# TSCSet: A Crowdsourced Time-Sync Comment Dataset for Exploration of User Experience Improvement

**Zhenyu Liao**[1]*, **Yikun Xian**[2]*, **Xiao Yang**[1], **Qinpei Zhao**[1], **Chenxi Zhang**[1], **Jiangfeng Li**[1] †
[1] Tongji University, Shanghai, China
[2] Rutgers University, New Jersey, USA
102982liaozy@tongji.edu.cn, siriusxyk@gmail.com,
doris.yx@gmail.com, qinpeizhao@tongji.edu.cn,xzhang2000@163.com, lijf@tongji.edu.cn

## ABSTRACT
Time-Sync Comment (TSC) is a type of crowdsourced user review embedded in online video websites, which provides better real-time user interaction than traditional user comment type. Various TSC-related problems and approaches have been studied to improve user experience by taking advantage of special characteristics of TSCs such as strong time reliance. However, there are three major drawbacks to these TSC researches. First, they did not explicitly show advantage of TSC features over the traditional features in terms of users' experience. Second, the experiments were conducted on some inconsistent TSC datasets crawled from different source, which makes the effectiveness of their methods less convincing. Third, the methods were manually evaluated by a limited number of so-called "experts" in these experiments, so it is hard for other researchers to obtain the data labels and reproduce the results. In order to overcome these drawbacks, this paper aims to explore the usefulness of TSC data for for the improvement of user experience online by exploiting the TSC pattern inside a new dataset. Specifically, we present a larger-scale TSC dataset with four-level structures and rich self-labeled attributes and formally define a group of TSC-related research problems based on this dataset. The problems are solved by adapted state-of-the-art methods and evaluated through crowdsourced labels in the dataset. The result can be regarded as a baseline for further research.

### Author Keywords
Crowdsourced time-sync comment; hierarchical structured dataset; episode representation learning; storyline prediction

## INTRODUCTION
As an essential application in Internet, online video websites have been gaining increasing popularity in recent years. In addition to watching videos, users now would like to have more interaction with other audience. Time-sync comment (TSC; also called *Danmu* in Chinese and *Danmaku* in Japanese) is a new textual information on video content that has been applied to many online video websites such as Acfun[1] and Bilibili[2] in China, Niconico[3] in Japan and so forth. TSC is a type of crowdsourced user review organized by video playback time. Audience can bulletin TSCs at any time to share their feelings and opinions about the corresponding video shot. Other audience will see the TSC at the same playback time, so they may react with their own opinions through their TSC. This kind of interaction enriches description of videos and provides a brand new kind of textual data in the analysis of video content and movie reviews. Some substantial researches on TSC have emerged since 2014 including video tagging [35], highlight detection [38] and video key frame recommendation [6]. These studies showed that TSC data embodies great potential in semantic analysis on videos. However, their work on TSC only focused on specific scenarios, lack of a full scale analysis of online clients' experience based on a common dataset. Furthermore, most experiments were conducted on their selected individual datasets. Therefore, these assessments are incomparable on a common ground. As such, a standardized dataset and a complete assessments concerning different aspects are needed.

Based on above motivation and intuition, we present a large scale dataset of TSC in this study, and aim to explore the different levels of assessment methods regarding online users' experience. As shown in Figure 1, TSC data collected from online video websites are reorganized into a four-level structure, season-episode-comment-user, in order to capture hierarchical relation between TSC and other elements. Our full-scale TSC dataset contains nearly 900 seasons of animations, over 17K episodes and over 32 million comments generated by approximately three million users. It not only exceeds any existing TSC datasets, but also contains more abundant attributes, some of which can be regarded as labels as well, such as "tags" and "likes". By taking advantage of this dataset, we are able to study how TSC data can be used to improve user experience through state-of-art techniques in many problems. Our main contributions include:

*These authors contributed equally to this work.
†Corresponding Author


*IUI 2018, March 7–11, 2018, Tokyo, Japan.*

http://dx.doi.org/10.1145/3172944.3172966

[1] http://www.acfun.cn (accessed December 11, 2017)
[2] https://www.bilibili.com/ (accessed December 11, 2017)
[3] http://www.nicovideo.jp/ (accessed December 11, 2017)

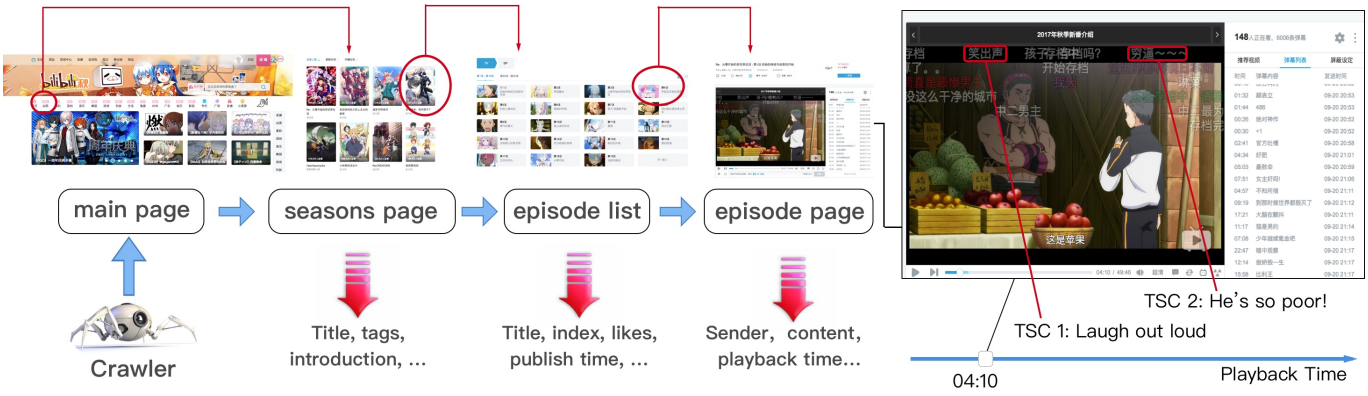



Figure 1: An overview of dataset structure and how our crawler obtained TSC data from Bilibili.com

- Presenting a large-scale TSC dataset with four-level structure and enriched self-labeled attributes collected from the most popular TSC-enabled online video website Bilibili.com through a customized web crawler.

- Formally defining four TSC-related research problems based on our dataset covering topics of high practical value such as representation learning, season tagging, season recommendation and storyline classification.

- Proposing solutions to these problems using state-of-the-art techniques and effective assessment that can be used as baselines for future study.

The rest of this paper is organized as follows. We first summarize existing work on TSC in the past few years, followed by detailed information about the datasets including how the data is obtained and basic statistics. Then, we define a series of TSC-related problems based on this dataset, followed by solutions and evaluations. Finally, the paper is concluded with discussion a summary of our work and some possible future work on this dataset.

## RELATED WORK

*Crowdsourcing.* Large-scale data is everywhere, but usually unlabeled in real world. Crowdsourcing emerging at the right moment provides a suitable way to label data by crowdworkers [2]. For instance, ImageNet [8] and Microsoft coco [21] are two successful crowdsourced datasets in computer vision that lead to numerous research achievements. Crowdsourcing also benefits research areas like natural language processing where human linguistics annotation is expensive and time-consuming [28]. Although noise and bias can hardly be avoided during crowdsourcing process, some advanced techniques are proposed to deal with resulting problems including multiple annotations [34], incorrect annotation [36] and debiasing [7], etc. In this paper, TSC is also a type of crowdsourced data and is similarly generated by "crowdworkers" (users who watch online videos), but one major difference is that these users naturally express their reviews on videos rather than be delibrately asked to label data like real "crowdworkers".

