The Pennsylvania State University
The Graduate School

CHARACTERIZATION, UNDERSTANDING, AND PREDICTION OF TEMPORAL DYNAMICS OF WEB CONTENTS

A Dissertation in
Information Sciences and Technology
by
Yiming Liao

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2020

The dissertation of Yiming Liao was reviewed and approved by the following:

Dongwon Lee
Associate Professor of College of Information Sciences and Technology
Dissertation Advisor, Chair of Committee

Xiang Zhang
Associate Professor of College of Information Sciences and Technology

Suhang Wang
Associate Professor of College of Information Sciences and Technology

Lin Lin
Assistant Professor of Department of Statistics

Mary Beth Rosson
Professor of College of Information Sciences and Technology
Director of Graduate Programs for College of Information Sciences and Technology

Abstract

Time series data is ubiquitous in various domains, such as users' behavioral change over time on e-commercial platforms and content popularity evolution on social media. Analyzing sequential data and characterizing temporal dynamics make it possible to uncover causal factors leading to future series data. Besides, the ability to detect potential temporal patterns contributes to predicting future changes and developing timely strategies, providing a wide range of applications (e.g., improving user retention and content recommendation). In this dissertation, I will show our path along with characterization, understanding, and prediction of the temporal dynamics of web content.

First, we start with two case studies about characterizing temporal patterns on two different domains. In the first study, we investigate users' funding behavior patterns on crowdfunding platforms (e.g., Indiegogo and Kickstarter). Employing time series clustering methods, we discover four distinct temporal funding patterns on both platforms. In the second study, we examine news articles' long-term popularity patterns. Although the majority of news articles are only prevalent for a very short time, there are a small fraction of news articles displaying much longer popularity, thus named as evergreen news articles. Motivated by the fact that evergreen news articles maintain a timeless quality and are of consistent interest to the public, we analyze evergreen articles and shed light on their long-term popularity. Second, to help readers better understand event evolution, journalists usually summarize a compact but complete storyline from thousands of news articles for each event, which is both time-consuming and labor-intensive. In order to deliver more accurate news timelines to the audience, we develop a fast and effective news timeline summarization algorithm to achieve state-of-the-art performances in both quality and speed. Third, we go beyond characterization and understanding the temporal evolution of web contents for prediction. More specifically, we introduce a temporal translation based framework to learn dynamic graph embeddings. Our framework allows us to train all graph snapshots simultaneously while still preserving the temporal constraint in learning, making it scalable to industrial-level graphs.

Finally, we close this dissertation by discussing the limitation of our works and signaling potential directions.

Table of Contents

List of Figures

List of Tables

Acknowledgments

First and foremost, I would like to thank my advisor, Prof. Dongwon Lee, for his guidance during my Ph.D. journey at Penn State. This dissertation would not have been possible without his invaluable advice, encouragement, and patience. He has granted me the opportunity and freedom to explore many ideas I was interested in, even if we had several disagreements in the first few years while it usually turned out that he was right. I truly appreciate his forbearance on my willfulness. I will be eternally grateful for his sharing of knowledge in both research and life, and sincerely wish that one day I could approach his level.

I also would like to thank my other committee members, Prof. Xiang Zhang, Prof. Suhang Wang, and Prof. Lin Lin, for their many insightful suggestions and inspiring discussions on my dissertation. Without the input from all of them, I could not have made the journey this far.

During my study at Penn State, I have been extremely fortunate to work with many amazing collaborators. I would like to thank Prof. Kyumin Lee and Than Tran for helping me start this journey smoothly. I would also like to thank Dr. Eui-Hong (Sam) Han and Shuguang Wang for their great support at the Washington Post in the summer of 2017 and 2018. Specifically, I am indebted to Everdeen Mason, Sophie Ho, Greg Barber, and their journalist team at the Washington Post for providing precious feedback for our work. I also thank Dr. Darío García García and Dr. Erik Brinkman for their fabulous mentorship at Facebook AI Applied Research in the summer of 2019.

I would like to thank all my talented fellow lab mates: Dr. Hau-Wen chang, Jiasheng Zhang, Stephanie Winkler, Thai Quang Le, Hae Seung Seo, Limeng Cui, Adaku Uchendu, Tracy Shen, Spyke Krepshaw, Maryam Tabar, for making this journey enjoyable. I am honored to be in the PIKE family.

I would also like to thank my fiancée, Xin Zheng, for her unconditional love and endless support that helped me go through ups and downs in this journey. Her smile always lights up my life.

Finally, I would like to express my sincerest gratitude to my parents, Caixing Liao

and Guxia Zhao. They have offered me constant encouragement and immeasurable support during my study, and their selfless love made the whole process possible.

Part of this dissertation is based upon work supported in part by NSF CNS-1553035, NSF CNS-1422215, NSF IUSE-1525601, and Samsung GRO 2015 awards. Any opinions, findings, and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsors.

Dedication

This dissertation is dedicated to my family.

Chapter 1 | Introduction

During the Information Age, the Web is growing rapidly with large quantities of contents being generated and changing. Since an increasing number of people interact with each other and share content on the Internet, the Web is highly dynamic and constantly evolving over time. Consequently, temporal dynamics on the web include users' behavioral dynamics and content dynamics. Users' behavioral dynamics include users' purchasing histories on e-commercial sites and users' posts on social networks, while content dynamics involve contents' popularity over the Internet and event evolution.

Characterizing, understanding, and even predicting temporal dynamics contribute to developing timely strategies and have a wide range of applications. For example, users' temporal behaviors could reflect users' dynamic relationships with platforms, which is quite valuable information for platforms to improve user retention. Knowing users' behavioral changes enables platforms to provide better personalized services, like, adjusting recommendation strategies at times [2]. As an example of content dynamics, content popularity has long been considered as a major research area. Understanding how content propagates over social media and even predicting content popularity brings benefits in various domains, such as news recommendation [3], rumor identification [4], and event tracking [5].

Thus, this dissertation is focused on mining temporal dynamics of web content, understanding temporal evolution on the web, and predicting how temporal data changes over time.

1.1 Motivation

Plenty of existing efforts have been devoted to mining web data. For instance, researchers found interesting patterns in users' browsing behaviors [6], query histories [7], and activity logs [8]. Similar studies were conducted to characterize content dynamics, such as structural changes of web pages [9] and content propagation over social media [10]. Besides characterizing and understanding web dynamics, in order to leverage web dynamic patterns and bring prophetic guidance to the society, research focuses has gradually moved to predict web dynamics, for example, predicting users purchasing intention [8] and content popularity [11].

Nevertheless, there is still much work to be done in uncovering the reasons behind web dynamics. Only by examining the changes of web data over time can we figure out more and deeper factors impacting the changes. For example, previous studies in crowdfunding platforms have shown different funding strategies or behaviors between occasional and frequent backers, e.g., frequent backers are found to be more likely to invest fast-growing projects [12]. However, none of the previous crowdfunding studies has investigated the temporal dynamics of users' funding behaviors, indicating users' behavioral changes over time. To fill up the gap, we investigate users behavioral dynamic patterns on crowdfunding platforms in Chapter 2. As another illustration, although news popularity prediction gains lots of attention in both industry and academia [3, 13, 14], few of them investigate the long-term popularity evolution of news articles. Thus, we take the first step to explore news articles' long-term popularity in Chapter 3.

Along with the rapid development of the world, an increasing number of news articles are published daily, describing various changing events. To prevent readers from getting lost in information floods, automatic news timelines are emerging to both release journalists' burden and help the audience to understand event evolution. However, existing works only focus on the quality of generated timelines and ignore the generation speed. Thus, we propose to develop a fast and effective news timeline summarization approach and build a real-time news timeline summarization system on an industrial-level news corpus in Chapter 4. In addition, a similar scalability issue is observed in predicting graph evolution, where modeling sequential modeling of individual node embedding changes is very expensive due to the temporal constraint in training. Thus, in Chapter 5, we develop a temporal space translation

based graph embedding model that is scalable to industrial-level graph data.

1.2 Characterization of Temporal Patterns

Temporal patterns encode the evolution logic of time series data. Identifying temporal patterns is the first step to learn how web content evolves over time while characterizing temporal patterns contributes to uncovering possible factors that impact the temporal changes. For example, user behaviors may vary with time, making it difficult for web platforms to provide accurate services with unchanged strategies. As users' satisfaction with platforms is more likely to be gradually changing, knowing users' temporal behavioral changes enables platforms to develop timely user retention strategies. Thus, we propose to analyze a large amount of backer data from two of the largest crowdfunding platforms and discover four distinct temporal backing patterns on both platforms. Besides user behavioral data, web content popularity is another important temporal data. Figuring out how web content popularity evolving with time signals the interest change of the audience and can help the information providers produce more relevant content and better deliver their messages. Therefore, we investigate news popularity patterns. In particular, with an observation that there are a small fraction of news articles that are read across years though the majority of news articles are only viewed for days or weeks, we propose to examine news articles' long-term popularity and unearth several distinctive characteristics of these *evergreen* articles.

1.3 Understanding Event Evolution

As discussed, identifying temporal patterns help web platforms or information providers better understand users and deliver improved services in a timely manner. Accordingly, characterizing temporal patterns and uncovering their correlation with user behavior and content popularity mainly benefit users in an indirect way. To directly support users in understanding content evolution, we focus on the task of *news timeline summarization*, which summarizes important news over major events across the timeline, and produces both concise and complete daily summaries in chronological order. News timelines make it easy for the audience to gain key insights and understand the evolution of news events. News events, however, can

span multiple years and involve thousands of news articles, introducing difficulties for journalists to manually generate news timelines. Although many automatic news timeline summarization algorithms are emerging in recent years, generating timelines from a large number of news articles not only has quality issues but also encounters the challenge of generation speed, which all existing works ignored. To fill up the gap, we propose a fast and effective news timeline summarization approach, allowing us to deliver more high-quality news timelines to the audience.

1.4 Prediction of Graph Evolution

Going beyond characterizing and understanding temporal evolution, we target at predicting temporal data change. More specifically, we focus on the temporal graph embedding. Because of the evolving nature of graphs, graphs are generated by a series of temporal interactions between nodes, where users' interest may vary with time while items' semantic meaning could also drift over time. Therefore, it is necessary to encode graph evolution patterns in node embedding. Due to the time constraint that requires to process edges in chronological order, modeling sequential interactions between nodes can be expensive, making it difficult to learn embeddings on large graphs. Thus, in this work, we propose a novel temporal graph embedding framework to efficiently learn node embeddings for large graphs. Specifically, we adopt a temporal translation-based model to simultaneously embed different graph snapshots and encode the topological structure evolution of graphs by aligning temporal translation spaces over time. For one thing, simultaneously embedding different graph snapshots makes our approach scalable in learning embeddings for large temporal graphs. For another, by learning the changes in the temporal space, our approach allows to predict future embeddings rather than directly using the embedding of the final state, leading to better performances in prediction.

