Highlight detection in live streams using audience reactions with transformer language models

Maximilian Lukas Gutsche

MSc. Dissertation

Department of Artificial Intelligence

Institute of Linguistics and Language Technology

Faculty of Information and Communication Technology

University of Malta

September, 2022

Supervisors:

Dr. Marc Tanti, Institute of Linguistics and Language Technology, University of Malta

Prof. Dr. Eneko Agirre, Computer Science Faculty, University of the Basque Country

Submitted in partial fulfilment of the requirements for the degree of

Master of Science in Human Language Science and Technology (HLST)

FACULTY OF
INFORMATION AND COMMUNICATION TECHNOLOGY

UNIVERSITY OF MALTA

Declaration

Student Name:	Maximilian Lukas Gutsche
Course Code:	CSA5310 HLST Dissertation
Title of work:	Highlight detection in live streams using audience reactions with transformer language models

Signature of Student: [illegible]

Date: September 12, 2022

Supervisors

Dr. Marc Tanti

Institute of Linguistics and Language Technology

University of Malta

and

Prof. Dr. Eneko Agirre

Computer Science Faculty

University of the Basque Country

Acknowledgements

During the inception, planning and execution of this research endeavor which culminated in this master thesis, a multitude of people have supported me in my work. I would like to express gratitude for their continued assistance and encouragement, lending me a helping hand whenever I needed it:

To Dr. Marc Tanti, my supervisor from the University of Malta, who from the beginning showed great interest in my research activities and provided invaluable feedback for every part of my journey. His commitment to his students is unrivaled and paired with his knowledge this makes him a great teacher to work with.

To Prof. Dr. Eneko Agirre, my supervisor from the University of the Basque Country, who encouraged me in my research interest and was always quick with most useful insights to help me out when I required his assistance.

To Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI) who provided access to their cluster for me to train my deep learning models.

To Sepideh, Sarah and Bernadeta, who shared the experience of writing a thesis with me and were wonderful companions for exchanging troubles and doubts, finding encouragement and spending a great time together.

Des Weiteren gilt meine Dankbarkeit:

Meiner Mutter Karoline, die mich in all meinen Bestrebungen vorbehaltlos unterstützt und mir dabei hilft meine Ziele zu erreichen.

Schließlich, meiner Großmutter Elisabeth, die mir eine Bleibe für die Zeit der Fertigstellung meiner Masterarbeit gab und immer ein aufmunterndes Wort für mich bereit hielt.

Abstract

Livestreaming of e-sports events has become very popular in recent years, with millions of people watching livestreams of competitions and commenting synchronously in chat rooms. As an effect of the rise of e-sports, there is a demand for match highlight videos. These videos, which consist of the most exciting moments in a match, help followers of the sport to stay up-to-date with or relive past games. Since their manual creation is time intensive, automatic and semi-automatic approaches for highlight detection in live streams have been devised. In this work, we suggest a novel transformer based approach to highlight detection. We employ the audience reactions found in live stream chat in order to find gripping segments of livestreams. To this end, we suggest an approach which combines contextual transformer embeddings with additional temporal features of the chat. We pre-train a language model for the domain of live stream chat in the game League of Legends and employ it on this task. For training this transformer language model, we collect a corpus from a popular livestreaming platform which contains audience reactions to competitive League of Legends matches. With our new model, we achieve an improvement over the state of the art of 0.01 f-score. We provide a new corpus for the domain and make available our pre-trained language model, which we call TwitchLeagueBert.

Contents

1 Introduction

Competition in sports has always attracted audiences for watching the spectacle of the best athletes in their field fighting for a title. This fascination has also endured through the digitization of many fields, including sports events. Nowadays, e-sports events, which are competitions in computer gaming, fascinate millions of people who do not only watch in person at a venue, but also follow live broadcasts from home.

One of the platforms which have specialized in livestreaming of computer game content is Twitch[1]. It is the largest platform with many competitors in the field like YouTube live, Huya TV and trovo[2]. These services all operate similarly, providing a live stream of content on the internet for viewers to watch. In addition, many platforms also offer the option to discuss the content concurrently in a chat room. These chat rooms provide an immediate way for viewers to interact with the broadcaster and other viewers, making the watching experience interactive. In Figure 1 we show an example of livestreamed League of Legends gameplay with the live chat room.

With many computer games being released every year, the potential for competitive gaming is huge. League of Legends (LoL) [3] has evolved into one of the most played and followed games, with 180 million active players in 2022 and a record audience of 99.6 million viewers for the world championships in 2018. It is a multiplayer online battle arena (MOBA) where two teams of five players compete against each other. Each player controls one character, a so-called hero, and their collective goal is to destroy the enemy base [4].

With this huge interest in playing and following competitive gaming, the demand for content besides the actual game and the long-running live streams is given. Thus, highlight videos of e-sports matches also enjoy popularity. These videos compile the most gripping scenes of a match into a shorter format, allowing fans to rewatch matches and follow missed events. A popular way of distributing these videos is through the video publishing

[1] https://twitch.tv
[2] https://escharts.com/platforms
[3] https://www.leagueoflegends.com/en-us/
[4] https://en.wikipedia.org/wiki/League_of_Legends

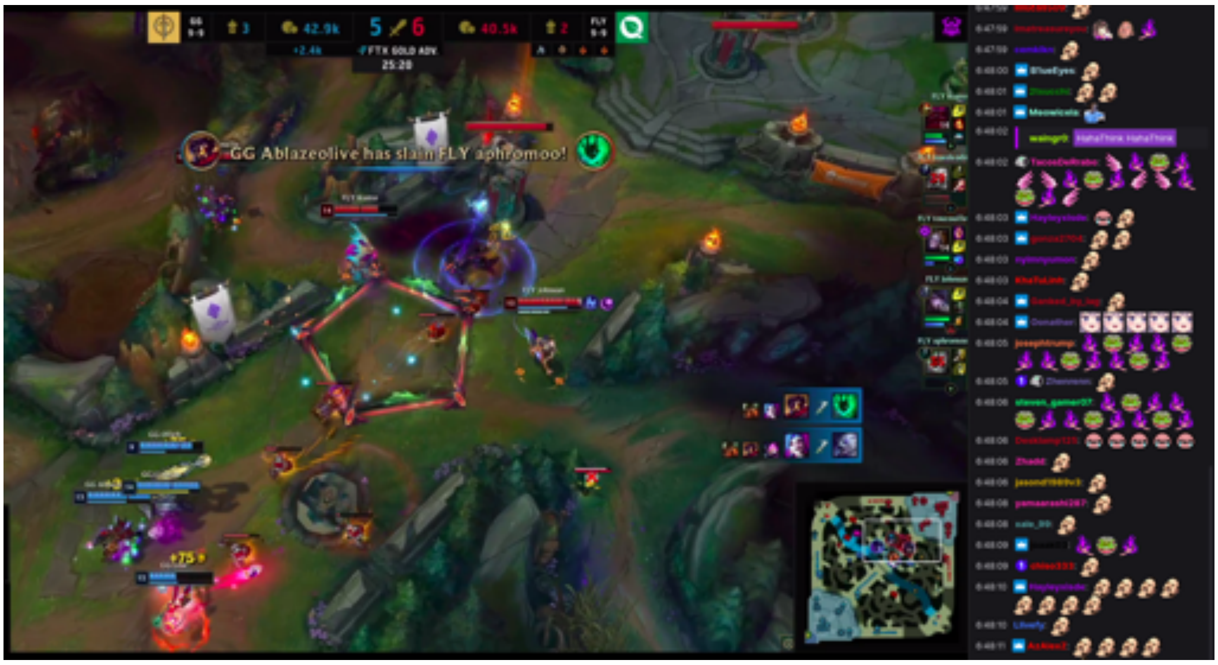

Figure 1: Watching experience of a League of Legends tournament stream on Twitch. On the left, we see the live stream video and on the right there is the live chat panel. We see a big fight with multiple heroes, resulting in the defeat of one hero. The chat room is full of emotes expressing different sentiments towards the fight.

platform YouTube[5]. Many channels, dedicated to uploading highlight reels, like Onivia[6] and Kaza[7], provide a well accepted service to the LoL e-sports community. Highlights in LoL competitive matches typically include scenes with many kills, important in-game events, tactical choices of heroes and gear, as well as the conclusion of a game where the winner is decided. However, not each one of these scenes is of interest to the audience and there are other actions that are highlight worthy. Thus, choosing entertaining highlights is not trivial.

Identifying highlights and cutting a video can be time-consuming and requires experts who have to be familiar with the game as well as the audience in order to provide high-quality videos. Thus, there is a potential in machine-aided or fully automated generation of these videos for speeding up the process and allowing organizations who may not have a team of highlight annotators to provide a good service to their audience.

In recent years, there have been efforts in devising techniques for automatic analysis

[5]https://www.youtube.com/
[6]https://www.youtube.com/c/OniviaLECLCSLCKLPLHighlights
[7]https://www.youtube.com/c/LoLEsportsHighlights

of live streams and creating highlight videos from them. These techniques employ the video content [14] [19] [40], audience reactions in the form of chat messages [16] [34], broadcaster commentary and facial expressions [37] and combinations thereof [6] [11] [25]. For creating highlight videos from a longer live stream, interesting segments have to be identified, cut from the live stream footage and stitched together into a video. Typically, the order of the highlights is preserved. This means that the correct choice of highlight segments is most important. While the obvious carrier of information is the video itself, it has been shown that using audience chat reactions improves the results on the highlight detection task [7] [11] [25].

For processing textual information, different machine learning techniques have been prevalent over the last decade. Currently, the most successful deep learning approach is the transformer architecture [20]. Typically, applying a transformer to a task involves the self-supervised pre-training of a language model on a large amount of text and then fine-tuning it to a specific task on a smaller, annotated dataset [35].

In this work, we aim at investigating how transformer models can be employed in creating highlight videos from League of Legends e-sports live streams. We focus on the audience reactions which are present in live stream chat.

For this investigation, we raise the question of which model configurations are helpful for the highlight detection task. We consider different aspects of chat messages like their content, temporal relation to the live stream video and how to encode this information. We test if an approach, previously devised for a recurrent neural network architecture, is also suitable for transformers and present a new approach which we test with different pre-trained language models.

Additionally, we seek to answer whether a specifically pre-trained language model on League of Legends live stream chat is better suited for this task than a general pre-trained language model. In order to answer this question, we collect a corpus of English live stream chat and train our new language model which we call TwitchLeagueBert on this corpus. We test it against RoBERTa[27].

We structure our work as follows: In Chapter 2 we provide an overview of the current research in the field of highlight detection on live streams. In order to provide a better understanding of the livestreaming domain and to gain insights into how live stream chat is used, we also consider other work, which generally outlines communication behavior of viewers and their use of language. Additionally, we provide information on current transformer techniques and how they can be employed, which we will leverage later on.

Next, in Chapter 3 we analyze the highlights dataset, which we use as our labeled training corpus and show how we collect our pre-training corpus which we call Twitch LoL corpus. In Chapter 4 we provide considerations for baseline models, which help us gauge model performance, and their features. We show how we set up pre-training for TwitchLeagueBert and describe our highlight detection models. All these models are trained and evaluated in Chapter 5 where we present and compare their performance in order to determine which transformer models are applicable to the highlight detection task. We summarize our findings in Chapter 6 along with providing future research directions in this field.

2 Background

To start off this investigation, we use this chapter to provide an overview of the fields of highlight detection and transformer models. To this end, we give an overview of live stream viewer interactions and describe which style of language is found in live stream chat in Section 2.1. For the central task of this thesis, we explore in Section 2.2 how this and other information is leveraged for the highlight detection task, and which techniques have been employed on it in the past. Finally, in Section 2.3 we introduce transformer models and how they are applied as the central technique which we employ on the problem of highlight detection.

2.1 Research on Livestreaming and Language of live stream chat

Like many other social events, e-sports livestreaming is a field of study in academia. In an early study, Kaytoue and Silva [22] characterize the e-sports livestreaming community and find that ranking of live streams can be based on viewer count. Like traditional sports events, many of the e-sports event are held on weekends, emphasizing the entertainment nature of the sport. Their research is based around viewer numbers and characteristics derived from those, especially popularity of live streams and their ranking between them. They find a ranking method for live streams and predict popularity. Here, live stream chat is not yet considered.

From the study of live stream interactions, we take a turn into the field of chat research on live stream audiences. Here, we identify work that links the chatting behavior of audience members and groups to the events that occur in e-sports live streams. Through the presentation of work characterizing gaming audiences and massive chat rooms, we draw conclusions to how messages can be interpreted and bundled for processing. Finally, the importance of emotes, small images which can be used inline in text chat to express emotions with one single symbol, and their relevance is shown. These are easy and fast to insert with standardized code words and provide common ground for expressing emotions in this text-based medium.

Bulygin et al. [5] use audience chat behavior to investigate common reactions to the content of the live stream in e-sports event. They show which topics are typically expressed by game audiences in reaction to which types of events, and that there is a direct correlation between events and audience reactions as a whole. They model topics in the chat in different stages of a tournament, providing insight on when which topics are prevalent. Viewers tend to express boredom when there is a lull in gameplay, while cheering and reactions to stream and game specific events are linked to events. General discussions about teams, players and other topics are often found. Their analysis is based on a Dota tournament, a competition in a game which is similar to the subject of this work.

This finding, that chat reactions correlate with events in a live stream, is also shared by Musabirov et al. [31] who employ different sentiment analysis approaches to livestreamed real-live events. They mostly use the meanings of emotes for finding the polarity of live stream comments. In general, the averaged sentiment scores across comments in a defined time frame, match the events which are shown on screen.

Barbieri et al. [1] investigate the usage of Twitch emotes by creating a Bi-LSTM (bi-directional long short term memory) model for emote prediction. Its main task is predicting emotes from their message context. Although they only use single messages as context, the model achieves improved results over bag of words (BOW) and Skip-gram based baselines. Their second task of "trolling" prediction leverages Kappa-based emotes as labels for utterances which show trolling, thus making it an easier variant of the first task. Nevertheless, they show that emotes depend on their context and serve as an important communicative device in the Twitch chat.

