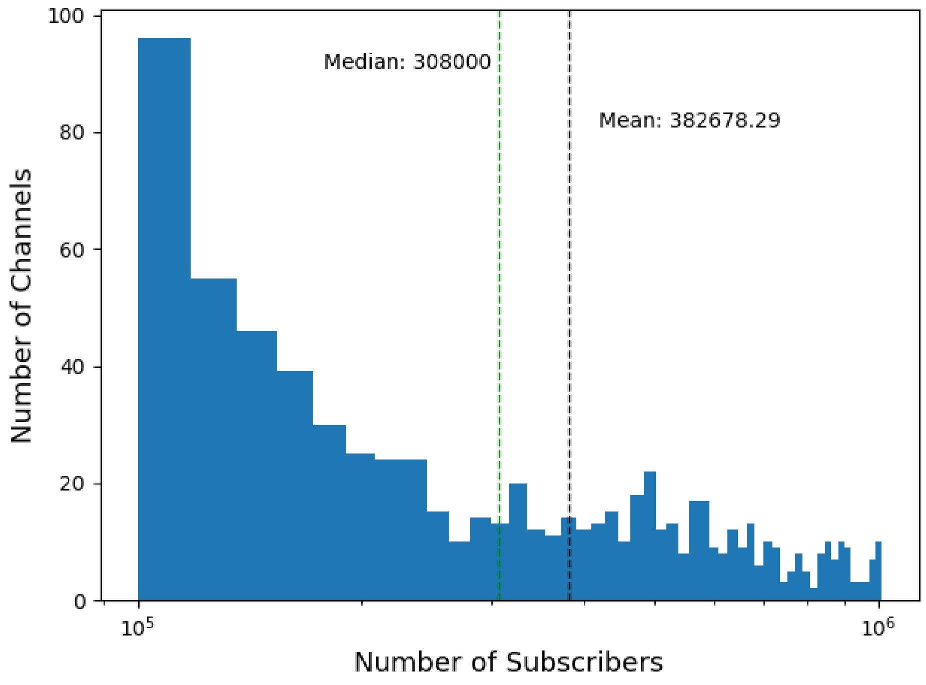


Figure 3: Subscribers Histogram - Nonpolitical Channels
The channel with the most subscribers has 1.010.000 subscribers and the channel with the least subscribers 100.000.

Since it is not possible to sample YouTube videos or channels randomly, because YouTube does not offer access to the full set of channels, alternative ways to sample YouTube channels had to be found. A previous study by Figueiredo, Benevenuto and Almeida employed a semi-random sampling technique by randomly sampling a set of words and proper names from a lexical ontology, and then picking the first search results on YouTube for each entity [46].
However, an additional challenge is the enormous variety of channels and channel content on YouTube. For a good, generalizable machine learning classification between political and nonpolitical content, ideally examples from all kinds of nonpolitical content are provided. The sampling method should therefore make sure that the diversity of content on the platform is reasonably represented in the sample. One option to do this is to use the categories that YouTube offers.
According to their API documentation, YouTube has two different category

systems in place to sort its content. On the video level, the uploader of the video can manually set the category to which he or she thinks the video belongs. The categories are:

1. Film & Animation
2. Autos & Vehicles
3. Music
4. Pets & Animals
5. Sports
6. Travel & Events
7. Gaming
8. People & Blogs
9. Comedy
10. Entertainment
11. News & Politics
12. Howto & Style
13. Education
14. Science & Technology
15. Nonprofits & Activism

On the channel level, the users can not set their category themselves, but instead YouTube classifies them according to the following categories:

1. Best of YouTube
2. Creator on the Rise
3. Music
4. Comedy
5. Film & Entertainment
6. Gaming
7. Beauty & Fashion
8. Sports
9. Tech
10. Cooking & Health
11. News & Politics

Unfortunately, it is not possible to query the YouTube Data API to return these channel categories, which are called *GuideCategories*, for a specific channel. It is only possible to ask for a list of channels that belong to one specific GuideCategory. Luckily, the website *channelcrawler.com* [47] offers a free search engine for YouTube channels. One can filter based on channel popularity, language, country, and also category. The categories by which one can filter are not based on the GuideCategories, but rather on the YouTube video categories. The author of the tool assigns each channel a category based on the majority of the 10 most recent videos' categories. Furthermore, it is possible to enter search terms, so that related channels are found. This would technically allow for the sampling method by Figueiredo et al [46] which in this case presented to be too time-intensive: Because the ChannelCrawler's automatic language detection tool was very inaccurate, every search result needed to be manually checked whether first, the channel language is indeed English and second, whether its content is indeed unpolitical. Employing Figueiredos sampling technique was therefore disregarded because of time reasons. Instead, I used the following procedure:

For every nonpolitical category (i.e. excluding 'News & Politics' and 'Nonprofits & Activism') I filtered the search for channels in English language and a subscriber count between 100.000 and 1.000.000. The results were by default ordered by their most recent video in descending order. All search queries were conducted on the 13th of April 2020. I then picked the first results that fulfilled the previously mentioned conditions from this list. In the 'Education' category, channels that educated about cultural topics such as the bible and religion were disregarded in order to have a clear separation between political and nonpolitical examples. To have a sufficient number of popular channels represented in the dataset, the results were also ordered by subscriber count in descending order and the first dozen results were picked from this list. In total, 60 channels were collected for each category, resulting in 780 nonpolitical channels.
The resulting nonpolitical data set has a mean subscriber count of $382,184$ and a median of $308,000$. The distribution is shown in Figure 3. The mean number of channel views is 99.227.448 and the median $46,406,315$. While the general distribution is similar to the one in Figure 2 and the mean is close to the mean of the political channels, channels below 100,000 subscribers are missing because they were filtered out in the ChannelCrawler search results. In an analysis of the dataset, I found that there is an overlap of two channels between the nonpolitical and the political data. These two channels were therefore removed from the nonpolitical data. Four more channels were removed. Two because they were clearly political, and for two other channels, one educating about laws, and another about political science, I changed the label to 'Educational' and thus moved them from the nonpolitical channels to the political channels. This reduced the number of nonpolitical channels to 774 and increased the number of political channels by two. More information about these steps can be found in Appendix B.

4.2.3 Classification Features

Having a dataset of labeled positive and negative examples is necessary for training a machine learning classifier, but it is certainly not sufficient. Every machine learning classifier needs a set of *features* in a machine-readable form to base any decision on. The features can be measurements taken from sensors, images and video recordings, or inherent characteristics of the objects one wants to classify. In the case of YouTube channels, the content that a channel produces is of course the main feature, but some channels upload thousands of videos, so it is with my available resources not possible to take a channel's full video content into account. Since the visual information from the videos is presumably a minor decision factor for distinguishing political videos, I instead focused on more language-based features of the videos. For deciding which features to use, the study by Faddoul, Chaslot and Farid on classifying YouTube recommendations regarding their conspiratorial nature was used as an orientation [7]. For their classification task they use the following features:

1. the (auto-generated) captions of a video
2. the video snippet of the video (title, description, tags)
3. the top 200 comments of the video (without replies and sorted by YouTube's relevance metric)
4. a feature vector created through classification of the comments with Google Perspective API

The classifier trained on these features performed well enough such that it could be used to label channels gathered in their study. Based on these promising results, I made use of the same features, with some differences. First, since my approach classifies channels and not videos, all features were aggregated from multiple videos. Instead of using the captions from one video, multiple captions plus the video snippets were concatenated. Comments were treated in the same way, but instead of the top 200 comments, only 100 were downloaded because of resource limitations.

To be able to process the data with the available computational resources, the amount of videos from which features are aggregated is set to the three most popular videos of each channel. While the most popular videos do not necessarily best represent the political nature of the channel, they are, next to the most recently uploaded video of the channel, the most likely videos to be watched by new viewers and therefore the most influential. One could also choose to pick the three most recent videos of each channel, but there might be fewer comments under those videos, which are presumably an important feature for the classification.

The Google Perspective API to classify each comment was not used because the quota restrictions of this API are so limited that classifying the given amount of data would take weeks. Furthermore, this feature's regression coefficient had the lowest weight from all features in Faddoul's classifier and might thus not be that influential on the final classification [7].
Instead of this comment feature vector, I made use of some other features that are either channel-related or harness network information. First of all, each channel is given a description by the owner, which was concatenated to the video snippets to create one text document. Additionally, a channel can also give information about related channels: Each user has the option to present other channels on their channel homepage[13]. Some use this feature to refer to their own other channels on YouTube if they have multiple channels, but some users present channels they recommend their viewers to check out because they are similar to their own channel. This list of *related channels* gives valuable information about what the channel owner likes or what they think their content is similar to.
Furthermore, for some channels, information about their subscriptions is publicly available. On YouTube, subscribing to a channel means that one wants to be kept up to date when this channel uploads new content. Being able to see the list of subscriptions of a channel enables inferences about what kind of content the person behind the channel personally consumes or finds interesting.
Another information source on how channels are connected are the comments. Some channel owners frequently leave comments under videos of other channels, which gives information on who is watching the videos of that channel. In a similar vein, simple mentions of other political channels in the comment under a video can also give information about the political orientation of the video uploader. This would rest on the assumption that commenters mention channels with similar content more often than other channels. It is also possible that users discuss a variety of political YouTubers in the comments and extracting such a feature would contain no discerning information. This feature was considered to be analysed for efficacy but dismissed for other reasons, as explained in Section 4.3.1.
In summary, the following features were added:

1. The channel description (added to the snippets)
2. Channel subscriptions
3. Related channels of the network
4. Comments by known channels

[13]See for example `www.youtube.com/googlecode/channels` (accessed: 10.08.2020).

All of these features need to be prepared such that they are in a form that the machine learning algorithm can interpret them. The *encoding* of these features will be explained in the following section. But first, the methods for how this data was collected are presented here.
In the beginning, the id's of each channel's top three most popular videos were scraped with Selenium. With the YouTube Data API I downloaded the respective video snippets (title, description, keywords and statistical data), the top 100 comments (sorted by relevance) under each video with the comment author names, the general channel description and information about which channels they feature in their 'Related Channels' list and who they subscribe to. If available, the English closed captions for each of the videos were downloaded with the help of the *dotnet* package *YouTube Explode* [48]. The YouTube Data API could not be used for the captions due to their API quota limit.
Between the time the channels were labeled for [6] and the time I gathered the feature data, some channels were deleted or they deleted all of their videos. This made it impossible to get more data for them, so they were removed from the dataset. The final set of labeled political channels for the binary classification task was thus reduced to 732 and for the multi-label classification task to 713.

Missing Data Through the previously described process, one should find for each channel three video snippets, three captions, three sets of 100 comments each, and a list of related channels and subscriptions. However, not for all channels were three videos available and not for all videos from the channels existed all features. Some videos on YouTube do not have closed captions, some have comments disabled or sometimes the channel owner simply deletes all comments under a video. It is also not possible for each channel to see what other channels they subscribed to, since most people set this list to 'private' in their settings. Approximately a third of the channels do not list any related channels on their channel page. Table 2 shows a comparison of the missing data for all political channels and nonpolitical channels.
The missingness of these features is not equal for the two classes of channels. For the political channels, 4.8% have no captions for their top 3 videos and only 1.6% have no comments for any of them. In contrast to this, the nonpolitical channels have no captions in 19.8% and no comments in 7% of all cases. The missing captions are easily explained as there are many music

videos and other videos without spoken words in the set of nonpolitical channels.

	political	nonpolitical
total number of channels:	734	774
#channels with 2 video snippets:	2 (0.3%)	0
#channels with 1 video snippet:	0	0
#channels with no video snippets:	0	0
#channels with 2 captions:	138 (18.8%)	135 (17.4%)
#channels with 1 caption:	53 (7.2%)	103 (13.3%)
#channels with 0 captions:	35 (4.8%)	153 (19.8%)
#channels with 2 comment sets:	27 (3.7%)	36 (4.7%)
#channels with 1 comment set:	4 (0.5%)	8 (1%)
#channels with 0 comment sets:	12 (1.6%)	54 (7%)
#channels with 0 related channels:	232 (31.6%)	253 (32.7%)
#channels with 0 subscriptions:	511 (69.6%)	538 (69.5%)

Table 2: An overview of the missing features for the two classes.

4.3 Data Preprocessing and Feature Extraction

This subsection explains how the feature data was processed so that they are interpretable by the machine learning algorithms. First, the encoding of all features based on the relations between channels is described and then the text-preprocessing pipeline is outlined.

4.3.1 Network Features

The simplest way to encode relations between channels in vector form is to create a connectivity matrix $A \in \{0,1\}^{n \times m}$ such that

$$A_{ij} = \begin{cases} 1, & \text{if there is a connection from channel i to channel j} \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

For the n channels in the dataset (including unlabeled channels described in Section 4.4.5), each row vector A_i would then encode the feature vector for channel $i = 1...n$. For the choice of m two options exist. Since some channels had e.g. 'related channel' connections to channels that were not part of this dataset, these outgoing connections could either be included or excluded. Depending on this choice, the dimensionality m of the feature vector could either be lower or equal to n, or a finite number greater or

equal to n, depending on how many of these outgoing connections exist for that feature.
Some features, namely the *cross-channel comments* and *channel mentions* first had to be extracted from the comments that were gathered earlier. Due to the way these two features were extracted, the corresponding feature vectors could be of maximal length n. The following paragraphs give more details about each network feature.

Related Channels The most promising network feature was the list of related channels that a channel owner can present on their main page. Two thirds of all channels do recommend at least one other channel in this way. If only recommendations to channels included in the dataset were used to create the feature vectors, the length of the vectors was 2,401. If also connections to channels outside of the dataset were considered that number was 15,606. This feature will also be referred to as ‘associations’ in this report.

Subscriptions This *subscriptions* feature is unfortunately the most sparse, since most channel owners set their list of subscriptions to ‘private’ such that this information is not available to the public. Channels that do make their subscriptions public are on average subscribed to 151 other channels with some subscribed to up to 970. This leads to a relatively high dimensionality of 113,614 for the feature vector if unknown channels are included. If only subscriptions to channels contained in the dataset are regarded the length of the vector is 4,431.