1.5 Overview of this Dissertation

Figure 1.1 gives the thesis structure and the remaining of this thesis is organized as follows: we introduce two case studies on characterizing users' behavioral dynamics and content dynamics in Chapter 2 and Chapter 3 respectively. Towards a better understanding of web dynamics, we propose a novel fast and effective news timeline

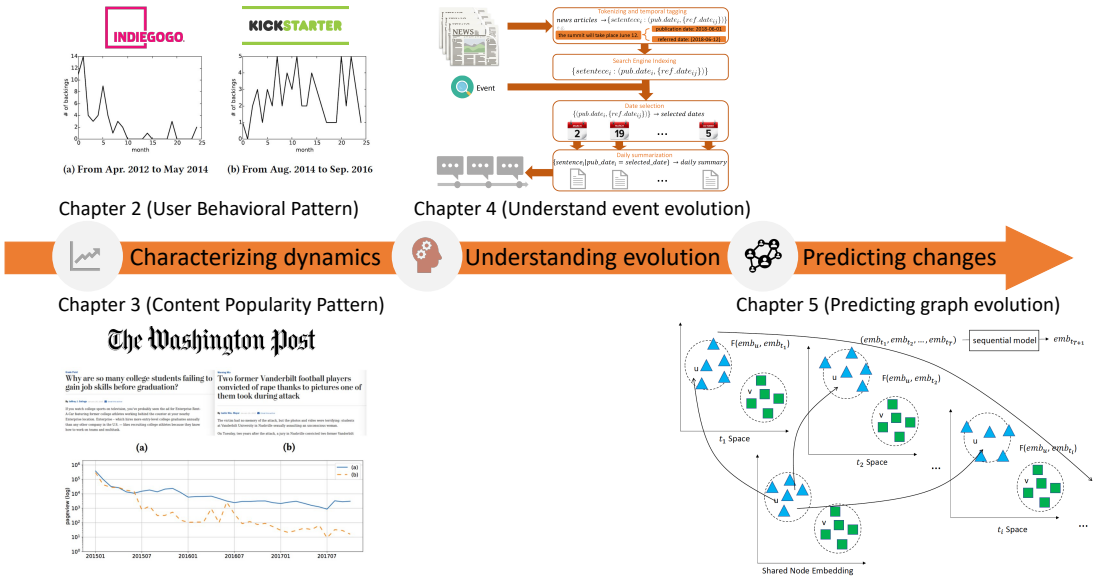


Figure 1.1: Thesis structure

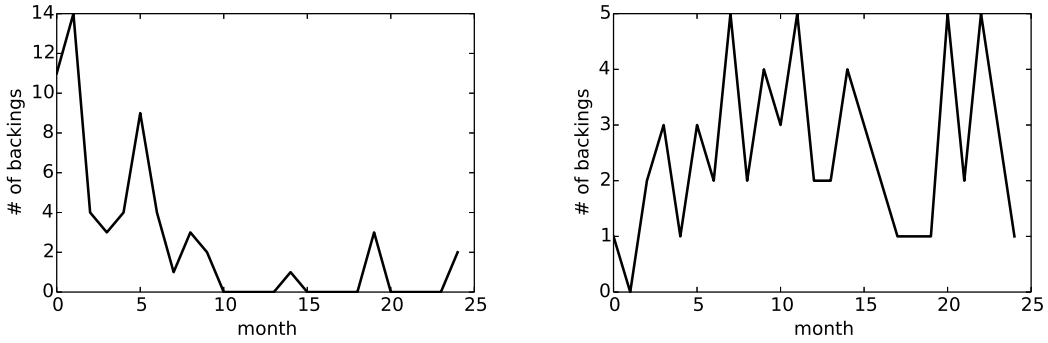
summarization model in Chapter 4. In Chapter 5, we go beyond learning temporal patterns for prediction and present our work in developing a scalable dynamic graph embedding model. We conclude our work in Chapter 6.

Chapter 2 | Characterizing User Behavioral Pattern

This section is corresponding to our work "Understanding Temporal Backing Patterns in Online Crowdfunding Communities" [15].

2.1 Introduction

Online crowdfunding platforms such as Kickstarter and Indiegogo have opened up a new avenue for entrepreneurs to raise funding, overcoming the barriers of small start-up companies [16]. In such platforms, users, known as *backers*, are given the opportunity to pledge funds to join entrepreneurs, known as *creators*, to bring their favorite projects into life. Taking Oculus Rift for example, the virtual-reality



(a) From Apr. 2012 to May 2014 (b) From Aug. 2014 to Sep. 2016

Figure 2.1: Kickstarter user example

gaming headset had reportedly raised $2.4 million dollars via crowdfunding, which clearly displays the power of crowdfunding [17]. According to Kickstarter's report[1], around March 2017, 12 million backers have pledged approximately $2.9 billion dollars for more than 120k projects. Among these 12 million backers, however, only less than 4 million of backers have backed two or more projects. Similarly, less than 30% of user retention rate has been reported on other two crowdfunding platforms, DonorsChoose.org [18] and Indiegogo (Section 3).

As a motivating example, see Figure 2.1, where actual backing numbers of two backers on the Kickstarter platform are plotted. Note that two users show radically different backing behavior for the same duration of 24 months over different years. The user in (a) shows active backings in the early stage but gradually backs less, while the user in (b) more or less remains active throughout. Clearly, identifying such users as (a) early and encourage them to remain active is beneficial to crowdfunding platforms. To increase such backer retention rate, therefore, we argue that monitoring and understanding backers' behaviors is indispensable.

Previous studies have pointed it out that there are distinguished differences between: (1) occasional backers, who back only a small number of projects and mostly join to support their friends' projects, and (2) frequent backers, who invest on many projects and show clear backing interests [12, 19]. In addition, users' first backing behaviors are shown to be closely correlated with user retention and even can be used to predict which backers will return [18]. Nevertheless, none of the previous crowdfunding studies has investigated on the "temporal dynamics" of users' backing behaviors over multiple years. Users' temporal backing behaviors could reflect users' dynamic relationships with platforms, which is quite valuable information for platforms to improve backer retention. For example, if a platform observes a backer gradually becomes less active, platforms may take timely actions, such as special promotion, to restore their interests. Furthermore, knowing backers' temporal behaviors enables platforms to be able to provide better personalized services, for instance, adjusting recommendation strategies at times.

Therefore, in this work, we propose to analyze users' temporal backing behaviors on online crowdfunding platforms. In particular, we intend to investigate the following three research questions:

- **RQ1:** Do users have any temporal backing patterns?

[1] https://www.kickstarter.com/help/stats

- **RQ2:** If so, what are the characteristics of those backing patterns and possible factors impacting those patterns?
- **RQ3:** Can we identify users' backing patterns at early stages?

In answering these research questions, we make the following main contributions:

- To our best knowledge, we are the first to analyze users' temporal backing behaviors on crowdfunding platforms.
- We discover four distinct temporal backing patterns on two most popular crowdfunding platforms with hundreds of thousands frequent backers.
- We analyze the characteristics of each pattern from various aspects, including success rate and pledged money, and explore possible factors impacting users' backing patterns.
- We validate our analysis by showing the possibilities to build early prediction models in inferring all four temporal backing patterns.
- Based on the analysis, we achieve encouraging results in identifying those who will progressively become inactive at very early stages.

The rest of the work is organized as follows. In Section 2.2, we review related works on crowdfunding platforms. Section 2.3 introduces our data collection strategy and the experimental dataset. The approach to uncover users' temporal backing behaviors and its results are given in Section 2.4. In Section 2.5, we analyze characteristics of backing patterns and explore possible factors impacting users' behaviors. Early prediction models are discussed in Section 2.6 and we conclude the work with a discussion on limitations and future work in Section 2.7.

2.2 Related Works

In this section, we review crowdfunding research work in three categories: (i) analysis of crowdfunding platforms, (ii) project success prediction and recommendation, and (iii) crowdfunding backer behaviors.

There are a rich set of studies on analyzing crowdfunding platforms [20–25]. For example, studies have examined the dynamics of crowdfunding projects [26],

revealed motivations of both creators and backers [23], and made comparisons of different crowdfunding platforms [27]. With the rise of social media, researchers noticed the close relationship between crowdfunding platforms and social networks, and started leveraging social media activities to study crowdfunding [12, 19, 28]. For example, social media have been found to be quite helpful in project promotion and strongly correlated with project outcomes [28]. In addition, other factors, such as geography, project updates and rewards, have also been proved to be closely connected with crowdfunding projects [29–31].

Aimed at helping creators get funded and backers find favorable projects, research has focused on using machine learning algorithms to do project success prediction [32–35] and project recommendation [12, 19, 36]. Besides static and dynamic features on crowdfunding platforms, both [33] and [19] leveraged social network features to achieve state-of-the-art performance in project success rate prediction and project recommendation respectively.

Although various studies have been conducted on crowdfunding platforms, few works have investigated on backers' behaviors. Previous studies have shown different backing strategies or behaviors between occasional and frequent backers. For example, frequent backers are found to be more likely to invest fast-growing projects [12]. [16] classified backers into three categories, immediate backers, delayed backers and serial backers. In addition, only based on one's first backing, researchers showed the possibility to predict whether that user will come back or not [18].

In our work, instead of uncovering different backing behaviors between occasional and frequent backers, we intend to focus on frequent backers, who carry out more activities and are more valuable for platforms, and dig into their temporal backing behaviors, which has rarely been explored before.

2.3 Dataset Description

Aimed at understanding user behaviors in online crowdfunding communities, we collect a large amount of user data from two of the most popular crowdfunding platforms, Kickstarter and Indiegogo. To make our analysis more reliable, we have collected all projects that are available and thereby, to our best knowledge, obtain the largest datasets on both platforms (comparison of raw datasets is given in Table 2.1). In this section, we introduce data collection strategy, data cleaning process

and overview of our dataset.

2.3.1 Dataset Collection

Due to the disparate strategies of Indiegogo and Kickstarter for displaying user backing histories, we adopt two different collection methods for these two platforms.

2.3.1.1 Kickstarter Dataset

Kickstarter prevents users from obtaining backer list of certain project by only displaying limited backers' avatars without profile URLs. Fortunately, however, we can find backers' profile URLs in project comments page, and with one's profile URL, we are able to collect the complete project backing history of that user. At the end, we have gathered 233,534 ended projects, spanning from April 2009 to December 2016, and corresponding comments. Then we collected users' full backing histories through user profile URLs from comments, which resulted in more than 500,000 backers.

2.3.1.2 Indiegogo Dataset

Different from Kickstarter, Indiegogo displays projects' backer lists with some anonymous backers but only shows recent backings of each user on their profile URLs. Hence, we directly collected Indiegogo's project backer lists after gathering all available ended 124,292 projects, which spans from January 2008 to January 2017. Identifying users by profile URLs, there are around 2,300,000 unique backers. Although we cannot guarantee to obtain full backing history of each backer and each user's backing history in our dataset ought to be viewed as a sample of the true history, the influence of sampling should be negligible as we have a large number of projects.