Emotes do not only help to emphasize a concept in a message in chat communication, but are also used for abbreviation purposes. Usually, the need for brevity arises from chat rooms with many concurrent users writing a high number of messages [23]. These so-called massive chats, with 10s of thousands of participants, are the subject of investigation in [10] in comparison to smaller chat rooms with fewer than 2000 participants. They investigate the techniques that are used by crowds to keep communication working at a large scale. To this end they identify voices present in chat, which may or may not coincide with participants and use counts of lexical items present in messages to define where concepts are repeated. As a result, they identify three practices of coherence in massive Twitch chat: Shorthand, voice-taking and bricolage. Short handing is found to be used for generally accepted terms, the meaning of which is agreed upon by a community. Certain recurring game events may be expressed in that way, emote usage falls into that category as well. A similar, agreed-upon concept is used in voice-taking, which occurs when chat participants talk in a way that is associated with a certain emotional expression, character, or idea. Users adopt this certain voice and thus express their opinion through a common voice, repeating the same emotes, phrases or even entire messages, giving it more importance and recognition than any single individual utterance. We see this in the chat window of Figure 1 where the same emote is repeated by multiple different users. This phenomenon helps to gauge general points of view in live chat. Finally, bricolage, combining a limited repertoire of terms into repeating messages, is found in massive chats. This is identified as a technique for making messages more readily understood, as the items used are known by the community and easily decoded. This technique is also called copypasta by the community and typically is an expression of boredom [31]. These findings suggest that massive chat is more than individuals expressing their emotions and ideas. Thus, massive chat as present on many e-sports Twitch channels allows for summarizing individual utterances under common topics, resulting in a less fractured and more readily understandable chat room as a whole. However, this kind of understanding is only granted when the community and their particular way of expressing themselves is adopted by the reader.

Some work in this area has been conducted on another type of video content with user comments. So-called time sync video comments [47] are added by users to pre-recorded video material and displayed for other users. In contrast to livestreaming, these comments are not created at the same point in time as the airing of the video, but are nevertheless considered reactions to consumed video content. A temporal difference is, that comments are shown for a fixed amount of time on screen. We thus consider work on these videos [48] [28] relevant to our topic. This is because they also research text based reactions to video content. While the immediacy of live reactions is lost and the way of displaying reactions is different, they nevertheless allow for drawing parallels to live stream audience reactions.

2.2 Live stream highlight detection

The central information carrying part of a live stream is the video. Many approaches to highlight detection use the video information in their models. This seems to be the obvious source, as the live stream experience is centered around video consumption. Typically, individual frames of a video are classified as a highlight or non-highlight, which when combined produce shorter segments from a larger video [40]. In this scenario, feature extraction from frames and highlight detection may be part of separate modules and improved with attention mechanisms [19]. Gunawardena et al. [14] extract static and dynamic features from sequences of frames and combine these into self-organizing maps to classify highlights.

As the term highlight can be subjective to the viewers, it is of interest to also include the audience reaction to provide more information to an automatic system for highlight detection. To this end, chat messages are commonly used in the live stream setting. Chu and Chou [6] explore the use of chat statistics like message counts and train a support vector machine to add more information to their video-based highlight detection. Xian et al. [48] incorporate meaning by using Latent Dirichlet Allocation (LDA) on keywords extracted from time sync comments on videos to detect shot boundaries and highlights. Fu et al. [11] go beyond that and create a recurrent neural network (RNN) for chat and combine it with a convolutional neural network (CNN) for visual information. They try to exploit some semantic information and show that character-based models work better than word-based models. Their idea is extended with word embeddings by [16] based on Word2Vec. They use an RNN model with pre-calculated embeddings for words. Song et al. [39] even generate live stream chat specific embeddings for emotes from Twitch and achieve to incorporate even more word meaning into their system for extracting "epic" moments.

Another approach uses video data combined with in-game event data extracted from screen [21] and extract highlights based on changing win-loss probabilities in a match. Facial expressions of individual streamers are also among the features which have been investigated for highlight detection [37]. In our data this information is not always available, and we thus do not consider this kind of information. What would be interesting to address though is the e-sports commentator's voice. Ringer and Nicolaou [37] and Sun et al. [41] show how this information can be used.

Although we see a variety of techniques and features used, when it comes to using audience chat reactions in these models, all the approaches face similar issues. In the following, we shall outline alignment of chat and video content and the unit of classification problems with respect to the task of highlight detection.

2.2.1 Aligning chat and video content

While video is made up of individual frames, text chat is based on messages. On Twitch, both of these can be linked using the time stamps of when they were created. For a video frame, this is the time when it is created in a live stream. For a chat message, this means when it is displayed in the chat room. Although it is tempting to simply use the time stamp information, a chat message is generally regarded as being a reaction to a past event in the live stream or being part of a general discussion in the chat room which may be less closely linked to the live stream content [10]. So, there certainly is some form of delay between video and chat messages. When chat messages are to be used for the segmentation of the video, a link needs to be established. A fixed offset or time window is often determined empirically, which provides a simple way of linking messages to video.

However, this may fall short when the reactions are more or less delayed. A less delayed reaction can be a short message with only an emote or few words, whereas a more delayed reaction may be a well articulated comment on the action in the live stream. It is favorable to be able to identify which comments are linked to which video segment as accurately as possible, in order to provide the correct information a classification algorithm needs for the segment to classify it as a highlight or non-highlight.

Ping and Chen [34] identify messages with similar topics and subsume each topic under a lexical chain, which aims at representing a conversation or reaction. They use the time stamp of the first comment to link the whole chain to the video.

2.2.2 Unit of classification

Having explored the different types of information which are useful for highlight detection, one of the other fundamental questions of this work is how a live stream can be segmented into highlights. Ideally, a video is cut into highlight and non-highlight parts directly. However, in practice the features of the live stream are calculated for a certain unit or span of units of where multiple ones make up a highlight segment[40]. This is done to create highlights of differing lengths, which is preferred. Here, a unit of interest may be a video frame [11] or a time window in the broadcast in terms of n seconds [28]. Chu and Chou [7] extract chat-based information in one second windows and combine those with additional information from the video.

Abstracting away from the technical aspect of the medium, a video can also be viewed as made up of a series of shots or scenes and automatic approaches for segmentation of video into shots are researched [26]. Chu and Chou [7] define a shot based on the changes in color histograms between frames, while [48] extract shots based on latent discriminant analysis of key words found in chat messages.

Of course, segments can also be viewed from the chat side and be encoded in terms of chat characteristics like message density, semantic information and conversations within the chat room [34].

In this thesis, we take these issues into account theoretically. We provide our models with implicit and explicit information for aligning chat and video, with context and temporal clues. We choose different types of units for classification, which are not only based on temporally fixed length segments. We elaborate on this in Chapter4. Our models are based on transformer language models, which we explore in the following section.

2.3 Fine-tuning transformer language models

These key concepts of live stream highlight detection inform our approach for fine-tuning transformer models for this problem. We choose this technology, because in natural language processing (NLP), pre-trained transformer language models have been found to be very effective starting points for learning many tasks. Language models allow for learning of general knowledge about language patterns from text [9]. The transformer [45] is one architecture of models used for language modeling.

A transformer model is a neural network with multiple encoder-decoder layers. In contrast to previous approaches such as (bidirectional) recurrent neural-networks (RNNs)[29], the input sequence is processed all at once and not word by word. This allows the transformer to encode the context of each word in a generalized sequence representation in parallel for faster processing. Additionally, the attention mechanism allows for learning which part of the input sequence is relevant to the meaning of each individual word.

Typically, a transformer language model is pre-trained on a self-supervised task. This pre-training leverages vast amounts of texts which are available through corpora or web crawls, which humans have generated. This allows the language model to encode representations of texts which are helpful for many downstream tasks [9].

In order to employ a transformer model for a specific NLP task like sequence tagging or text classification, a pre-trained language model is fine-tuned to the specific task. This is done by adding a task-specific neural network on top of the pre-trained transformer part of the model. Then, a smaller amount of annotated data for a specific task used to train the task specific neural network or the whole model is supplied. Now, the transformer layers of the pre-trained model can be trained in conjunction with the neural network specific to the task. This means that the language model keeps training on the task at hand. Alternatively, the underling language model can be left as-is and the transformer weights are “frozen”, keeping them the same as they were after pre-training. In this second approach, called feature-based downstream adaptation, the language model representation is merely used to derive vectors from the input text and the task-specific architecture is supplied with these vectors [20].

With fine-tuning, the weights of the transformers can be trained with a specific task, leading to learning textual representations that are helpful for this setting. However, this is time and resource intensive [27]. Feature-based adaptation on the other hand does not backpropagate the loss through the transformer layers, leading to a less computationally heavy task while leveraging the contextual embeddings of the transformer language model at the expense of performance. For both implementations, task-specific neural architectures can be added.

2.3.1 Transformer implementations

A variety of different transformer models with different architectures have been introduced [20]. Each one tries to either cater for specific down-stream tasks or improve the architectures for language modelling performance, lower amounts of pre-training or simplifications.

The widely used BERT model is a transformer model that allows for bi-directionality in the input by attending simultaneously over all input tokens [9]. This enables dependencies, long and short distance, to be learned between tokens which yields a context dependent representation. By using masked language modeling (MLM) and next sentence prediction (NSP), this implementation aims at even capturing dependencies between sentences. This means that pairs of sentences are provided during pre-training. For tokenization, byte pair encodings (BPE) and WordPiece encodings are used. This is applied to a corpus of roughly 16GB in size containing Wikipedia articles and the BookCorpus.

With this technique, the authors show, not only how to use attention in a fast and effective way, but also provide a language model that can be trained on a pre-training task and then fine-tuned to specific tasks.

Based on the BERT architecture, one optimized model is RoBERTa which aims at implementing a simplified, more effective pre-training architecture [27]. The authors here do away with NSP and use byte-level BPE encodings in order to avoid unknown tokens and minimize vocabulary size. They show empirically that MLM alone is enough for pre-training and yields comparable if not better results than the original BERT architecture. Here, MLM is implemented with dynamic masking of the input, showing even more varied training examples to the model. Additionally, a bigger corpus of around 160GB in size is used for even higher performance.

Another variant of the transformer architecture is BART which seeks to improve upon BERT and RoBERTa by means of more varied mutations of input text [24]. Here the idea is to define a number of transformations including the classic MLM task among others like re-ordering of tokens, replacing multiple tokens and deleting tokens entirely from the input. The challenge now is to recreate the original input. In this setting, the data of BERT is used with the training objective of RoBERTa yielding competitive results, but also resulting in a larger model. It mainly succeeds in translation and abstractive summarization. Finally, a so-called classification head is not needed for this implementation, as its purpose is sequence generation.

The final model in this comparison is ELECTRA. With this model, Clark et al. [8] present a smaller, more efficient transformer which is competitive with respect to the other implementations. ELECTRA is conceptualized as a discriminator whose task is to tell "real" from "fake" tokens in the input. In this case, the original input is corrupted by a small BERT-based MLM model which generates sentences which may have some tokens replaced. By trying to determine which tokens are fake and which ones are real, the problem is defined over all tokens making instead of only the masked ones. It is trained on the same data as BERT with the same objective as RoBERTa for the generator.

In Chapter 4 we will discuss the theoretical decision to employing one of these architectures on the task of highlight detection. Later, we evaluate the effectiveness of the chosen model on this problem in Chapter 5 in different setups.

2.3.2 Transformers in applications

These transformer architectures have improved the state of the art in many NLP tasks [9], including various natural language understanding tasks included in the GLUE evaluation dataset [46] and also other kinds of tasks including sequence tagging, for example named entity recognition [49] and sequence classification [2].

For each task, usually a different input format is chosen [20]. This is important to give the model the specific information it needs to perform best. For sequence classification, typically a single input sequence is used[46]. For other tasks like information retrieval, that require more context, multiple inputs can be encoded. This allows for a search query and a longer passage of where the information can be found to be input and separated by special tokens. For our work, these ideas are useful, as we want to classify sequences of chat messages. We want to encode them in context, allowing for structural information to be present by delimiting them with specialized tokens as well. This is discussed in detail in Chapter 4.4.

2.3.3 Transformers in different domains

While it is evident that transformer models pre-trained on English web-crawled text perform well on tasks using English in a general setting, domain-specific applications suffer from the different writing style that is present in these domains. Examples of specific domains are medical publications, computer science publications, poetry, judicial texts and many more. Here, not only do texts contain a different, specialized vocabulary, but also sentence and utterance structures may differ. This poses a challenge when trying to fine-tune a language model pre-trained on Wikipedia articles and books [9]. Previously, we explore that live stream chat language contains its own peculiarities 2.1 thus adding another domain to the aforementioned list.

In order to overcome the lack of domain-specific vocabulary, it is suggested to expand the transformer vocabulary by adding additional tokens and thus improving coverage of the words used in a specific domain. Gu et al. [13] show that, while pre-training directly on data from a different domain is most beneficial, adding additional tokens to a pre-trained language model does improve its performance in a different domain. Furthermore, Gururangan et al. [15] show that continual pre-training also improves domains-specific performance. They use a pre-trained language model and employ it in a pre-training task on domain-specific text. Finally, Yao et al. [50] develop a method for combining these ideas into a continually pre-trained language model with expanded vocabulary. For the vocabulary expansion, they average existing token embeddings of word pieces and average them to create a new token, which is then added to the vocabulary. This yields a model, which contains domain-specific vocabulary and is exposed to in-domain text to learn from it. This approach proves to be effective in the medical and computer science domains.

Another approach of adapting transformer LMs to a specific domain is shown by [12] on Twitter and Twitch chat corpora. Here, they employ models for offensive language detection trained on tweets on Twitch chat messages to find insulting chat participants. To this end, they analyze the similarity of the two domains and conclude that training on Twitter data can help performance of this task in the Twitch setting. With this approach they find additional data to construct a larger training set which helps in task performance, where not enough task-specific training data is available for a certain domain. This is interesting as much more work has been conducted on Twitter in comparison to Twitch live stream chat. Although the task of highlight detection is not applicable to Twitter, this work gives an indication that the language used on both platforms has something in common and that using annotated data from a different domain can be helpful.

2.4 Conclusion

Because livestreaming of e-sports events is a relatively new phenomenon, the field of research around it is still evolving. In section 2.1, we have explored some approaches towards the user behavior on livestreaming platforms, tying into the highlight detection task. We have seen how users employ emotes and other stylistic means in live stream chat to express themselves, and how crowds communicate in massive chat rooms with thousands of users. Additionally, we have explored emote usage and user voices in live stream chat. In the following chapters, we use these considerations in order to devise techniques for highlight detection. As we have pointed out, there are a few approaches to the problem present in current research. We have seen how chat and video alignment is approached, which units of classification are used, and seen which machine learning techniques have been employed in the past.