Cross-Channel Comments A channel on YouTube can not only be used to publish videos, but also functions as a user account with which one can comment videos. With the *cross-channel comments* feature I wanted to examine which channel owners write comments under the videos of other channels, because it shows that they watch and engage with that channel’s content. For each comment returned by the YouTube API, the id of the author’s channel is delivered as well, which allows for a precise identification of who commented under whose videos. For every channel in the dataset, I checked whether any of the comments on the three most popular videos were written from a channel account that also appeared in the dataset. The *cross-channel comment* feature vector thus signifies which owners from channels in the dataset commented at least one of the three most popular videos. It is important to note again that only the first 100 comments (sorted by relevance) were gathered, so not every cross-channel comment was necessarily found. The feature vector had a length of 506 which means that at the time the data was gathered, at least 506 comment interactions happened between two channels from this project's dataset.

Channel Mentions Under the assumption that commenters will mention channels similar to the one that is currently watched, I wanted to extract mentions of channel names from the comments. For this purpose, a regular expression with each channel's title was used to identify its occurrence in any of the comments in the dataset. This method unfortunately quickly showed to be unsuccessful because many of the channel titles were either so specific that is unlikely that they are referred to by their full title (e.g. 'Patriots' Soapbox News Network LIVE 24/7' or 'David Wilcock — Divine Cosmos (OFFICIAL)') or they are too generic that people often use the title without specifically referring to channel with that title (e.g. 'Patriot Act', 'amazon'). Using a simple regular expression to extract this feature is therefore overly imprecise, but other methods are potentially very complex and do not necessarily promise better results. For these reasons I dismissed the feature completely.

4.3.2 Text Preprocessing

All text-based features were extracted by simply concatenating relevant parts from the data gathered through the YouTube Data API and other methods. The *snippets* consist of the channel description, channel title, video titles, video descriptions and video tags. To create the *captions* for the channels, the closed captions from the three most popular videos were concatenated, and the same was done for the 100 comments for each video to create the *comments* feature.
This way, up to three text documents with varying lengths existed for each channel. These documents did not have to be encoded in vector form directly, because the library that was used for text classification in this project receives the text as input and the encoding happens internally (see Section 4.4.1). To facilitate an effective encoding, they nevertheless had to be preprocessed which I did in the following way:

Since comments and snippets are written by humans (in contrast to the auto-generated captions) they contain spelling mistakes that needed to be corrected. Running extensive spell-checking on texts of this size is very time expensive which is why I decided to employ only a very basic spell-check. With a regular expression any letter that appears more than two times after

	Captions		Comments		Snippets	
Processing Step	Total Tokens	Distinct Tokens	Total Tokens	Distinct Tokens	Total Tokens	Distinct Tokens
Before preprocessing	9,047,641	129,026	8,704,924	508,404	964,501	106,399
Spellchecking	9,047,641	128,759	8,704,924	494,233	964,501	105,909
HTML tag removal	9,047,431	127,918	8,477,714	438,931	964,475	105,311
URL removal	9,047,408	127,882	8,476,210	437,302	934,885	86,209
Special symbols removal	8,997,105	85,339	8,262,769	227,757	890,885	49,610
Stop word removal	4,367,255	85,173	4,401,594	227,587	642,747	49,450
Lemmatization	4,367,255	75,135	4,401,594	217,622	642,747	45,282
	mean	median	mean	median	mean	median
Tokens per document	2864	1745	2916	2724	364	316

Table 3: The effects of each processing step on the corpus of each text feature.

another is replaced with just two occurrences of that letter. The misspelling 'lettter' thus turns into the correct form 'letter' and 'amaaaazing' becomes 'amaazing' which is still not correct, but at least merges all instances of this stylistic device with different amounts of repetitions to one token.
Comments and snippets also contained links and HTML-tags that added unnecessary noise to the data that were therefore identified with regular expressions and deleted. After this step, all text was set to lowercase and all newlines and tabs and any special characters, including punctuation, were removed.
Next, the text was tokenized and 157 stop words from the NLTK library [49] were removed. The stop words were processed in the same way as the text data to assure they match correctly. As a final step, the tokens were lemmatized with a lemmatizer from the NLTK library. I measured the effect each preprocessing step has on the corpus by counting the number of total tokens and also distinct tokens (which represents the dimensionality) before and after applying each step. Additionally, the mean and median number of total tokens per data point was calculated. The results are collected in Table 3.
Removing special characters and punctuation has the largest effect with regards to dimensionality reduction as it cuts the number of distinct tokens down by up to 50%. This shows that the tokenizer creates different tokens for differently written words such as 'sub-process' and 'subprocess' but also

that it incorrectly identified words as different tokens if they are adjacent to punctuation such as ‘house’ and ‘house.’.
The table also shows that a large part of the text consists of stop words as the total number of tokens is roughly cut in half when they are removed. Only the snippet data is not as affected by this which can be explained by the fact that these texts are not completely natural language, but rather consist of many keywords and tags from the titles and descriptions of the videos. This is also the reason why removing HTML tags and URLs from the text has the largest effect on the snippets. Since many content creators link their profiles on social media sites in their channel or video descriptions, a large part of the text is removed in this step.
An interesting fact is that the number of distinct tokens in the comments data is almost three times as large as for the other two text features. This can be explained by the large diversity of authors of the comments, which is reflected in the large vocabulary. Furthermore, the authors of these texts likely put less effort into the correct spelling of their comments than the authors of the video titles and description, leading to more spelling mistakes and thus a larger vocabulary.

4.4 Classifier Architectures

Similar to the classifier employed in the study by Faddoul et al. [7], the classifier used in this project is also an ensemble of multiple classifiers. This section describes the different classifiers employed for each feature, the ensemble build from the combination of them, and the hyper-parameter optimization for each of these components. Before any classifiers were trained, 15% of the data was split off to be saved for estimating the risk of the trained models.

4.4.1 Text Classifiers

On all text features the *FastText* classifier [25] was trained and evaluated. At the core of the FastText classifier are three components. First, words are represented in this model as a bag of character n-grams, which means that a word such as ‘rhetorics’ is split into ‘<rh’, ‘rhe’, ‘het’, ‘eto’, ‘tor’, ‘ori’, ‘ric’, ‘ics’, and ‘cs>’ if n equals 3 [50]. The ‘<’ and ‘>’ symbols are used as markers to differentiate between word beginnings or ends and short words. This representation enables the classifier to calculate the word embedding of an unseen word from its sub-components.
The word representations are then averaged into a hidden layer which serves

as text representation, or embedding layer. The dimensionality of this layer is 100 by default. These embeddings can be reused to initialize other models if needed. Lastly, the embedding vector is fed into a linear classifier with a softmax layer to compute the label probabilities.
The FastText classifier has four main hyper-parameters to optimize, which is the number of epochs for training, the learning rate, the n in the word n-grams, and the embedding size. Except for the embedding size, I varied each of the parameters while keeping the others parameters at their default values and ran a 5-fold cross-validation on the training data for each setting. The performances were compared by measuring the F1-scores for both classes and calculating the mean weighted by the support for each class in the validation set. Figure 4, 5 and 6 show the resulting mean F1-scores on the training and validation data for each parameter value that was tested. For reasons of brevity, the graphs show only the cross-validation results for the caption-feature.
While changing the learning rate or the n-gram from the default value to any other value decreased the quality of the model, training it with for enough epochs was essential to get a high precision and recall. Running the same cross-validation for the other text-based features validated this result. Since a larger number of epochs leads to longer training times, the final value for this parameter should be as small as possible, while still maximizing the validation score. The optimal number of epochs for the captions, comments and snippets feature are 45, 45, and 50 respectively.
A final model was then trained for each feature on the full training data. In the training data, missing data was removed, so the model is only training on examples with non-empty captions, comments or snippets respectively. When this model later predicts label probabilities for an example where the respective feature is missing, it will predict *Null* as a value.
The final model was evaluated by measuring the precision, recall and F1-score on the testing set. The results are presented in table 4.

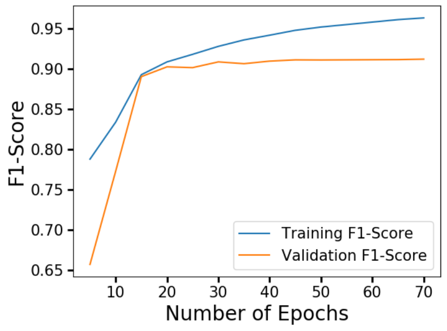


Figure 4: Varying the number of epochs on the caption dataset. Default value = 5.

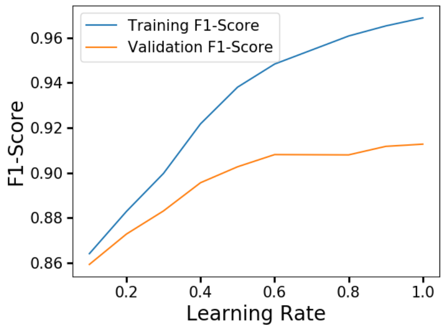


Figure 5: Varying the initial learning rate on the caption dataset. Default value = 1.

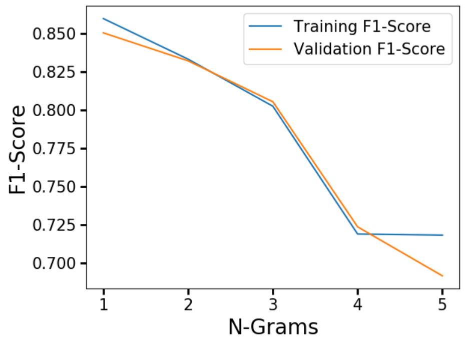


Figure 6: Varying over different n-grams on the caption dataset. Default value = 1.

	Captions		Comments		Snippets	
Metric	P	NP	P	NP	P	NP
Precision	0.914	0.887	0.927	0.921	0.956	0.928
Recall	0.901	0.902	0.914	0.933	0.915	0.963
F1-Score	0.907	0.894	0.921	0.927	0.935	0.945
Average F1-Score	0.902		0.924		0.940	
Accuracy	0.902		0.924		0.940	

Table 4: The performance metrics for the respective best text classification models. P stands for the label ‘political’ and NP for ‘nonpolitical’.

Multi-Label Text Classifiers To achieve a multi-label classification with the *FastText* library, the parameters of the *predict* function can be changed such that a vector with the probabilities for each label is returned. An example is assigned with all labels for which the probability is higher than a given threshold. Unfortunately, the probability vectors that the model predicts are the output of a softmax function which means that the probability values sum up to 1[14]. This makes the choice of the threshold difficult because for a channel that has multiple labels with similar (high) probability, the absolute probability values in the output vector are rather low.
For the hyper-parameter optimization with cross-validation, I evaluated the models with the *Mean Average Precision* (MAP), where the absolute probabilities for each label do not matter, but only the ordering is relevant. However, the interpretation of the MAP is not intuitive and therefore I also evaluated the models according to precision, recall and the F1-score for each label. For this evaluation, I chose a different threshold for each prediction by sorting the predicted class probabilities in descending order and setting the cut-off value at the largest difference between two probabilities. If the class probability predictions are for example $[0.5, 0.4, 0.04, 0.01, 0.01, ...]$ only the two most likely labels are predicted because the largest decrease in predicted class probability is between the second and third label.
The hyper-parameter search indicated that for the multi-label condition, the number of epochs for each feature needed to be much higher to achieve good results. Caption, comment and snippet classifiers had to be trained with 500, 500 and 400 epochs respectively. Table 5 presents the performance on the testing set when the classifiers were trained on the full training set. The difference between the text classifiers is in the multi-label classification more pronounced than in the binary classification, as the classifier trained on the snippets performs considerably better than the other two. Although having the smallest amount of data regarding the tokens per document, the classifier trained on the snippets showed the best performance of all text-based classifiers.

4.4.2 Network Classifiers

For the network features a logistic regression was chosen that receives as input the connection vector, i.e. the binary vector which shows to which other

[14]The documentation of the package suggests that also an option for a 'true' multi-label classification with independent probability values exists, but due to a bug in the library that is not yet fixed at the time of writing (see `www.github.com/facebookresearch/fastText/issues/994`), this was not available to me.

	Captions	Comments	Snippets
MAP	0.541	0.61	0.708
Average Precision	0.32	0.43	0.61
Average Recall	0.34	0.38	0.53
Average F1-Score	0.32	0.36	0.54

Table 5: The metrics for the respective best text classification models in the multi-label condition. The averages are weighted by support of the respective class.

channels the respective channel has connections, and outputs a confidence score, i.e. the signed distance of that data point to the decision hyperplane [23]. In the binary task, the confidence score represents how likely the channel belongs to class ‘political’ and in the multi-label task there is one confidence score for each class. For the training of these classifiers, I did not remove instances with zero connections because the bias of the logistic regression adapts to the class imbalance and enables the classifier to also make predictions about zero vectors.
As described in Section 4.3.1, the network feature vectors could be constructed in a way that either includes unknown channels or excludes them. I conducted hyper-parameter searches for each condition, varying over the regularization strength of the regression. The parameter settings that gave the best F1-score on the validation data were used to train a model on the full training data. These models were evaluated on the test set and the resulting classification metrics can be found in Table 6.

	Associations		Associations All		Subscriptions		Subscriptions All		CC Comments	
Metric	P	NP	P	NP	P	NP	P	NP	P	NP
Precision	1	0.65	0.971	0.681	0.84	0.595	1.0	0.562	1.0	0.504
Recall	0.394	1.0	0.478	0.988	0.292	0.949	0.291	1.0	0.148	1.0
F1-Score	0.565	0.788	0.641	0.806	0.433	0.731	0.451	0.58	0.504	0.67
Average F1-Score	0.683		0.728		0.589		0.58		0.448	
Accuracy	0.715		0.748		0.636		0.629		0.534	

Table 6: The metrics for the respective best network classifier models, either trained on vectors covering only channels from the dataset or also including external (*all*) channels. P stands for the label ‘political’ and NP for ‘nonpolitical’.