*Video Tagging.* Traditional video tagging problem aims to automatically generate tags for entire videos and content-based methods have been widely studied and successfully applied [27, 31, 33]. When it comes to TSC, the first substantial research was done by Wu *et al.* [35] who aimed to extract keywords from TSCs as the tags of video shots through a temporal and personalized topic model. Yang *et al.* [40] additionally conducted thorough analysis on TSC features such as semantic relevance, real-time, interdependence and noise data. Xu *et al.* [39] extracted representative comments instead of keywords from TSC because complete sentences convey more meaningful information in terms of video content. However, datasets in these studies didn't provide tags of videos clips and the methods were only evaluated by limited number of "experts".

*Popularity Prediction.* Video popularity analysis is to study relation between video popularity and video-relevant features and how to leverage them to predict popularity in the future. Some traditional features include intrinsic statistical properties of popularity distributions [5], average viewing rate over some time [25] and "rich get richer" behavior [4], etc. Wu *et al.* [37] firstly investigated the co-relation between volume of TSC and popularity based on number of replays and bookmarks of videos. He *et al.* [12] then introduced concept of Herding Effect on TSC and predicted popularity by considering multiple factors including popular videos, popular TSCs, newly updated videos, uploaders influence and video quality. The work was further extended by He *et al.* [11] to detect leading TSC and predict growth of future TSC. However, characteristics of TSC comments were not fully taken into account, *e.g.* semantic meaning behind comment and user sentiment toward videos reflect upon video popularity trend.

**Key Frame Extraction.** Video key frame extraction is a process of presenting a summary of an entire video with a small set of representative frames [29]. Traditional methods mainly analyze image information and relation among frames [9, 22, 30]. Another research topic of TSC is to extract key frames in the video. Xian *et al.* [38] first tried extracting highlight shots from video based on temporal topics of TSC. Li *et al.* [20] similarly proposed to detect event in video using

TSC by considering sequence of user behaviors. Lv *et al*. [23] took advantage of special properties of TSC and represented TSC as semantic vectors and then split video to extract and label meaningful video clips by mapping semantic vectors to manually-defined labels. Recently, Chen *et al*. [6] made use of both video images and TSCs for key frame extraction and recommendation. However, the same issue as video tagging is that there is no labeled key frames of video available in these datasets.

These researches did a nice job on enhancement of traditional methods to solve problems like video tagging and key frame extraction using TSC information. However, as abovementioned, most datasets in these studies did not provide any ground truth like tags and key frames. The experiments could only be evaluated by a small group of so-called "experts" and hence the results may be hard to reproduce and not very convincing.

In this paper, we redefine TSC-related research problems in reference to the existing work by taking advantage of our crowdsourced self-labeled dataset. Furthermore, deep learning based methods [19] are adapted and applied to solve these problems in consideration of its tremendous success in many research areas. The breakthrough of deep learning happened in 2012 when AlexNet [17] was proposed to solve image classification on the well-known large-scale ImageNet [8]. New techniques have been invented to make deeper neural network easier to train in order to achieve better performance. For example, residual structure [10] enables neural network to be substantially deeper and works effectively in both visual and non-visual tasks. In the recent work of TSC, Chen *et al*. [6] also used convolutional neural network to analyze video frame together with TSC information. In the area of natural language processing, word embedding techniques such as Word2Vec [24] and Paragraph2Vec [18] outperforms traditional methods like topics models because they can preserve semantic meaning and syntactic relationship. Deep learning was also successfully applied to solving problems including the sentiment analysis of rotten tomato movie reviews [15] and topic categorization of news articles [16] and video caption generation [32], which are similar topics to TSC problems in this paper.

## DATASET OVERVIEW AND STATISTICS DESCRIPTION

In this section, we will basically answer following questions. Where and how is TSC data collected and organized into the dataset? What properties do our dataset have as compared with previous work? What are the statistical information hidden in our dataset?

### Dataset Overview

A customized web crawler is designed and implemented to collect TSC data in the animation section of Bilibili.com, the most popular TSC-enabled website in China. Firstly, the crawler starts with requesting seasons list in animation section. Information about seasons can be obtained according to the seasons list, which is in JSON format. Then, the crawler enumerates every season in the list and requests their episodes list through another API where each episode uniquely corresponds to one TSC file ID. This enables crawler to download TSC files of XML format according to their file ID. Attributes and content of every single TSC can be retrieved in these XML files and they are stored in a local database, together with season and episode details. Due to the well-functioning APIs provided by Bilibili.com, the data we obtained are almost structured and few data-cleaning works is required during the collection.

Using the crawler mentioned above, we collected two kinds of dataset in this paper for different scenarios: *BL-906* and *BL-D-64*. BL-906 is a larger-scale dataset containing 906 seasons and over 30 million TSCs collected during the period from Aug. 23rd, 2017 to Sept. 3rd, 2017. However, it is not dynamic, *i.e*. changes of TSCs with time are unable to track because the crawler only stores available TSC data when it visits corresponding episode pages. To make up this defect, BL-D-64 is another TSC dataset with dynamic property that is being collected since Sept. 30th, 2017 and will be kept updated in the future. Particularly, a modified crawler is developed to update TSC records in this dataset every two hours in order to keep all the history copies of TSCs in the dataset. The number of seasons in BL-D-64 is relatively smaller than that in BL-906 because we only track hot animations on the air where TSCs were frequently updated during the period.

Therefore, our datasets are different from all existing ones featured by four following characteristics.

1. *Hierarchical Data Structure.* This dataset consists of data organized as a four-level structure (season-episode-comment-user) compared with traditional two-level structure (episode-comment).
2. *Large Data Scale.* This dataset has larger scale than any previous ones in terms of the number of episodes (videos), TSCs, users, etc.
3. *Multi-Dimensional Data Attributes.* This dataset contains more abundant attributes such as season tags, episode coins (likes) and user history that can be regarded as labels for supervised learning problems.
4. *Dynamic Data Update.* This dataset is "dynamic" since it records variation of TSCs for each episode as time elapses while all previous datasets are "static" that only save a single copy at a certain timestamp.

Table 1 illustrates the scale of the dataset and its comparison with previous datasets. In this paper, our work is mainly based on BL-906. The usage of the other dataset will be discussed in the last section. Both datasets are accessible from `https://github.com/Viscount/IUI-Paper`.

### Statistics Description

Statistical results of the dataset will be described according to four levels: seasons, episodes, comments and users.

#### *Season-level Statistics*

There are 906 seasons in total, all of them are valid, *i.e*. these seasons are associated with a unique set of episodes and TSCs in the dataset. Meanwhile, in a season, there are two useful information: 1) a short passage of animation introduction, and 2) a set of tags of animation styles. There are totally

| | BL-906 | BL-D-64 | Paper[35] | Paper[23] | Paper[12] | Paper[11] | Paper[39] | Paper[6] | Paper[40] |
|---|---|---|---|---|---|---|---|---|---|
| Source | bilibili | bilibili | acfun.tv | bilibili | acfun.tv | acfun.tv | iqiyi | bilibili | acfun.tv |
| # Seasons | **906** | 64 | None | None | None | None | 2 | None | None |
| # Videos | **17,870** | 716 | 16,414 | Unknown | 3,623 | 6,506 | 6 | 7166 | 120 |
| # TSCs | **32,949,297** | 7,413,517[(i)] | 1,103,884 | 133,250 | 60,956 | 1,704,930 | 234,003 | 11,842,166 | 227,780 |
| # Users | **3,240,478** | 1,482,120 | 382,752 | Unknown | 278,520 | 320,000 | Unknown | 1,133,750 | Unknown |
| Tags | Yes | Yes | No | No | No | No | No | No | Yes[(ii)] |
| Likes | Yes | Yes | No | No | Yes | No | No | No | No |
| Dynamic | No | **Yes** | No | No | No | No[(iii)] | No | No | No |

[(i)] The count comes from the copy of BL-D-64 by Dec. 31, 2017.
[(ii)] It contains videos from only three categories: music, sport and movie.
[(iii)] It only shows number of TSCs varying with time.