We also provided an overview of how transformer models are pre-trained, fine-tuned and used on NLP tasks. Furthermore, we have introduced ideas to adapt models to a different domain. This is helpful in order to limit the effort that is needed in devising a specific language model for live stream chat.

With these theoretical investigations, we can next describe the data we use and approaches we take towards highlight detection.

3 Data

As is common for NLP and Machine Learning, in this work, too, a big focus lies on the data. In this project, we use two different datasets which were collected from the platform https://twitch.tv. As our goal is to find highlight segments using audience reactions, we employ datasets which contain chat messages of live streaming content. First and foremost, we use an annotated dataset of one LoL e-sports tournament, which contains video and chat information alongside annotations for highlight segments. We call it the "highlights dataset", and it was collected and published by Fu et al. [11]. Secondly, we collect a dataset of Twitch chat messages from LoL e-sports matches for the purpose of training a language model which is specialized on Twitch chat language. This second dataset is called "Twitch LoL corpus" and features a variety of English speaking LoL e-sports channels.

In the following, we describe and analyze these two datasets with respect to the task of highlight detection.

3.1 Highlights dataset

In order to train and evaluate a supervised model for highlight detection, we need an annotated dataset of e-sports events which contains chat messages of audience reactions with their timestamps of submission as well as annotated highlight and non-highlight segments of these events.

3.1.1 Dataset overview

For their work on highlight detection, Fu et al. [11] collect and annotate a dataset of the League of Legends tournaments "North American League of Legends Championship Series" (NALCS) and "League of Legends Master Series" (LMS). The former was aired in English and predominantly contains English chat messages, while the latter was broadcast in traditional Chinese, resulting in predominantly Chinese chat messages. Both tournaments are the Spring series of 2017.

This NALCS dataset is harvested from the Twitch channels nalcs1[8] and nalcs2[9]. Both channels have been deleted since the time of data collection. In addition, the gold standard annotations for highlights are obtained from the ONIVIA YouTube channel[10]. These videos are matched by means of automatic image comparison to the original stream video, thus identifying which segments of a match are present in the published highlight video on the video publishing platform YouTube. As the original live stream was aired with a frame rate of 30 frames per second (fps), the granularity of these annotations is 1/30 of a second or one frame of video time.

So, we can express a highlight segment as a number of consecutive video frames. This exact representation is displayed in Figure 2.

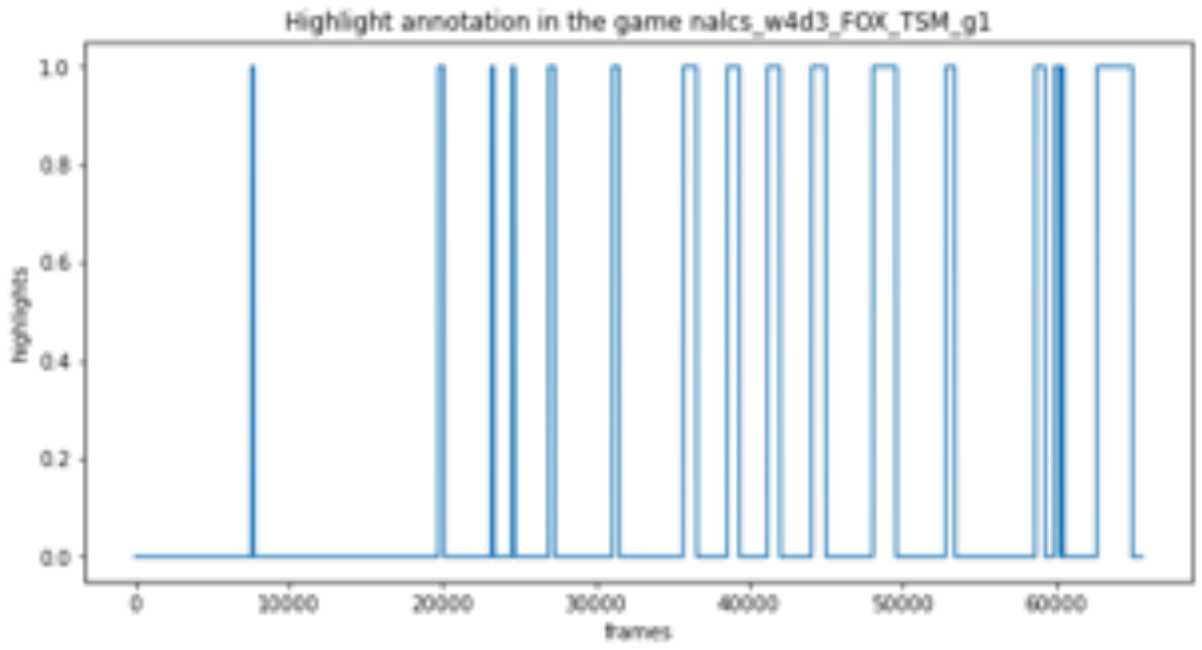

Figure 2: Highlight annotations for one game.

The messages from the live chat are matched to the video with their time stamps of when they were published. For the following analysis, this means a strict correspondence of messages to video content, where in reality we assume that reactions are delayed from the video.

[8]https://www.twitch.tv/nalcs1
[9]https://www.twitch.tv/nalcs2
[10]https://www.youtube.com/channel/UCPhab209KEicqPJFAk9IZEA

tournament	train	val	test	total
NALCS	128	40	50	218

Table 1: Number of games in the dataset splits as reported by [11]

This dataset uses only the matches isolated from the longer running live streams of the whole event. Each match is played in a best of three format with up to three individual games. There are 221 games in the group stage and 24 games in the playoffs played, according to liquipedia.net[11].Only the matches that were played during the nine weeks of group stage are considered in this dataset. For their experiments, Fu et al. [11] use the first and third game, if there is one, of each match as training samples, the second match of weeks one through four as validation samples and the second match of weeks five through nine as test samples.

Matches and games are referred to by their shortened name, denoting the tournament, week and day of play, the teams which played the match and the game number which was played. Thus, "nalcs_w4d3_FOX_TSM_g1" denotes the NALCS tournament, week four, day three, the match between team "Echo Fox" (FOX) and "Team Solo Mid" (TSM) and the first game of this match. With this system, Fu et al. [11] can individually identify each game.

In this present work, we focus on the English portion of this dataset. This leaves us with 218 out of 321 videos which we use for our experiments.

Fu et al. [11] report the game distributions as listed in table 1.

3.1.2 Dataset analytics

Now we turn to characterizing the data within this dataset in more detail. It contains video content of around 165.5 hours in length with a total of 1 733 448 chat messages recorded. Games range from 30 minutes to 50 minutes of duration. There are 3 106 highlights in total in the dataset. We find 14.45 highlights per game on average with a standard deviation of 3.69 and between 5 and 30 highlights.

The shortest highlight merely lasts for 0.267 seconds or 8 frames while the longest one has a duration of more than seven and a half minutes or 13 460 frames. The mean

[11]https://liquipedia.net/leagueoflegends/LCS/North_America/2017/Spring

highlight duration is 20.10 seconds with a standard deviation of 20.36 seconds. This observation suggests that a highlight cannot simply be characterized by its length. We find, however, that 95.5% of all highlights are shorter than one minute (1800 frames).

Although our main focus lies on chat activity and content, we also report how much of the total runtime of video content is occupied by highlights. In the dataset, we find that highlights make up 10.5% of the total runtime, while the number of chat messages in highlight segments amounts to 16.6% of all chat messages. We can thus conclude, that chat activity is higher during more exciting parts of a match.

3.1.3 Criticism and notes

Han et al. [16] criticized the approach taken by [11] for constructing the highlight annotations. While it is helpful to use professional annotations for highlight segments, the subjective character of these segments makes using a singular YouTube channel and studio less reliable and less generalizable. This present dataset has the risk of encoding the personal preferences towards highlight creation of this video production studio.

During our inspection of the highlight detection dataset, we found that some highlight segments seem to be impossibly short. Segments which last less than one second probably do not convey a lot of interesting information. We manually inspected some of these short highlights and found possible problems with the automatic video matching algorithm. In the case of match "nalcs_w5d2_TL_DIG_g3" we see screen tearing in the original YouTube video[12]. We believe this can cause some unreliable detections in the highlight mapping process. With some more analysis, we can find out how short a highlight can possibly be and filter out improbable highlights by length.

Additionally, we see games without any highlight annotations in two cases, "nalcs_w8d3_TSM_FOX_g1" and "nalcs_w8d3_TSM_FOX_g2", where we suspect that the video matching algorithm failed. We can find highlight videos for these two games on YouTube[13]. We find files for these matches in the dataset, but they appear empty.

These issues probably impact the performance of automatic highlight annotation sys-

[12]https://www.youtube.com/watch?v=zWdtNCGZDRM&t=293s

[13]highlight videos for "nalcs_w8d3_TSM_FOX_g1" https://www.youtube.com/watch?v=onMKVMeXbI4 and "nalcs_w8d3_TSM_FOX_g2" https://www.youtube.com/watch?v=ZoC3qcv6Gr0

game name	highlight video on YouTube
nalcs_w5d2_C9_TMS_g1	https://www.youtube.com/watch?v=uh0npPZsUUc
nalcs_w6d3_IMT_NV_g1	https://www.youtube.com/watch?v=U9msQYsWcOo
nalcs_w6d2_FOX_C9_g1	https://www.youtube.com/watch?v=JF2o7xIg0no
nalcs_w6d2_FLY_NV_g1	https://www.youtube.com/watch?v=TyZRsAT6P5k
nalcs_w5d2_C9_TMS_g3	https://www.youtube.com/watch?v=xtNhxrG2U5A

Table 2: Missing games in the dataset. We can find highlight videos on YouTube.

tems, which are trained and validated on this dataset. However, to research the model setups which work for highlight detection, we expect this dataset to be helpful. For comparability with previous work, we choose to keep the original dataset.

In the NALCS Spring 2017 tournament group stage, there were 221 games in total. However, in the dataset, we only find 218 games. We do not know how to explain this discrepancy. The chat content for these games is present in the dataset, however no highlight annotations are present. These games are excluded from the experiments. We list the names and corresponding videos in Table 2.

3.2 Twitch Chat League of Legends corpus

For training a live stream chat specific language model, we additionally collect a dataset containing chat messages from LoL e-sports streams. Because the language style [1] and communication patterns [10] in live stream chat rooms differ from text found on the web, we create a domain specific dataset with this work. Although we find a sizable number of chat messages in the highlight detection dataset, it is not enough data for training a BERT-like language model. Additionally, we want to provide chat content that is not found in the highlight detection dataset, in order to make the evaluation as fair as possible. We focus on English as the language of this dataset.

During the process of collecting relevant chat messages, we select appropriate channels for downloading videos of past broadcasts, also called videos on demand (VODs), and then use this information to access the recorded chat of these videos. Finally, we clean the data by filtering out live stream videos that do not meet our criteria for useful information and discard messages that we deem not helpful. We will now present the process in detail

3.2.1 Dataset overview

Our Twitch chat League of Legends corpus is collected from 9 different twitch channels, which focus mainly on LoL e-sports content. The dataset contains 3110 videos of live streams with 89 960 287 messages including around 387 652 810 tokens. The total runtime of the live streams amounts to about 17 583 hours. These live streams were originally broadcast between December 2014 and March 2022.

3.2.2 Data collection

Because we want to include very specific content into our dataset, we rely on selecting appropriate videos of League of Legends e-sports broadcasts. We identify channels which air these broadcasts by consulting lists of LoL tournaments on `liquipedia.net`[14]. From these lists, we extract websites that contain information about each individual tournament, including a "Streams" section in most cases. For simplicity, we select the first link that appears in these sections, usually denoting the channel of the English live stream. This process yields a list of 25 broadcasters on Twitch. We then manually check if their predominant stream language[15] is English and if the channel focus appears to be LoL e-sports events. We eliminate 16 channels which do not fit these criteria and end up with a list of 9 channels which we use to download VODs from.

Next, we turn to the twitch.tv API[16] in order to discover all the past broadcasts of these previously collected channels. From nine channels, we select 5 954 videos for downloading chat messages from.

The official twitch.tv API does not provide functionality for downloading chat messages of past broadcasts. Thus, we employ an external tool, called TwitchDownloader[17], which provides a command line interface (CLI) for this task. This tool allows the download of all chat messages in VODs which we determined in the previous step. We do not save the video content, as our focus lies on the chat language.

[14] lists of important LoL tournaments: `https://liquipedia.net/leagueoflegends/S-Tier_Tournaments`, `https://liquipedia.net/leagueoflegends/A-Tier_Tournaments`

[15] `https://help.twitch.tv/s/article/languages-on-twitch?language=en_US#streamlang`

[16] `https://dev.twitch.tv/docs/api/`

[17] `https://github.com/lay295/TwitchDownloader/`

3.2.3 Data cleaning

Although we made informed decisions about which VODs to include in our download process, we find that some videos are not suitable for our purposes. In order to improve the quality of our corpus, we filter out certain live stream recordings and messages.

Filtering live streams For filtering VODs, we compute some information about them. We count the number of messages in each live stream (message_count), determine if a VOD is a rerun of a previous event, and record the duration of the videos.

Then we exclude all videos which are shorter than one hour to maximize the chance of including actual games in the live stream. Usually, there is a show around matches and while a game can be as short as 30 minutes, including the show aspect, the one-hour limit is reasonable.

Next, we inspect the titles of the VODs. These can help us to identify repeated broadcasts, as they usually contain keywords such as "REBROADCAST" or "RERUN". We create a list of such keywords by inspecting the stream titles manually and taking note of apparent patterns. Consequently, this list is applied to filter out 946 reruns. Alternatively a statistically driven approach can be applied, using message_counts, as rerun videos typically have fewer comments than the original one. On average, there are 3 698 in a rebroadcast and, 23 002 chat messages in an original video. These numbers were determined using the previous criterion, but clustering could be employed for filtering VODs.

Finally, we also eliminate all the VODs which do not have any comments. This excludes 1881 videos.