2.3.2 Frequent Backers

Despite a large number of unique backers identified on both platforms, the majority of them only back very limited projects. Taking Indiegogo backers for example, only 536,727 backers (23%) back more than twice, which is consistent with the previous study (26%) on another crowdfunding platform [18]. In order to reduce the

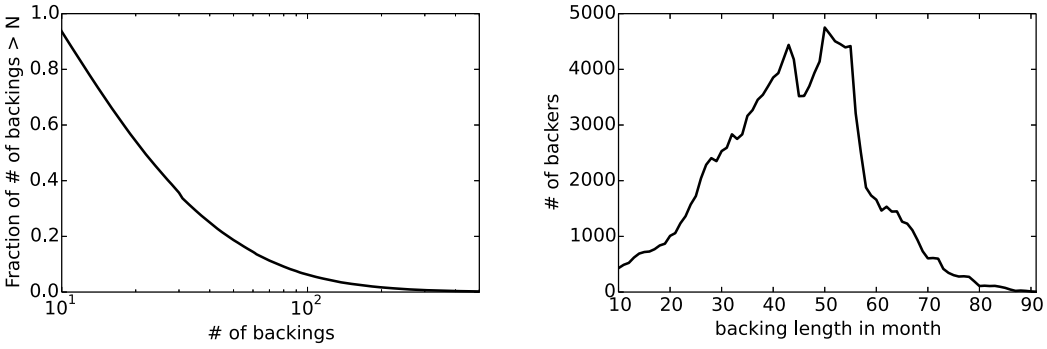
Data source	Platform	# of unique users	# of projects
Ours	Kickstarter	**508,850**	**233,534**
Ours	Indiegogo	**2,314,199**	**124,292**
[33]	Kickstarter	146,721	168,851
[26]	Kickstarter	N/A	59,115
[17]	Kickstarter	239,000	N/A
[25]	Indiegogo	N/A	47,139

Table 2.1: Raw Dataset Comparison

Platform	# of users	# of projects	# of backings per user
Kickstarter	150,122	174,938	38.0
Indiegogo	12,528	41,450	15.4

Table 2.2: Our frequent Backer Dataset

impact of occasional backers and better understand users' backing behaviors, we filter out those infrequent backers and only keep those frequent backers who have backed more than 10 times. As a result, we have 150,122 Kickstarter and 12,528 Indiegogo frequent backers. Table 2.2 shows overall statistics of our frequent backer dataset and Figure 2.2(a) illustrates the complementary cumulative distribution function (CCDF) of Kickstarter users' backing numbers. For each backing number N, CCDF of users' backing numbers shows the proportion of backers who invest more than N projects, for example, there are more than 20% of Kickstarter backers



(a) CCDF of users' backing numbers

(b) Backing Length Distribution

Figure 2.2: Kickstarter Frequent Backer Distribution

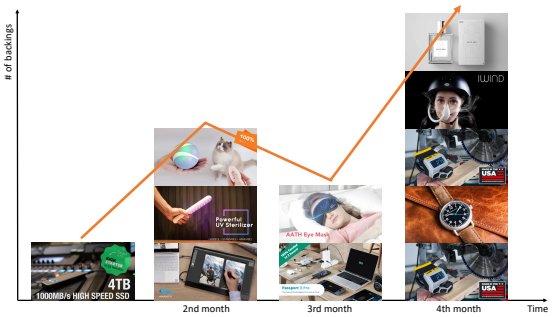


Figure 2.3: Example of backing behavior

supporting at least 40 projects.

In our work, we analyze users' backing behaviors during their lifetime, which is defined as the observed backing history in our dataset. As shown in Figure 2.2(b), the majority of backers have lifetimes lying between 2 and 5 years, which means users' baking histories in our dataset are reasonably long enough for temporal backing behavior analysis.

2.4 Temporal Backing Patterns

In this section, we intend to answer *RQ1: do users have any temporal backing patterns?* We start by introducing our definition of users' temporal backing behaviors, then describe our approach to cluster those temporal behaviors and finally discuss about the clustering results.

2.4.1 Temporal Backing behavior

With the purpose of uncovering users' possible backing strategies, we monitor users' backing numbers over time and define a user's temporal backing behavior as a backing number function of time, $N(t)$. Although the exact timestamp when one user back a certain project is not available in our dataset, we propose to use the middle time between each project's start and end time as its backers' joining time. Since projects' average duration is around 1 month and most of our backers have more than 2 years lifetimes, it should be reasonable to ignore the several days' shifting. Knowing the timestamps of all backings, each user's backing behavior can be viewed as a time series, $N(t) = \{n_1, n_2, ...\}$, which models users' backing

numbers through time. For example, Figure 2.3 shows one possible temporal backing behavior, where the user tends to back an increasing number of projects over time.

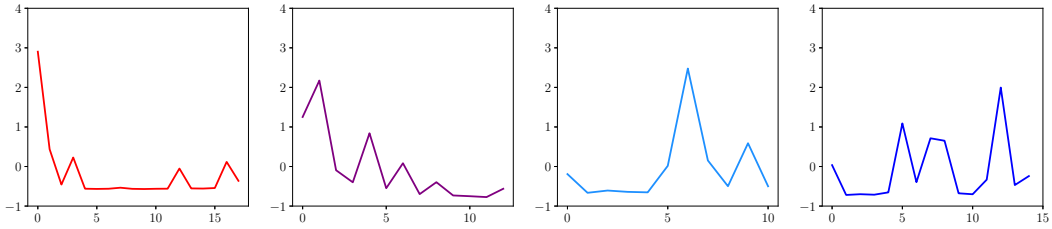
2.4.2 Dynamic Time Warping Clustering

Having converted users' temporal backing behaviors into time series, we investigate whether there exist any backing patterns. Because different backing behaviors correspond to different shapes in time series, we propose to use time series clustering to find distinct patterns. Since only the shape of time series matters, we do Z-normalization on all time series at first to make comparisons between two time series meaningful [37]. Then, in order to allow slight shifting in time and warping in shape, we take advantage of Dynamic Time Warping (DTW) [38] clustering to find discriminative patterns.

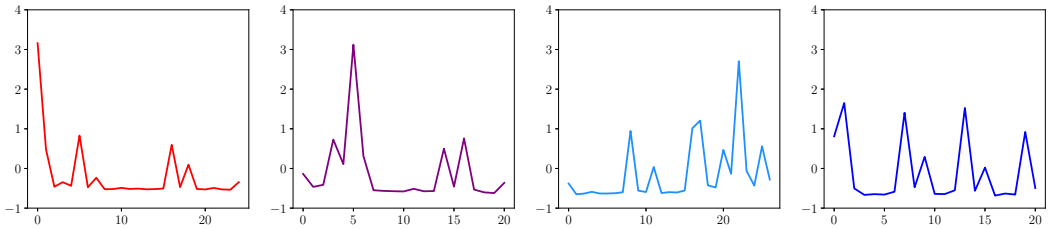
More specifically, we adopt Dynamic Time Warping with Dtw Barycenter Averaging (DBA) [39], a state of the art time series averaging method, to do the clustering. As for parameter settings, we try various time window sizes and cluster numbers, and run DBA-based DTW 100 times with different kmeans++ initialization [40] for each set of parameters and keep the clusters with minimum inertia. Finally, we find the most discriminative patterns when DTW window size is $\frac{1}{3}$ of time series length and the cluster number is 4.

2.4.2.1 Effect of backing length

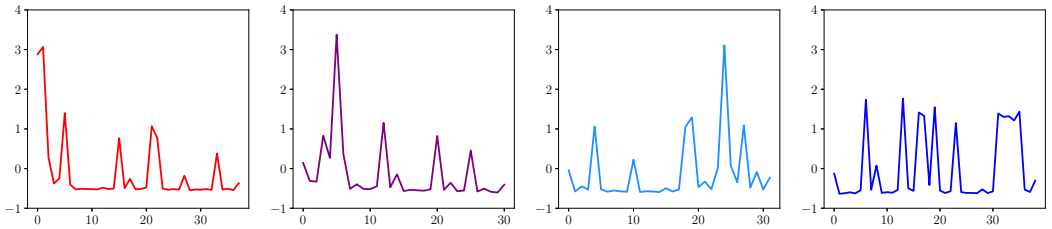
Due to the fact that backing history length varies with users, it is difficult to compare two time series that differ too much in length (e.g. one user with 2-year backing history while another with 5-year backing history). Thus, we intend to study the effect of users' backing length by running clustering on different backing length groups and the results are shown in Figure 2.4. Apparently, similar backing patterns are discovered in all groups, and there is no distinct pattern that is exclusively found in some groups. As a result, it should be reasonable to normalize each user's backing history into the same length (*Binning*): 1) we regard each user's first and last project backing time as one's lifetime start and end time respectively; 2) evenly split one's lifetime into 30 intervals and count the number of backings in each interval; 3) apply Z-normalization on the time series.



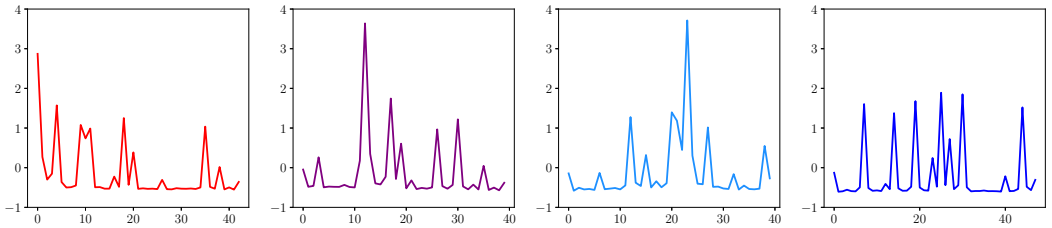
(a) 10-20 months



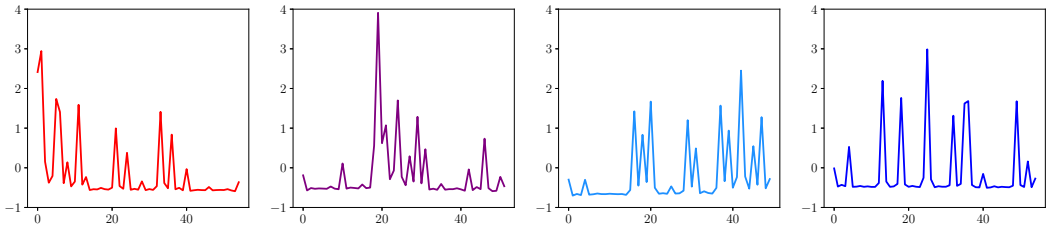
(b) 20-30 months



(c) 30-40 months

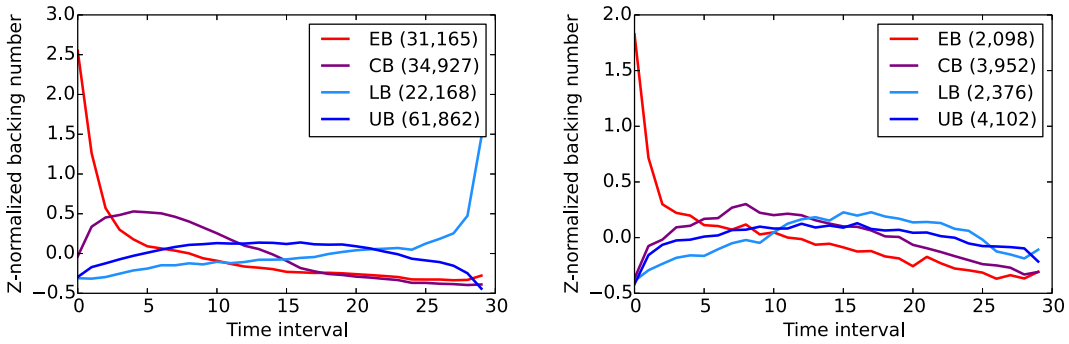


(d) 40-50 months



(e) 50-60 months

Figure 2.4: Patterns in various backing length groups. X-axis stands for months. Y-axis stands for Z-normalized backing number.