Table 1: Brief summary of this dataset and comparison with previous datasets.

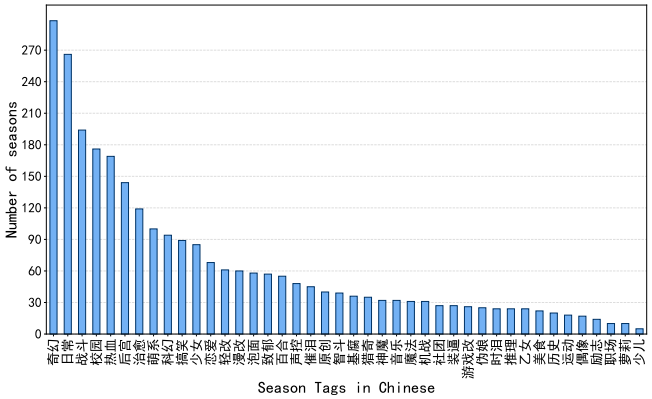



Figure 2: Distribution of number of seasons per tag

43 different tags and each season is assigned around 3 tags on average. The distribution of number of seasons per tag is shown in figure 2.

### *Episode-level Statistics*

There are 17,870 episodes in total and average 19.88 episodes per season. Each season includes one or more episodes and the distribution of number of episodes per season is shown in figure 3a. Aggregated video length of all episodes is 47,835 hours and over 97.63% episodes have video length less than 30 minutes, so only these episodes are considered in this section and its distribution is shown in figure 3b. One useful attribute of episode is "coins" representing how many users like the episode and distribution of ratio of coins within episode is shown in figure 3c.

### *Comment-level Statistics*

There are 32,949,297 TSCs in total, average 36,651 TSCs per season and average 1,857 TSCs per episode. Distribution of number of TSCs per season, number of TSCs along with episode index and number of TSCs along with video playback time are respectively shown in figures 4a, 4b, 4c.

Generally, TSC is featured by following properties:

- *Succinctness.* On average each TSC has 8.71 Chinese characters which is even shorter than Sina Weibo (20 Chinese characters) and Tweet length (30 characters). The distribution of various TSC length is shown in figure 4d. This is because audiences prefer to express their feelings with a few words when watching videos. Otherwise, longer comment may have a negative impact on audience watching experience, for example, to fail to fast read long comments without pausing video.
- *Slanginess.* TSC is usually made of Internet buzzwords among young audiences. Top 40 frequent segmented TSC words are shown in figure 4e. For example, the most frequent word is "233" which is actually not a Chinese word, but a textual expression representing "laugh". Meanwhile, these phrases are not of fixed length because there could be lots of trailing "3"s, and hence it is critical to separately preprocess TSC.
- *Herding Effect.* A user's new comment will be affected by existing comments within a time range. Herding effect on TSC has been deeply studied in [12], so detail will not be covered in this paper.

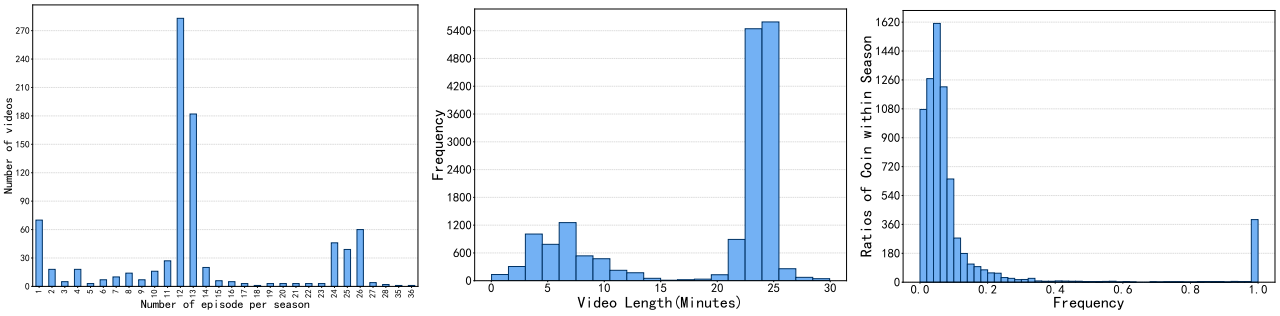
### *User-level Statistics*

Each TSC is associated with a sender ID that is regarded as visible "audience" or "user" in this paper. We will ignore those who watched the video but didn't leave a comment. We also assume one user is somehow interested in the video if and only if he/she has published at least one TSC for the video. In this dataset, there are totally 3,240,478 such users, the average number of TSCs per user published is 10.17 and average number of seasons watched by a user is 2.66.

## TSC-RELATED PROBLEMS

In this section, we formally define four TSC-related research problems to explore user experience improvement by exploiting features and patterns inside time-sync comments. These problems include TSC-based representation learning, season multi-tagging, season recommendation and storyline classification. The significance of selected problems mainly lies in two aspects below:

- Problems like representation learning are the essential researches in machine learning area[3]. Moreover, problems like tagging and recommendation are always hot topics in TSC-related studies.




(a) Distribution of number of episodes per season (limited to # episodes ≤40)

(b) Distribution of video length per season (limited to length ≤30 minutes)

(c) Distribution of ratio of coins within season

Figure 3: Episode-level Statistics

- All the problems are related with a certain business scenario in real world. TSC-enabled online video websites like Bilibili.com will benefit from breakthroughs in the research of these problems.

Formally, the four-level structure of the dataset $\mathbb{D}$ is composed of $\mathbb{S},\mathbb{E},\mathbb{C},\mathbb{U}$, four universal sets respectively representing seasons, episodes (videos), TSCs and users. Basically, each season contains a set of episodes, each episode contains a set of TSCs, and each user can publish a set of TSCs. A TSC $c \in \mathbb{C}$ is defined as a quadruple, i.e. $c \doteq (e,r,t,u)$ where $e \in \mathbb{E}$ is the episode the TSC associated with, $r$ is textual content of user review, $t$ is playback time of TSC in the video, and $u \in \mathbb{U}$ is the one who published the TSC. Let $C_{\cdot|u} \doteq \{c \in \mathbb{C} | c.u = u\}$ be the set of TSCs published by user $u$, $C_{\cdot|e} \doteq \{c \in \mathbb{C} | c.e = e\}$ be the set of TSCs associated with episode $e$ and $C_{\cdot|s} \doteq \{c \in \mathbb{C} | c.e \in s\}$ be the set of TSCs associated with season $s$. Let $E_{\cdot|u} \doteq \{e \in \mathbb{E} | \exists c \in \mathbb{C}, c.e = e, c.u = u\}$ to be the set of episodes where user $u$ published any TSCs and $E_{\cdot|s} \doteq \{e \in \mathbb{E} | e \in s\}$ be the set of episodes associated with season $s$. Let $S_{\cdot|u} \doteq \{s \in \mathbb{S} | \exists c \in \mathbb{C}, c.e \in s, c.u = u\}$ be the set of seasons where user $u$ has published at least one TSC in any episode that belongs to the season.

### TSC-based Representation Learning Problem

*Scenario Description.* One practical problem is how to represent as a vector a set of TSCs associated with an episode or a user? Raw TSC data can hardly be used directly for further analysis in tagging, population detection, key frame extraction, etc, due to special properties of TSC including succinctness and slanginess. Meanwhile, temporal relation between TSCs should be captured in vector representation because the it reflects upon influential factors including previous TSCs and recent video frames. The problem aims to automatically learn feature representation of TSC set and capture abovementioned properties.