Filtering messages Another way of cleaning the corpus is to inspect the content of the chat messages. Typically, chat moderation on Twitch is aided by bots. These automated systems detect spam messages, offensive language, can provide basic information about the live stream or the topic and are used for promotional purposes with repeating messages. Messages of these bots appear in the normal chat room. We argue that these predetermined messages are not part of audience reactions. This is why we try to filter

username	**video count raw**	**video count filtered**
Riot Games	2361	1058
ESL_LOL	1020	193
LPL	913	733
LCK	664	647
GarenaEsports	346	48
RiotGames2	291	103
LEC	256	241
RiotLAN	90	76
lolpacific	12	11
total	**5953**	**3110**

Table 3: Number of videos collected from different Twitch LoL e-sports channels before and after applying filters. The video counts are summed up by channel on which they are broadcast.

out messages by these bots. In order to do so, we collect a list of bots and their nicknames from two websites[18]. We look up the names of bots and determine their user profile. Then we store this information to filter messages which were sent by these users.

Bots can also be triggered to send a message by stream viewers by sending commands via the chat room. These messages are typically preceded by an exclamation mark, "!". We filter them out along with hyperlinks, user mentions preceded by "@", extraneous white spaces and empty lines.

We are left with a dataset which contains one chat message per line of live stream chat from LoL e-sports events.

3.2.4 Dataset analytics

In our final, cleaned dataset there are 3110 VODs remaining. The only language found as stream language[19] is English. This is expected as we searched for English channels in the first place. For ESL_LOL about $\frac{1}{5}$ of VODs were removed, while for other channels like LEC almost all the VODs remained in the dataset (see table 3). This can mean that the former channels broadcast more reruns, have a less active chat or many shorter VODs.

In order to better understand the kind of data we are dealing with, some superficial

[18]common livestreaming bots on Twitch: https://blog.lvlupdojo.com/the-most-common-twitch-bots-e07bed06538, https://www.streamscheme.com/best-twitch-bots/

[19]https://help.twitch.tv/s/article/languages-on-twitch?language=en_US#streamlang

	token count
max	487
min	1
mean	4.27

Table 4: Token counts per message in the Twitch LoL corpus

measures of the data are computed. We take a look at the messages and use the TweetTokenizer from the Natural Language Toolkit (NLTK) [3] for quickly tokenizing the messages in the corpus. With this information, we find the longest and shortest messages as listed in Table 4. Unsurprisingly, the shortest messages are only one token in length, consisting of one emote or single word exclamations. The longest messages, 487 tokens long, contain copypasta messages. With an average of 4.27 tokens, we generally find more short messages in the dataset. This is also expected in live stream environments and especially in e-sports.

We provide a download of this corpus at `https://huggingface.co/datasets/Epidot/twitchlolcorpus`.

3.3 Concluding remarks

For our work, we explored two datasets of Twitch chat. The highlights dataset was described and analyzed, the annotation scheme was shown, and we presented some statistical insights. We described the collection of the Twitch LoL corpus from Twitch and presented statistical measures on it. While we acknowledge the problems of the highlights dataset, we decide to use it for our work because of availability and comparability considerations. The second dataset is our Twitch LoL dataset, the starting point for our language model TwitchLeagueBert. In the following chapter, we describe how we use these datasets for highlight detection and language model training.

4 Methods

Having decided on our datasets, we can now turn to the technical considerations behind the highlight detection task and our experiments. This part involves the design of the different techniques we want to apply.

As a first step, we explore different approaches based on features that we identify in Chapter 2.2 and standard machine learning techniques. We exploit statistical and semantic characteristics of the chat message data in order to create baseline models to compare against our transformer setups. This is done to gain further insights into how different features of our data impact the ability to detect highlight segments in e-sports event live streams from chat messages. Next, we derive different deep learning models for this purpose from these insights. The core of these models is a pre-trained language model, which we fine tune for interpreting the live stream chat text in context. We train a pre-trained language model on the Twitch LoL corpus described in 3.2 with masked-language-modeling.

All our models are implemented using Python with the code is available GitHub[20].

4.1 Highlight detection definition

As we have established in Chapter 2, the central task of this project is to find segments in of livestreamed LoL e-sports matches. Because the inherent medium of these matches is video, we identify a frame as the smallest unit in our considerations. We divide our videos V into segments S of n frames each. We provide context C for each segment.

Each chat message, M_t is linked to a frame F_i through the timestamp t of seconds and milliseconds since the beginning of the video. In turn, each chat message contains a sequence of tokens $T_1 \dots T_n$ which make up the chat message.[21]

Thus, we can define our highlight detection task as a sequence classification task, assigning a label l of highlight or non-highlight to each segment. We define this binary label to be 1 if a segment is part of a highlight and 0 if it is not.

We provide an overview of these definitions in Table 5.

[20] `https://github.com/maugl/chat-highlight-detection`
[21] Tokenization is dealt with in different ways for different implementations. See Sections 4.3.2 and 4.4

V	live stream videos of the dataset
S	segments of the live stream video consisting of n frames each
F_i	i^{th} frame of the live stream video
$M(F_i)$	set of messages linked to Frame F at the i^{th} frame
$E(M)$	set of emotes found in a set of messages
m_t	message at timestamp t of a live stream video
t_n	nth token of a message m_t
l	binary label for a segment being part of a highlight or not
T	set of tokens in a sequence of messages

Table 5: Notation for the highlight detection task defined as a sequence classification task.

4.2 Baselines

Since we deal with data which can be interpreted as a time series, we deploy algorithms which can treat the chat messages as such. To this end, we encode multiple features as running averages over a fixed window of frames. This process yields smooth time series which we can apply different algorithms to. We try to exploit these features to predict when highlights occur in the live stream. Additionally, we use TF-IDF and Twitch chat embeddings[39] which we use as features for logistic regression and a Naive Bayes prediction.

4.2.1 Features

For testing out different features which we use to predict highlights from time series, we turn to the work of Liaw and Dai [25] and Chu and Chou [6].

For each of the following measures, we compute a result for each frame of video. We choose a window of n frames centered around this frame and compute the relevant measure for this frame. In this way, we keep the temporal resolution of the video and are able to make fine-grained predictions for highlight segments. Additionally, we take the context of each frame into consideration. These computed features are shown in Figure 3.

Message density In their experiments, Liaw and Dai [25] compute the number of messages over a fixed window of time, which we call "message density". We know that audiences more readily share their emotions when exciting gameplay happens. Thus, we

for more detail.

expect an increase of messages in highlight segments. Chu and Chou [6] in contrast, use raw message counts instead.

$$\text{message\_density}(i, w) = \sum_{j=i-w/2}^{i+w/2} |M(F_j)| \tag{1}$$

Emote density This second feature is closely linked to the previous idea of audiences sharing their excitement. We do not take the number of messages, but the number of emotes in a certain window, into account here. We employ a list of emotes that we collected from the Twitch API for emotes[22] and compare the strings found in chat messages to this list to determine the number of emotes in chat messages.

$$\text{emote\_density}(i, w) = \sum_{j=i-w/2}^{i+w/2} |E(M(F_j))| \tag{2}$$

Message diversity Another feature we investigate is basically the normalized entropy of chat messages. Liaw and Dai [25] call this feature "diversity" and calculate the Shannon Entropy over the tokens T in a window of frames, normalizing this by dividing the result with "its possible maximum value". With normalized entropy, it is assumed that when a highlight occurs, the audience reacts to that with a response that contains more similar tokens than in other segments of the live stream [25].

Copypasta density The feature that we implement not from an existing implementation, but from a theoretical idea, is copypasta density. We try to see if copying and pasting of the same text decreases when a highlight occurs. For computing copypasta density, we use the simple idea of comparing similar n-grams found in different messages. This is done in order to allow for partial matches where the same sentence is pasted 3 or 5 or 10 times within one message. Thus, we identify n-grams which are used within a live stream from unigrams, bigrams and trigrams up to a number k of n-grams. We then count how many times each of these are repeated and sum this up over a window of n

[22] https://dev.twitch.tv/irc/emotes/

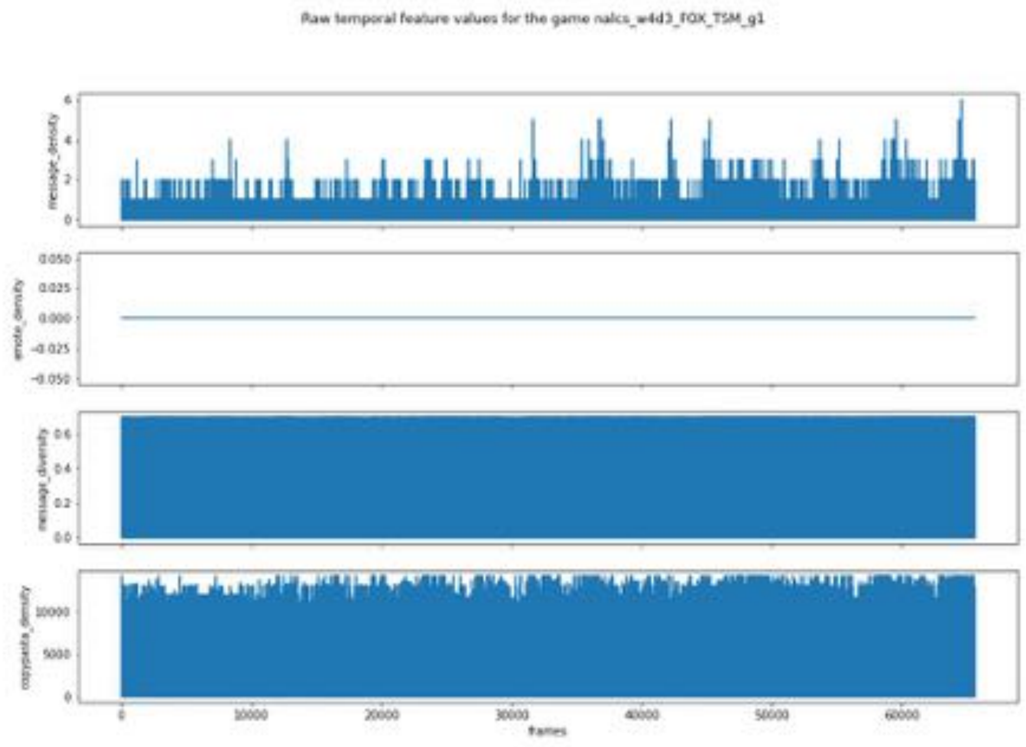

Figure 3: Temporal features calculated on an example match "nalcs_w4d3_FOX_TSM_g1"

frames. We set a threshold for how many times an n-gram has to occur before it factors into the count.

Smoothing the density features Each of these features produces a continuous output over the length of a live stream video or part of it. With relatively small windows for computing these, we find that the features are quite volatile. To make the output and highlight detection more gradual, we resort to smoothing the features with a running average over the signal. The running average is the cumulative sum over the last N frames for each frame. We denote the output of the density function $f(i)$ at frame with the index i of the total live stream.

$$\text{mvg\_avg}(i, N) = \frac{\sum_{j=i-N}^{i} f(j)}{N} \tag{3}$$

Feature scaling Additionally to smoothing the features, we scale them to the range between 0 and 1. This is done with the min-max scaling approach. It is defined in Equation 4 as implemented in Scikit-learn [33]. Here $min()$ and $max()$ are functions to find the minimum and maximum values in our variable to scale, X. mn and mx are the minimum and maximum values to scale to, 0 and 1 in our case.

$$X\_scaled = \frac{X - min(X)(mx - mn) + mx}{max(X) - min(X)} \tag{4}$$

TF-IDF Term frequency inverse document frequency (TF-IDF) is an established way of representing tokens as a vector by comparing their overall frequency in a corpus to their frequency in a document. In this way, their importance for a document is expressed and can be used to represent this document in contrast to others. We use this to provide a vector representation for segments of livestream chat, which we treat as documents. We calculate TF-IDF, see Equation 5, by multiplying term frequency tf defined as the count of a token t in a given document d by the inverse document frequency idf. idf is defined in Equation 6 with n as the number of documents in the corpus and $\mathrm{df}(t)$ as the number of documents containing t[33].

$$\text{tf-idf}(t, d) = \mathrm{tf}(t, d) \times \mathrm{idf}(t) \tag{5}$$

$$\mathrm{idf}(t) = log\frac{1 + n}{1 + \mathrm{df}(t)} + 1 \tag{6}$$

In this application, the corpus of documents is defined as the collection of segments of combined chat messages.

Twitch chat embeddings With our final semantic feature we rely on the work of [39] who compute Word2Vec[30] embeddings for Twitch chat with a focus on emotes. We load these embeddings and encode a sentence as the mean of all of its token vectors following [17].

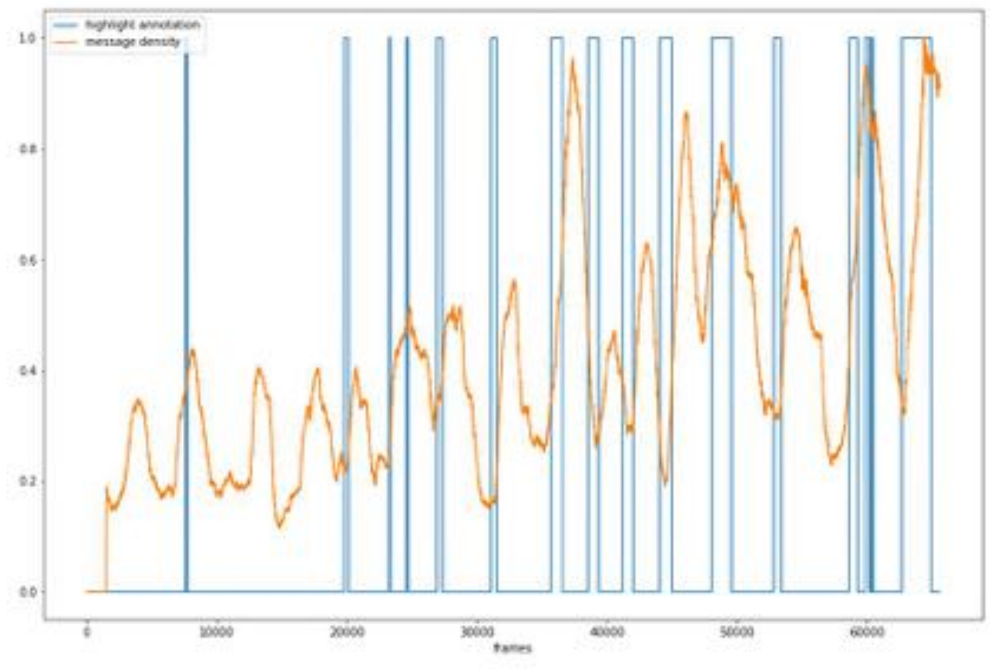

Figure 4: Smoothed and scaled message density and highlight annotations for game "nalcs_w4d3_FOX_TSM_g1". Smoothing is done by applying moving average with $N = 1500$ to message density counts. We then apply min-max scaling to it

4.2.2 Classification

Peak prediction algorithms With this approach, we assume that highlight segments appear when certain features reach a peak. This idea is backed by the plot in Figure 4. We clearly see peaks in message density following highlight annotations. Admittedly, there are peaks where no highlight is annotated, but the feature appears to provide a good indication. Given this observation, we compare two algorithms for finding peaks.