For all relation-based classifiers a large difference between precision and recall stands out. Most of the classifiers are able to correctly label all non-political channels in the dataset as ‘nonpolitical’ and most channels labeled

as ‘political’ indeed belong to the class ‘political’. If political channels are considered the positive class, all networks have, however, a large number of false negatives since the recall values for political channels and precision for nonpolitical channels are quite low. If not only internal connections, but also external ones are included for the features, the association classifier performs better with regards to accuracy and F1-score, while the classifiers trained on subscriptions do not gain from the inclusion of external connections.
For the final classifiers that are included in the ensemble, I decided to train the association classifier with all known relations, while restricting the subscription feature to only relations to other channels in the dataset. The reason for this is that including all subscriptions significantly increases the dimensionality of the data, and does apparently not increase the performance.
Although these classifiers perform considerably worse than the text classifiers, I argue that they can still be useful for the ensemble classifier. The strength of these classifiers lies in their high precision for the political channels. The line between political channels and nonpolitical channels can be very blurry if only the manner of the channel authors’ language is considered. Some political channels actively try to hide their political nature and pretend to be e.g. purely for entertainment, in order to reach a larger audience [35]. The text classifiers therefore do not necessarily pick up on these nuances. However, these channels can be recognized by examining their position in the network of political channels. As [35] has shown, especially politically radical channels are networking extensively. The relation-based classifiers could thus be useful in the ensemble, by filling in the gaps of the text classifiers.

Multi-Label Network Classifiers For a multi-label classification of the network features, the class *OneVsRestClassifier* of the *Scikit-Learn* package [23] was used. This class serves as a wrapper around a classifier, like in this case the logistic regression, and then trains one classifier of that type for each label in the training set. Each sub-classifier learns a binary decision between examples with that specific label and all other examples. The output of such a model is a vector with the class probability for each label. They do not necessarily sum up to 1 as a channel could have two labels, each with a predicted probability of 0.99.
For the logistic regression that is used inside the *OneVsRestClassifier* I chose the same parameter configurations that showed the best results for the binary classification. To get an overview of their performance, the resulting

wrappers were trained on the full training set and evaluated on the test set. As Table 7 shows, the recall values for the different labels are remarkably low. This can be explained by the method with which the classifiers are trained and the lack of data for many channels. The *OneVsRest* class trains one logistic regression model for each label, using all channels with that label as positive examples, and all others as negative ones. For relation features like the subscriptions, two thirds of the channels have a feature vector of only zeros because they have no relation to other channels and are thus indistinguishable from each other for the classifier. The majority of these channels will likely always belong to the negative class, and therefore receive no labels at all. If we assume that the missingness of relations to other channels is approximately uniformly distributed among the different classes, then in the case of the subscription feature, two thirds of all channels from each class cannot be recognized by these classifiers, which is reflected in the low recall values.

	Associations	Subscriptions	Cross-Channel Comments
MAP	0.575	0.56	0.507
Average Precision	0.71	0.62	0.34
Average Recall	0.19	0.15	0.04
Average F1-Score	0.28	0.15	0.08

Table 7: The metrics for the respective best network classifier models in the multi-label condition. The averages are weighted by support of the respective class.

4.4.3 Ensemble

To create a final ensemble classifier out of the individual classifiers, each of the sub-classifiers predicts the probability or confidence score of the instance being political for each instance in the training data as well as the testing data. Importantly, even though they are now applied to the testing data, they were only trained on the training data. The resulting scalar values, either values between 0 and 1 for the text classifiers, or real numbers for the logistic classifiers, where positive values indicate the label ‘political’, are then concatenated and standardized, such that the mean is *zero* and the standard deviation is *one*.

As it was explained earlier, some sub-classifiers predict *Null* if they do not receive input if e.g. a channel has no captions available. These missing

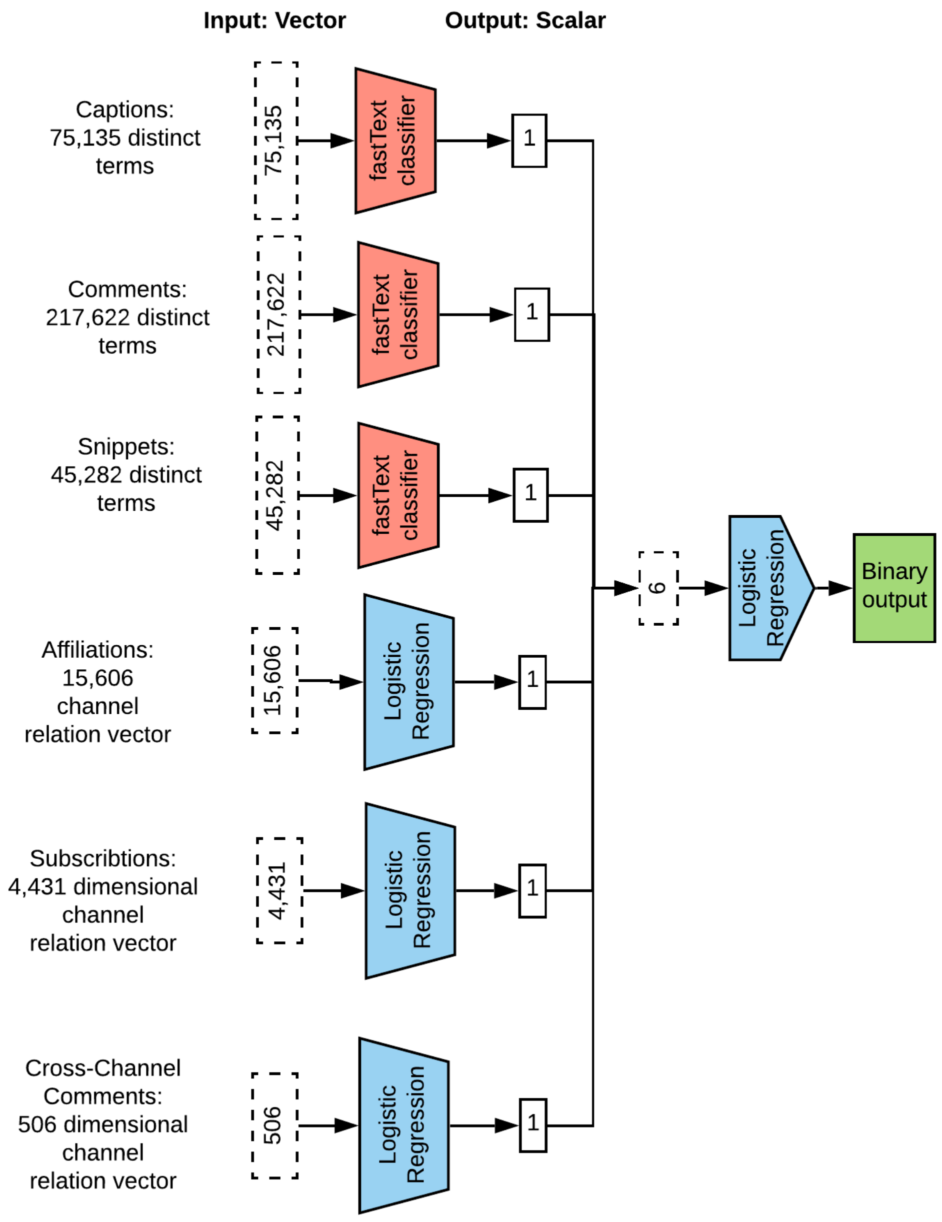

Input: Vector
Output: Scalar
Captions: 75,135 distinct terms
75,135
fastText classifier
1
Comments: 217,622 distinct terms
217,622
fastText classifier
1
Snippets: 45,282 distinct terms
45,282
fastText classifier
1
Affiliations: 15,606 channel relation vector
15,606
Logistic Regression
1
Subscribtions: 4,431 dimensional channel relation vector
4,431
Logistic Regression
1
Cross-Channel Comments: 506 dimensional channel relation vector
506
Logistic Regression
1
6
Logistic Regression
Binary output

Figure 7: The architecture of the binary ensemble classifier. The input to the FastText classifier is actually a string of words and not a vector, but the size of the vocabulary is depicted here to show inherent dimensionality of the data.

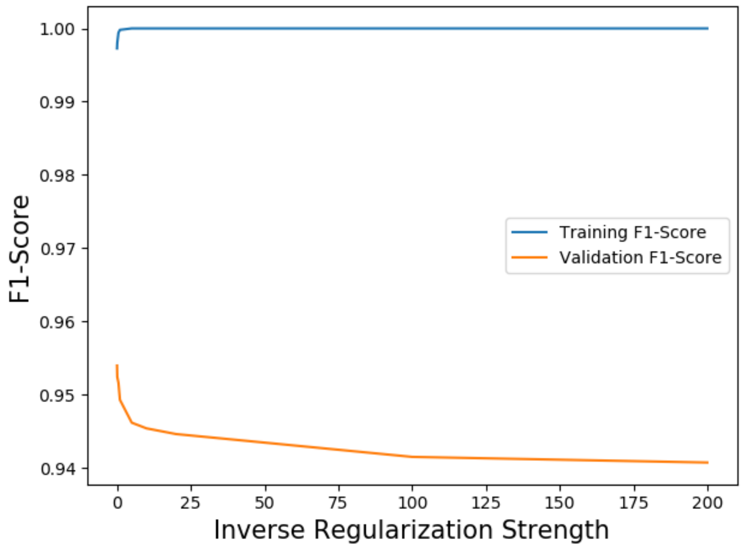


Figure 8: 5-fold cross validation of the Ensemble Logistic Regression with varying regularization strength.

predictions are here imputed with a *k-Nearest-Neighbors imputation* with $k = 3$. This means that for a channel with a missing prediction, the three most similar channels based on existing features are found and the missing value is replaced with the mean prediction for these channels.
Figure 7 shows the final architecture of the supervised baseline ensemble. A logistic regression classifier was trained on the six-dimensional prediction vector resulting from the sub-classifiers. Because the sub-classifiers were not trained on the testing set, it could be used to evaluate the final ensemble classifier. Before that, the regularization hyper-parameter of the final logistic regression was tuned with cross-validation on the training data. For this cross validation, each sub-classifier and the final logistic regression were only trained on the training fold and then their F1-score measured on the validation fold. Figure 8 shows that for every value above 0.1 for the regularization parameter, the mean training F1-score was 1, but the validation F1-score only decreased. Therefore, a small parameter value of 0.01 was chosen for the final logistic regression classifier.
Training the ensemble on the full training data and evaluating it on the test data resulted in a F1-score of 0.965 (see Table 8) and an ROC-AUC of 0.994 (see Figure 9)

	Ensemble Classifier	
Metric	P	NP
Precision	0.962	0.967
Recall	0.962	0.967
F1-Score	0.962	0.967
Average F1-Score	0.965	
Accuracy	0.965	

Table 8: The classification metrics for the binary ensemble classifier. 'P' is the class 'political' and 'NP' 'nonpolitical'. The average F1-score is weighted by the support of the respective class (122 NP, 105 P).

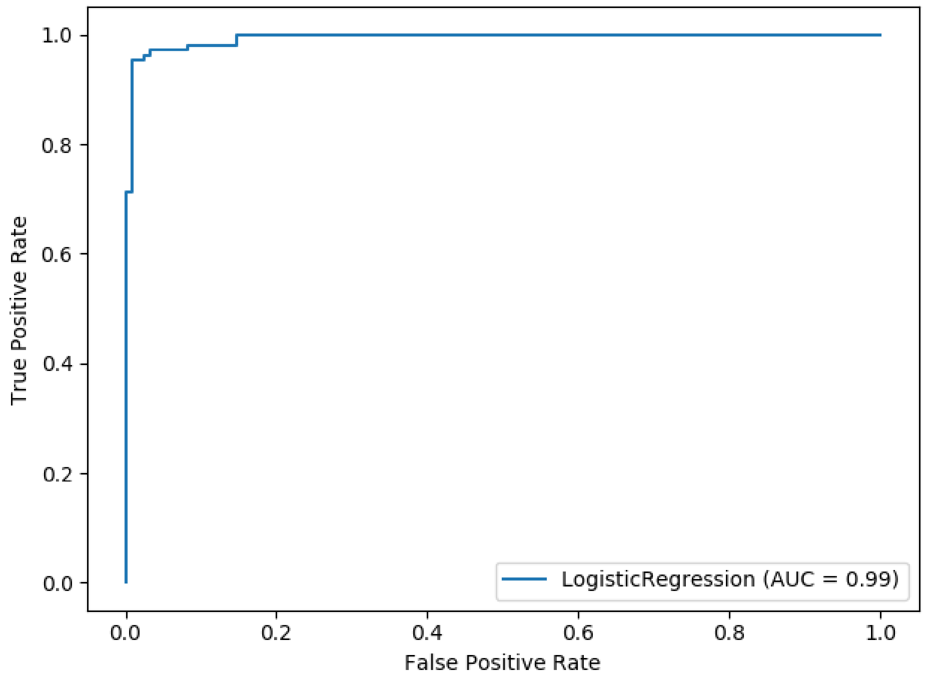
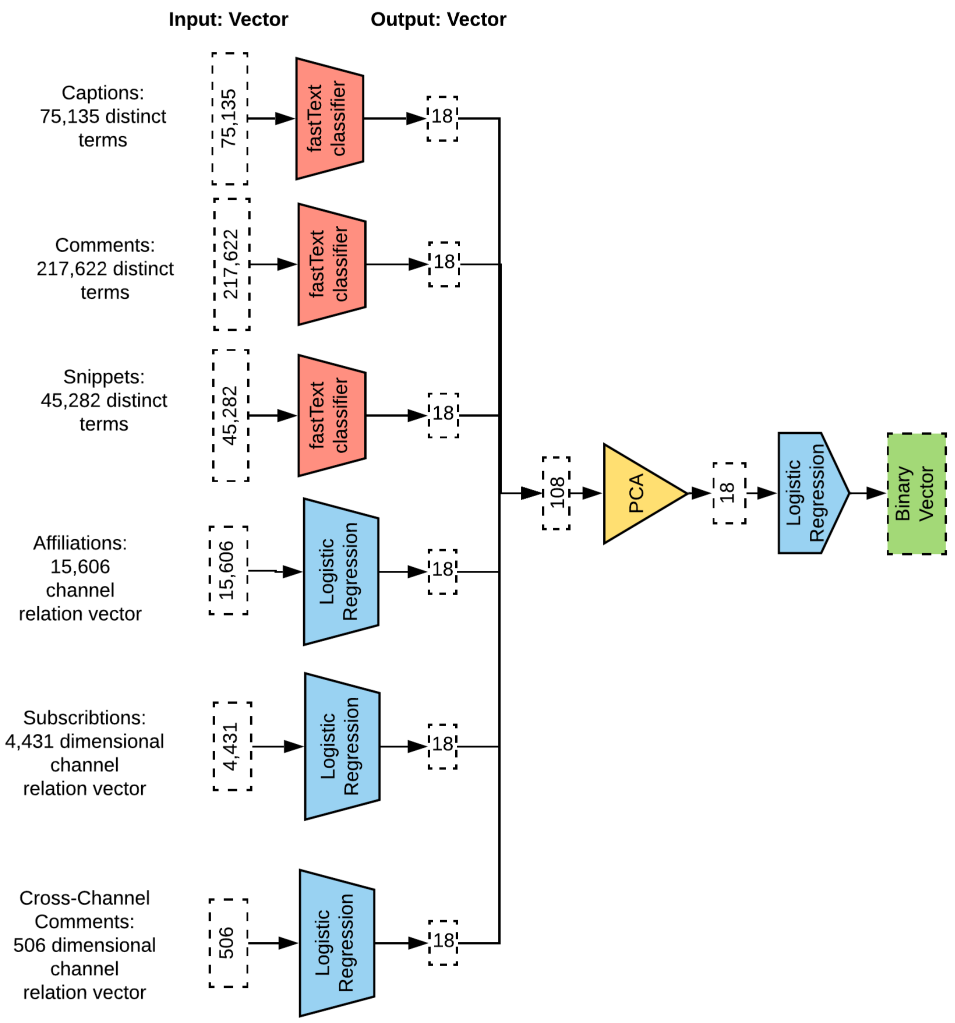


Figure 9: The Receiver-Operator Characteristics of the ensemble classifier on the test data.