*Problem Definition.* Given any TSC set $C \in \mathbb{C}$, the goal is to represent $C$ as a vector $\mathbf{C}$ such that

1. Given length $d \in \mathbb{N}^+$, $|\mathbf{C}| = d$.
2. Given some similarity measure $\rho$ and error $\varepsilon$, if $C$ is similar to another TSC set $C'$, then their vector representations are also similar, i.e. $\rho(\mathbf{C},\mathbf{C}') < \varepsilon$.
3. $\forall c_i, c_j \in C$, if timestamp $t_i$ of $c_i$ and timestamp $t_j$ of $c_j$ are swapped leading to a new TSC set $C'$, i.e. $c_i = (e_i,r_i,t_j,u_i), c_j = (e_j,r_j,t_i,u_j) \in C'$, then $\mathbf{C} \neq \mathbf{C}'$.

The first constraint ensures that any TSC set $C$ can be mapped to a vector of the same length $d$. The second constraint states that any vectorized TSC sets will remain similarity under some similarity measure. The third constraint means that temporal relation of every single TSC matters in $\mathbf{C}$. A special case is when $|C| = 1$, the goal is to vectorize a single TSC.

### Season Multi-Tagging Problem

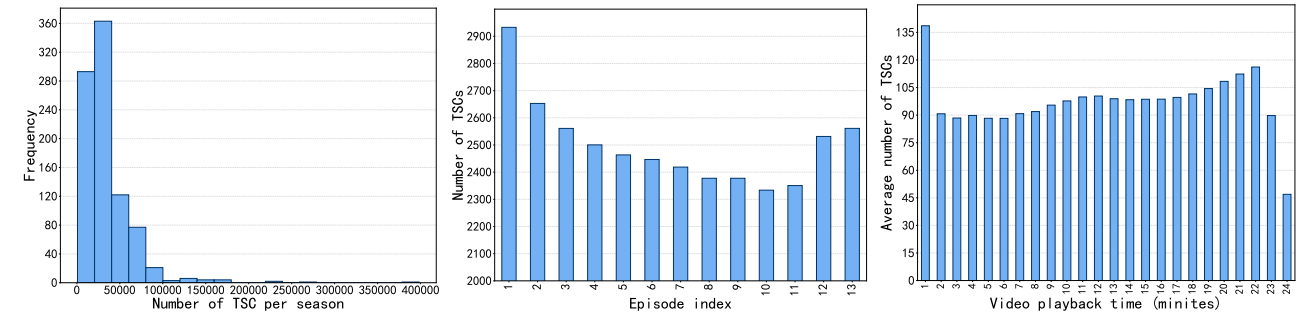
*Scenario Description.* In Bilibili.com, most seasons are assigned with predefined labels based on genre and plot, hence these tags can assist users in searching seasons easily by keywords. For example, the well-known animation *Neon Genesis Evangelion* is tagged with "Sci-Fi", "Action" and "Robot Fight". However, one issue is that all these tags are labeled manually by editors and it takes much time and effort to repeat on thousands of seasons. According to our observation, TSCs vary from each other according to different season tags, so it is possible to infer tags for a new season from its TSC data automatically.

*Problem Definition.* Given a set of $m$ training examples $\{(x_1,y_1),\ldots,(x_m,y_m)\}$ drawn from an unknown distribution $D$, the goal is to learn a classifier $f : X \to Y$ with maximum classification score. $x_i \doteq C_{\cdot|s_i}$ is the TSC set associated with season $s_i$, $y_i \in \{0,1\}^k$ where $k$ is number of tags, $X$ is the space of TSC set $\mathbb{C}$ and $Y$ is the space of binary vector of size $k$. Classification score here is defined as

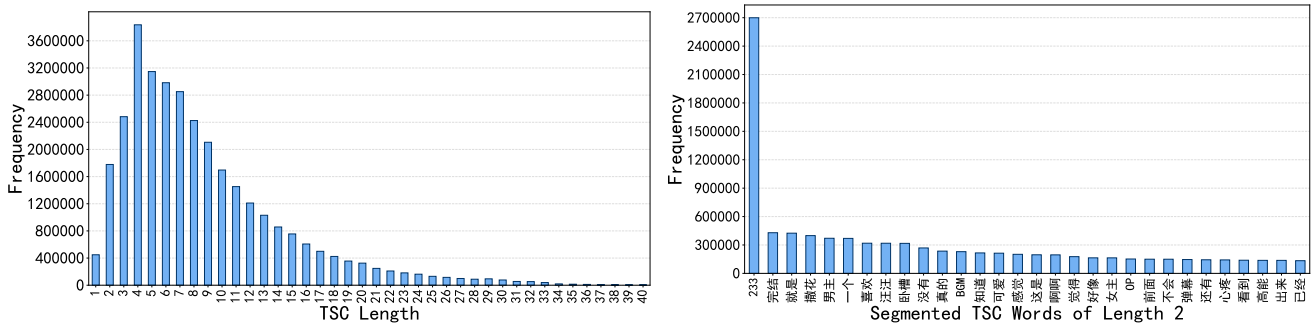
$$s_C(f,D) = \sum_{(x_i,y_i)\in X\times Y} D(x_i,y_i) \sum_{j=1}^{k} \mathbb{1}(y_i^j, f_i^j(x)) \qquad (1)$$

### Season Recommendation Problem

*Scenario Description.* Recommendation is becoming a fundamental function for any content-sharing website in recent years and it can help users autonomously explore the content in a website with better experience. For example, hundreds




(a) Distribution of number of TSCs per season (b) Number of TSCs changing with episodes (limited to the seasons of length 12 or 13) (c) Number of TSCs changing with video playback time (limited to 24-min length)




(d) Distribution of TSC length (limited to length $\leq 40$) (e) Frequency of segmented TSC words of two Chinese characters (Top 30)

Figure 4: Comment-level Statistics

of animations will be newly released every quarter and users have to watch many reviews before determining which animation to follow. Recommender system can provide a list of new animation seasons that may interest users based on their preference and history. Most existing recommender systems exploit ratings and comments to infer user preferences through collaborative filtering or content-based approach. For TSC-based online video website, larger-scale TSC reviews provide rich detail on video contents, especially overall audience reaction, and hence can be taken into consideration to better improve the quality of recommender system.

*Problem Definition.* Given a set of $n$ users $U = \{u_1, .., u_n\} \subseteq \mathbb{U}$ and an associated season set $S_{\cdot|u_i}$ for each $u_i \in U$, the goal is to learn a function $f : \mathbb{U} \times \mathbb{S} \rightarrow \{0, 1\}$ with maximum recommendation score. In this setting, each $u_i$ is attached with TSC history $C_{\cdot|u_i} \subseteq \mathbb{C}$ that can be taken into consideration by $f$. Recommendation score is defined as

$$s_R(f, \ell) = \sum_{u \in U} \sum_{s \in \mathbb{S}} \ell(s, f(u, s) | S_{\cdot|u}) \tag{2}$$

where $U$ is the space of user set, $S$ is the space of season set, and $\ell(\cdot)$ is some score function to measure to what level season $s$ interests user $u$ according to history $S_{\cdot|u}$.