(a) Kickstarter (b) Indiegogo

Figure 2.5: Clustering results (with # of backers in parentheses)

2.4.3 Clustering Results

After binning and clustering, we do pairwise averaging on the 30-interval time series per group to obtain smoothed clusters. As shown in Figure 2.5, similar backing patterns have emerged on both crowdfunding platforms, which are also consistent with clustering results on different backing length groups in Figure 2.4. We name the 4 distinct backing patterns as follows:

1. **Early backer (EB)**: those backers back a lot at the very beginning of their lifetime but gradually lose interest and seldom back later.

2. **Cautious backer (CB)**: although the majority of their backings still happen at the beginning, they are more cautious than early backers, as they try a few projects before conducting massive investments.

3. **Late backer (LB)**: contrary to early backers, they back limited projects at first, but gradually back more.

4. **Uniform backer (UB)**: they actively back all the time with their backings almost evenly distributed through their lifetime.

Two matters should be noted here: 1) Kickstarter clusters look much more smooth than Indiegogo clusters; 2) there is a slight difference between LB on both platforms, i.e., LB on Kickstarter shows a more sharp increase at later months than LB on Indiegogo does. The first issue results from the fact that Indiegogo dataset is

smaller (with respect to # of backers) and sparser (with respect to # of backings per user) than Kickstarter dataset. As for the second issue, it may be because our Indiegogo dataset does not guarantee containing users' full backing histories and lacks some projects. However, we can still see an obvious increase in the backing pattern of LB on Indiegogo, which is matched with Kickstarter's.

2.5 Understanding Backing Patterns

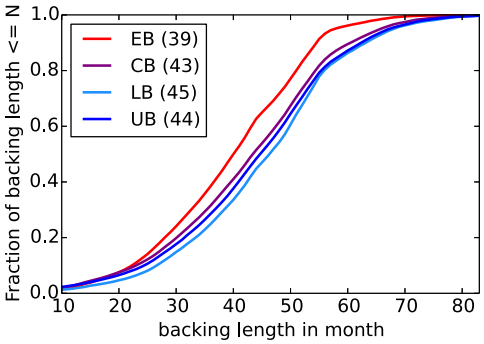
Figure 2.5 indicates an interesting phenomenon that both EB and CB invest a lot at the beginning, but gradually lose interest and become less active, while UB is active in backing all the time and LB even supports an increasing number of projects over time. As EB and CB account for a large proportion (more than 40%) of frequent backers, figuring out why they have become inactive will not only help those backers out of the possible problems they are trapped in, but also contribute to the development of crowdfunding platforms. Therefore, we intend to do a quantitative analysis on the characteristics of these 4 groups of backers and propose some empirical answers to *RQ2: what are the characteristics of those backing patterns and possible factors impacting those patterns?* Note that here, we only show results on Kickstarter dataset but similar results are found on Indiegogo dataset as well, except for the success rate[2].

This section starts with analyzing characteristics of backing patterns, and based on these characteristics, we propose some possible factors explaining users' behaviors or backing patterns.

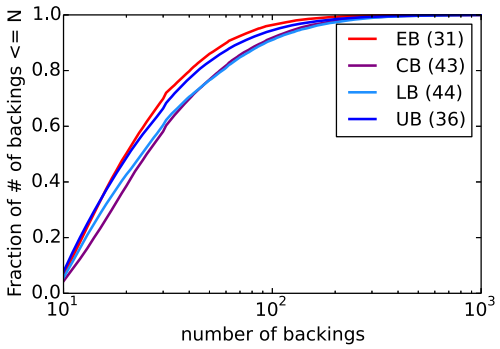
2.5.1 Characteristics of Patterns

When analyzing characteristics of backing patterns, cumulative distribution functions (CDF) are frequently adopted for displaying the distribution differences among those four groups in various aspects. For a certain value X, CDFs show the fraction of backers who have values less than or equal to X. To quantitatively measure the differences between distributions of two backing patterns' samples, we conduct

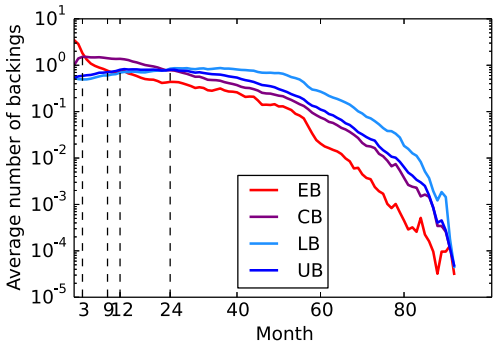
[2]Besides *All-or-Nothing*, Indiegogo gives one more choice, *Take-it-All*, to creators, which means backers are still able to get their rewards even if the goal amount is not reached. With more than 90% of Indiegogo projects starting with flexible funding, we cannot determine whether projects are successful or not on Indiegogo dataset.



(a) CDF of backing history length



(b) CDF of total backing number



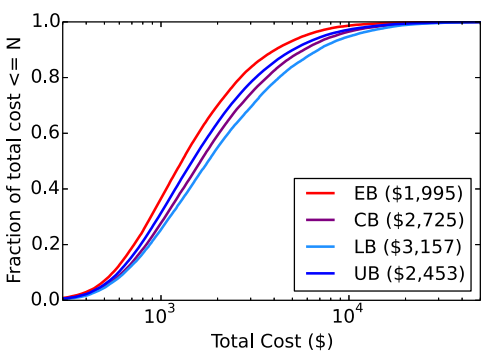
(c) Average backings per month

Figure 2.6: Backing length and backing number of each pattern (with average value in parentheses)

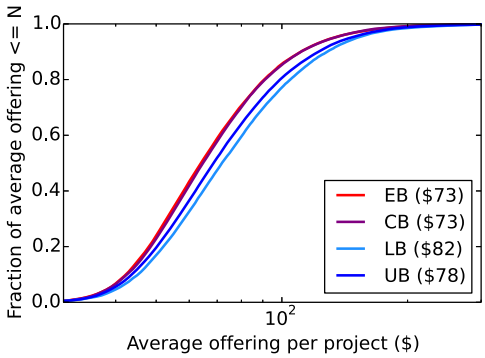
Kolmogorov-Smirnov test [41], a wildly used significance test for checking whether two samples are drawn from the same distribution. Accordingly, for any pair of backing patterns in any situation, KS tests reject their samples being drawn from the same distribution with p-value less than 0.01.

2.5.1.1 Backing length and backing number

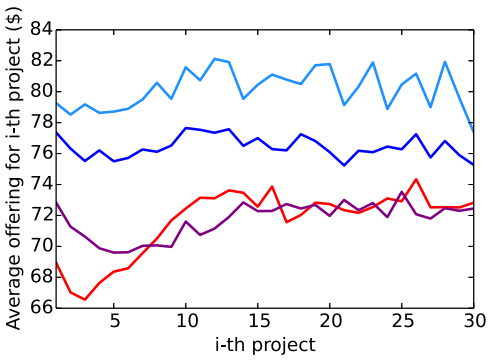
Figure 2.6(a) presents the CDF of backers' lifetime backing length. As expected, EB has obvious shorter lifetime backing length than other three groups, both in general and on average, in that it gradually loses interest in backing from the beginning. Despite the fact that CB has similar backing length distribution as UB, its backing number is apparently larger than UB's as indicated in Figure 2.6(b), which suggests that CB is as valuable as UB. CB could even contribute more to the



(a) CDF of total cost



(b) CDF of average offering



(c) Average offering for each backer's i-th project

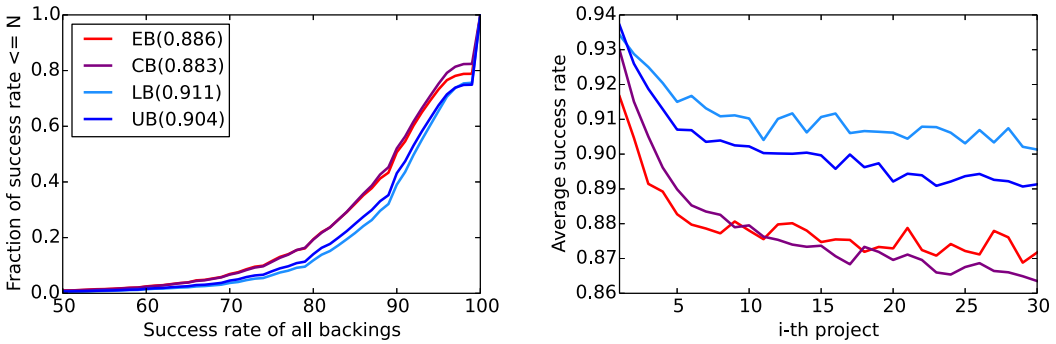
Figure 2.7: Backing cost (with average value in parentheses). Cost refers to the money spent on successful projects, while offering just indicates the prices that users are willing to offer for the projects, which may or may not be successful.

platforms if it remained active during the second half backing history. In addition, we examine the change of each group's backing number over time and find some inflection points. As shown in Figure 2.6(c)[3], on average, EB is actively backing during the first 12 months, while the passion of CB lasts 24 months, two times longer than EB's, after cautiously investing 3 months at the beginning.

2.5.1.2 Project Cost

Besides participation duration and backing numbers, the amount of pledged funds is another important measurement for the contribution of users to the platforms. We

[3]Note that it is the overall trend that matters not the absolute values, which are relatively small due to the data sparsity (not all users invest projects in every month).



(a) CDF of success rate

(b) Average success rate for each backer's i-th project

Figure 2.8: Success rate (with average value in parentheses)

compare different groups' spending behaviors and make several observations. First, as shown in Figure 2.7(a), on average, EB spends the least, whereas LB contributes the most. Considering the proportion of each group on the platform, UB, even if ranked 3rd on average spending, invests 151 million dollars in total, which is almost 1.5 times more than the second CB does. Because of the small number of LB, it only pledges 62 million dollars, which is slightly more than EB does. The fact that EB and CB pledge around 30% of funds among frequent backers implies that they are indispensable for the platforms. Second, Figure 2.7(b) demonstrates that EB and CB prefer offering smaller pledging fund per project, while LB and UB are willing to spend a little bit more on each project. Interestingly, both Figures 2.7(b) and 2.7(c) suggest EB and CB have similar average offering per project. Last but not least, Figure 2.7(c) shows that EB offers less money for each project when starting investments but gradually increases its offerings. We can see a similar pattern in CB's average offerings over projects as well. Taking together with their temporal backing patterns, this phenomenon may occur due to their limited budget. At the beginning, in order to support multiple projects, EB and CB have to lower down their offering per project, but with their backing numbers gradually decreasing, they are able to afford more offering per project.