4.4.4 Multi-Label Ensemble



Input: Vector
Output: Vector
Captions: 75,135 distinct terms
75,135
fastText classifier
18
Comments: 217,622 distinct terms
217,622
fastText classifier
18
Snippets: 45,282 distinct terms
45,282
fastText classifier
18
Affiliations: 15,606 channel relation vector
15,606
Logistic Regression
18
Subscribtions: 4,431 dimensional channel relation vector
4,431
Logistic Regression
18
Cross-Channel Comments: 506 dimensional channel relation vector
506
Logistic Regression
18
108
PCA
18
Logistic Regression
Binary Vector

Figure 10: The architecture of the multi-label ensemble classifier. The input to the FastText classifier is actually a string of words and not a vector, but the size of the vocabulary is depicted here to show inherent dimensionality of the data.

The main difference between the two ensemble models is that in the multi-label condition, the sub-classifiers do not return scalar values, which can be combined into a 6-dimensional vector, but rather a vector of length 18.

These vectors are concatenated into a 108-dimensional vector. *Principle Component Analysis* [51] was employed as a feature reduction technique to reduce this vector to an 18-dimensional vector in order to see whether it improves performance. Preliminary tests showed that this is the case and thus the feature reduction was included in the architecture of the multi-label ensemble classifier (see Figure 10. The reduced vector serves as the input for the final layer in the multi-label ensemble classifier, which is again a *OneVsRestClassifier* with logistic regression.
Another difference was the imputation of missing predictions from the sub-classifiers. Because the higher dimensionality of the sub-classifiers' predictions complicated the application of a k-Nearest-Neighbors imputation, a simple constant imputation was applied. If a sub-classifier predicted *Null*, the predictions were replaced by the uniform probability, i.e. 1/18 for every class.
A hyper-parameter search for this ensemble resulted in the same value for the regularization parameter as in the binary case. The performance of the multi-label ensemble classifier on the test set is given in Table 9. If compared with the individual snippet classifier in Table 5, the results show that the ensemble performs better according to the MAP, but worse in terms of the average F1-score. The snippet classifier alone cannot be used for the classification on the whole dataset, because the full snippet data does not exist for every channel. I therefore implemented a second ensemble that only consists of the snippets, comments and associations classifiers because they showed the best individual results. Table 10 lists the metrics of the smaller ensemble trained on the full training set and evaluated on the test set. Even though it is MAP score is even lower, the F1-score is on par with the snippet classifier.

	Multi-Label Ensemble
MAP	0.712
Average Precision	0.60
Average Recall	0.52
Average F1-Score	0.53

Table 9: The metrics for the ensemble classifier in the multi-label condition.

	Reduced Multi-Label Ensemble
MAP	0.696
Average Precision	0.63
Average Recall	0.53
Average F1-Score	0.55

Table 10: The metrics for the ensemble classifier consisting of the comment, snippet and association sub-classifiers in the multi-label condition.

A more detailed overview of the smaller ensemble classifier's precision and recall values for every label can be found in Table 11

Label	Precision	Recall	F1-Score	Support
AntiSJW	0.64	0.78	0.70	41
AntiTheist	0.57	0.50	0.53	8
AntiWhiteness	0.00	0.00	0.00	0
Conspiracy	0.62	0.62	0.62	8
Educational	1.00	0.08	0.14	13
LateNightTalkShow	0.00	0.00	0.00	1
Libertarian	0.50	0.60	0.55	5
MRA	0.80	0.57	0.67	7
MissingLinkMedia	0.50	0.60	0.55	5
PartisanLeft	0.65	0.69	0.67	16
PartisanRight	0.68	0.68	0.68	31
Provocateur	0.00	0.00	0.00	6
ReligiousConservative	0.20	0.25	0.22	4
Revolutionary	1.00	1.00	1.00	2
SocialJustice	0.45	0.28	0.34	18
Socialist	0.33	0.67	0.44	3
StateFunded	0.80	0.67	0.73	6
WhiteIdentitarian	0.29	0.50	0.36	4

Table 11: Multi-Label Classification Report. The support for a label tells how many channels with that label appear in the test set. The total support is 178.

4.4.5 Ensemble Self-Training

For each ensemble classifier I created a wrapper class that implements self-training as described in [31]. The basis for this semi-supervised method

was a set of unlabeled data points. For the experiments described in this report, the unlabeled data was taken from the Recfluence dataset [6]. Since the launch of the Recfluence project, over 100.000 unlabeled channels were discovered through gathering recommendations. To reduce the run-time of the further experiments, only the top 5000 most recommended channels from this large set were taken and used to train the classifiers.
For these channels, all features such as the captions, comments etc. were collected. During this process, channels that were either deleted or had no videos available were removed from the dataset. These were 311 channels. Additionally, some of these channels were already included in the labeled dataset, so the 102 duplicate channels were also removed. The resulting set of unlabeled channels thus consisted of 4587 channels. A breakdown of missing features is presented in Figure 12.
In the multi-label classification task, only unlabeled channels that the supervised ensemble classifier predicted as 'political' were used in the self-training algorithm.

	Unlabeled Channels
Total number of channels:	4587
#channels with 2 video snippets:	20 (0.4%)
#channels with 1 video snippet:	19 (0.4%)
#channels with 2 captions:	900 (19.6%)
#channels with 1 caption:	516 (11.2%)
#channels with 0 captions:	724 (15.8%)
#channels with 2 comment sets:	236 (5.1%)
#channels with 1 comment set:	96 (2%)
#channels with 0 comment sets:	192 (4.1%)
#channels with 0 related channels:	1502 (32.7%)
#channels with 0 subscriptions:	3100 (67.6%)

Table 12: An overview of the missing features for the unlabeled data.

The *SelfTrainer* wrapper is a class that creates an instance of one of the ensemble classifiers and additionally loads the unlabeled data. Originally the classifier is just trained on the labeled examples. Step by step, the classifier then predicts the labels for each unlabeled channel and the channels where these predictions had the highest confidence are added to the training set. In the next iteration, the classifier thus has a larger training set to learn from.
There are multiple ways to implement how many of the unlabeled examples

are transferred to the training set in each iteration. For the experiments in this project, two versions were tested. In the first version, the stop condition of the self-training algorithm is set to a fixed number, which means that also in each iteration a fixed number of examples is added to the training set. The number of iterations was set to six and in each iteration, the 16.6% most confidently labeled examples are transferred to the training set, such that the set of unlabeled data is empty when the algorithm finishes. In the second version of the SelfTrainer, a confidence threshold was applied. Only examples for which the classifier predicts a class probability higher than this threshold are added to the training data. The algorithm ends if none of the predictions fulfill this criterion. This way the classifier only learns on its high-confidence predictions.

In the multi-label condition, there were two ways of calculating the confidence of a prediction. Since the ensemble classifier gives a confidence score for each label, one could either take the mean or maximum of all confidences. I decided to go with the mean confidence because a high mean confidence indicated an overall good prediction and reduced the risk of channels with wrongly predicted labels in the training data.

4.5 Comparison Experiment

The main research goal of this project was to find out whether the use of unlabeled data in a semi-supervised learning approach leads to better classifiers than supervised learning does.

Comparing algorithms effectively is not a trivial task. Thomas Dietterich explains in [52] the difficulties of making a reliable statement about the differences in performance between algorithms. First, it is important to distinguish between classifiers, which are functions that assign examples to one or multiple classes, and algorithms that construct classifiers from a set of examples and their corresponding classes.

To evaluate two classifiers, a common method is to hold out a part of labeled data and measure the performances of the classifiers on this test set.

Since the supervised method and the semi-supervised method presented in this report are algorithms that produce classifiers, we want to know, which algorithm produces the better classifiers. To train one classifier with each algorithm and comparing their performance on a test set is unfortunately not sufficient because the result of that comparison can be wrong by chance due to statistical variations.

There are usually four different sources of variation that could influence the result of a single comparison [52]. First, there is variation from the randomly

drawn test data. A classifier might perform better on one specific test set, while being actually worse if it was applied to the whole set of channels. Similarly, the randomly drawn training set can influence the performance of the resulting classifiers. Thirdly, some algorithms such as neural networks have internal randomness that can have an impact as well. Lastly, there is usually a certain rate of random mislabeling in the data.
The result from a single comparison between one classifiers from each algorithm is therefore statistically not very meaningful. The question arises how two algorithms can be compared.
Dietterich [52] suggests, that if the set of labeled data is large enough, one test set and a number of disjoint training sets should be created, on which each one classifier is trained. They can then be evaluated on the test set and their performances compared with a statistical test.
Unfortunately, the available data in this project did not allow for completely disjoint training sets because of its small size and therefore a resampling method was applied. A fraction of 15% of all labeled channels was retained as testing set and ten different training sets were created by sampling 90% of the remaining data ten times. Similarly, for the semi-supervised algorithms, ten sets of unlabeled channels were created with the same method but using the set of unlabeled channels to sample from.
This way two comparisons were performed, one for the binary classification task, and one for the multi-label task. In each condition, ten classifiers created from the supervised ensemble were compared with ten classifiers from the SelfTrainer algorithm, using the ensemble classifier to classify the unlabeled data. For the statistical comparison, a t-test was used to check the null hypothesis that the resulting classifiers from the two algorithms perform equally well.

5 Results

During the project, multiple experiments and tests were conducted. Results from cross-validation experiments were already presented in the Methods section. Results from the exploration of the data and the results of the main comparison experiment are presented separately in the following subsections.

5.1 Data Exploration

Some of the features were visualized to get an insight into the characteristics of the gathered data. The next subsection presents networks of related channels, subscriptions and cross-channel comments and the following one gives a visualization of the predictions of the different sub-classifiers of the ensemble classifier.

5.1.1 Network Features

To investigate the potential of the feature sets that are based on relations between channels, the channel relations were visualized as networks. For each relation feature, a graph was created that contains the channels as nodes and each edge represents a connection between them. These connections can either be a mention in the 'related channels' section, a subscription, or a comment from one channel under the video of another. Although all of these relations are directed, i.e. from channel i to channel j but not necessarily the other way around, the directionality is not depicted in the graphs to not clutter the image too much. The position of the nodes in the graph is determined via the Fruchterman-Reingold algorithm [53].

For each feature two networks were created and illustrated. In one, the colors of the nodes represent the binary classes 'political' and 'nonpolitical' and in the other, they represent the ideological labels. Since each channel can have multiple labels, for these visualization, only the most salient label was selected according to the schema found in appendix D of [6]. Nonpolitical channels were not included in the second network unless otherwise noted.

The goal of these visualizations was to get an impression of how clustered the channels are in the political network. For these features to be of discriminative value for an algorithmic classification, the hypothesis is that a channel is more likely to be connected to other channels of the same class.

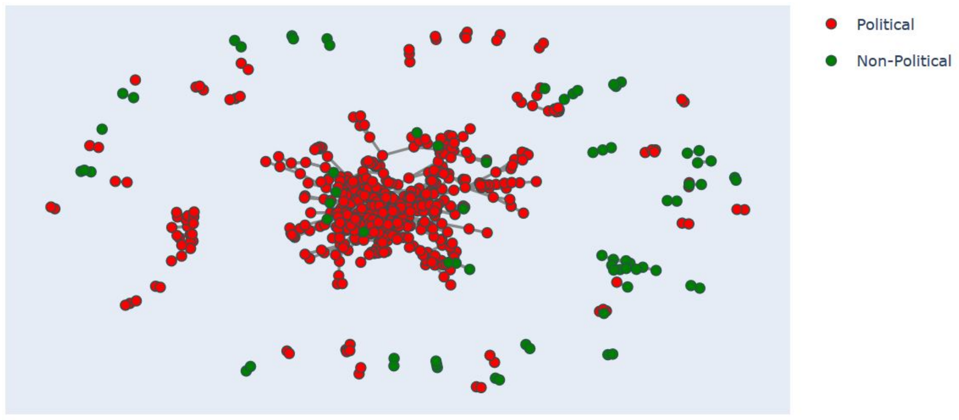


Figure 11: Association network of political and nonpolitical Channels. An edge from channel i to channel j means that i lists j in their 'Related Channels' section.