### Storyline Classification Problem

*Scenario Description.* According to Freytag's pyramid, the storyline of a drama can be divided into five parts: exposition, rising action, climax, falling action, and denouement [14]. Similarly, every season composes of a complete storyline and each episode of the season belongs to one of these five parts. For example, there are 26 episodes in the well-known animation *Neon Genesis Evangelion*. Episode 24 is considered as a climax because of the depiction of final battle while episode 4 is considered as exposition, for the content in this episode is just Shinji's (lead character) daily life. To simplify the problem, we only consider two aggregated parts: whether one episode is climax or not within a season. From our observation, the content of each episode is highly correlated to user feedback such as TSCs and "likes" that is equivalent to "coins" in our dataset. A very practical problem that can we predict whether an episode is climax or not based on history of associated TSCs?

*Problem Definition.* Given a set of $m$ seasons $S = \{s_1, .., s_m\}$ where each season $s_i$ is associated with an ordered set of episodes $E_{\cdot|s_i}$ and an ordered set of labels $y_i \in \{0, 1\}^{|E_{\cdot|s_i}|}$ indicating whether an episode $e_{ij} \in E_{\cdot|s_i}$ is climax or not within the season $s_i$, the goal is to learn a function $f : \mathbb{E} \rightarrow \{0, 1\}$ with maximum storyline classification score $s_{SC}$. In this setting, $f$ can takes as input all episodes $e'$ prior to the current episode $e$ and their associated TSC sets $C_{\cdot|e'}$ and outputs predicts whether $e$ is climax or not. Storyline classification score is defined as

$$s_{SC}(f) = \sum_{s \in \mathbb{S}} \sum_{e_i \in E_{\cdot|s}} \mathbb{1}(f(e_i), y_i) \tag{3}$$

## METHODS AND EVALUATIONS

In this section, we will solve four problems by adapting state-of-the-art methods and then evaluate the effectiveness of TSC

in improving performance. The pipeline of the complete procedure from data crawling and preprocessing to problem solving can be divided into four steps. First, we crawl TSC data from Bilibili and save the four-level raw data into a relational database. Second, we perform a thorough Chinese textual preprocessing on raw TSC data. This includes removing punctuation, auxiliary word and emojis, replacing slangs with standard words through regular expression, for example substituting "233333" to "233", and splitting every comment into Chinese words. Third, we conduct basic statistics on TSCs to find interesting properties just as shown in the previous section. Finally, we try to solve four selected problems as follows.

#### Representation Learning Problem

*Method.* We mainly adapt *paragraph2vec* [18] to the TSC representation learning problem because it can learn continuous vector representation of fixed length as well as take two important factors into account as known as word order in the document and semantic meaning of each word. Analogous to documents and words, we can regard TSC set of an episode as document since temporal dependency of TSC plays as an important role as order of words. Nevertheless, there are two questions worth thinking about.

- What is the granularity in the dataset, *i.e.* taking each TSC as a whole or splitting TSC into words?
- How to choose suitable vector dimension for TSC representation?

For the first question, we prefer to split TSC into words and reconstruct all split words in a TSC set as a "document" in *paragraph2vec*. This is based on following observation with respect to TSC according to our statistical findings in the paper. Although TSC itself is short, taking TSC as a whole may lose semantic information. Some single word in TSC occurs frequently, but they usually mixed with other less-frequent words. For example, let's assume there are two comments $r_1$ and $r_2$ and both of them contains some top frequent word $w_2$, *e.g.* "233". Then, it is most likely that $r_1$ and $r_2$ have similar semantic meaning, *e.g.* laughing or interesting video clip. It implies that if we regard $r_1$ and $r_2$ as a whole, *paragraph2vec* will treat them as two different "words" and fail to capture similarity between them from context. Therefore, it is better to split each TSC into separate words and treat each of them as a "word" in document. For the second question, in reference to parameter setting in *GloVe* [26], vector size is typically selected from 100 to 300 because dimension less than 100 will lead to very poor performance and dimension greater than 300 makes no significant improvement. We conducted a preliminary experiment on one-vs-42 binary classification in order to evaluate four different choices of vector dimension for TSC representation, *i.e.* , $d = 100, 200, 256, 300$. The result showed that classification performance of $d = 100$ was the worst and other three settings make no big difference in performance. As a rule of thumb, we chose $d = 200$ as the global setting for vector dimension in *paragraph2vec* in consideration of both performance and simplicity.We will discuss evaluation of TSC vector representation in detail through season tag classification problem in the next section.

#### Season Multi-Tagging Problem

*Method.* Since each season can be assigned multiple tags, this is actually a multi-label classification problem and we developed two methods to deal with it. The first one is to split the large problem into $K$ binary classification subproblems. For each subproblem, we train a shallow neural network with consistent structure. The input is 200-dimensional encoded vector, followed by three fully-connected layers of size 256, 128 128, and each layer is also connected with one Relu layer and one dropout layer. The advantage of this one-vs-all mode is that each subproblem is very easy to solve even if the labels are imbalanced. However when $K$ is large, it will take lots of time and memory to train and store $K$ binary classifier so it's not suitable for large scale case. The second approach is to create only one deeper neural network to solve this entire problem, and this turns out to be a hard problem. We are inspired by residual network[10] and adapt it to this problem. As shown in figure 5, the input also takes 200-dimensional encoded vector, followed by three residual substructures of size 256, 512, 256. Each residual substructure takes $\mathbf{x}_{in}$ as input and output $\mathbf{x}_{out}$ of the same size:

$$h_1 = p(\sigma(W_1\mathbf{x}_{in})) \tag{4}$$

$$h_2 = p(\sigma(W_2 h_1)) \tag{5}$$

$$\mathbf{x}_{out} = p(\sigma(W_3 h_2 \oplus h_1)) \tag{6}$$

where $\sigma(\cdot)$ is Relu activation layer, $p$ is dropout layer with keeping probability 0.1, and $\oplus$ is elementwise addition.

The output of the network is a fully connected layer of size $K$, the same size as number of total tags. Since labels are mostly imbalanced, a modified weighted cross entropy loss is used here. Assume the output of instance $\mathbf{x}_i$ is $\hat{\mathbf{y}}_i$ and the real label vector is $\mathbf{y}_i$, the loss is computed as weighted cross entropy:

$$\ell_i^k = -\mathbf{y}_i^k \log(\sigma(\hat{\mathbf{y}}_i^k)) - (1-\mathbf{y}_i^k)\log(1-\sigma(\hat{\mathbf{y}}_i^k)) \tag{7}$$

$$L(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{mK}\sum_{i=1}^{m}\sum_{k=1}^{K} w_k \ell_i^k \tag{8}$$

where $\ell_i^k$ is the cross entropy loss of class $k$ of instance $i$, $m$ is sample size and $w_k$ is the weight for class $k$:

$$u_k = \log(\frac{1}{m}\sum_{i=1}^{m}\mathbb{1}[\hat{\mathbf{y}}_i^k = 1]) \tag{9}$$

$$w_k = 2 - u_k / \min_k\{u_k\} \tag{10}$$

*Evaluation.* We conduct two sets of experiments to interpret both effectiveness of representation learning using *paragraph2vec* and performance of our method for all-in-one multi-label classification. All models are implemented and tested in Tensorflow [1]. Specifically, minibatch size is 128, learning rate is $10^{-4}$ and will decay by 5% every 5000 steps, and solver is Adam optimizer.

First, we apply shallow net to three different encoding of textual information of season or episode for one-vs-all binary classifications (42 classes), which is used to reflect upon the usefulness of representation method working for episode encoding. In detail, "Intro2vec" method only utilizes season introduction to encode it into vector through *paragraph2vec*.