2.5.1.3 Success Rate

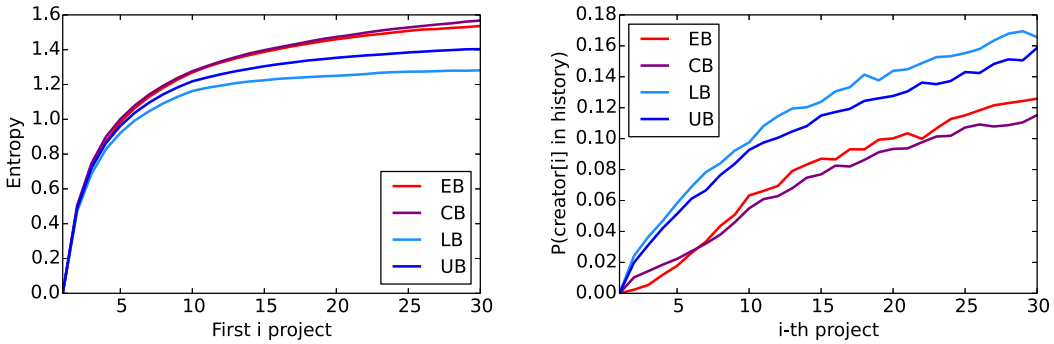
Next, we investigate on the success rate of each backing group. Surprisingly, 22.4% of the 150K frequent backers (with at least 10 backings) in Kickstarter dataset have never failed in backing, even though their average backing number is 18. As for each backing group, 25.1% of UB, 24.4% of LB, 21.2% of EB and 17.6% of CB achieve the perfect backing histories. Although we analyze frequent backers, who have up to 90.5% success rate on average, there are still some differences among these four groups. Figure 2.8(a) shows that LB and UB are better at backing and have higher success rate than other two groups, which is consistent with the ratios of perfect backers in each group to some extent. Moreover, looking at each group's average success rate at i-th project from Figure 2.8(b), we can see a clear gap between success rates of EB and CB and those of LB and UB. In addition, we note the slopes of CB and EB's curves are relatively smaller, indicating a faster drop in their success rates. Considering temporal backing patterns of EB, the low success rate may result from the fact that EB invests very frequently at the beginning without caring much about its backings' outcomes, while other three groups, especially LB, gain more experience before conducting massive investments and certainly reach much higher success rate in backings.

2.5.1.4 Categories and Creators

Previous studies have shown that topical preference and connections with creator play an important role in users' backing behaviors [19, 23]. Category entropy is applied to measure users' topical preference and defined as follows:

$$CatEntropy(u) = -\sum_{i=1}^{C} \frac{n_i}{N} log_2 \frac{n_i}{N} \tag{2.1}$$

In the equation, C is the number of categories, N is the total number of projects backed by user u and n_i is the number of backed projects under category i. Compared with uniformly backing all 15 project categories on Kickstarter, where category entropy is around 4, Figure 2.9(a) indicates that all four groups have strong preferences in backing categories, which is consistent with previous studies. We also notice that EB and CB have larger entropies than other two groups in general, which means they have broader interests in backing projects. As expected,



(a) Average entropy of each backer's first i projects

(b) Probability of each backer's i-th project being created by his/her past supported creator

Figure 2.9: Categories and creators

Figure 2.9(b) shows that there are big chances that frequent backers will support creators whom they have supported before. In addition, the probability of LB and UB funding a past supported creator is at least 40% higher than that of EB and CB during the first 20 backings. Although EB and CB's lower probability at the first a few backings may result from their more frequent backings in a short time at the beginning, they are still less likely to fund a past supported creator than LB and UB after a longer time, say after 20 backings.

2.5.2 Causes of Patterns

Having observed various characteristics of users' temporal backing patterns, we seek potential factors impacting users' behaviors. Since users' backing behaviors can rely on diverse factors, including personal backing strategies, we just provide some general empirical analysis here and leave user studies for future investigation. Specifically, we focus on the impact of project outcomes and creators in this section.

2.5.2.1 Project Success

In Section 2.5.1.3, we have identified the huge differences among each group's backing success rate that success rates of EB and CB are not only much lower than those of LB and UB, but also drop faster than other two groups at the early stage. Considering EB and CB gradually lose interests in backing from the beginning, the relatively low success rate should play an important role in their later inactivity.

Thus we argue that EB and CB may be discouraged by experiencing relatively low backing success rates and failing several projects initially, and thereby progressively become less active in backing. Based on this finding, in order to encourage users to be active, platforms may recommend some easily successful projects to those who have a bad start initially in backing.

2.5.2.2 Connections with Creators

Another interesting observation comes from Section 2.5.1.4. Note that both LB and UB are more likely to fund their past supported creators, indicating stronger connections between them and their creators. During the first 20 backings, after which EB and CB gradually become inactive, LB and UB show at least 40% higher probability to support a recognized creator. Because of LB and UB's willingness to return to fund past supported creators, there could be some long-term connections being established between them and their creators. Accordingly, favorable relationship between users and creators may increase the probabilities of users being a returned backer, and encourage users to continue backing on the platforms. Hence, both platforms and creators should pay more attention to building up mutual trust between backers and creators.

2.6 Early prediction

In addition to the differences in the dynamics of backing numbers, we have shown clear distinctions among these 4 groups in various aspects, including project cost and success rate, in the previous section. To validate the insights from the previous section, we build up models on users' early backing features to see whether we are able to predict users' later backing patterns. Apparently, the earlier we can detect the loss of users' interests, the better we may help them stay active by taking actions such as promotion. Therefore, we turn to *RQ3: can we identify users' backing patterns at early stages?* and investigate on building early prediction models in this section.

at i-th finished project	1	2	3	4	5	6
Kickstarter	18	104	166	220	271	320
Indiegogo	26	162	252	327	396	465

Table 2.3: Duration in days at i-th finished project

2.6.1 Set-up

To validate our analysis, we build 4-class classifiers to predict users' backing patterns at first. Then, based on the observation in section 2.5.1.2 that EB and CB pledge around 30% of funds among frequent backers and had the potential to contribute even more if they stayed active, we propose to build binary classifiers to distinguish EB and CB from UB and LB as well.

2.6.1.1 Prediction Time

Section 2.5.2.1 points out the influence of project outcomes on users' future behaviors. As such, it should be reasonable to make inferences after observing a few project outcomes. Table 2.3 displays Kickstarter and Indiegogo users' duration in days[4] when their i-th project is finished. As shown in Figure 2.6(c), since 9th month, EB has already been less active than all other three groups, which might be too late for user retention. Consequently, we propose to predict users' backing patterns at the end of their 1st to 4th project on Kickstarter and correspondingly, at the end of 1st to 3rd project on Indiegogo.

2.6.1.2 Features

After setting up the prediction time, we extract the following features from users' behavior data during the time period to build classifiers. Note that, when clustering users' backing patterns, we only used Z-normalized time series of users' backing numbers in their entire lifetimes. However, here we only utilize users' first a few months data to extract features.

1. **Temporal (4 features)**: As indicated in Figure 2.6(c), even when just joining the platforms, different groups of users behave differently in backing.

[4]By the end of i-th project, how long have that user been on the platform?

Therefore, dynamics of users' backing numbers should be valuable features. In addition, we include the number of backed projects[5], duration in days, mean and standard deviation of the time difference between two adjacent backings.

2. **Backer (3 features)**: In sections 2.5.1.2 and 2.5.1.3, we observe that averaging offering per project and success rate vary with different groups in the first a few backings. Therefore, we formulate backer features, including backing success rate, mean and standard deviation of backed money per project.

3. **Creator (4 features)**: Sections 2.5.1.4 and 2.5.2.2 discuss about the potential encouragement of favorable connections between backers and creators. Thus, based on the assumption that experienced creators may be good at establishing favorable relationship with backers, we also extract features from creators, involving their past backing numbers, backing success rate, creating numbers, and creating success rate.

2.6.1.3 Criteria

Because of the imbalance of our data, we evaluate the multi-class classification results using two popular measures–i.e., macro-F1 score and G-means, to be defined as follows:

$$Macro\text{-}F1 = \frac{1}{N_c} * \sum_{i=1}^{N_c} F1_i \tag{2.2}$$

$$G\text{-}means = \sqrt[N_c]{\prod_{i=1}^{N_c} recall_i} \tag{2.3}$$

with

$$F1_i = \frac{2 * precision_i * recall_i}{precision_i + recall_i}$$

$$precision_i = \frac{\sum TP_i}{\sum TP_i + \sum FP_i}$$

$$recall_i = \frac{\sum TP_i}{\sum TP_i + \sum FN_i}$$

[5]At the end of i-th project, users may back more than i projects, though their outcomes are not available then.

at i-th finished project	1	2	3	4
Random Baseline	0.50	0.50	0.50	0.50
Kickstarter	0.71	0.78	0.81	0.82
Indiegogo	0.63	0.70	0.73	0.75

Table 2.4: Binary classification results (ROC AUC)

In the equations, N_c stands for the number of classes, TP for True Positive, FP for False Positive, and FN for False Negative. For 4-class classification, random baselines will score 0.25 on both metrics.

As for binary classification, Area Under the ROC Curve (AUC) is employed to measure the possibility that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one. Thus, a random baseline will have 0.50 of AUC score.

To handle our imbalanced data, via preliminary testing, we chose the AdaBoost algorithm [42] with a decision tree classifier and under-sampling [43], which has given the best performance on 10-fold cross validation.

2.6.2 Results

2.6.2.1 Predicting backing patterns

The results are given in Figure 2.10. Generally, based on our proposed features, we can achieve promising performance on inferring users' future backing patterns when they just complete first 3 to 4 projects. As expected, backer and creator features contribute to the prediction, especially at the end of their first or second project, which validates our analysis in the previous section. With more user backings being observed, although overall performance is improved significantly, temporal features play a more essential role in the prediction and other two features only lead to smaller increments. Consistent with the fact that success rate is indeterminable on Indiegogo dataset as mentioned in section 2.5, backer and creator features show very limited improvement in the prediction.