Related Channels Figures 11 and 12 depict the graph visualization of the associations feature. The graphs contain only channels that had connections (in our out-going) to other channels in the dataset. Both graphs show very prominently a core network of channels that connects the majority of political channels in the dataset.
In Figure 11 one can see that some nonpolitical channels have ties to political channels, but mostly they are connected to each other in pairs or small clusters. What also becomes very clear is that the nonpolitical channels are much less connected than political channels, as noticeably fewer nonpolitical channels appear in the graph.
If only the relations between political channels are considered, as in Figure 12, some clustering is recognizable. Channels that mostly produce videos on conspiracy theories seem to be tightly connected and separated from the main cluster. Socialist and other left-leaning channels form one large section in the lower right of that core cluster, and reactionary and other more right-leaning channels are grouped in the top left. Connected are these two sections mostly via a cluster of anti-theist channels.

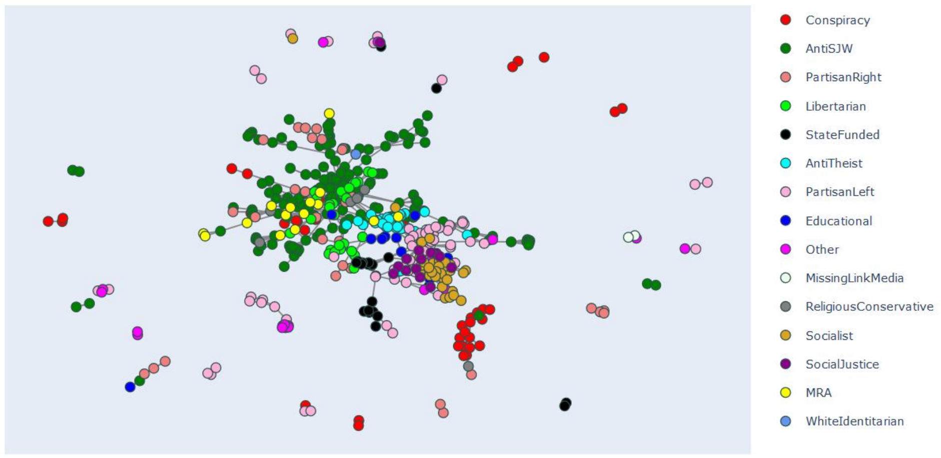


Figure 12: Association network of political channels. An edge from channel i to channel j means that i lists j in their 'Related Channels' section.

Subscriptions Looking at the subscriptions of political and nonpolitical channels reveals that channels' subscriptions are much more mixed and not as separated by political nature or ideology. Figures 13 and 14 display the nodes in-degree, i.e. the number of incoming ties as node size. The channels that were most often subscribed by other channels from the dataset were "Sargon of Akkad" (53), "StevenCrowder" (41) and "PowerfulJRE" (40), all of which are labeled, among others, with the 'AntiSJW' tag.
Both subscription networks show that cluster boundaries are heavily overlapping and in the case of the nonpolitical channels in Figure 13 a cluster is not even recognizable. The overlapping boundaries may partially be explained by the larger amount of connections for this feature that causes all nodes to be drawn closer together by the Fruchterman-Reingold algorithm. In the case of political/nonpolitical classes, it is also reasonable to expect that political channels subscribe to popular non-political channels and vice versa, which explains why the two classes are not very separated.

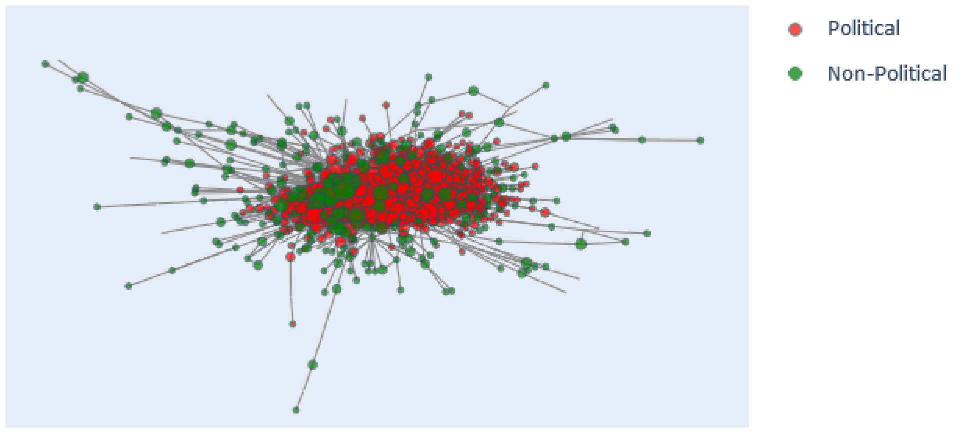


Figure 13: Subscription Network of Political And Nonpolitical Channels. An edge from channel i to channel j means that i subscribes to j.

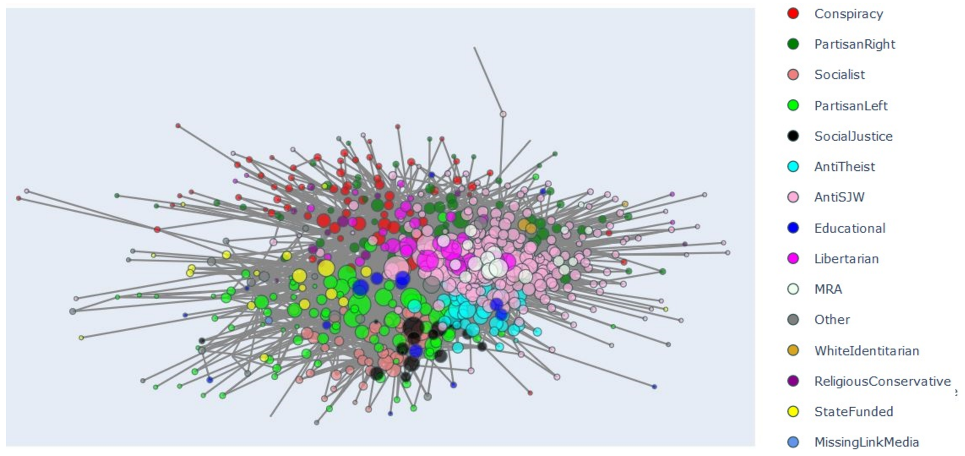


Figure 14: Subscription Network of Political Channels. An edge from channel i to channel j means that i subscribes to j.

Cross-Channel Comments Figures 15 and 16 depict the *cross-channel comment* feature and show the clearest clustering among the network features. Channel owners almost exclusively interact through comments with channels that are ideologically similar to themselves. Unfortunately, one can also see in Figure 15 that nonpolitical channels have almost no cross-channel comments to or from other channels. The graph includes 282 (36

nonpolitical) out of 1508 channels because only 124 of them have comments from other channels in the dataset under their videos.
The disparity between political and nonpolitical channels with regards to some of the network connections can be explained by the data acquisition process of the nonpolitical channel data. Since these channels were picked from completely different categories, it is unlikely that the channels heavily interact with each other in the comments or subscribe to each other. A large network of nonpolitical channels was therefore not expected. Unfortunately, this introduces some bias into the dataset for the binary classification task, as most of the channels that have no connections at all will be predicted to be nonpolitical because the classifiers learn that political channels have more connections. This explains the classifiers' high recall values for the nonpolitical class and low recall values for the political class seen in Table 6.
However, this problem is expected to be less relevant with a larger number of unlabeled channels because the proportion of channels with absolutely no connections is likely to go down when more channels are included in the dataset, making the network features more useful.

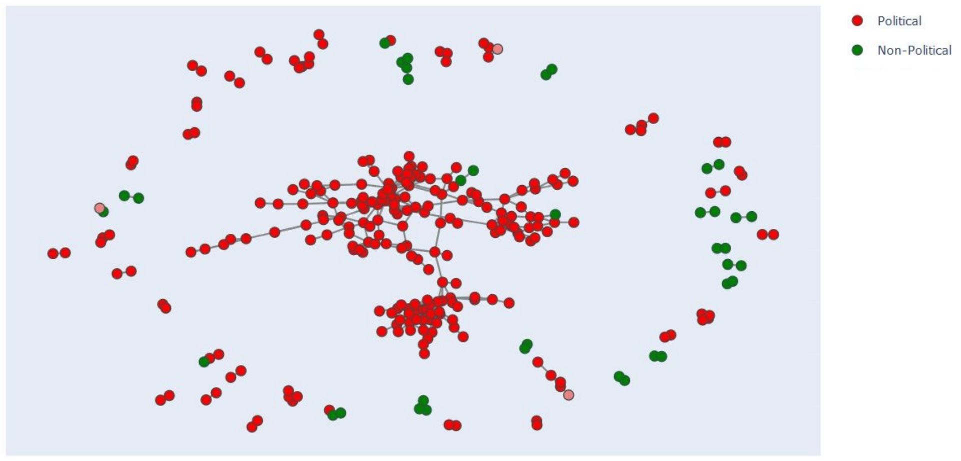


Figure 15: Cross-channel comment network of political and nonpolitical channels. An edge from channel i to channel j means that i left a comment under one of j's three most popular videos and this comment appeared in the first 100 most relevant comments.

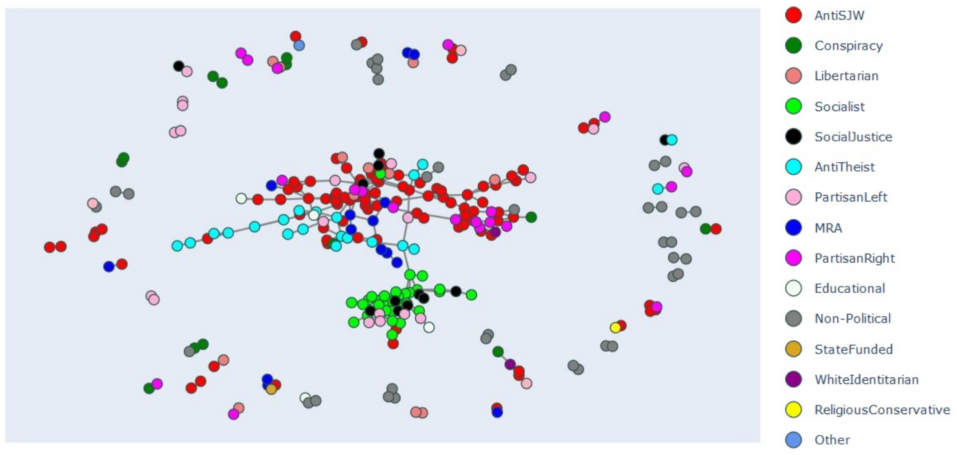


Figure 16: Cross-channel comment network of political and nonpolitical Channels. An edge from channel i to channel j means that i left a comment under one of j's three most popular videos and this comment appeared in the first 100 most relevant comments.

5.1.2 Ensemble Predictions

The vector of predictions from the ensemble's sub-classifiers that is served as input to the final logistic regression was visualized in 3d space to get an insight into how separated the classes are from each other. For the binary classification task, the predictions from the three text classifiers were separately plotted from the three network feature classifiers. Figures 17 and 18 depict the training data. For the text classifiers, the dimensions correspond to the predicted probability that the channel belongs to the class 'political', while they show the confidence score for the relation classifiers. The probability predictions were centered around zero which is why they do not fall into the range (0,1).
The predictions of the text classifiers on the training data are very clearly separated, which is not surprising since all classifiers have already seen these examples. For the relation data it is also expected that many data points fall together on one point of the feature space since they do not have any relations to other channels.
With regards to the test set, which the individual classifiers have not yet seen, the two classes are not as clearly separated. Some of the channels from the nonpolitical class fall in the space of the top right quadrant while also three of the political channels are deemed more nonpolitical by the

classifiers. For the relation classifiers, no overlap between the classes is observable except of course for those channels that fall together on the point where each channel without relations falls onto. The network classifiers can apparently very reliably separate political from nonpolitical channels as long as they have some information on what other channels they are related to.

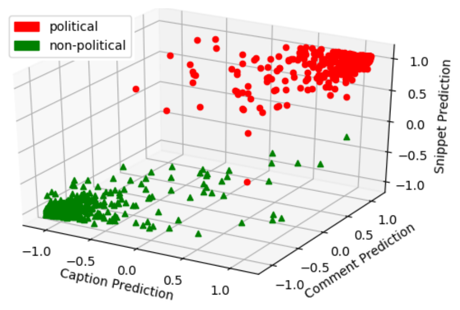


Figure 17: The class probability predictions of the caption, comment and snippet classifier on the training set.

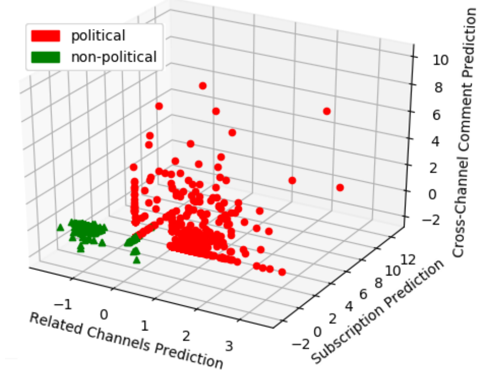


Figure 18: The class probability predictions of the associations, subscriptions and cross-channel comments classifier on the training set.

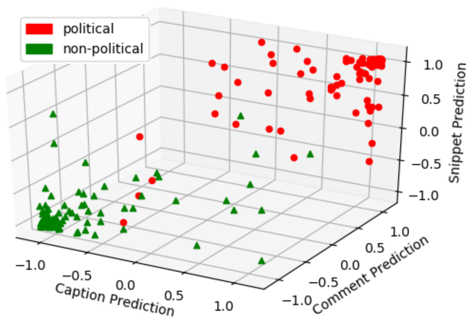


Figure 19: The class probability predictions of the caption, comment and snippet classifier on the testing set.