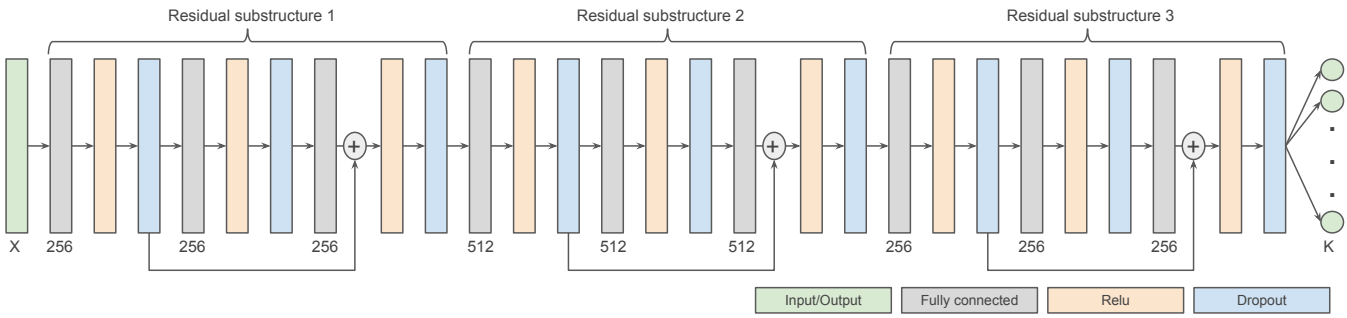



Figure 5: Deep neural network structure for multi-tagging problem

| One-vs-42 binary classification (ShallowNet) | | | | |
|---|---|---|---|---|
| | Accuracy | Precision | Recall | F1 Score |
| Intro2vec | 0.9288 | 0.1591 | 0.0511 | 0.7735 |
| MeanW2v | 0.9691 | 0.8660 | 0.5807 | 0.6952 |
| TSC2vec | **0.9820** | **0.9262** | **0.7522** | **0.8302** |
| 42-in-one multi-label classification | | | | |
| | Accuracy | Precision | Recall | F1 Score |
| ShallowNet | 0.9365 | **0.6110** | 0.4538 | 0.5208 |
| DeepNet | **0.9382** | 0.6107 | **0.5138** | **0.5581** |

Table 2: Experiment results of multi-tagging problem

"MeanW2v" method takes TSC set of each episode and encodes it using traditional *word2vec*. "TSC2vec" method that is proposed in this paper also takes TSC set of episode but encodes through *paragraph2vec*. The vector dimension for three cases keeps 200 all the time. Then, we use the shallow network to train 42 binary classifiers mapping these vectors to tags. The entire results of 42 classifiers are shown in figure 6. It is obvious that "TSC2vec" outperforms the other two methods in all metrics evaluation (accuracy, precision, recall, F1). Therefore, we can conclude that TSC data is really useful in representing episode or season.

Second, we compare the proposed deep net to the same shallow net in all-in-one multi-label classification problem. As shown in table 2, the result implies that this is a relatively harder problem than binary classification since the shallow net performs much worse than the previous case. Although our deep net performs pretty same as shallow net in terms of accuracy and precision, it is much better in recall and F1 score. This implies that the model can predict more correct tags among ground truth and it's more useful in practical scenarios.

### Season Recommendation Problem

*Method.* In season recommendation problem, four solutions are proposed, including collaborative filtering, content-based recommendation as well as two methods using TSC data.

- **Collaborative Filtering** Collaborative filtering is a traditional method in recommendation problems. We prefer user-based collaborative filtering in this problem, for it's easier to get users' watching history than to compute the similarity between two seasons. For a target user $u$, we calculate a score to estimate how a certain season $s$ interests him, defined as

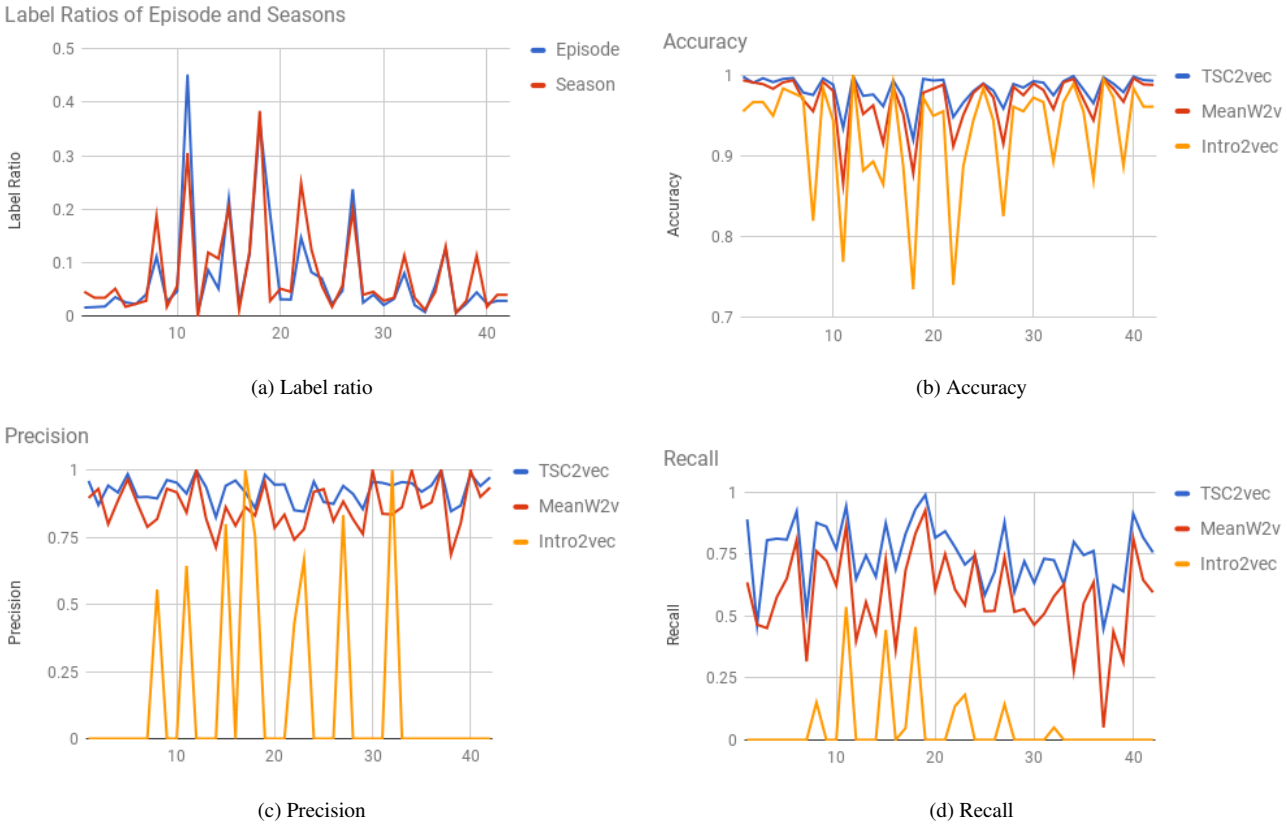
$$score_{u,s} = \sum_{n \in N} \sum_{s \in \mathbb{S}} sim(u,n) * S_{s|n} \tag{11}$$

where N is the space of $u$'s neighbors who have watched at least one same season as $u$. $sim(u,n)$ is the cosine similarity between $u$ and $n$, and $S_{s|n}$ is the watching history of neighbor $n$ on season $s$. Finally, all the scores are normalized into [0,1], a threshold $w$ is set and seasons are recommend to the target user if their score is greater than $w$.

- **Content-based Method** In content-based method, recommending an item is based upon a description of the item and profiles of users' interests. In this case, we extract semantic features of a season by applying Paragraph2vec model on its introduction text. Seasons are represented by a 200-dimension vector. For each user, a classifier is trained to represent his profile of interest. Finally, we can get the results through his classier that whether a certain season is recommended to this user.