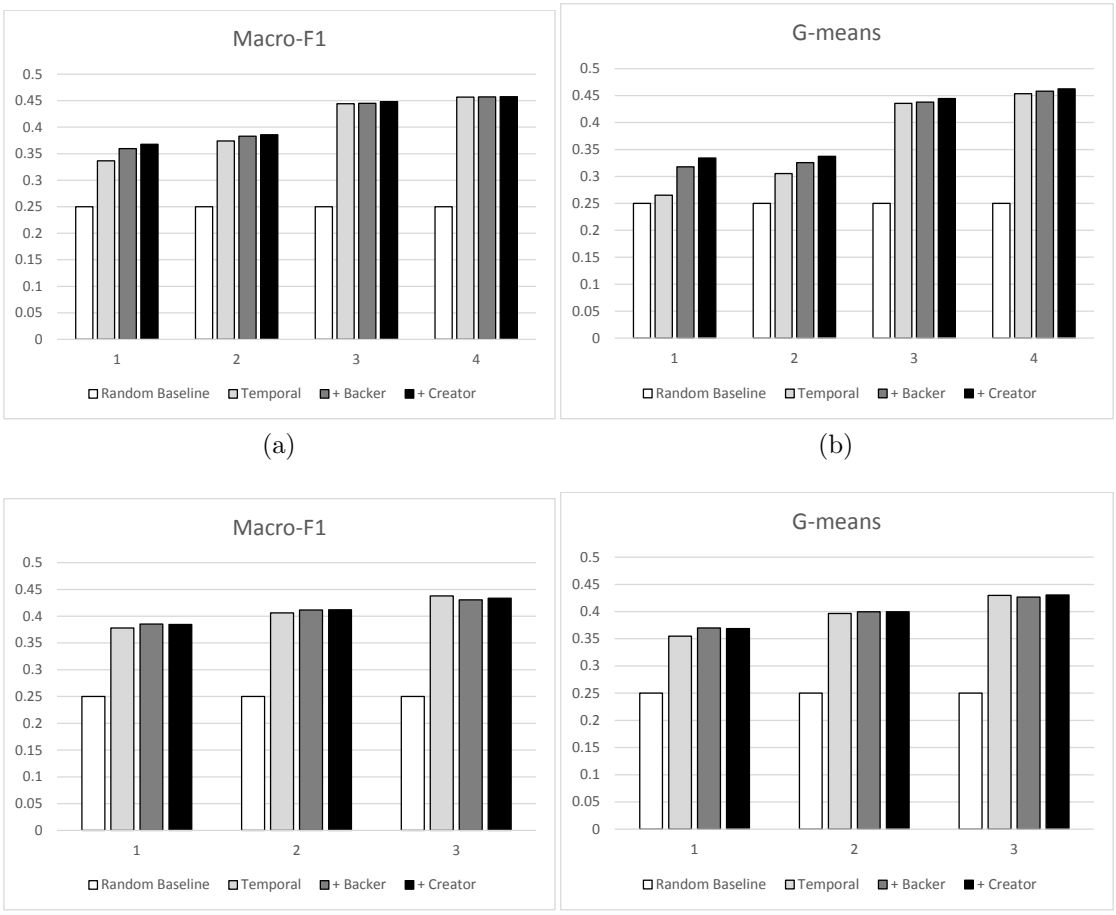


Figure 2.10: Predictive performance for inferring users' future behavior patterns at the end of first few projects. (a) and (b) show results on Kickstarter dataset, while (c) and (d) display Indiegogo's.

2.6.2.2 Identifying early backers and cautious backers

Having validated our analysis and corresponding features, we build a binary classifier to identify EB and CB from all frequent backers for user retention. As shown in Table 2.4, at the end of each user's 1st project, we are able to tell whether that user will be EB and CB with 0.71 AUC on Kickstarter and 0.63 AUC on Indiegogo. When one's finished project number reaches 4, AUC increases to 0.82 and 0.75 respectively. Additionally, when joining the platforms for just 5 months, which corresponds to the time of 3rd project finished on Kickstarter and second project finished on Indiegogo, EB and CB can be identified very well by the built classifier,

with 0.81 AUC and 0.70 AUC, correspondingly.

2.6.2.3 Discussion

Although 4-class classification models do not achieve superior performance in backing pattern inference, we show the possibility of building predictive models, which confirms the relationship between users' early activities and their future behaviors. Based on the assumption that platforms may do some user retention to help potential EB and CB stay active, we formulate the identification of EB and CB as a binary classification problem and obtain very promising results. At the end of users' 3rd project, we are able to identify EB and CB with 0.81 AUC on Kickstarter and 0.73 AUC on Indiegogo. Certainly, there is still much room for improvement in EB and CB identification. For example, we can incorporate more user behavioral features, such as users' view and clicking histories. We leave this for future study. Throughout these preliminary studies, we emphasize that there are four distinct backing patterns, which are proved to be strongly related with users' very early behaviors, and show encouraging results in identifying those who tend to progressively become inactive at early stages.

2.7 Limitations

Despite encouraging results, our work is not without limitations. First, as usual for data-driven study using social media data, our study is based on a small fraction of real-life datasets sampled from two crowdfunding platforms. Due to the limitation of programming APIs, we had to adopt two different data collection strategies that may have introduced two types of sampling bias: (1) For Kickstarter dataset, as we obtained users' profile URLs from projects' comments, backing histories of those backers who seldom leave comments are missing. However, we hypothesized that frequent backers care more about their backings than occasional backers and therefore should be more active in leaving comments. (2) On Indiegogo, in each project's backer list, there are lots of anonymous users that we could not identify. Therefore, our dataset might be from those users who are less concerned about their privacy. In addition, we could not obtain users' full backing histories, which may affect some users' temporal backing patterns and make them act as noises in our dataset. As to these potential issues in our datasets, we attempted to mitigate

the effect by obtaining relatively large-scale datasets and verified that there were consistent results on both platforms. Nevertheless, to be able to generalize our findings further, we plan to collect less-biased datasets and repeat our study on other types of crowdfunding platforms (e.g., Change.org) as well.

Second, in this work, we mainly focused on the relationship between users' temporal backing behaviors and project features. However, it is worth noting that user features, including users' demographics and logging histories, could be equally informative. Therefore, we intend to explore connections between users' temporal backing behaviors and user features.

Third, although our investigation was based on quantitative analysis, some unforeseen factors affecting users' backing behaviors may not be reflected by the collected data. For example, users' backing behaviors might be influenced by their friends'. Accordingly, conducting user studies can be a good complement to fully understand users' temporal behaviors. Such a study will not only validate our findings in this work but also provide potential causes of backers' actions.

2.8 Summary

Having a better understanding of the backers' dynamic and temporal behaviors allows crowdfunding platforms to provide better services and improve customer retention. In spite of such an importance, however, prior studies lack the exploration of the temporal dynamics of backer behaviors. As such, this work aimed to take a step to investigate on backers' temporal behaviors.

To analyze backers' temporal behaviors, we have collected large-scale datasets from two of the most popular crowdfunding platforms, Kickstarter and Indiegogo. Employing time series clustering methods, we discovered four distinct temporal backing patterns on both platforms (**RQ1**), including Early Backer (EB), Cautious Backer (CB), Uniform Backer (UB), and Late Backer (LB). Among these patterns, both EB and CB show decreasing interests in backing, whereas UB constantly invests on projects and LB even progressively supports an increasing number of projects. Driven by the research question why backers have such temporal backing patterns (**RQ2**), we conducted a quantitative analysis, explored the characteristics of these patterns in various aspects, and proposed possible factors affecting users' backing behaviors.

In our findings, we noted that the outcomes of backers' first a few projects are closely correlated with their future backing behaviors. For instance, initial low success rate may discourage users from investing further. In addition, we found that the connections with project creators could be an another factor impacting users' backing behaviors–that is, favorable relationship between backers and creators seems to encourage backers to keep supporting the same creators and/or invest on more projects. In order to validate our analysis, we extracted related features from users' first few backings, and showed promising results in early predicting all four temporal backing patterns (**RQ3**). Moreover, aimed at helping platforms with respect to user retention, we proposed to build binary classification models to identify two types of backers–EB and CB–at a very early stage and achieved encouraging performance.

Chapter 3 | Characterizing News Articles' Long-term Popularity Pattern

This section is based on our collaboration with the Washington Post and the work is corresponding to our work "Characterization and Early Detection of Evergreen News Articles" [44].

3.1 Introduction

Articles that consistently gain traffic over time, named as ***evergreen*** articles, are important to newsrooms because they signal a topic of significant importance to readers. News sites want to continue to serve these articles to new readers over time, whether that is through re-promoting on their main social media channels, linking frequently in news articles, or optimizing them for search engines so they continue to surface for readers' search queries. Evergreen articles can also provide authoritative, reliable information during repeating major events. For example, an article about the history of solar eclipses can be used for all such events, or an article about how to treat a common cold is good for months out of every year. From a traffic perspective, evergreen articles provide an important baseline that news sites can rely on for low traffic days and serve as a good source for recirculation traffic.

Journalists at one of top 10 US daily newspapers, the Washington Post (denoted as WaPo), whom we interviewed, currently rely on their memory for keeping track of evergreen news articles. Some newsroom editors in WaPo manually maintain a short list of evergreen articles and use them for right occasions. Many good evergreen articles never have a chance to be served because they are not remembered

by journalists and editors. Journalists also try to identify evergreen articles by checking articles' traffic history by selecting potential candidates and manually reviewing them. While the traffic performance pattern of evergreen articles needs to be thoroughly understood based on a time series analysis, at present, journalists do not have a good or agreed-upon definition of evergreen articles in terms of time series of traffic performance patterns. They do not have easy-to-use query tools to explore time series data. Furthermore, even after identifying potential candidate evergreen articles, they face a daunting challenge of manually reviewing a large number of candidate articles.

Even though we only interviewed journalists at WaPo for this study, we believe that the utility of evergreen articles and challenges associated with identifying evergreen articles are universal across different news organizations. We believe that the characterization of evergreen articles and automatic detection of evergreen articles, especially at early stages, are important tasks with research challenges and practical benefits.

Prior work includes popularity analysis conducted in various domains, such as news articles [3, 14, 45], videos [46, 47], shared images [48, 49] and online series [50]. Because the lifespan of news articles is found to be short [3, 45], existing studies mainly focused on the short-term popularity (***trending***) prediction. However, there are some observations that certain topics of news articles have a longer life cycle [51],and inferring popularity of long-term popular contents (***evergreen***) is very difficult [11]. To our best knowledge, none of the prior works have systematically examined the characteristics of evergreen articles and their automatic identification.

To fill up the gap, this work starts with the first research question: how can we reliably identify evergreen news articles at a large scale **(RQ1)**? Consistent with journalists, we leverage news articles' temporal traffic histories and propose a flexible parameterized definition to capture evergreen news articles with different temporal dynamics. Having obtained a dataset of evergreen news articles using the parameterized definition, then, we investigate on **RQ2:** what are the characteristics of evergreen news articles? Based on the insights learned in our analysis, finally, we move to tackle **RQ3:** can we identify evergreen news articles at early stages?

In answering these research questions, we make the following main contributions:

1. To our best knowledge, this work is the first work to study news articles' long-term popularity.

2. We proposed a parameterized definition to capture evergreen news articles on the basis of their historical page view data, which is validated by journalists.

3. Detailed analysis has been conducted to characterize evergreen news articles in various aspects.

4. Based on our analysis, we build machine learning models for the early detection of evergreen news articles and achieve promising results with extensive experiments, especially in production evaluation from journalists.

3.2 Related works

Since predicting news articles' popularity has long been considered as an important research area, there are a lot of existing efforts to this problem. However, unlike other online contents, such as videos and photos, the life spans of news articles tend to be shorter and their view counts often decrease faster [3, 11]. As a consequence, the majority of previous studies lies in the scope of *short-term* news articles' popularity analysis. In this section, we review related works and contrast them to our work under two categories: (1) news articles' popularity prediction; (2) long-term popularity prediction for other online contents.