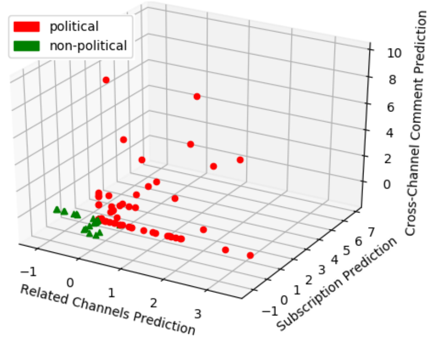


Figure 20: The class probability predictions of the associations, subscriptions and cross-channel comments classifier on the training set.

Multi-Label Classification Since the sub-classifiers in the multi-label classification do not predict a single probability value or confidence score, but one for every class, their predictions are not as easily translated into 3d. Principal Component Analysis (PCA) was employed as a feature reduction technique to project the predicted values from each sub-classifier to only three dimensions. The predictions that are visualized here stem from the ensemble consisting of the comment, snippet and associations sub-classifiers. Figures 21 and 22 show these predictions for the training and testing set respectively. Again, the color of each data point refers to the most salient label of each channel.

On the training set, the data points form relatively well-formed clusters. The final ensemble classifier is trained to separate each example with label i from all examples without that label. If channels with the same label appear close together in the PCA-reduced 3d space, we can infer that they are likely also separable for the logistic regression in the ensemble classifier.

In Figure 22 it can be seen that the data is still grouped according to their classes, but in general much closer together. Interestingly, there are some 'outliers'. Two socialist channels and one channel from the class 'AntiSJW' appear far outside of the main cluster. Unfortunately, this distance is not easily interpretable. The position of the two socialist labels might be explained by the fact that they are also labeled 'Revolutionary' which is a rather rare label in the dataset.

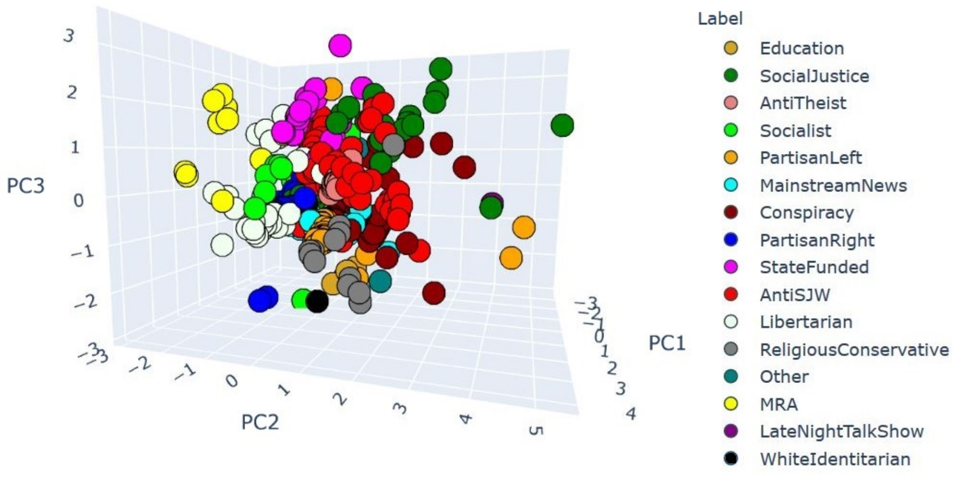


Figure 21: Multi-Label Class Probability Predictions of Ensemble Sub-Classifiers - Training Set. The axes show the three leading principal components.

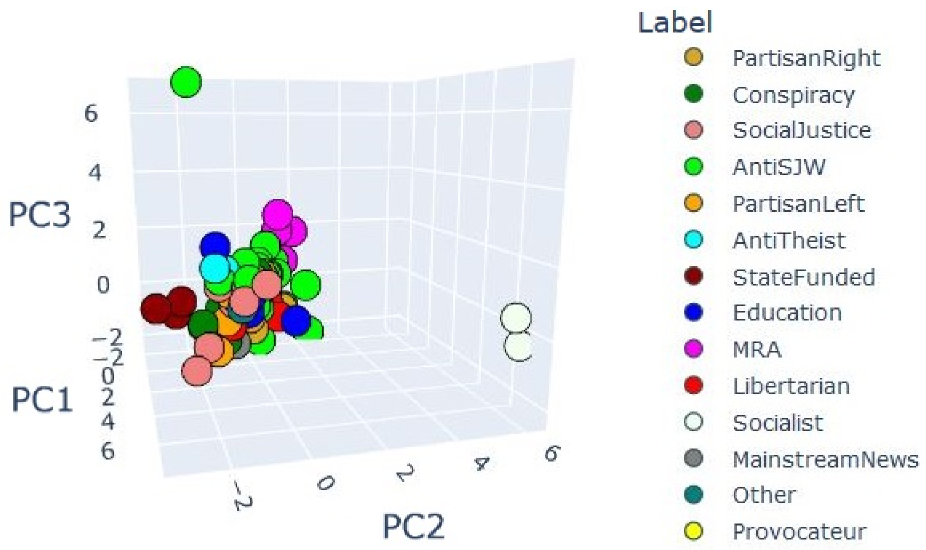


Figure 22: Multi-Label Class Probability Predictions of Ensemble Sub-Classifiers - Testing Set. The axes show the three leading principal components.

5.2 Comparison Experiment

Before I compared the classifiers resulting from the self-training, I first measured multiple variables during the process of the algorithm to get a better understanding of it. In each iteration of the self-training algorithm, the confidence score of the classifier's predictions on the unlabeled data was recorded. To see whether more training data makes the classifier better at discerning edge cases in the examples, the data points closest to the decision boundary were compared over time. Figure 23 shows that, indeed, the confidence score of the 'most difficult' cases rises steadily with an increasing number of training data.

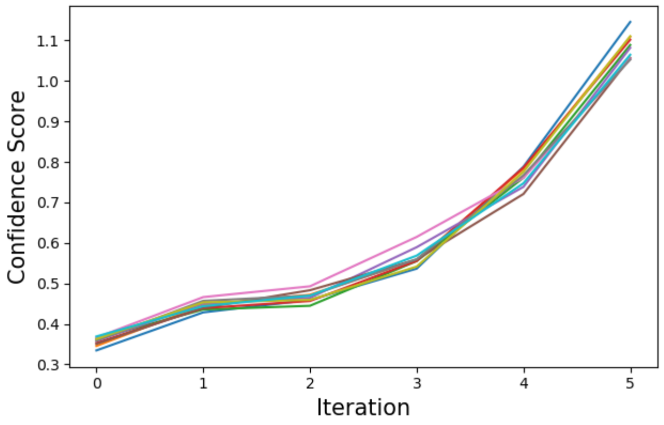


Figure 23: Mean confidence scores of 10% lowest confidence predictions. Ten different runs of the self-training algorithm with fixed number of iterations.

Additionally, the classifier's performance was evaluated on the test set after each iteration to see whether it is able to extract additional information from the unlabeled data. In Figure 24 the performance trajectory of 10 different runs of the self-training algorithm with fixed number of iterations can be seen. On average, the first iteration slightly increased the classifier's performance, only to drop with every further iteration. In some cases, an increase in performance is visible in some of the last iterations, but the negative trend is still apparent.

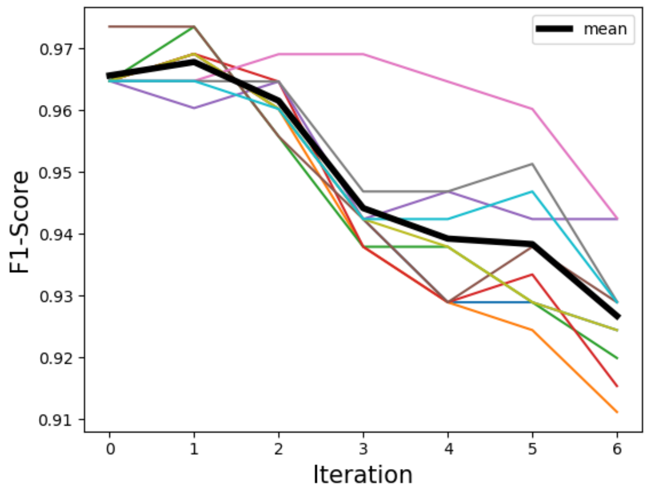


Figure 24: Performance on test set after each training iteration. Ten different runs of the self-training algorithm with fixed number of iterations.

Because the initial improvement of the classifier indicates that at least its high confidence predictions can be used for successful self-training, I implemented a version of self-training with a confidence threshold. Two thresholds were tested, 95% and 90%. Figure 25 and 26 present the plots for the respective test scores. If only predictions with a confidence over the threshold of 95% are chosen for training, the performance increases in some algorithm runs, and decreases in others but on average the F1-score on the test set only slightly increases. Overall, the fluctuation of the scores is much lower in this condition. The reason for that is that only a small number of examples are added to the training set. On average, 311 self-labeled examples are used for training in about 9 iterations of the algorithm. This increases the number of examples in the training set by 24.3%.
If the confidence threshold is lowered to 90% around 3259 unlabeled channels are used to train the classifier in the algorithm, increasing the number of training examples by a factor of 2.54. Unfortunately, the quality of the predictions suffers slightly, as Figure 26 shows.
Similar results could be observed for the multi-label classification task. If the self-training algorithm was run with a fixed number of iterations, the performance of the test set decreased with every iteration almost without exceptions (see Figure 27). For the self-training version with a confidence threshold, different thresholds had to be chosen than in the binary classification task. Instead of confidence thresholds of 90% and 95%, 80% and 83%

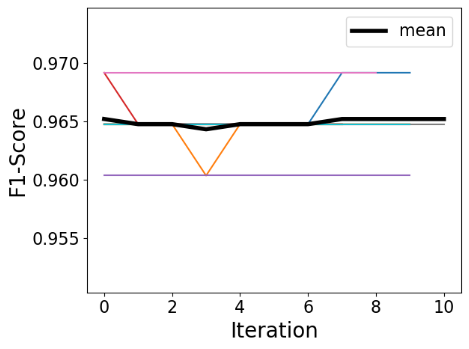


Figure 25: Ten different runs of the self-training algorithm with a confidence threshold of 95%.

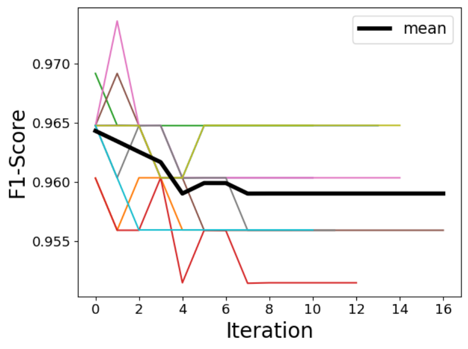


Figure 26: Ten different runs of the self-training algorithm with a confidence threshold of 90%.

were tested because the mean confidences were lower and they varied less, which is why a few percent increase already meant that either all unlabeled channels are added, or only a few dozen. But even for multiple runs of the algorithm with 83%, there were significant differences in the run time as Figure 29 shows. In some cases, the algorithm stopped after only a few iterations or it continued for over 10. The difference in added training examples is also significant. For runs that stopped early, between 3 and 8 examples were added. For all other runs, the number was between 2102 and 2118. Apparently, there were some examples in either the unlabeled or labeled set, that facilitated an increase in prediction confidence for a majority of the training examples.

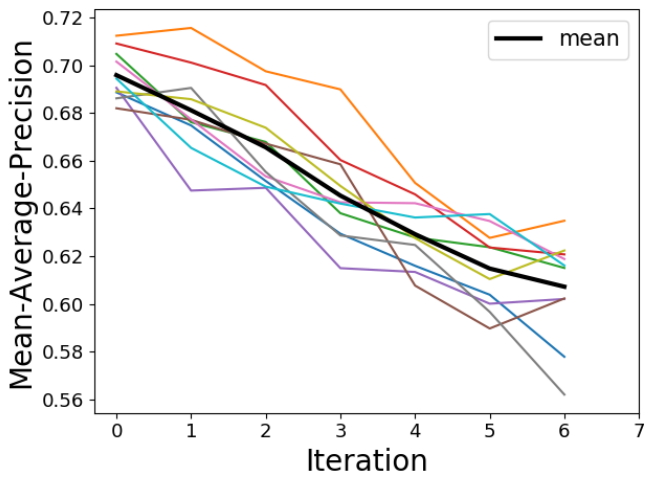
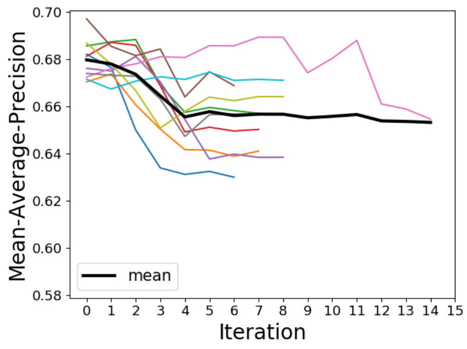


Figure 27: Ten different runs of the self-training algorithm on multi-label data with fixed number of iterations.



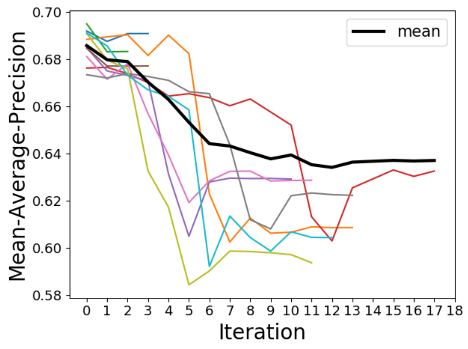


Figure 28: Ten different runs of the self-training algorithm with a confidence threshold of 80%.

Figure 29: Ten different runs of the self-training algorithm with a confidence threshold of 83%.

Main Results - Binary Classification All classifiers were evaluated again after convergence of the self-training algorithm and their scores on the test set compared with the supervised classifiers' scores. Figure 30 displays these final scores and the mean score of every condition. The normality

	Self-Training 6 Iterations	Self-Training 90% Threshold	Self-Training 95% Threshold
Ensemble SL	9.4692081e-10	0.0023552	0.6718178

Table 13: The P-values resulting from the statistical comparisons of the binary classification algorithms.

of the distribution of scores was checked with a Shapiro-Wilk test [54] before testing whether the mean performances of the self-trainer classifiers are significantly different from the classifier with a standard Student's T-Test [55]. The results (see Table 13) confirm that only self-training with a threshold of 95% performs equally well as simple supervised learning. For the two other conditions, the null hypothesis that the two algorithms perform equally well is rejected.