We use a method to identify peaks by contrasting them to their surroundings implemented in SciPy[23] as a simple and fast method. This algorithm relies on its parameters for how much a peak must stand out compared to its surroundings to output a signal. Through preliminary tests, we find that only certain parameters have great effect on our dataset. We test the performance when altering the following parameters:

[23] `https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.find_peaks.html#scipy.signal.find_peaks`

- **prominence** defines, how much higher a peak has to be compared to the surrounding signal.
- **relative height** defines how high a peak should be.
- **width** sets the minimum and maximum width of the peak to be detected.

With these and a fixed offset parameter to account for message lag, we predict highlight segments from peaks in our feature signals.

In contrast, we also use a more elaborate approach called real time peak predictor (RTPP) which leverages a moving mean distribution of the feature signal and produces a positive output when there is a sudden change in the signal. This jump needs to be higher than a threshold defining the number of standard deviations that the signal needs to jump to produce an output. The algorithm has a lag parameter built-in which influences how many time steps, in our application frames, contribute to the moving mean distribution. This algorithm was suggested on `https://stackoverflow.com` [44] and has been used in various publications. The Python implementation that we use for our experiments can be found in the same thread[24].

Highlight prediction with semantic features While we see a use for temporal features for highlight prediction [25], we also find that [11] [18] [39] show that semantic features of the audience reactions benefit our task. We explore this with some standard ML-methods for text classification and with help from a tutorial[25]. We concatenate together n frames worth of chat messages and treat these as our text samples we want to classify. The ground truth label for each segment is given by the first frame's label. With a step of 30 frames, we classify each second in the 30 fps videos, like Fu et al. [11] do. In order to obtain features for our machine learning approaches, we apply TF-IDF over this corpus and translate each text representation into a vector representation using the Word2Vec model of Song et al. [39]. Our highlight prediction in this context classifies chunks of n frames as highlight or non-highlight. For the logistic regression algorithm, we

[24] `https://stackoverflow.com/a/56451135`

[25] `https://scikit-learn/stable/tutorial/text_analytics/working_with_text_data.html`

use the implementation of Scikit-learn [4]. Here, we employ the default model as provided by this implementation and feed TF-IDF vectors to the algorithm for classification. The same approach is taken for the Naive Bayes classifier, whose implementation also stems from Scikit-learn. Instead of TF-IDF, we use the averaged Word2Vec document vectors, which are loaded using zeugma[26]. We balance the dataset by over-sampling the minority class of highlight to match the number of samples in the majority class.

4.3 Transformer language model

From the simpler baseline models, we want to turn to deep learning models for highlight detection. As we have examined in Chapter 2, a variety of model architectures have been used on this task, among them CNN and RNN. In this work, we focus on transformer models [45] as the next step in the development of these models that are used for encoding tokens in context. Our model of choice is the RoBERTa architecture [27].

As we have seen, the basis of these transformers are pre-trained language models. While there are a variety of different English language models available, we suspect that they perform poorly on Twitch chat data. To verify this claim, we pre-train a transformer model based on the RoBERTa training regime on the Twitch LoL corpus described in Chapter 3.2 and compare its performance to a pre-trained RoBERTa model that has been trained on a more general web collected corpus[27]. We follow the methodology of this approach with respect to the tokenizer training, data pre-processing and model architecture as well as training parameters. We use the implementation by huggingface[27] with default hyperparameters. We call our model "TwitchLeagueBert".

4.3.1 TwitchLeagueBert - RoBERTa for Twitch chat modeling

We use the RoBERTa architecture over plain BERT, for several reasons. Firstly, RoBERTa promises reduced complexity by removing token types not needing to specify which tokens belong to which input sentence. As our domain contains many shorter messages, we do not see a benefit in encoding different tokens as parts of different sentences. In

[26] https://github.com/nkthiebaut/zeugma

[27] https://huggingface.co/docs/transformers/model_doc/roberta#transformers.RobertaForMaskedLM

the live stream chat, messages quickly follow each other and do not necessarily appear in a sensible order on a micro level. However, since it has been shown that there are multiple voices present in big chat rooms which are embodied by different users [10], we expect a benefit to separate these individual short messages by the sentence beginning and sentence end tokens.

While we believe it necessary to keep the original order of messages within a training sample, predicting the exact order is probably not helpful. We ground this idea in the fact that in live stream chat users send their messages, as part of a reaction to the action on screen, an expression of partisanship for a team or other kind of copypasta behavior or messages that are part of an ongoing discussion. If the messages appear as a reaction, the order in which different viewers react to the same event is irrelevant, as they most likely do not reference each other. In the copypasta case, the chat messages usually do not have a reference to anything going on in the livestream, possibly a lull in gameplay. In the last case, with a high volume of messages, the exact order will not be important, only that responses to earlier messages, appear later. However, these will most likely not appear in messages that exactly follow each other. This reasoning, together with the fact that it has been shown that the NSP task does not necessarily improve model performance for our purposes, the RoBERTa training method makes sense in this regard.

Additionally, the tokenizer that RoBERTa uses is much more robust, allowing for encoding emojis and words that have never been encountered before. Because of the non-standardized language that is found in livestreaming chat, these features are preferred.

We understand that RoBERTa has been trained on a larger corpus than the one we use (160GB vs. 1.9GB). Thus, the increased score that RoBERTa shows over BERT [27] may not directly transfer to our setup. However, in the increased training data RoBERTa has been presented with much more varied language, which we believe to be an advantage. Another issue that we want to mention is the one of societal bias. We know that language models pick up social biases from the data they learn from [32], especially as their size increases [42]. Additional considerations in this direction may be fruitful to create highlights that are catered towards a wide range of people from different communities. For now, we focus on which techniques we have to employ to make transformer models usable for the highlight detection task.

4.3.2 Model definitions

Tokenizer As the first step in language model training, the input text is prepared to be fed to the transformer model. Transformers need input tokens split and converted into numerical values. These so-called input_ids are determined by the tokenizer. In this implementation, we use a byte-level byte pair encoding (BPE) for the tokenizer, following [36]. We train it with a vocabulary of 50 000 tokens.

Model structure Following [27], our model has an input length of 512 tokens. It comprises 12 layers with self-attention with a total of 81 966 416 parameters. The output of the model are logits for each token of the input.

4.3.3 Data structuring for MLM training

Because we use the same architecture and setup as [27], we also structure the corpus that we use for training similarly. In order to provide the model with messages in context, we concatenate all the chat messages and group them into groups of pre-defined length. The length is measured in number of tokens after tokenization and is determined by the maximum sequence length for the final model. This procedure results in no truncation, and text simply overflows to the next group of messages when there is a cut in the middle of a message. Additionally, we do not pay attention to which stream the message was harvested from. A new stream may begin in the middle of one group. This lazy grouping procedure can result in many split messages and some incorrect contexts. For illustration purposes, we include Figure 5 which shows the split message "$< s >$ c9 skins where :) $< /s >$" spanning two different groups. We can also see how the beginning and end tokens "$< s >< /s >$" are used to delimit messages.

4.4 Highlight detection models

For investigating how transformer models can be used for the downstream task of highlight detection in live streams, we devise different architectures with different models and

```
["for the emperor</s><s>ResidentSleeper</s><s>SJOKZ rbzSleeper</
   s> <s>PEOPLE BLAMING DRAFTS LUL</s><s>top gap</s><s>JG DIFF</
   s> <s>YEP next game battle of the bronze YEP</s><s>farm KEKW
   </s><s>Topgap BigBrother</s><s>5Head</s><s>INF FeelsDankMan</
   s><s>-700 KEKW</s><s>BibleThump BibleThump BibleThump
   BibleThump BibleThump</s><s>heyyy</s><s>why on earth would
   you put Perkz on Viktor.</s><s>c9",
"skins where :)</s><s>D OMEGALUL TA</s><s>NA SO BAD WINNING
   DRAFT MEANS NOTHING TO THEM KEKW</s><s>brain gap</s><s>Gaming
   </s><s>quantPickle quantPickle?</s><s>Why can DWG group with
   3 wild cards? That's unfair WutFace</s><s>tournament draft
   but bad execution MikeHogu MikeHogu</s><s>AYAYAQQQAA</s><s>
   Optimistic about NA KEKW</s><s>C9 got clapped</s><s>NA JUNG"]
```

Figure 5: Example for data preparation for language modeling. Chat messages are joined together by special tokens "$< s >$" and "$< /s >$" into sequences and cut to a fixed length.

compare their performance. Theoretically, the models should take several dimensions of the problem into account and solve the following problems:

- **semantic features:** As we work on chat messages, we need to encode their meaning. To this end a model should be able to encode words, emotes and emojis. Additionally, phenomena like chants and copypasta may be detected through contextual clues.

- **Temporal features:** Another important factor in language in general and live messaging is timing. The ideal model should have a mechanism to encode temporal patterns that evolve over time, like the features defined in Chapter 4.2.1. From previous work [25], we know that these changing features can help with the task at hand.

- **Message and live stream linking:** To address the lag between the live stream broadcast and the audience reactions, a mechanism for the model to dynamically link the time when a highlight occurs and when the reaction emerges in chat, should be devised.

- **Space efficiency:** We aim to align a sequence of messages to their respective time stamps or frames in the video. However, chat activity can vary over time, resulting in more or less dense spans of time in terms of message and token counts. Thus, there needs to be a way to efficiently pack messages in sparse chat segments and dense ones equally while preserving temporal features. This consideration stems from the practicality of having constrained input length for transformers.

In the following, we define our models theoretically and discuss the design choices with respect to the above described criteria.

4.4.1 Transformer model for sequence classification

For our first model, we directly lean on the idea introduced by [11] with their L-Word-LSTM. We replace the recurrent neural network part in their setup with a transformer model and keep the rest similar. This model solves the highlight detection problem with a sequence classification approach. Each frame is assigned a sequence of messages in a window of n frames following it. They choose each k^{th} frame of the input for classification and feed it to an LSTM-based RNN which outputs a prediction for the sequence.

As this task is a sequence classification problem, we can choose an established architecture for the transformer model. The classification head for sequence classification comprises a dropout layer and a simple feed-forward layer. The loss function used in this setup is the mean squared error (MSE). The target is a binary label (0 or 1). We use huggingface's AutoModelForSequenceClassification[28] with "roberta-base" and our TwitchLeague-Bert as the transformer models. We call this model setup the "AutoModel" because of the implementation we employ. In Figure 6 we show a diagrammatic view of this approach.

In this setup, the continuous stream of messages is associated with the frames in the video when they were sent in live stream chat. We take 210 frames worth of messages and concatenate them, including sequence delimitation tokens. Then, these sequences are tokenized, padded and truncated from the beginning to the maximum model input size. We truncate from the left because [11] find that the later messages in a sequence are more helpful for this task.

In terms of semantic encoding of the messages, we expect the transformer model to be able to find the most important features for this task by fine-tuning. As we provide

[28]https://huggingface.co/docs/transformers/v4.20.1/en/model_doc/auto#transformers.AutoModelForSequenceClassification

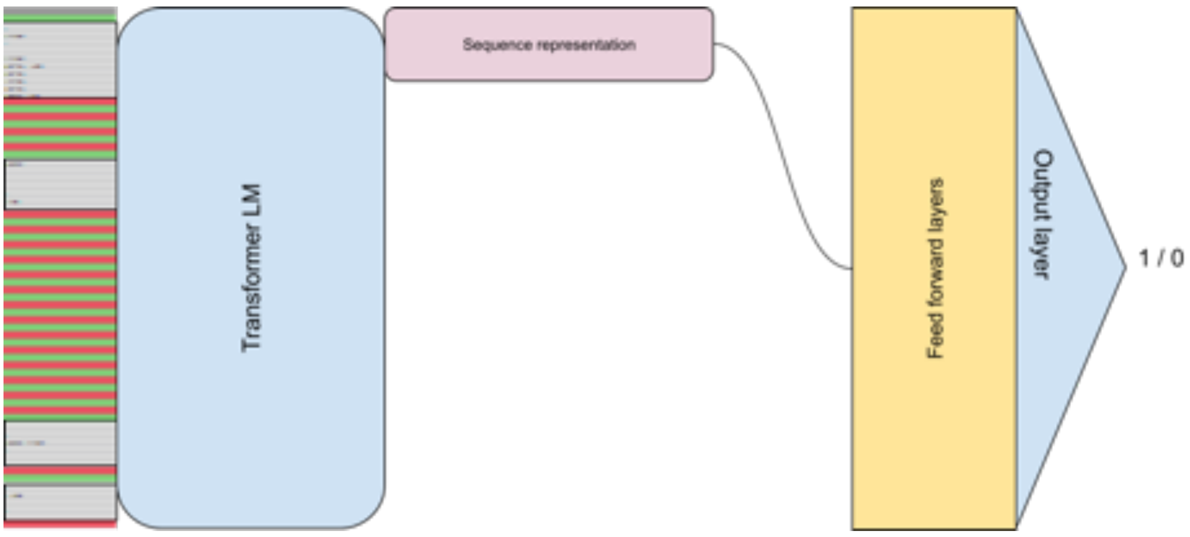

Figure 6: Model schema for sequence classification (AutoModelForSequenceClassification): The input tokens for delimitation tokens (green/red), and other tokens (grey) are encoded into a sequence representation to be classified by the feed forward and output layers.

a sizable sequence of messages to the model, temporal cues should be picked up as well, especially with empty frames included in the input. This makes this implementation less space efficient in terms of transformer input, with a higher ratio of sequence beginning and end tokens to content tokens. There is no explicit frame and message linking mechanism, but truncation from the left leaves out the earliest messages, resulting in an implicit linking based on the number of messages in a segment. It is left entirely to the model to learn linking from it.