3.2.1 Predicting Popularity of News

As one of the first investigations on popularity prediction of online contents, [11] discovers that news stories exhibit a much shorter lifespan than videos and usually become outdated after hours. Besides inferring future popularity by historical time series, recent works take an advantage of news articles' content information in prediction [3, 52]. To better understand user participation in news' propagation, researchers are also interested in predicting the number of users' comments [13, 14, 45, 53], number of votes [11, 14, 54] and number of tweets [55]. As noted earlier, due to the short lifespan nature of news, almost all of these existing works focus on predicting news articles' popularity within a short time. The most related work to ours is [56], where they define *long shelf life* news articles–i.e., those taking at least 80 hours to reach 60 percentage of their total page views in the lifetime, and *short shelf life* news articles–i.e., those taking less than 8 hours. Despite targeting at

analyzing long shelf life news articles, however, [56] fails to consider the temporal dynamics of news articles' long-term popularity. As a result, [56] cannot capture long-term popular "evergreen" news articles, which consistently attract high traffics over their lifetime across many years. In this work, taking temporal dynamics into consideration, we propose a reliable measurement to capture long-term popular news articles and perform a systematic analysis on them.

3.2.2 Predicting Long-Term Popularity

Although few works in news domain study long-term popularity prediction, there are some works focusing on long-term popularity prediction of other online contents, such as videos [47, 57] and paper publications [58]. To model popularity evolution, viewing content propagation as a stochastic process is one of the most common methods. For example, [58] adopts the reinforced Poisson process in paper citation network and [59] models the cascading in social network as a Hawkes process. For complex systems such as video platforms, where content propagation is difficult to model, researchers turn to time series approach and feature driven approach. In time series approach, often, early popularity series are used for future predictions. For instance, [47] assigns varying weights to videos' historical popularity series via a multiple-linear regression model to predict future popularity . Due to the dependency on the historical popularity, time series approach usually requires an extended period of observations and suffers from the so-called *cold-start* problem. A feature driven approach is proposed to address such issues. Diverse types of potential features that may impact popularity are incorporated into the prediction, such as text features [60], author features [61] and meta features [46]. Similar to videos, news articles have various traffic sources, including search engine and social media, which makes the propagation also hard to model. Hence, this work adopts the feature driven approach with historical popularity series to predict news articles' long-term popularity.

Compared with other online contents, news stories generally are more time sensitive. Only a very small fraction of published news articles exhibit high popularity patterns over many years, which makes the problem more challenging. Understanding why these small number of articles are consistently of interest to the public will benefit both journalists and news sites. To our best knowledge,

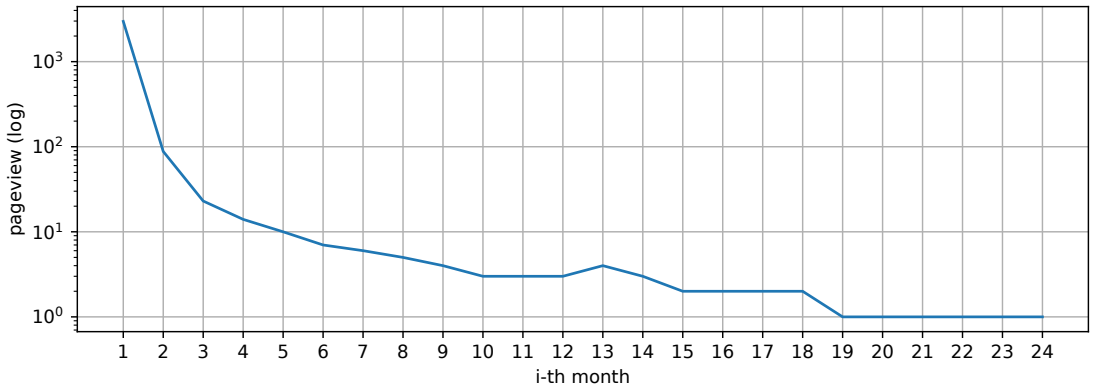


Figure 3.1: Median page views of news articles published from January 2012 to December 2015

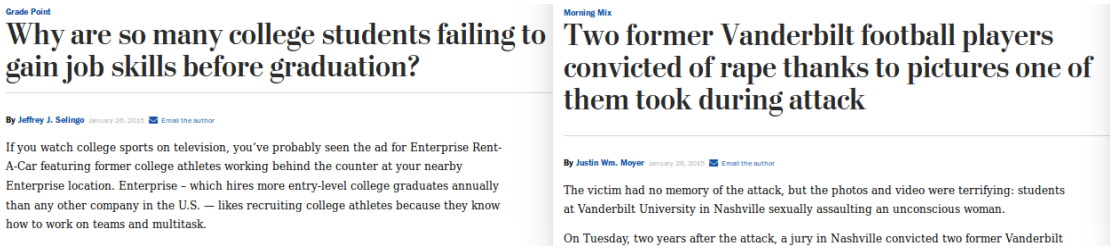
our work is the first to systematically define evergreen articles, to examine their characteristics, and to predict them at early stages.

3.3 RQ1: Defining Evergreens

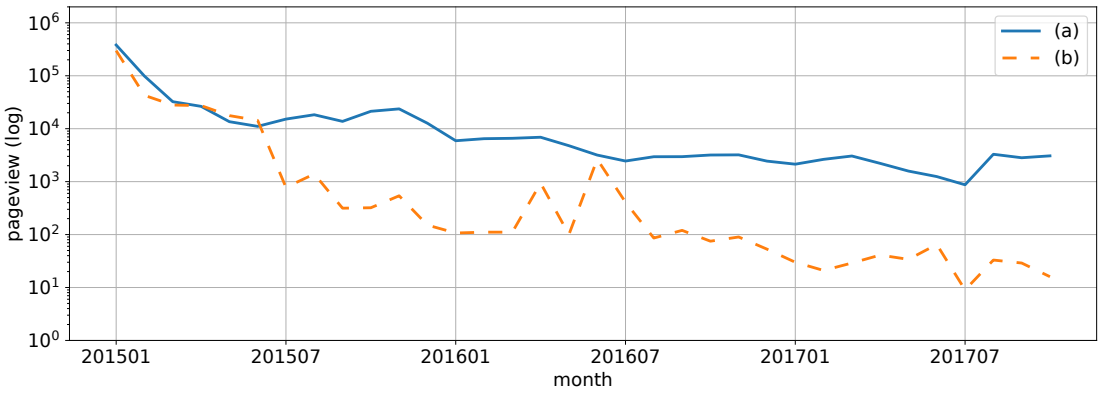
This section first introduces the dataset used in the later analysis and experiments. Then, we answer *RQ1: how can we reliably identify evergreen news articles at a large scale?* by proposing a parameterized definition of evergreen news articles.

3.3.1 WaPo News Article Dataset

Our dataset contains more than 500,000 news articles published by WaPo from January 2012 to November 2017. For each article, we dumped its monthly page view data from its publication date to November 2017. The *median* page view of each article's i-th month after initial publication is shown in Figure 3.1. Because the distribution of news articles' page views in i-th month is highly skewed, where most page views are zero, we present median page views, instead of average page views here. Note that articles' monthly page views in Figure 3.1 drop faster than log-linear and down to around 20 at 3rd month already, which is consistent with previous observation [11] and confirms the short lifespan of news articles.



(a) evergreen article (b) non-evergreen article



(c) two news articles' monthly page views (Y-axis on a log scale)

Figure 3.2: Articles with similar initial traffic data show different long-term popularity pattern. For example, article (a) consistently receives high traffics in long term, while (b)'s monthly page views drop dramatically over time.

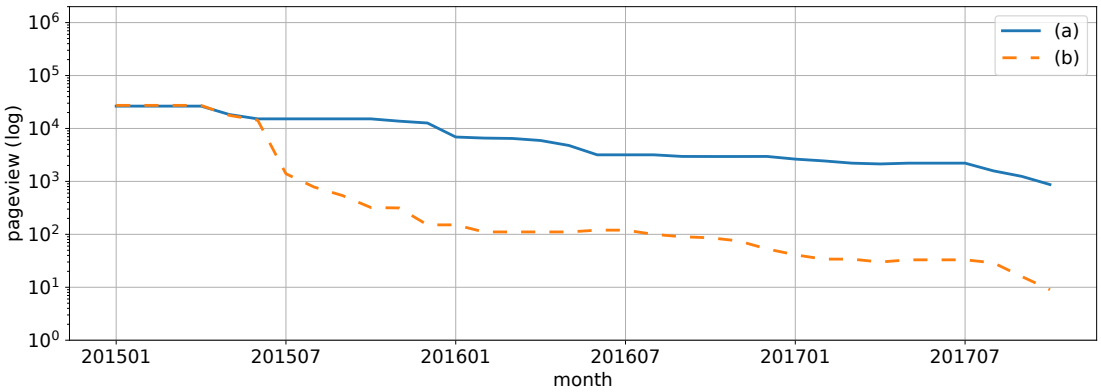


Figure 3.3: Median filtered page views of two example articles from Figure 3.2(c).

3.3.2 Definition of Evergreen Articles

Consulting journalists at WaPo, we set the observation time of each article up to 2 years after their initial publication, which is significantly longer than the period studied in prior studies. As shown in Figure 3.1, the popularity of news articles drops dramatically and down to tens at 3rd month. Since most news articles become outdated after 3rd month, the first 3-month traffic data is viewed as a news article's *initial traffic* to define trending articles. More specifically,

Definition 3.1 (Trending Article) *During the observation period, we sort news articles by their initial first 3-month traffics and consider the first k ranked articles as* **top-k trending** *articles.*

As a motivating example to capture evergreens, in Figure 3.2, we present two news articles published in January 2015 and their monthly page views till October 2017. Note that, despite receiving similar page views in the first a few months, these two articles exhibit radically different long-term traffic patterns. Article (a) maintains high traffic long after its publication and receives more than 5,000 page views a year later, while the traffic of article (b) drops dramatically to around 100 after a year. Clearly, article (a) is consistently of more interest to readers, thus fitting the definition ofan evergreen.

Although most news articles have a short lifespan similar to article (b) in Figure 3.2 and exhibit a fast decaying traffic pattern after the initial publication, we still observe occasional peaks long after their publication. Interviewed with a domain expert at WaPo, we found out these peaks were mainly resulted from 1) journalists at WaPo promoted these articles; and 2) these articles were associated with occurring events. These sudden traffic peaks are usually caused by stochastic events[1] and very difficult to predict. Accordingly, to better capture long-term popular news articles, we propose to use *median filters* to remove those sudden peaks. Examples of smoothed traffic pattern are shown in Figure 3.3. In addition, because of news articles' trending nature, initial traffics of news articles generally are significantly higher than the rest, up to several orders of magnitude. Therefore, we ignore initial traffic information when identifying evergreen news articles. Traffic

[1]Even though some events occur more regularly than others, such as seasonal festivals and holidays, and thus might be predictable, the impact of events on news articles' long-term popularity is beyond the scope of this work. We leave it for future study.

patterns of evergreen articles should not decrease too fast. To get smooth traffic series, we adopt accumulated traffic series and use the following *normalized* metric to measure a traffic pattern,

Definition 3.2 (Accumulated Traffic Ratio (ATR)) $ATR_i = \frac{\sum_{j=1}^{i} \hat{pv}_j}{\sum_{k=1}^{n} \hat{pv}_k}$, *where* i *is the* i_{th} *month after an article is published.*

When an article has constant traffic, its ATR starts low but will increase linearly, as described by blue line in Figure 3.4 (a). At the same time, since trending articles' traffics mostly fall in a few months, their ATR starts high but increases slowly. If we look at the area under ATR for an evergreen [2] article vs. a trending article, evergreen article will have smaller area.