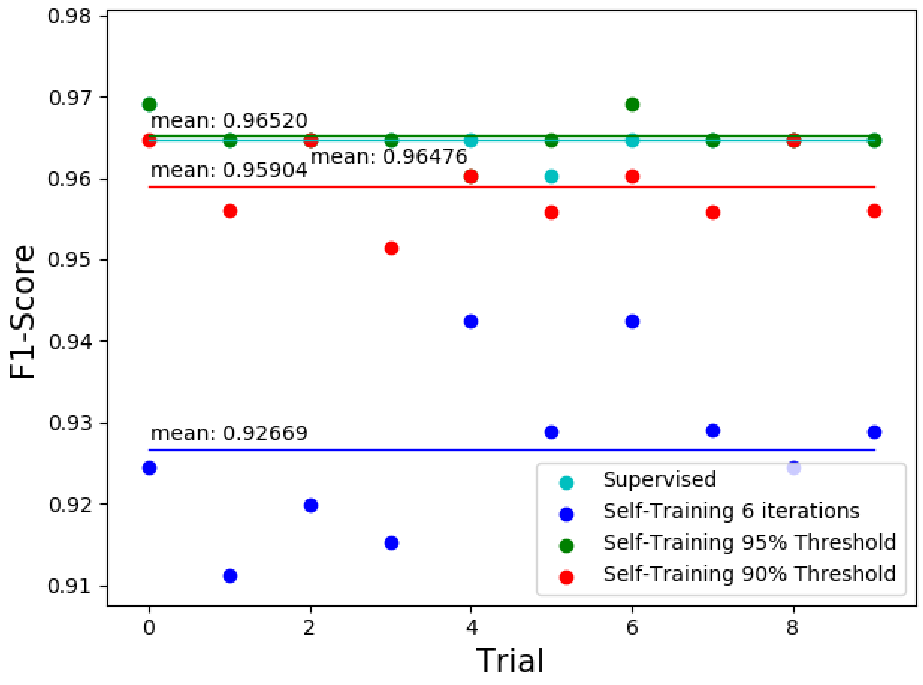


Figure 30: The performance on the test set by the classifiers trained only on the labeled data versus the classifiers trained with self-training algorithm.

Main Results Multi-label Classification The same experiments were performed for the multi-label classification task. Figures 27, 28 and 29 illustrate that the same negative trend that could already be seen for the binary classification appears again. Each step in which additional channels are labeled and added to the training decrease the quality of the classifier. This time, the effect is even stronger as before, as Figure 31 demonstrates. Some differences to the binary classification can be observed. First of all, the classifiers trained with a higher confidence threshold performs worse than those with a lower threshold. Both of them are even outperformed by the self-training algorithm with a fixed number of iterations, but only if the reduced ensemble is used inside the wrapper class. Interestingly enough, the reduced ensemble performed better when trained with self-training than the larger ensemble in the same condition, although it performs slightly worse when trained in a supervised manner only.
On average, the MAP scores for classifiers trained with the self-trainer algorithm were always significantly worse than the supervised classifiers, as Table 14 shows. The two versions of the ensemble classifiers perform equally well, as the p-value is larger than 0.05 and the null hypothesis is thus not rejected.

	Reduced Ensemble SL	Self-Training 6 Iterations	Self-Training Reduced Ensemble 6 Iterations	Self-Training 90% Threshold	Self-Training 95% Threshold
Ensemble SL	0.6359	1.253e-09	0.0022	3.804e-07	8.651e-05
Reduced Ensemble SL	-	2.7409e-09	0.00646	1.592e-06	0.000150

Table 14: The P-values resulting from the statistical comparisons of the multi-label classification algorithms.

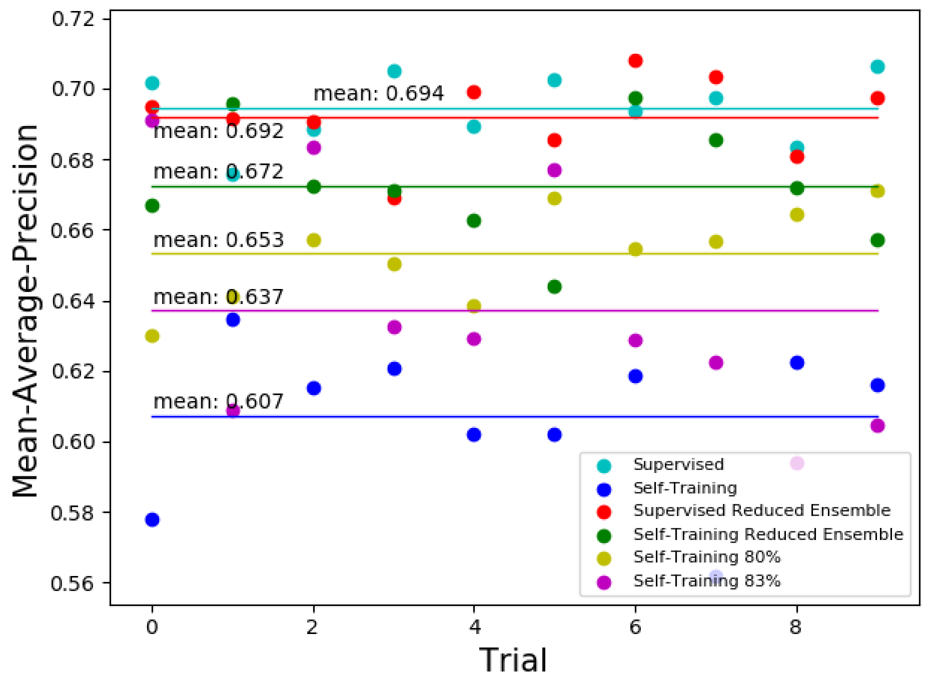


Figure 31: The performance on the test set by the classifiers trained only on the labeled data versus the classifiers trained with self-training algorithm.

6 Discussion

With this Master project my aim was to create a classifier with which political YouTube channels could be labeled, in order to facilitate research on the personalized YouTube algorithm. Since only little training data for such channels is available, the main research question of this project was whether unlabeled data could help with the training of machine learning classifiers in the domain of YouTube channels. For this purpose, the semi-supervised learning method *self-training* was tested and compared with a standard supervised learning approach.
Given the results of the experiments, the unlabeled examples appear to be of only very limited value for the training process of algorithmic classifiers, at least when used in the form of self-training. Most models that were trained with the self-training algorithm performed worse than the models that were trained exclusively with the labeled data points. As previous research has shown, self-training does not always work successfully, but it is usually applicable when the base classifier's high confidence predictions are mostly correct [43, 44].
The experiments carried out as part of this project could confirm this assumption. Only in one condition, where the most confident predictions of the already very accurate base classifier were used for training, did the performance increase slightly, but not significantly. However, this modest improvement came with a large reduction in unlabeled data that was actually used for training. In the multi-label task, not even the high confidence predictions helped in the classification because the base classifier was apparently not accurate enough. In none of the experiments did the utilization of the self-training algorithm significantly improve the classifier performance.
These results could be explained by the specific characteristics of YouTube channels' content. Especially for the multi-label classification task, it is possible that examples that lie close together in the feature space do not share the same label. The exploration of the ensemble data demonstrated that the text classifiers can not completely distinguish the classes. Two channels that produce videos about the same topic, but do so from two opposing perspectives, will still have a large overlap in their vocabulary, which text classifiers interpret as semantic similarity. This is especially the case when content creators cite each other and discuss the content of other channels, which is commonly done on YouTube. In these situations, even high confidence predictions of the text classifiers are not necessarily correct.
Already expecting this challenge, I included the relation-based features with the hope that textually similar channels can be disambiguated. While the

relation classifiers were indeed able to discern channels that were connected to other channels, some channels have no relations and thus unfortunately no feature information was available for these channels. The exploration has shown that only if that condition is fulfilled, channels can be reliably classified by the relation classifiers. Given that the self-training algorithm did not prove to be successful, the relation features were apparently not often enough available to clearly separate the classes in feature space and thus the class borders were not *smooth* enough for self-training to work.
As an open question remains, whether the best classifiers could still be used for the large-scale classification of the data obtained by the experiment on personalized recommendations. For this purpose, the binary classifier would be used to first filter out political channels that would then be labeled by the multi-label classifier. While the supervised binary classifier has shown excellent performance, even the multi-label classifiers that performed best on the test data still did not display very high accuracy. In particular, the low recall values imply that many relevant channels are missed in the analysis.
When interpreting the performance of the algorithmic classifiers, one should keep in mind that also human classification of political YouTube channels is not clearly defined. Political orientation is not a natural category that can be recognized by human raters but is to some degree always a subjective judgement. Even though the raters in [6] defined guidelines according to which the channels are labeled, a rater can still make 'errors' in their classification. To evaluate the reliability of human codings, the *Intraclass Correlation Coefficient* (ICC) can be computed. The ICC gives a measure for how well the labels actually capture information about the constructs that they are supposed to represent, rather than just noise from the coding process [56]. For 5 out of the 18 labels in the dataset used for this project, the ICC is rather low [6]. This means that the channel labels that were used for training the classifiers contained noise that the classifiers inadvertently learned. This naturally affected their performance when they were tested on the (also noisy) testing set.
A clear limitation of the ensemble classifier created for this project is that only up to three videos from each channel were used for determining the political ideology of the entire channel. These three videos do not necessarily represent the political content of the channel well. Gathering more videos could thus improve the accuracy of the predictions.
Another limitation is concerned with the normalization of the probability output vectors in the text classifiers. The implementation of the text classification library uses a softmax function, which makes the probability vector sum up to 1. The output of the text classifiers is used as the input for the

logistic regression in the ensemble, but because of this normalization, they do not necessarily produce consistent output. For a channel with 'Partisan-Left' as the only label, the text classifiers might predict a probability of 0.9 for that label and very low probabilities for all other labels. However, for a channel with the labels 'Partisan-Left' and, in addition, 'Educational', the predicted probability for these two labels will likely be around 0.5. Even though both channels belong to the same class, the logistic regression in the ensemble classifier interprets the second channel as less likely to be labeled 'Partisan-Left'. The noise introduced through this normalization presumably affects the accuracy of the ensemble and could be mitigated by a different implementation of the text classifiers.

Another challenge for successful classification, which may explain the relatively weak performance, and especially the low recall, is the class imbalance in the training data. Since the classifier learns in a 'One-vs-Rest' manner in which the training set is split into multiple binary classification tasks, large imbalances are created for labels that do not occur often. In the training data for the multi-label task, the worst splits had a ratio of 3 to 603. Proper handling of these imbalances with methods such as resampling [57] could improve the classifiers' performance.

Despite the non-optimal performance, there would exist ways to make the classifier usable for subsequent studies on YouTube channels and videos. For instance, Faddoul et al. [7] multiply the count of videos classified as conspiratorial with the class probability that the classifier outputs for the videos, which correlated with the actual likelihood of videos containing conspiracy theories. This way, not every video was correctly identified, but statistically, the count of classified videos was approximately correct. Similar approaches could be followed when the classifiers from this project are used in the analysis of further experiments.

With regards to the goal of the algorithmic classification, one difficulty arises from the transductive nature of the classifier. The model created in this project cannot classify channels that are not part of the dataset because the relation feature vectors do not necessarily cover those channels. If a new channel is added to the dataset, the subscription and cross-channel comment feature vector have to be created again, and the classifiers retrained. The channels collected daily through the experiment described in Section 4.2.1 can therefore not be classified every day without completely retraining the ensemble classifier. A more convenient option would be to collect data over a longer time-window and then classify and analyse them retrospectively.

7 Conclusion

With this Master project, I set out to design an empirical experiment on personalized YouTube recommendations that would be able to detect structural political biases in the YouTube recommendation system - if they exist. The labeling of the data resulting from such an experiment was identified as the key challenge for this undertaking. Since unlabeled data is readily available and has in previous studies helped with the training of algorithmic classifiers, the central goal of this project was the exploration of semi-supervised learning for the classification of YouTube channels.

To have an effective baseline to which a semi-supervised learning approach could be compared, I adapted a classifier used in [7] to identify videos spreading conspiracy videos, to the purposes of channel classification. As an additional feature for this classification, I gathered and extracted what I call *network features* that consist of relations between channels. An exploration of these features revealed a tight network of political YouTube channels, where channel owners consume and interact heavily with content that is similar to what they themselves create. Especially the network of related channels showed that producers of political content do not exist in a vacuum but actively connect to similar YouTubers.

The labeling of political channels was split into two sub-tasks. In a first binary classification task, channels that produce videos on political and social issues were distinguished from channels with nonpolitical content such as sports, movies, music, etc. In a second classification task, the channels identified as political by the first classifier were each labeled with possibly multiple tags defined by Ledwich and Zaitsev in [6]. These labels were created to explicitly capture the different categories that emerged in the political landscape of YouTube.

For each task, an ensemble classifier that consisted of sub-classifiers trained on six different feature sources (video closed captions, comments, video and channel snippets, related channels, subscriptions and cross-channel comments) was constructed. They were trained and evaluated on labeled examples that were published in [6].

As a comparison to these baseline classifiers, unlabeled data was used with a technique called self-training to train multiple classifier models that were also evaluated on a test set created from the labeled data. The classifiers are easily comparable because in self-training, the model trained on the labeled data in a supervised manner is further training itself with the unlabeled data. The comparison is thus facilitated because a possible positive effect of the unlabeled data can be observed directly.