4.4.2 Transformer model with additional features

While this first model architecture relies entirely on the outputs of the transformer model to encode all useful information, for our second setup we explore adding time series features separately to the language model part. To achieve this, we also use a pre-trained transformer to encode the textual data, but leave out empty frame sequences, compressing the data and removing temporal spacing between the individual chat messages. In order to compensate for this loss of information, we compute features as the ones described in Chapter 4.2.1 and add them to a classifier along with the transformer embeddings for the input sequences.

Again, we define the task for this model as sequence classification, predicting highlight or non-highlight for one or more chunks in the center of the input sequence. We define a chunk in terms of the number of tokens T per chunk. Each chunk should contain the same number of tokens, cutting at the end of messages. This means, we split each input sequence into k number of chunks and classify over the number $window\_size$ of chunks by providing $window\_size$ number of targets to be predicted. Context is given by the number of leading and trailing chunks defined as $context\_size$. Because we use the number of tokens as our measure to calculate chunk boundaries, we lose the direct connection of the label to each frame. Thus, we re-calculate the label as the average label found in the frames associated with a chunk. This leads to more coarse than frame level labeling. In evaluation, we have to adjust the results to bring them to the same granularity as the reference.

In this way, we add contextual information, over which we do not classify directly. Also, with this setup, we provide a *step* to classify over each chunk only once. With $window\_size = 2$ and one chunk of context before and after the classified chunks, the classification of a sequence would result in two labels per example. Together with $step = 3$, each chunk is classified and context for classification is given. This model architecture is shown in Figure 7 and is called "temporal" model as it encodes temporal features explicitly.

When combining the temporal features and the token embeddings, we take the mean of the token representations and concatenate them with values of the additional features. These features are scaled "by removing the mean and scaling to unit variance" [4]. For normalization, we add a dropout layer with 0.3 of dropout. As a loss function, we use mean squared error (MSE).

With this setup, the direct association between video frames and prediction is removed from the input messages. We trade this loss of information for more efficient packing of input tokens when leaving out empty frame representations. Subsequently, there is more textual context information present. This information is compressed, based indirectly on the number of messages that are found in a span of stream chat, providing more information even if few messages are sent in chat. When there are many messages in

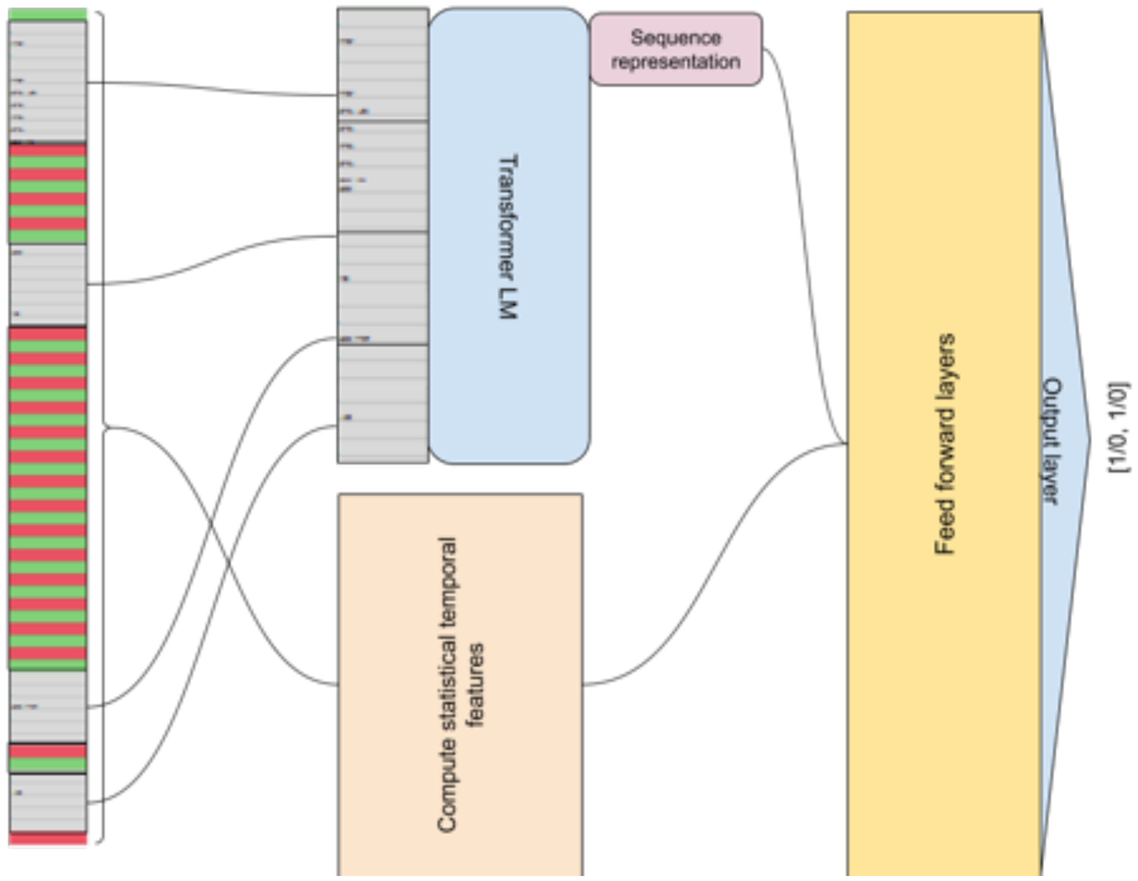

Figure 7: Model schema for transformer with temporal features: The input tokens for delimitation tokens (green/red), and other tokens (gray) are encoded separately, into a sequence representation and explicit features to be classified by the feed forward and output layers. The input sequence is split into chunks and classification is done over $window\_size = 2$ chunks with $context\_size = (1, 1)$ of chunks of context.

a short period, this process effectively shortens the live-stream segment that we classify over. Again, the linking of the stream action, video frames and the chat messages is modeled indirectly and must be learned by the model.

With these ideas, we find a tradeoff in every setup. The more complex an architecture becomes, the longer data pre-processing and training will take. Here, we see the challenge as devising the most simple, yet most powerful model.

4.5 Defining evaluation

While we can create different models to address the highlight prediction task, we want to quantify how well they do. Ultimately, the quality of highlights lies in the eye of the viewer. On video publishing platforms like YouTube, we find highlight videos and how well they are received by the community. On the streaming platform Twitch, we find user-generated clips which can also be regarded as highlights and may inform how well a model does at this task. Although these videos provide an indication at how good highlights are, there is still a lot of possibility for variation in personal preference. Some viewer would prefer a longer or shorter highlight or would deem a scene in the video not highlight worthy while the whole video is generally appealing. Thus, we see that using highlight videos as a reference for the quality estimation of this task, is problematic because of the inherent subjectivity of the task. Following this argument, training and evaluating a system on a dataset of these videos is inherently imprecise. These considerations play into the choice of evaluation measure and how precise prediction needs to be. Song et al. [39] tested acceptability of automatically generated videos of "epic moments" which outperformed the expert annotations. Thus, we will evaluate the system empirically on the dataset, but point to actual viewer feedback as a more suitable way of measuring the quality of highlights.

4.5.1 Evaluation on highlight boundaries or segments

Generally, with the task of highlight detection, we have two possible predicted values to analyze. We can define this task as boundary prediction, where the beginning and the end of a highlight need to be detected. Encoding these boundaries is straight forward by simply using a time stamp of seconds since the video start or the frame number. This is exemplified by looking at a highlight segment in the game "nalcs_w4d3_FOX_TSM_g1" as displayed in Figure 2. Here, we find the beginning of the first highlight at 252.93 seconds or 7588, frames and the end at 257.56 seconds or 7727 frames. In contrast, defining segments to classify over, we provide a series of parts which in conjunction make up a highlight segment. In the simplest case, we count all contiguous segments which are classified as highlight towards the same highlight. Both of these approaches come with their own advantages and disadvantages when it comes to evaluating the quality of the highlight detection.

Boundary evaluation If we quantify how well the two boundaries of a highlight segment have been detected, we ground our evaluation in two predictions. Depending on the time measurement, we can achieve a fine granularity with the prediction, possibly resulting in highlight segments with very precise cuts. This precision, however, comes with the problem of defining an appropriate error term. With a large search space of frames/milliseconds and few boundaries in a video, it is important to show how far off a prediction has been in comparison to the gold standard. This is done in object recognition by calculating intersection over union. We can define it over our time intervals and see how many milliseconds or seconds fall within our defined highlight segment. Alternatively, we calculate error measures like mean square error (MSE) on the milliseconds or frames by which the predictions are off. Measures like these are more commonly used as error terms or objectives for machine learning models rather than evaluation measure, as they do not provide a lot of insight about the task performance. Another point with the boundary prediction is the problem of less data points to evaluate. The less information we use to quantify the performance of a technique, the less reliable it is. While predicting boundaries directly is much more closely related to how a human would cut a video, for evaluation purposes it poses some challenges.

Segment Evaluation The second approach, for task definition and evaluation, is to quantify over segments which are individually classified. Here, we can influence the granularity and number of evaluation points by choosing a tighter or less tight window for the task. In practice, this window is determined empirically. Each of these segments directly influences the performance score and thus can be more gradual. This, however, can lead to disjointed predictions, which can be accounted for by applying post-processing to the results of the classification task. One approach is to apply smoothing, like a running average of predictions, over the predicted segments. This is a relatively cheap way to mitigate inconsistencies with disjoint highlight segments. Together, these considerations also mean that this second approach of defining segments is less precise, as with larger segments a boundary may not be able to be hit exactly as defined in the gold standard.

4.5.2 Evaluation measures

In previous works, the latter approach in conjunction with precision, recall and f-score has been dominant. For the purpose of comparability, we use these measures as well and provide more in depth analysis of the results of our experiments. Following [11], we define the set of correctly predicted highlight segments as S_pred and the set of gold standard highlight segments as S_gold. For precision (P), recall (R) and F1-score (F) we use the definitions in Equations 7 and 8.

$$P = \frac{|S_{gold} \cap S_{pred}|}{|S_{pred}|} \qquad R = \frac{|S_{gold} \cap S_{pred}|}{|S_{gold}|} \tag{7}$$

$$F1 = \frac{2PR}{P + R} \tag{8}$$

4.6 Conclusion

In order to address the highlight detection task on live stream chat, we have presented a variety of models. We showed how we calculate temporal features over segments of chat message, especially message density as well as semantic ones with TF-IDF and Word2Vec embeddings. For peak prediction, we introduced SciPy's peak prediction and the real time peak prediction algorithm. Additionally, we explained how the semantic features can be used in Naive Bayes and logistic regression models.

Furthermore, we mentioned how to use a transformer based deep learning architecture for the highlight detection problem. We presented a pre-training approach for Twitch-LeagueBert, a transformer language model pre-trained on Twitch chat from League of

Legends e-sports live streams, and theoretically explored two architectures the model can be used in. Finally, we showed why we chose RoBERTa as a comparison model in these architectures.

In addition, we discussed different approaches for evaluation and defined how we compute precision, recall and f1-score on segments of highlight predictions. Next, we move on to training and evaluating the models we have presented in this chapter.

5 Experiments and Results

Now, with different models defined for the main subject of investigation, detecting highlights with live stream chat data, we train and test them in the following chapter. To this end, we use the dataset introduced by Fu et al. [11] and described in Chapter 3.1 as a training and testing corpus with the previously defined train, validation and test sets. First, we evaluate the techniques as presented in Chapter 4.2 starting with the peak prediction algorithms and next move to semantic features with machine learning techniques. Finally, we train and evaluate different transformer based models on the highlight detection task. To judge the quality of these techniques, we compare these results to the ones reported by [11].

5.1 Baseline investigations

As a baseline for our transformer models, we evaluate less complicated setups. These setups are chosen because of their conceptual and computational simplicity and are informed by previous work. First and foremost, we use message density as an information source. From our first plots of message density and the other features as displayed in Figure 3, we quickly see that message density has the most variation and highest correlation with highlights. Thus, we focus on this feature as the central information metric for our peak prediction algorithms. In the following, we describe the training and evaluation parameters we use.

5.1.1 ScipyPeaks

For the ScipyPeaks predictor we find some parameters by hand through trial and error, in order to judge which parameters may provide best results and also run a more commonly used grid search with cross validation with five folds over a parameter grid. Surprisingly, we find two different configurations, which are found in Table 6. The hand-picked configuration fares better not only in validation, but also in the test dataset. With this configuration, SciPyPeaks achieves 0.32 f-score with 0.40 precision and 0.27 recall.

parameter name	hand-picked	grid search
prominence	0.2	0.14
relative height	0.35	0.7
width	[600, 5000]	[600, 5000]
shift	0.9	0.3
width_scale	0.8	-

Table 6: Parameter configurations for the SciPyPeaks predictor

5.1.2 Real-time peak predictor

With the real-time peak predictor, we also use a grid search with five-fold cross validation to search over a parameter grid. The parameters used are: ”*lag*” = 30, ”*threshold*” = 2.0 and ”*influence*” = 0.7. These parameters achieve 0.17 f-score with 0.3 precision and 0.12 recall. We see that this predictor is significantly worse in finding highlights, resulting in low recall, while the precision is relatively high.

5.1.3 Semantic predictors: logistic regression and multinomial naive bayes

Our last two baselines are the models with semantic features. Here, we use a span of messages from 7 (210 frames) seconds of stream for each document, which we determine by a parameter grid search with five-fold cross validation. With the default parameters, the Scikit-learn models reach the best performance in training[4]. Specifically, the logistic regression classifier with Word2Vec word embeddings reaches an f-score of 0.40 with 0.29 precision and 0.66 recall while the multinomial naive Bayes classifier with TF-IDF features reaches 0.41 f-score with 0.30 precision and 0.65 recall as displayed in Table 7. These two classifiers easily outperform the other baselines, especially in retrieving highlight documents correctly.

model name	precision	recall	f-score
SciPyPeaks	0.40	0.27	0.32
RealTimePeakPredictor	0.30	0.12	0.17
Logistic regression	0.29	0.66	0.40
MultinomialNaiveBayes	0.30	0.65	0.41

Table 7: Precision recall and f-score results for the baseline models on the test set.