- **Season-level TSC-based Method** In our proposal, we use TSC data instead of season's introduction to train the Paragraph2vec model. To bridge from a single TSC review to a representation of a season, we firstly combine all the TSCs in this season into one document order by episode index and their position in video playback time. A corresponding vector representation for every season can be calculated by Paragraph2vec model, which is also a n-dimension vector ($n$=200 as mentioned in content-based method). After that, we learn a classifier to represent every user with season representation vector and his watching history, then predict the label of new seasons through this classifier.

- **Episode-level TSC-based Method** The effectiveness of classifier training in season-level method may be affected by the limited number of seasons (only 906 seasons). Thus we expand the season-level method to episode-level (17870 episodes in dataset). That is, we apply Paragraph2vec model with combined TSCs in one episode, and get vector representation for every episode through this model. We feed these episode vector representations for classifier training instead of season vector representations.




(a) Label ratio

(b) Accuracy

(c) Precision

(d) Recall

Figure 6: Experimental results for 42 binary classification subproblems

| | Method | Accuracy | Precision | Recall |
|---|---|---|---|---|
| Naive Bayes | *A* | 0.9385 | 0.5489 | 0.6197 |
| | *B* | 0.6112 | 0.1199 | 0.6459 |
| | *C* | **0.9593** | **0.7134** | 0.3076 |
| | *D* | 0.9210 | 0.5086 | **0.6579** |
| Neural Network | *A* | 0.9385 | 0.5489 | **0.6197** |
| | *B* | 0.9326 | 0.3359 | 0.2577 |
| | *C* | 0.9577 | 0.6976 | 0.3030 |
| | *D* | **0.9655** | **0.7178** | 0.4353 |

Table 3: Experimental results on season recommendation problem. *A*,*B*,*C*,*D* represent Collaborative Filtering, Content-based Method, Season-level TSC-based Method, Episode-level TSC-based Method.

*Evaluation.* To evaluate forementioned solutions in season recommendation problem, a series of experiments are conducted on current dataset. At first, 2717 users who sent over 500 TSCs in our dataset are selected to avoid cold start problem. For every user, his watching history on 906 seasons is represented as a 906-dimension vector with 0/1 label. 80% of them are selected as a train set, and the other 20% are test set. There are some parameters configuration and model selection issues here. In our experiments, $w$ in Collaborative Filtering method is set to 0.5. As for classifier selection, we chose two different models: Naive Bayes and Neural Network. Specifically, for Neural Network, learning rate is $10^{-5}$, hidden-layer size is 256, and solver is L-BFGS.

Three typical metrics in classification problems are used to evaluate the effectiveness of methods: accuracy, precision and recall rate. We found that the partition of train set and test set has an important effect on the result. Thus, we choose top 500 users on precision to evaluate the overall effectiveness of methods. The result of experiments are shown in Table 3. Compared with two methods without TSC data, season-level and episode-level TSC-based methods archive higher precision and accuracy, especially when using Neural Network as classifier. We also tried other models, such as SVM and decision tree, but the effectiveness of them are far worse then these two models. However, which is the best classifier model in this problem still needs further study.

### Storyline Classification Problem

*Method.* According to Herding Effect, highlight of current episode depends not only on current TSC set, but also is highly relevant to previous ones. In order to capture such history dependency, we introduce LSTM [13] to solving the problem of predicting whether current episode is highlight or not within some given season. The model takes as input a sequence of TSC sets of episodes, followed by two fully connected layers of sizes 256, 128 respectively and each of them is connected with Relu activation layer and dropout layer with 0.3 keeping

probability. An LSTM layer is then added to the network of cell size 128 connected with another dropout layer with 0.3 keeping probability. A final output layer of size two is connected to LSTM layer, followed by cross entropy loss. Considering that number of episodes varies from seasons to seasons, we enable the network to take dynamic size of inputs and outputs by feeding with a prior variable indicating the sequence length. We call this method All-in TSC History Dependent Model (*ATHDM*) because it incorporates cross entropy loss of every single episode within a season. Suppose $y_i^j$ is the climax label for episode $i$ in season $j$ and $\hat{y}_i^j$ is the predicted label, the loss function $L(\mathbf{y},\hat{\mathbf{y}})$ is defined as:

$$L(\mathbf{y},\hat{\mathbf{y}}) = \frac{1}{|S|}\sum_{j=1}^{|S|}\sum_{i=1}\ell(\hat{y}_i^j, y_i^j) \tag{12}$$

where $|S|$ is number of seasons and $\ell(\cdot)$ is cross entropy loss.

*Evaluation.* We first clarify how climax label can be obtained in our dataset. Basically, climax episode can be measured by number of coins from users. However one issue is that it is infeasible to set a fixed threshold for all episodes because number of coins differs from season to season. Therefore, a variant of ratio of coin is introduced as label $y_i^j$ in this experiment:

$$y_i^j = \mathbb{1}[\frac{n_i^j}{\sum_{i=1}^{L_j} n_i^j} > \theta] \tag{13}$$

where $L_j$ is number of episodes in season $j$, $n_i^j$ is number of coins in episode $i$ and $\theta$ is the global threshold indicating climax episode in a season. In reference to Figure 3c, $\theta$ is set to 0.1, 0.2 and 0.3 in order to verify robustness of our model under different settings of climax labels.

In terms of hyperparameters setting of *ATHDM*, batch size is 64, learning rate is $10^{-5}$ and decays by 10% every 5000 steps, and total number of steps is $10^5$. We compare the method to a basic model named TSC History Independent Model (*THIM*), which treats each episode independently and predicts binary label $\hat{y}_i^j$ of current episode without incorporating history TSC of previous episodes. Since it is similar problem of binary classification, we use the same neural network as tagging problem except hyperparameters setting, *i.e.* batch size=32, learning rate=$10^{-5}$ and number of steps=$10^5$. The results are shown in table 4, where we can find that *ATHDM* significantly outperforms *THIM* in terms of accuracy, precision, recall and F1 score. Thus we can draw conclusion that highlight of an episode depends both on current TSC content and all previous history.

| $\theta$ | Method | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|---|
| 0.1 | THIM | 0.8692 | 0.6034 | 0.5412 | 0.5706 |
| | ATHDM | **0.9023** | **0.7317** | **0.6186** | **0.6704** |
| 0.2 | THIM | 0.9512 | 0.5172 | 0.2500 | 0.3371 |
| | ATHDM | **0.9536** | **0.5500** | **0.3667** | **0.4400** |
| 0.3 | THIM | 0.9743 | 0.1667 | 0.0370 | 0.0606 |
| | ATHDM | **0.9768** | **0.4000** | **0.0741** | **0.1250** |

Table 4: Experimental results on storyline classification problem with $\theta = 0.1, 0.2, 0.3$.

## CONCLUSION AND FUTURE WORK

In this paper, we presented a new crowdsourced TSC dataset with larger scale and richer attributes compared with existing work. In order to improve user experience on online video websites, we explored TSC-related problems by exploiting special characteristics and patterns behind TSCs. Specifically, four research problems were formally defined and deep learning techniques were adapted and applied to these problems. Unlike most previous researches where experiments were manually evaluated by a very small group of "experts", all methods in this paper were evaluated through labels in the dataset. The results showed that TSC-enhanced methods perform better than the methods based on the traditional way, and can be used as a baseline for further studies. One limitation of the dataset is that it only contains textual data but no visual information like video frames due to copyright restriction of video websites.

In the future, our work can be extended in two directions. The first track is to develop more effective algorithms to solve four proposed problems by incorporating better TSC features and additional visual information like video frames. For example, the sequential character of TSC can be considered in the model for season recommendation problem. The second track is to explore more research problems by using characteristics in this dataset. Some possible problems are listed below.