Unfortunately, the unlabeled channels did not improve the performance on the testing set. A reason for this could be that the discriminative features used for the classification do not fulfill the so-called 'semi-supervised smoothness assumption', i.e. that two channels that have very similar features belong to two different classes. Unavoidable noise in the labeled dataset, stemming from the very difficult process of classifying political categories by hand, could explain why the smoothness assumption is violated.
Next to this exploration of classification methodology, I implemented a scraper that is able to gather personalized recommendations from YouTube, enabling experiments on the YouTube recommendation system that have not yet been conducted according to the literature. One such experiment was designed and proposed in this Master Project. The experiment is deployed under the name 'UserScrape' in the environment of the Recfluence project with support from the author Mark Ledwich. The project analyses the 'flow' of recommendations in-between political YouTube channels over time. As soon as enough data from the UserScrape experiment is collected the results will be analyzed. The Appendix of this report points to the repositories where the code used in this project and all data that was collected can be found.
This project took the first steps towards understanding personalised recommendations on YouTube, and also how semi-supervised learning can be used for this. However, further work is needed to gain deeper insights.
From the comparison experiments in this project, no wide-reaching statements about the effectiveness of semi-supervised learning in general can be made, as only one out of many methods was tested. There are at least two other methods that promise success with regards to the problem at hand. Da Silva et al. [43] use the network structure from Twitter follows and retweets for a *graph-based* semi-supervised approach. The network features that were extracted for this project could be used in the same way.
Another technique that could be explored is *Co-Training* in which two classifiers trained on different feature sets teach each other in a similar procedure as it happens in the self-training algorithm. According to Didaci et al. [44], this is very suitable for ensemble classifiers built from multiple feature classifiers, as it is the case in this project. However, this procedure operates under similar assumptions as self-training, so it might suffer from the same problems.
In addition to other semi-supervised techniques that could be examined further, it would also be worth exploring whether implementational variations in the classifier used as the comparison baseline would yield better, or qualitatively different, results. These variations could, for instance, include an

exchange of the algorithms used for the sub-classifiers as well as for the final ensemble classifier, the utilization of different combinations of features, or, instead, the application of an entirely different classifier architecture.
Further research should also explore the mechanisms that actually cause individuals to radically change their view points and behaviour through content consumed on platforms like YouTube. With this work, the focus was primarily on the recommendation algorithm, but how significant its role is in such a complex process in comparison to many other factors, is still unexplored and would require a multidisciplinary research approach.

References

[1] Y. Golovchenko, C. Buntain, G. Eady, M. A. Brown, and J. A. Tucker, "Cross-Platform State Propaganda: Russian Trolls on Twitter and YouTube During the 2016 US Presidential Election," *The International Journal of Press/Politics*, vol. 25, no. 3, pp. 357–389, 2020.

[2] R. K. Gibson and I. McAllister, "Do Online Election Campaigns Win Votes? The 2007 Australian 'YouTube' Election," *Political Communication*, vol. 28, no. 2, pp. 227–244, 2011.

[3] P. Lewis, "'Fiction Is Outperforming Reality': How YouTube's Algorithm Distorts Truth." *The Guardian*, 2018-02-02. Online available at `www.theguardian.com/technology/2018/feb/02/how-youtubes-algorithm-distorts-truth` (accessed: 14.08.2020).

[4] Z. Tufekci, "YouTube, the Great Radicalizer." *The New York Times*, 2018-03-10. Online available at `www.nytimes.com/2018/03/10/opinion/sunday/youtube-politics-radical.html` (accessed: 14.08.2020).

[5] M. Horta Ribeiro, R. Ottoni, R. West, V. A. F. Almeida, and W. Meira, "Auditing Radicalization Pathways on YouTube," *arXiv e-prints*, Aug. 2019. arXiv:1908.08313.

[6] M. Ledwich and A. Zaitsev, "Algorithmic Extremism: Examining YouTube's Rabbit Hole of Radicalization," *arXiv e-prints*, Dec. 2019. arXiv:1912.11211.

[7] M. Faddoul, G. Chaslot, and H. Farid, "A Longitudinal Analysis of YouTube's Promotion of Conspiracy Videos," *arXiv e-prints*, Mar. 2020. arXiv:2003.03318.

[8] F. Ricci, L. Rokach, and B. Shapira, *Introduction to Recommender Systems Handbook*, pp. 1–35. Boston, MA: Springer US, 2011.

[9] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich, *Recommender Systems: An Introduction.* Cambridge University Press, 2010.

[10] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen, *Collaborative Filtering Recommender Systems*, pp. 291–324. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

[11] J. Parsons, P. Ralph, and K. Gallagher, "Using Viewing Time to Infer User Preference in Recommender Systems," *AAAI Workshop - Technical Report*, July 2004.

[12] R. Burke, "Hybrid Recommender Systems: Survey and Experiments," *User Modeling and User-Adapted Interaction*, vol. 12, Nov. 2002.

[13] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using Collaborative Filtering to Weave an Information Tapestry," *Commun. ACM*, vol. 35, p. 61–70, Dec. 1992.

[14] S. Lee, J. Yang, and S.-Y. Park, "Discovery of Hidden Similarity on Collaborative Filtering to Overcome Sparsity Problem," in *Discovery Science*, pp. 396–402, Springer Berlin Heidelberg, 2004.

[15] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock, "Methods and Metrics for Cold-Start Recommendations," in *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 253–260, 2002.

[16] G. Shani and A. Gunawardana, *Evaluating Recommendation Systems*, pp. 257–297. Boston, MA: Springer US, 2011.

[17] R. Jiang, S. Chiappa, T. Lattimore, A. György, and P. Kohli, "Degenerate feedback loops in recommender systems," in *AIES '19*, 2019.

[18] E. Pariser, *The Filter Bubble: What the Internet Is Hiding From You.* Penguin UK, 2011.

[19] J. Davidson, B. Liebald, J. Liu, P. Nandy, T. Van Vleet, U. Gargi, S. Gupta, Y. He, M. Lambert, B. Livingston, and D. Sampath, "The YouTube Video Recommendation System," in *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, p. 293–296, Association for Computing Machinery, 2010.

[20] P. Covington, J. Adams, and E. Sargin, "Deep Neural Networks for YouTube Recommendations," in *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, p. 191–198, Association for Computing Machinery, 2016.

[21] Z. Zhao, E. Chi, L. Hong, L. Wei, J. Chen, A. Nath, S. Andrews, A. Kumthekar, M. Sathiamoorthy, and X. Yi, "Recommending What Video to Watch Next: A Multitask Ranking System," pp. 43–51, 09 2019.

[22] S. Menard, *Applied Logistic Regression Analysis*, vol. 106. Sage, 2002.

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011. Referenced Documentation (accessed: 14.08.2020):

1. `www.scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html`
2. `www.scikit-learn.org/stable/modules/generated/sklearn.metrics.average_precision_score.html`
3. `www.scikit-learn.org/stable/modules/generated/sklearn.multiclass.OneVsRestClassifier.html?highlight=onevsrest#sklearn.multiclass.OneVsRestClassifier`

.

[24] C. M. Bishop, *Pattern Recognition and Machine Learning.* Springer, 2006.

[25] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of Tricks for Efficient Text Classification," *arXiv e-prints*, July 2016. arXiv:1607.01759.

[26] T. Fawcett, "An Introduction to ROC Analysis," *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.

[27] K. Shin, J. Jeon, S. Lee, B. Lim, M. Jeong, and J. Nang, "Approach for Video Classification with Multi-label on YouTube-8M Dataset," in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, September 2018.

[28] Z. Wang, K. Kuan, M. Ravaut, G. Manek, S. Song, Y. Fang, S. Kim, N. Chen, L. F. D'Haro, L. A. Tuan, *et al.*, "Truly Multi-modal YouTube-8m Video Classification With Video, Audio, and Text," *arXiv preprint arXiv:1706.05461*, 2017.

[29] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," 2013.

[30] M. L. Hann, "Principles of Automatic Lemmatisation," *ITL Review of Applied Linguistics*, 1974.

[31] X. Zhu and A. B. Goldberg, "Introduction to Semi-Supervised Learning," *Synthesis lectures on artificial intelligence and machine learning*, vol. 3, no. 1, pp. 1–130, 2009.

[32] O. Chapelle, B. Scholkopf, and A. Zien, "Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]," *IEEE Transactions on Neural Networks*, vol. 20, no. 3, pp. 542–542, 2009.

[33] C. O'Donovan, C. Warzel, L. McDonald, B. Clifton, and M. Woolf, "We Followed YouTube's Recommendation Algorithm Down The Rabbit Hole." *Buzzfeed News.* Online available at `www.buzzfeednews.com/article/carolineodonovan/down-youtubes-recommendation-rabbithole` (accessed: 14.08.2020).

[34] J. Nicas, "How YouTube Drives People to the Internet's Darkest Corners." *Wall Street Journal.* Online available at `www.wsj.com/articles/how-youtube-drives-viewers-to-the-internets-darkest-corners-1518020478` (accessed: 14.08.2020).

[35] R. Lewis, "Alternative Influence," *Data & Society*, 2018.

[36] K. Munger and J. Phillips, "A supply and demand framework for youtube politics," *Preprint*, 2019.

[37] V. Simonet, "Classifying YouTube Channels: A Practical System," in *Proceedings of the 22nd International Conference on World Wide Web*, pp. 1295–1304, 2013.

[38] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan, "YouTube-8M: A Large-Scale Video Classification Benchmark," *arXiv e-prints*, p. arXiv:1609.08675, Sept. 2016.

[39] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale Video Classification with Convolutional Neural Networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[40] B. Banerjee, "Machine learning models for political video advertisement classification," Master's thesis, Iowa State University, 2017.

[41] L. Qi, C. Zhang, A. Sukul, W. Tavanapong, and D. A. Peterson, "Automated Coding of Political Video Ads for Political Science Research," in

2016 IEEE International Symposium on Multimedia (ISM), pp. 7–13, IEEE, 2016.

[42] Y. Dinkov, A. Ali, I. Koychev, and P. Nakov, "Predicting the Leading Political Ideology of YouTube Channels Using Acoustic, Textual, and Metadata Information," *arXiv e-prints*, p. arXiv:1910.08948, Oct. 2019.

[43] N. F. F. Da Silva, L. F. S. Coletta, and E. R. Hruschka, "A Survey and Comparative Study of Tweet Sentiment Analysis via Semi-Supervised Learning," *ACM Comput. Surv.*, vol. 49, June 2016.

[44] L. Didaci and F. Roli, "Using Co-training and Self-training in Semi-supervised Multiple Classifier Systems," in *Structural, Syntactic, and Statistical Pattern Recognition*, pp. 522–530, Springer Berlin Heidelberg, 2006.

[45] ThoughtWorks, "Selenium." GitHub repository, `https://github.com/SeleniumHQ/selenium` (accessed: 14.08.2020), 2018.

[46] F. Figueiredo, F. Benevenuto, and J. Almeida, "The Tube Over Time: Characterizing Popularity Growth of YouTube Videos," pp. 745–754, Jan. 2011.

[47] G. Reemer, "ChannelCrawler." `https://channelcrawler.com/eng` (accessed: 14.08.2020), 2019.

[48] A. Golub, "YouTube Explode." `https://github.com/Tyrrrz/YoutubeExplode`, 2016.

[49] S. Bird, E. Klein, and E. Loper, *Natural Language Processing With Python: Analyzing Text With the Natural Language Toolkit.* O'Reilly Media, Inc., 2009.

[50] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching Word Vectors with Subword Information," *arXiv e-prints*, July 2016. arXiv:1607.04606.

[51] M. E. Tipping and C. M. Bishop, "Probabilistic Principal Component Analysis," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 61, no. 3, pp. 611–622, 1999.

[52] T. G. Dietterich, "Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms," *Neural computation*, vol. 10, no. 7, pp. 1895–1923, 1998.

[53] T. M. Fruchterman and E. M. Reingold, “Graph Drawing by Force-Directed Placement,” *Software: Practice and Experience*, vol. 21, no. 11, pp. 1129–1164, 1991.

[54] S. S. Shapiro and M. B. Wilk, “An Analysis of Variance Test for Normality (Complete Samples),” *Biometrika*, vol. 52, pp. 591–611, Dec. 1965.

[55] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. Jarrod Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, İ. Polat, Y. Feng, E. W. Moore, J. Vand erPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python,” *Nature Methods*, 2020. Referenced Documentation (accessed: 14.08.2020): `docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_ind.html`.

[56] R. Landers, “Computing Intraclass Correlations (ICC) as Estimates of Interrater Reliability in SPSS,” *The Winnower*, June 2015.

[57] A. More, “Survey of Resampling Techniques for Improving Classification Performance in Unbalanced Datasets,” *arXiv e-prints*, Aug. 2016. arXiv:1608.06048.

8 Appendix

A

Scripts that were used to collect and preprocess feature data in addition with all scripts used to train and compare the classifiers and ultimately visualize the results can be found in `www.github.com/antonlaukemper/YouTube-Recommendation-ThesisProject`
Labeled and unlabeled data that was used for training the classifiers can be downloaded via `drive.google.com/file/d/1oKKUhTh1tt8OyZFoA4cgg_HhSM6Z9MSz/view?usp=sharing` or requested by mail at `anton@laukemper.it`.
The selenium scraper with which personalized YouTube recommendations can be gathered and the implementation of the experiment described in Section 4.2.1 can be found in the folder 'userscrape' in `github.com/markledwich2/Recfluence`.

B

During the acquisition of nonpolitical channels, each channel was briefly examined with regards to possible political content before it was added to the dataset. Some political channels were missed in this filtering and were only discovered during later processing steps. These channels were[15]:

1. `www.youtube.com/channel/UCwd_sSDZ8EQt6SEeOO2tBRA`
2. `www.youtube.com/channel/UCD4qMoFPUEzSn4CurFLJkIQ`
3. `www.youtube.com/channel/UCBizESL5Wcvcxs2eB6h0txQ`
4. `www.youtube.com/channel/UCOYm0tm472SIMVan6KSOuKg`
5. `www.youtube.com/channel/UC1iFTPspSKcb4vb1N7czmRQ`
6. `www.youtube.com/channel/UCMljRGC0eBJrxbUorWEnasg`

The first two channels in this list also appear in the set of labeled political channels and were thus simply removed from the set of nonpolitical channels. Channels 3 and 4 produce videos that can be considered political, but because I did not have the means for a proper labeling process with multiple labelers, these channels were also deleted. The last two channels were more unambiguously political and could be confidently labeled 'Educational', which I did after consulting the original labelers of [6].

[15] All accessed on 14.08.2020