5.2 Training TwitchLeagueBert

Before we train task-specific models for highlight detection, we pre-train the TwitchLeagueBert language model on our “Twitch LoL corpus”. We mainly follow the training procedure of RoBERTa, but providing input sequences of a maximum length of 128 tokens and a batch size of 64. We train 1 million steps (14.86 epochs). After 500 000 steps, we save a version of the model for later comparison which we call “TwitchLeagueBert-500k”. For the masked-language-learning objective, we mask 15% of tokens in each epoch. In terms of optimization, we use the same approach as RoBERTa. The training is done on an RTX6000 GPU and takes 7 days.

With this model, we conduct two tests to see if it has learned some of the text found in Twitch chat. We let the model fill some masked tokens and compute perplexity over the datasets.

In order to provide an indication about what TwitchLeagueBert is able to learn from the Twitch LoL corpus, we test the model with phrases which we construct. We provide a masked input sequence of tokens and let the model predict the missing token. We start with a sentence that aims to discover the model’s emote understanding. To this end, we provide the sentence “[emote] this is <mask>.” where we replace [emote] with the following emotes:

- LUL - Laughter. The emote version of Laugh Out Loud.
- ResidentSleeper - For when there’s a lull in action, a boring cut scene or event, or when someone literally falls asleep.
- WutFace - Used to express shock, disgust, or to note a loud, disruptive noise on stream.
- NotLikeThis - Used to express dismay at an outcome, usually due to bad luck or a misplay.

The definition for the emotes are taken from the official Twitch website[29]. TwitchLeagueBert provides predictions for the five most likely tokens to fill the mask. As we can

[29] https://www.twitch.tv/creatorcamp/en/learn-the-basics/emotes/

see in Table 8, generally adjectives are predicted, which is expected. We can relate these to the meanings of the emotes. For "LUL" we first see "hilarious", which exactly represents the meaning of the emote. The same is true for the other emotes. With RoBERTa, however, the results do not correspond to the filled in tokens. While we mainly find adjectives as well, we see that the meaning of the emotes is not captured. This difference is expected and provides a possible insight for highlight prediction models based on these pre-trained models.

emote	TwitchLeagueBert	RoBERTa
LUL	hilarious, sad, awful, amazing, boring	awesome, interesting, great, cool, bad
ResidentSleeper	boring, exciting, it, intense, so	not, me, better, good, for
WutFace	awful, horrible, amazing, terrible, cancer	awesome, great, me, interesting, hilarious
NotLikeThis	sad, embarrassing, painful, awful, ridiculous	private, experimental, JavaScript, personal, interactive

Table 8: Five best predictions for TwitchLeagueBert and RoBERTa for filling the masked sentence "[emote] this is <mask>." beginning with the respective emote.

For gauging how well these models understand the language style on the highlights corpus, we compute the perplexity on this dataset. Perplexity is a measure of how well a language model models a set of inputs. It describes how surprised a model is about an input sequence [38]. Thus, lower perplexity is better. TwitchLeagueBert achieves 12.96 and RoBERTa does 50.12 as presented in Table 9. Here again, we see that RoBERTa does not model the Twitch language style as well as TwitchLeagueBert, which is expected. Neither model has been trained on the highlights corpus.

model name	perplexity
RoBERTa	50.12
TwitchLeagueBert	12.96

Table 9: Perplexity measures for RoBERTa and TwitchLeagueBert on the highlights corpus.

5.3 Transformer model investigations

For the main part of our experiments, we now train and test the transformer models for highlight detection, which are defined in Chapter 4. We choose two approaches, one with sequence classification employing a similar strategy as [11] and another with an additional feature added. With both setups, we test "roberta-base" and "TwitchLeagueBert" for language models. We use the same dataset splits as with the baselines for training, validation, and testing.

5.3.1 Transformer model for sequence classification

We train the AutoModelForSequenceClassification[30] with "roberta-base" and "TwitchLeagueBert" as language models for a maximum of 10 epochs, with possible early stopping if there is no improvement after 3 evaluations. Evaluation on the validation set takes place 20 000 steps on the mean squared error loss, precision, recall and f1-score. The early stopping mechanism is based on f1-score. The learning rate is $2E-5$.

With this setup, we find in Table 11 that both TwitchLeagueBert models perform worse than the RoBERTa based model. We see that the scores on the validation set for TwitchLeagueBert-AutoModel and TwitchLeagueBert-500k-AutoModel are substantially lower than RoBERTa-AutoModel as shown in Table 10. We observe the same trend on the test set, which is shown in Table 11. Thus, we can see a trend in improved model performance when using TwitchLeagueBert which has been pre-trained for more epochs. However, a better performance with the RoBERTa model indicates, that the importance of the structure of messages is exaggerated compared to their meaning with this setup. We suspect, that RoBERTa can interpret the sequences of empty frames in comparison to frames with message contents better than TwitchLeagueBert resulting in a better performance.

5.3.2 Transformer model with additional features

For our second model architecture, we explicitly encode a temporal feature, namely message density, and supply it alongside with compressed chat content. Again, we base these

[30] `https://huggingface.co/docs/transformers/v4.21.3/en/model_doc/auto#transformers.AutoModelForSequenceClassification`

models on RoBERTa and TwitchLeagueBert. The models are trained for a maximum of 25 epochs with early stopping (3 evaluations) based on F1-score and evaluation after every epoch. We train with a batch size of 16 and a learning rate of $2E-5$. Each model is trained on a single RTX6000 GPU.

With this set of models, we observe TwitchLeagueBert performing better overall than RoBERTa. While TwitchLeagueBert-temporal achieves 0.55 precision, 0.44 recall and 0.49 f1-score on the test set, RoBERTa-temporal scores 0.43 precision, 0.24 recall and 0.31 f1-score (see Table 11). Additionally, when comparing TwitchLeagueBert-500k to the full model, we see an improvement for the latter over the former one. Here, precision is traded for recall, with TwitchLeagueBert-temporal achieving a higher overall f1-score. We see these results, both on the test and validation sets (see Table 10). It appears that the test set is represented better by the training set than the validation set, as we see overall higher scores in the test set, even if only marginally.

The greater performance can be traced back to one more explicit feature available to the model in comparison to the previous structure. A full ablation study may prove this. Apart from this, we see RoBERTa-temporal not achieving the same performance. We attribute this to the language model, which is not pre-trained on Twitch language style. With providing as much context as the model can fit, we seem to leverage TwitchLeague-Bert's better modelling of this writing style.

In order to be able to compare the results to the ones of the AutoModel, we need to bring it to frame-level precision. This is done by associating the number of frames with each predicted chunk, and then using the chunk label to label each frame. When evaluating, for TwitchLeagueBert this results in a lower score of 0.43 precision, 0.46 recall and 0.44 f1-score on the test set, RoBERTa-temporal scores 0.21 precision, 0.24 recall and 0.23 f1-score. As expected, we find a drop in performance compared to the coarser output here. Now, comparing the results to the AutoModel approach, RoBERTa seems to be able to access less structural information, resulting without being able to compensate for it with the message density feature. We thus assume the transformer model to be able to pick up on more temporal features than message density alone.

5.4 Comparability of models and results

While different techniques with different encodings are not always directly comparable, we use the same dataset for all the models in training and evaluation, providing us with some comparability. For the AutoModel approaches, we follow the same evaluation strategy as [11]. All the models that we train, perform worse than their [11] best chat model, L-Char-LSTM, with F1-score of 0.432. We thus conclude that this approach, which may be adequate for RNN training, does not yield good performance with transformer models. Our baselines also do not reach the level of their best model, which is expected. MultinomialNaiveBayes produces results, which are only slightly worse, suggesting a model with less complexity and more additional features may work well for this task.

The second model setup produces better results on segments which are formed by the number of chat messages in a fixed length sequence of tokens. Thus, we do not see direct comparability of the results. However, we can revert the predictions back to frame-level granularity, which is done by expanding the prediction of each chunk into the number of frames it contains. With this conversion, we see RoBERTa-temporal drop to 0.23 F1-score. For TwitchLeagueBert-temporal we see a drop to 0.44 f-score barely outperforming the classifier by Fu et al. [11].

model name	P	R	F
TwitchLeagueBert-500k-AutoModel	0.97	0.11	0.19
TwitchLeagueBert-AutoModel	0.82	0.13	0.22
RoBERTa-AutoModel	0.69	0.24	0.35
TwitchLeagueBert-500k-temporal	0.70	0.33	0.45
TwitchLeagueBert-temporal	0.53	0.44	0.48
RoBERTa-temporal	0.49	0.20	0.28

Table 10: Evaluation of the highlight detection models on the validation set. P: Precision, R: Recall, F: F-score.

5.5 Conclusion

Having trained, fined-tuned and evaluated our different models, we see that the TwitchLeagueBert temporal classifier outperforms all other setups. For the baselines, we showed our peak prediction algorithms with message density at the lower end of all models. This

model name	P	R	F
TwitchLeagueBert-500-AutoModel	0.97	0.11	0.20
TwitchLeagueBert-AutoModel	0.84	0.13	0.22
RoBERTa-AutoModel	0.67	0.24	0.36
TwitchLeagueBert-500k-temporal	0.73	0.34	0.46
TwitchLeagueBert-temporal	0.55	0.44	0.49
RoBERTa-temporal	0.43	0.24	0.31
TwitchLeagueBert-1000k-temporal-frame-level	0.43	0.46	0.44
Roberta-temporal-frame-level	0.21	0.24	0.23

Table 11: Evaluation of the highlight detection models on the test set. P: Precision, R: Recall, F: F-score.

did not come as a surprise as we focus on one single feature. However, it showed us, that the amount of messages present in live chat is a helpful feature for highlight prediction. With the semantic baselines, we found out that using word embeddings provides even better results, supporting the conclusion that actual message content is of greater value to this task than temporal information. We thus pre-trained and evaluated a transformer language model specifically for Twitch LoL chat, TwitchLeagueBert, and saw in examples of mask filling, that central information about emotes was picked up quite well in comparison to RoBERTa. Additionally, it reached lower perplexity in a held out corpus. Our two highlight detection model setups, showed disparity in performance. The AutoModel setup for sequence classification only outperformed the peak prediction baselines with the RoBERTa language model. With our second setup, we found that encoding additional features, like message density, in addition to textual information, resulted in better performance for the TwitchLeagueBert based model, suggesting, that it successfully combined the token embeddings with the temporal information. Finally, we conclude that training TwitchLeagueBert as a specialized in-domain language model is beneficial for the highlight detection task in comparison to RoBERTa.

6 Conclusion

In this master thesis, we set out to apply transformer models to the task of highlight detection in League of Legends e-sports live stream chat. To this end, we pre-train and fine-tune a transformer language model which we call TwitchLeagueBert and compare it to RoBERTa [27]. In order to do so, we collect a corpus from live stream chat on the platform Twitch and pre-train TwitchLeagueBert on it. For fine-tuning on the highlight detection task, we use the highlights dataset introduced by Fu et al. [11] and compare the results to their L-Char-LSTM model. In the following, we describe the outcomes of this thesis and point out potential for additional research in this area.

6.1 Outcomes

In order to address the highlight detection problem on Twitch chat, we decide to employ transformer deep learning models. As these models advance the state of the art in many NLP applications, we apply them to this task as well.

As the base step to providing a new approach to this task, we collect a sizable dataset that we use to pre-train our language model. Our Twitch LoL corpus comprises almost 90 million chat messages in a LoL e-sports setting, making it the largest corpus of this kind, that we know of. We provide a methodology for creating a dataset from Twitch chat and show how we clean and preprocess this corpus for further research. We make it available for download at `https://huggingface.co/datasets/Epidot/twitchlolcorpus`. This is a significant addition to resources in the line of live stream chat research, which hopefully provides opportunities to other members of the research community.

Next, we pre-train a RoBERTa based language model on the Twitch LoL corpus and name it TwitchLeagueBert. We follow along the same lines as Liu et al. [27] in order to pre-train this language model. We use this architecture over plain BERT [9] for its improved performance and lower complexity. We also argue that the byte-level BPE tokenizer is more robust and better suited for encoding live stream chat messages. We pre-train TwitchLeagueBert to 1 million iterations and provide examples showing its understanding of Twitch Emotes. We make it available at `https://huggingface.co/Epidot/TwitchLeagueBert-1000k`. With this contribution, we provide another useful tool in live stream language research. Thus, we advance this field of research from Word2Vec token embeddings which were made available by [39] to transformer language models.

Our main finding of this work, is the comparison of highlight detection models in LoL e-sports live streams on chat data. We introduce two approaches for using transformer models on this task and provide performance measures on them. We follow in the line of work of [11] and improve on their L-Char-LSTM with an f1-score of 0.44 over 0.432. Although we do not see the revolutionary improvement that we saw in other NLP tasks when transformers were introduced, we provide a first application of this technique to the highlight detection problem. Our technique successfully reproduces and barely outperforms the previous LSTM technique, showing potential for this kind of application. We additionally show that using a domain specific pre-trained language model is preferred over a more general one. We present these results on our English datasets. We also make our pre-processed datasets and models available for download at `https://huggingface.co/Epidot` providing opportunity for future research into this area.

6.2 Future research and limitations

In closing, we would like to mention areas of our work that allow for further research. Namely, we remark the application of multimodal models to the highlight detection task and different potential setups for transformer training on this task. Additionally, we discuss limitations with our approaches. Here, we mention the inherent subjectivity of highlight detection, as well as existing crowd feedback for highlights clips.

With the inherent multimedia character of live stream interaction consisting of video, audio and viewer generated live chat, it is not a far reach to use all of these factors in a multimodal approach for highlight detection. While we have seen combined models in past research, we have not leveraged this kind of information in this master thesis. Thus, an interesting future direction is applying our chat models together with audio and video analysis models to the task. As [11] and [25] suggest, combining different approaches is fruitful and beneficial to the task.

As discussed in Section 3.1, we found several issues with the highlight corpus. Thus, we encourage the creation of a cleaned up and improved version for future research which mitigates the mentioned problems.

In terms of chat based models and transformers, we have merely provided a first architecture that does work, but can probably be improved. We have not conducted extensive hyperparameter fine-tuning, nor have we explored adding multiple temporal features to our models. We see the potential for other model setups, that include an additional learning objective for detecting if a highlight boundary is close to the classified segment.