- *TSC Denoising Problem*. In order to prevent overwhelming TSC displayed on a video screen, the total capacity of TSCs of a video is fixed and dependent on video length. Therefore, TSCs are now being regularly cleaned by human (we have confirmed with Bilibili), which definitely costs lots of time and effort. It would be very useful to solve this denoising problem by learning a filter that can clean TSCs automatically. The filter observes variation of TSCs at different timestamps and learns latent rules similar to what human does.

- *TSC Generation Problem.* A common process for a user to publish a TSC is as follows. When the user watched some attractive shots or read some interesting TSCs by others, he/she is probably willing to write some comments and then publishes as TSC. Since the content is largely influenced by existing TSCs according to Herding Effect, is it possible to provide an intelligent user interface that is able to automatically generate some comments based on current video content for users to choose from?

- *Season Popularity Prediction Problem.* Episode of a season is usually released every week and users will leave feedback like TSCs accordingly. One interesting problem is to predict how popular a season will be by analyzing the trend of popularity among all previous episodes. Unlike some existing work that only studied the amount of likes or comments, our dataset tracks dynamic TSC sets of each episode at different timestamps, so it is possible to analyze season popularity through variation of TSCs of each episode.

**ACKNOWLEDGEMENT**

This study was funded by the National Natural Science Foundation of China (Grant No.61702372) and the Youth Science and Technology Foundation of Shanghai, China (NO.15YF1412600).

**REFERENCES**

1. Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, and others. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* (2016).
2. Daniel Archambault, Helen Purchase, and Tobias Hoßfeld. 2015. *Evaluation in the Crowd*. Springer.
3. Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. *Representation Learning: A Review and New Perspectives*. IEEE Computer Society. 1798–1828 pages.
4. Youmna Borghol, Sebastien Ardon, Niklas Carlsson, Derek Eager, and Anirban Mahanti. 2012. The untold story of the clones: content-agnostic factors that impact YouTube video popularity. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1186–1194.
5. Meeyoung Cha, Haewoon Kwak, Pablo Rodriguez, Yong-Yeol Ahn, and Sue Moon. 2009. Analyzing the video popularity characteristics of large-scale user generated content systems. *IEEE/ACM Transactions on Networking (TON)* 17, 5 (2009), 1357–1370.
6. Xu Chen, Yongfeng Zhang, Qingyao Ai, Hongteng Xu, Junchi Yan, and Zheng Qin. 2017. Personalized key frame recommendation. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 315–324.
7. Abhimanyu Das, Sreenivas Gollapudi, Rina Panigrahy, and Mahyar Salek. 2013. Debiasing social wisdom. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 500–508.
8. Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition, 2009.* IEEE, 248–255.
9. Yihong Gong and Xin Liu. 2000. Video summarization using singular value decomposition. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, Vol. 2. IEEE, 174–180.
10. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016b. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.
11. Ming He, Yong Ge, Enhong Chen, Qi Liu, and Xuesong Wang. 2017. Exploring the Emerging Type of Comment for Online Videos: DanMu. *ACM Transactions on the Web (TWEB)* 12, 1 (2017), 1.
12. Ming He, Yong Ge, Le Wu, Enhong Chen, and Chang Tan. 2016a. Predicting the Popularity of DanMu-enabled Videos: A Multi-factor View. In *International Conference on Database Systems for Advanced Applications*. Springer, 351–366.
13. Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
14. C Hugh Holman and William Harmon. 1992. *A handbook to literature*. Macmillan Publishing Company.
15. Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. 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)*, Vol. 1. 1681–1691.
16. Rie Johnson and Tong Zhang. 2014. Effective use of word order for text categorization with convolutional neural networks. *arXiv preprint arXiv:1412.1058* (2014).
17. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.
18. Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*. 1188–1196.
19. Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436–444.
20. Jiangfeng Li, Zhenyu Liao, Chenxi Zhang, and Jing Wang. 2016. Event Detection on Online Videos Using Crowdsourced Time-Sync Comment. In *7th International Conference on Cloud Computing and Big Data (CCBD), 2016*. IEEE, 52–57.
21. Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*. Springer, 740–755.
22. Tianming Liu, Hong-Jiang Zhang, and Feihu Qi. 2003. A novel video key-frame-extraction algorithm based on perceived motion energy model. *IEEE transactions on circuits and systems for video technology* 13, 10 (2003), 1006–1013.
23. Guangyi Lv, Tong Xu, Enhong Chen, Qi Liu, and Yi Zheng. 2016. Reading the Videos: Temporal Labeling for Crowdsourced Time-Sync Videos Based on Semantic Embedding.. In *AAAI*. 3000–3006.
24. Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

25. Siddharth Mitra, Mayank Agrawal, Amit Yadav, Niklas Carlsson, Derek Eager, and Anirban Mahanti. 2011. Characterizing web-based video sharing workloads. *ACM Transactions on the Web (TWEB)* 5, 2 (2011), 8.
26. Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.
27. Stefan Siersdorfer, Jose San Pedro, and Mark Sanderson. 2009. Automatic video tagging using content redundancy. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. ACM, 395–402.
28. Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y Ng. 2008. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the conference on empirical methods in natural language processing*. Association for Computational Linguistics, 254–263.
29. C Sujatha and Uma Mudenagudi. 2011. A study on keyframe extraction methods for video summary. In *International Conference on Computational Intelligence and Communication Networks (CICN), 2011*. IEEE, 73–77.
30. Ba Tu Truong and Svetha Venkatesh. 2007. Video abstraction: A systematic review and classification. *ACM transactions on multimedia computing, communications, and applications (TOMM)* 3, 1 (2007), 3.
31. Adrian Ulges, Christian Schulze, Markus Koch, and Thomas M Breuel. 2010. Learning automatic concept detectors from online video. *Computer vision and Image understanding* 114, 4 (2010), 429–438.
32. Subhashini Venugopalan, Marcus Rohrbach, Jeffrey Donahue, Raymond Mooney, Trevor Darrell, and Kate Saenko. 2015. Sequence to sequence-video to text. In *Proceedings of the IEEE international conference on computer vision*. 4534–4542.
33. Meng Wang, Richang Hong, Guangda Li, Zheng-Jun Zha, Shuicheng Yan, and Tat-Seng Chua. 2012. Event driven web video summarization by tag localization and key-shot identification. *IEEE Transactions on Multimedia* 14, 4 (2012), 975–985.
34. Peter Welinder, Steve Branson, Pietro Perona, and Serge J Belongie. 2010. The multidimensional wisdom of crowds. In *Advances in neural information processing systems*. 2424–2432.
35. Bin Wu, Erheng Zhong, Ben Tan, Andrew Horner, and Qiang Yang. 2014. 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*. ACM, 721–730.
36. Xian Wu, Wei Fan, and Yong Yu. 2012. Sembler: Ensembling Crowd Sequential Labeling for Improved Quality.. In *AAAI*.
37. Zechen Wu and Eisuke Ito. 2014. Correlation analysis between user's emotional comments and popularity measures. In *IIAI 3rd International Conference on Advanced Applied Informatics (IIAIAAI), 2014*. IEEE, 280–283.
38. Yikun Xian, Jiangfeng Li, Chenxi Zhang, and Zhenyu Liao. 2015. 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*. ACM, 31–36.
39. Linli Xu and Chao Zhang. 2017. Bridging Video Content and Comments: Synchronized Video Description with Temporal Summarization of Crowdsourced Time-Sync Comments.. In *AAAI*. 1611–1617.
40. Wenmian Yang, Na Ruan, Wenyuan Gao, Kun Wang, Wensheng Ran, and Weijia Jia. 2017. Crowdsourced time-sync video tagging using semantic association graph. In *2017 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 547–552.