With respect to the subjectivity of the highlight detection task, we argue that the quality of a highlight video lies in the eye of the viewer. Thus, conducting viewer acceptance studies, as [39] and [43] do, of the highlights produced by our model would provide more insight into about model performance. Additionally, the dataset used for this work consists of annotations by one singular video production company, making the reference annotation possibly skewed. This may be mitigated by combining highlight segments from multiple videos or even user-generated clips from twtich.tv.

These user generated clips are a way for viewers to save short segments of a live stream for distribution or later reference. As this possibility is given on live stream platforms, this is another possibly data point which may be accessed for automatically creating high-quality highlight videos. We suggest investigations into using user-generated clips in combination with model prediction for better highlight detection.

References

[1] Francesco Barbieri, Luis Espinosa-Anke, Miguel Ballesteros, Juan Soler-Company, and Horacio Saggion. Towards the Understanding of Gaming Audiences by Modeling Twitch Emotes. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 11–20, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4402. URL https://aclanthology.org/W17-4402.

[2] Iz Beltagy, Kyle Lo, and Arman Cohan. SciBERT: A Pretrained Language Model for Scientific Text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1371. URL https://aclanthology.org/D19-1371.

[3] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly Media, Inc., Sebastopol, first edition, 2009. ISBN 978-0-596-51649-9.

[4] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake Vanderplas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API Design for Machine Learning Software: Experiences from the Scikit-Learn Project. In *European Conference on Machine Learning and Principles and Practices of Knowledge Discovery in Databases*, September 2013. URL https://hal.inria.fr/hal-00856511.

[5] Denis Bulygin, Ilya Musabirov, Alena Suvorova, Ksenia Konstantinova, and Pavel Okopnyi. Between an Arena and a Sports Bar: Online Chats of eSports Spectators. *arXiv:1801.02862 [cs]*, December 2020. URL http://arxiv.org/abs/1801.02862.

[6] Wei-Ta Chu and Yung-Chieh Chou. Event Detection and Highlight Detection of

Broadcasted Game Videos. In *Proceedings of the 2nd Workshop on Computational Models of Social Interactions: Human-Computer-Media Communication*, 2015.

[7] Wei-Ta Chu and Yung-Chieh Chou. On Broadcasted Game Video Analysis: Event Detection, Highlight Detection, and Highlight Forecast. *Multimedia Tools and Applications*, 76(7):9735–9758, 2017. URL https://link.springer.com/content/pdf/10.1007/s11042-016-3577-x.pdf.

[8] Kevin Clark, Minh-Thang Luong, and Quoc V Le. ELECTRA: PRE-TRAINING TEXT ENCODERS AS DISCRIMINATORS RATHER THAN GENERATORS. In *International Conference on Learning Representations*, 2020. URL https://nlp.stanford.edu/pubs/clark2020electra.pdf.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

[10] Colin Ford, Dan Gardner, Leah Elaine Horgan, Calvin Liu, a. m. tsaasan, Bonnie Nardi, and Jordan Rickman. Chat Speed OP PogChamp: Practices of Coherence in Massive Twitch Chat. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, pages 858–871, Denver Colorado USA, May 2017. ACM. ISBN 978-1-4503-4656-6. doi: 10.1145/3027063.3052765. URL https://dl.acm.org/doi/10.1145/3027063.3052765.

[11] Cheng-Yang Fu, Joon Lee, Mohit Bansal, and Alexander Berg. Video Highlight Prediction Using Audience Chat Reactions. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 972–978, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1102. URL https://aclanthology.org/D17-1102.

[12] Zhiwei Gao, Shuntaro Yada, Shoko Wakamiya, and Eiji Aramaki. A Preliminary Analysis of Offensive Language Transferability from Social Media to Video Live Streaming. *The 34th Annual Conference of the Japanese Society for Artificial Intelligence*, 2020.

[13] Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. Domain-Specific Language Model Pretraining for Biomedical Natural Language Processing. *ACM Transactions on Computing for Healthcare*, 3(1):1–23, 2021. ISSN 2691-1957, 2637-8051. doi: 10.1145/3458754. URL `https://dl.acm.org/doi/10.1145/3458754`.

[14] Pawara Gunawardena, Oshada Amila, Heshan Sudarshana, Rashmika Nawaratne, Ashish Kr. Luhach, Damminda Alahakoon, Amal Shehan Perera, Charith Chitraranjan, Naveen Chilamkurti, and Daswin De Silva. Real-Time Automated Video Highlight Generation with Dual-Stream Hierarchical Growing Self-Organizing Maps. *Journal of Real-Time Image Processing*, 18(5):1457–1475, October 2021. ISSN 1861-8219. doi: 10.1007/s11554-020-00957-0. URL `https://doi.org/10.1007/s11554-020-00957-0`.

[15] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. Don't Stop Pretraining: Adapt Language Models to Domains and Tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.740. URL `https://aclanthology.org/2020.acl-main.740`.

[16] Hung-Kuang Han, Yu-Chen Huang, and Chien Chin Chen. A Deep Learning Model for Extracting Live Streaming Video Highlights Using Audience Messages. In *Proceedings of the 2019 2nd Artificial Intelligence and Cloud Computing Conference*, pages 75–81, Kobe Japan, December 2019. ACM. ISBN 978-1-4503-7263-3. doi: 10.1145/3375959.3375965. URL `https://dl.acm.org/doi/10.1145/3375959.3375965`.

[17] Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. Deep

Unordered Composition Rivals Syntactic Methods for Text Classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1681–1691, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1162. URL https://aclanthology.org/P15-1162.

[18] Ruochen Jiang, Changbo Qu, Jiannan Wang, Chi Wang, and Yudian Zheng. Towards Extracting Highlights from Recorded Live Videos: An Implicit Crowdsourcing Approach. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 1810–1813. IEEE, 2020. URL https://ieeexplore.ieee.org/abstract/document/9101800.

[19] Yifan Jiao, Zhetao Li, Shucheng Huang, Xiaoshan Yang, Bin Liu, and Tianzhu Zhang. Three-Dimensional Attention-Based Deep Ranking Model for Video Highlight Detection. *IEEE Transactions on Multimedia*, 20(10):2693–2705, October 2018. ISSN 1941-0077. doi: 10.1109/TMM.2018.2815998.

[20] Katikapalli Subramanyam Kalyan, Ajit Rajasekharan, and Sivanesan Sangeetha. AMMUS : A Survey of Transformer-based Pretrained Models in Natural Language Processing. *arXiv:2108.05542 [cs]*, August 2021. URL http://arxiv.org/abs/2108.05542.

[21] Seok-Kyu Kang and Jee-Hyong Lee. An E-sports Video Highlight Generator Using Win-Loss Probability Model. In *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, SAC '20, pages 915–922, New York, NY, USA, March 2020. Association for Computing Machinery. ISBN 978-1-4503-6866-7. doi: 10.1145/3341105.3373894. URL https://doi.org/10.1145/3341105.3373894.

[22] Mehdi Kaytoue and Arlei Silva. Watch Me Playing, I am a Professional: A First Study on Video Game Live Streaming. *WWW '12 Companion: Proceedings of the 21st International Conference on World Wide Web*, 2012.

[23] Konstantin Kobs, Albin Zehe, Armin Bernstetter, Julian Chibane, Jan Pfister, Julian Tritscher, and Andreas Hotho. Emote-Controlled: Obtaining Implicit Viewer Feedback Through Emote-Based Sentiment Analysis on Comments of Popular Twitch.Tv Channels. *ACM Transactions on Social Computing*, 3(2):1–34, May 2020. ISSN 2469-7818, 2469-7826. doi: 10.1145/3365523. URL `https://dl.acm.org/doi/10.1145/3365523`.

[24] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.703. URL `https://aclanthology.org/2020.acl-main.703`.

[25] Chieh-Ming Liaw and Bi-Ru Dai. Live Stream Highlight Detection Using Chat Messages. In *2020 21st IEEE International Conference on Mobile Data Management (MDM)*, pages 328–332, Versailles, France, June 2020. IEEE. ISBN 978-1-72814-663-8. doi: 10.1109/MDM48529.2020.00072. URL `https://ieeexplore.ieee.org/document/9162302/`.

[26] Rainer Lienhart. Comparison of Automatic Shot Boundary Detection Algorithms. In *SPIE Vol 3656*, pages 290–301, San José, 1999.

[27] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv:1907.11692 [cs]*, July 2019. URL `http://arxiv.org/abs/1907.11692`.

[28] Guangyi Lv, Tong Xu, Enhong Chen, Qi Liu, and Yi Zheng. Reading the Videos: Temporal Labeling for Crowdsourced Time-Sync Videos Based on Semantic Embedding. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1), March

2016. ISSN 2374-3468. URL https://ojs.aaai.org/index.php/AAAI/article/view/10383.

[29] Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. Extensions of Recurrent Neural Network Language Model. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5528–5531, May 2011. doi: 10.1109/ICASSP.2011.5947611.

[30] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. *arXiv:1301.3781 [cs]*, September 2013. URL http://arxiv.org/abs/1301.3781.

[31] Ilya Musabirov, Denis Bulygin, Paul Okopny, and Ksenia Konstantinova. Event-Driven Spectators' Communication in Massive eSports Online Chats. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI EA '18, New York, NY, USA, April 2018. Association for Computing Machinery. ISBN 978-1-4503-5621-3. doi: 10.1145/3170427.3188447. URL https://doi.org/10.1145/3170427.3188447.

[32] Moin Nadeem, Anna Bethke, and Siva Reddy. StereoSet: Measuring Stereotypical Bias in Pretrained Language Models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5356–5371, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.416. URL https://aclanthology.org/2021.acl-long.416.

[33] Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, and David Cournapeau. Scikit-Learn: Machine Learning in Python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[34] Qing Ping and Chaomei Chen. Video Highlights Detection and Summarization with Lag-Calibration Based on Concept-Emotion Mapping of Crowdsourced Time-Sync

Comments. In *Proceedings of the Workshop on New Frontiers in Summarization*, Copenhagen, Denmark, 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4501. URL http://aclweb.org/anthology/W17-4501.

[35] XiPeng Qiu, TianXiang Sun, YiGe Xu, YunFan Shao, Ning Dai, and XuanJing Huang. Pre-Trained Models for Natural Language Processing: A Survey. *Science China Technological Sciences*, 63(10):1872–1897, October 2020. ISSN 1674-7321, 1869-1900. doi: 10.1007/s11431-020-1647-3. URL https://link.springer.com/10.1007/s11431-020-1647-3.

[36] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models Are Unsupervised Multitask Learners, 2019. URL https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

[37] Charles Ringer and Mihalis A. Nicolaou. Deep Unsupervised Multi-View Detection of Video Game Stream Highlights. In *Proceedings of the 13th International Conference on the Foundations of Digital Games*, FDG '18, New York, NY, USA, August 2018. Association for Computing Machinery. ISBN 978-1-4503-6571-0. doi: 10.1145/3235765.3235781. URL https://doi.org/10.1145/3235765.3235781.

[38] R. Rosenfeld. Two Decades of Statistical Language Modeling: Where Do We Go from Here? *Proceedings of the IEEE*, 88(8):1270–1278, August 2000. ISSN 1558-2256. doi: 10.1109/5.880083.

[39] Hyeonho Song, Kunwoo Park, and Meeyoung Cha. Finding Epic Moments in Live Content through Deep Learning on Collective Decisions. *EPJ Data Science*, 10(1), December 2021. ISSN 2193-1127. doi: 10.1140/epjds/s13688-021-00295-6. URL https://epjds.epj.org/articles/epjdata/abs/2021/01/13688_2021_Article_295/13688_2021_Article_295.html.

[40] Yale Song. Real-Time Video Highlights for Yahoo Esports. *arXiv:1611.08780 [cs]*, November 2016. URL http://arxiv.org/abs/1611.08780.

[41] Yi Sun, Zhijian Ou, Wei Hu, and Yimin Zhang. Excited Commentator Speech Detection with Unsupervised Model Adaptation for Soccer Highlight Extraction. In *2010 International Conference on Audio, Language and Image Processing*, pages 747–751, November 2010. doi: 10.1109/ICALIP.2010.5685077.

[42] Yarden Tal, Inbal Magar, and Roy Schwartz. Fewer Errors, but More Stereotypes? The Effect of Model Size on Gender Bias. In *Proceedings of the 4th Workshop on Gender Bias in Natural Language Processing (GeBNLP)*, pages 112–120, Seattle, Washington, July 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.gebnlp-1.13.

[43] Muhammad Umair, Yungi Jang, and Jiwoo Park. LiveLight: Crowd Generated Highlights. Course Project Work of university students, 2016.

[44] J.P.G. van Brakel. Robust Peak Detection Algorithm Using Z-Scores, 2014. URL https://stackoverflow.com/a/22640362.

[45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

[46] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Glue: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *7th International Conference on Learning Representations*, 2019. URL http://www.scopus.com/inward/record.url?scp=85083952595&partnerID=8YFLogxK.

[47] Bin Wu, Erheng Zhong, Ben Tan, Andrew Horner, and Qiang Yang. Crowdsourced Time-Sync Video Tagging Using Temporal and Personalized Topic Modeling. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 721–730, New York, NY, USA, Au-

gust 2014. Association for Computing Machinery. ISBN 978-1-4503-2956-9. doi: 10.1145/2623330.2623625. URL https://doi.org/10.1145/2623330.2623625.

[48] Yikun Xian, Jiangfeng Li, Chenxi Zhang, and Zhenyu Liao. Video Highlight Shot Extraction with Time-Sync Comment. In *Proceedings of the 7th International Workshop on Hot Topics in Planet-scale mObile computing and Online Social neTworking*, HOTPOST '15, pages 31–36, New York, NY, USA, June 2015. Association for Computing Machinery. ISBN 978-1-4503-3517-1. doi: 10.1145/2757513.2757516. URL https://doi.org/10.1145/2757513.2757516.

[49] Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. LUKE: Deep Contextualized Entity Representations with Entity-aware Self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6442–6454, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.523. URL https://aclanthology.org/2020.emnlp-main.523.

[50] Yunzhi Yao, Shaohan Huang, Wenhui Wang, Li Dong, and Furu Wei. Adapt-and-Distill: Developing Small, Fast and Effective Pretrained Language Models for Domains. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 460–470, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-acl.40. URL https://aclanthology.org/2021.findings-acl.40.