With these observations, we propose the following parameterized operational definition of an evergreen news article.

Definition 3.3 ((α,β,γ)-Evergreen Article) *Ignoring the initial first p-month traffics*[3]*, we denote the monthly page view time series of an article* x *during the remaining observation period as* $PV = (pv_1, pv_2, ..., pv_n)$*. Then, first, we use: (1) the median filter with a window size* γ *to smooth the time series* PV *as* $\hat{PV} = (\hat{pv}_1, \hat{pv}_2, ..., \hat{pv}_n)$*. If the smoothed time series* $\hat{PV}$ *satisfies (2) average monthly page view at least* α *such that:* $\frac{1}{n}\sum_{i=1}^{n} \hat{pv}_i \geq \alpha$*, and (3) normalized area under ATR at most* β *such that:* $\frac{1}{n}\sum_{i=1}^{n} ATR_i \leq \beta$*, then, the article* x *is referred to as an* (α, β, γ)-**evergreen** *article.*

Note that α guarantees the minimum monthly page views, β controls the decaying rate of an article's page views, and γ is used to remove sudden page view peaks caused by unpredictable events. In our experiments, empirically, we set $\gamma = 5$, and explore different α and β values in the next part.

3.3.3 Tuning α and β

To guarantee each article to have at least 2-year traffic history, we only consider articles published from January 2012 to December 2015. After removing articles with traffic tracking errors, we have 242,429 news articles in total.

[2] Because the median filtering is employed to smooth page view series, occasional traffic peaks are removed and will not affect the area.

[3] After consulting with journalists in WaPo, we set p as 3 months in our subsequent experiments.

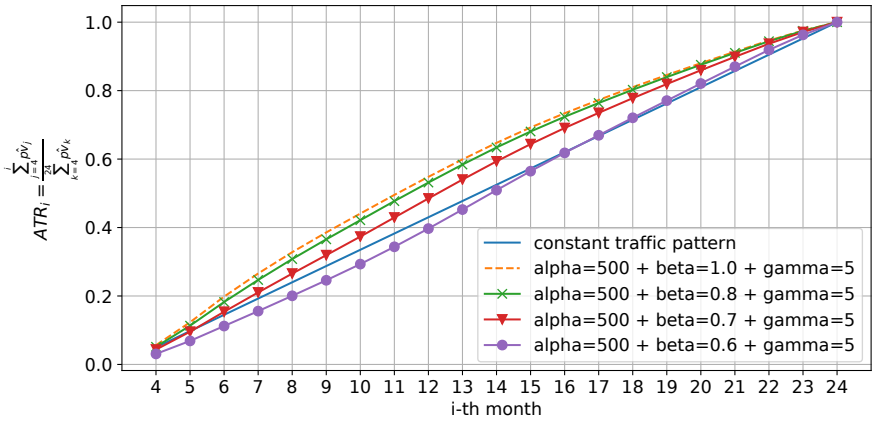


Figure 3.4: The impact of β on the decreasing rate of articles' page views

α	250-500	500-1000	>1000
Positive Rate%	3.33%	10.00%	25.00%

(a)

β	0.0-0.6	0.6-0.7	0.7-1.0
Positive Rate%	18.33%	11.67%	6.67%

(b)

Table 3.1: Journalist validation with different α and β

Figure 3.4 shows ATR of news articles with α=500 and β varying from 0.6 to 1.0. As indicated, controlling only the average monthly page view during the observation period is not enough to identify evergreens, because news articles still tend to gain more page views in the beginning months. Decreasing β helps filter out articles exhibiting faster declining patterns. To further validate the effect of α and β in capturing evergreens, we consulted domain experts at WaPo to manually label a few samples from each criterion[4]. More specifically, we sample 60 articles from each α with β=1.0 and each β with $\alpha \geq 250$, then mix them up for labeling. As expected, Table 3.1 shows the agreement of journalists on our definition that an article with larger α and lower β is more likely to be evergreen. Considering the rarity of evergreen articles, our definition is confirmed to filter out most non-evergreen articles and produce highly qualified evergreen datasets. Weighing both the quality and size of the dataset, we adopt (α=500, β=0.6, γ=5)

[4]Labeling details are similar to Newsroom Editor's Evaluation in the experiment section.

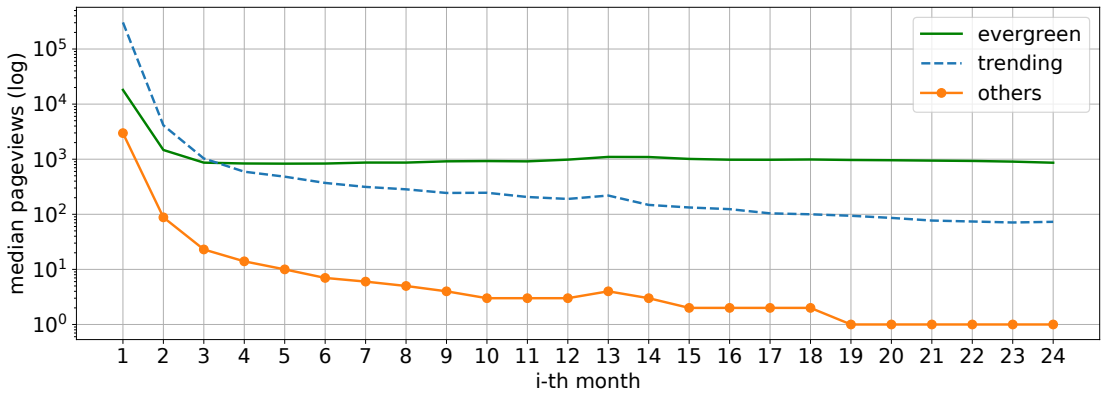


Figure 3.5: Median page view comparison between evergreen and trending news articles

as the criterion to finally obtain 1,293 evergreens out of 242,429 new articles in WaPo, a mere 0.5 % of all news articles, and use them as the base evergreen articles for further analysis and experiments.

3.3.3.1 Comparison with trending articles

To make a fair comparison, from January 2012 to December 2015, we select the same number of 1,293 trending articles, of which less than 10% are overlapped with evergreen articles. Then, the median page view comparison between evergreen and trending articles is shown in Figure 3.5. As expected, trending articles initially attract significantly higher traffics than evergreen articles, but quickly fade away from users' attention. On the contrary, evergreen articles are consistently of interest to the public, and generally obtain almost one order of magnitude higher monthly page views than trending articles after one year of publication.

3.4 RQ2: Characterizing Evergreens

Having identified evergreen news articles from traffic data, we turn to *RQ2: what are the characteristics of evergreen news articles?* In this section, we focus on characterizing evergreen news articles, and examine on possible factors correlating their long-term popularity.

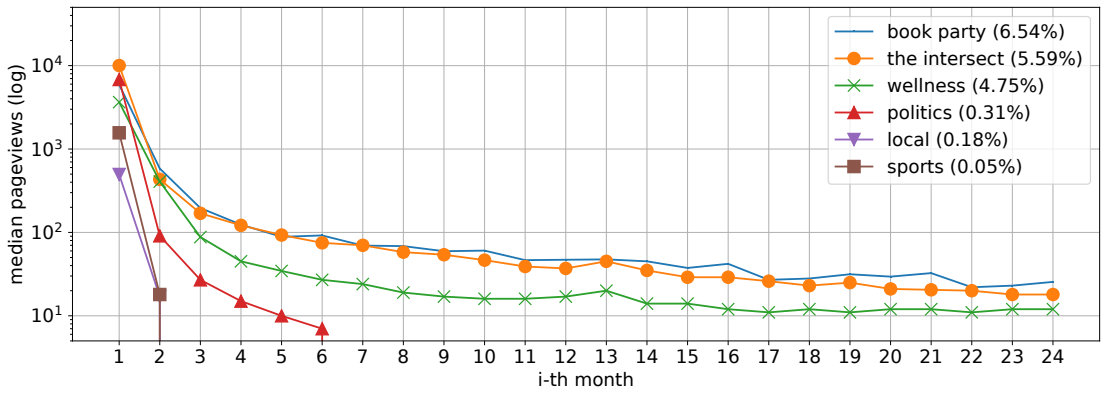


Figure 3.6: Monthly median page views of selected categories with evergreen ratios.

Rank	Category	Evergreen Ratio
#1	book reviews	6.54%
#2	Internet culture	5.59%
#3	wellness	4.75%
#4	perspective	4.25%
#5	investigations	3.54%
#6	health & science	2.78%
#7	parenting	2.73%
#8	life	2.60%
#9	real estate	2.58%
#10	finance	2.49%

Table 3.2: Top 10 categories with the highest evergreen ratios (category names are edited per similar contexts)

3.4.0.1 Category

In WaPo, each news article has been manually assigned a category by journalists, e.g., politics, opinions, and business. Previous studies show categories play an important role in identifying trending articles, where some categories tend to generate more viral articles than others [3]. Therefore, we investigate here if there exists a similar relationship between categories and evergreens. Excluding categories with less than 100 articles, we sort the remaining 127 categories by their evergreen ratio (i.e., a fraction of evergreen articles over all articles in a category) and present the top 10 categories in Table 3.2. As demonstrated, housing and health related

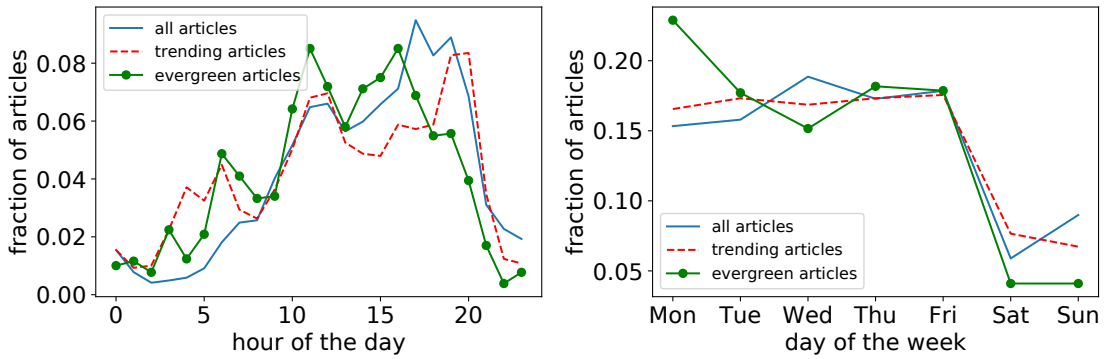
Rank	Top 10 words in each topic	Evergreen Topic Ratio
#1	health, diet, study, sugar, food, fat, disease, weight, calorie, body	9.13
#2	doctor, patient, hospital, health, treatment, physician, care, medicine, surgery, illness	6.05
#3	paint, wood, material, water, wall, tile, piece, brick, surface, floor	5.65
#4	brain, memory, welch, trauma, mind, body, love, emotion, activity, people	5.61
#5	scientist, human, bone, dinosaur, animal, specie, researcher, fossil, creature, study	5.06
#6	study, researcher, research, university, author, behavior, journal, finding, professor, effect	4.61
#7	kid, parent, child, school, teen, adult, parenting, mom, son, family	4.29
#8	illness, flu, symptom, strain, allergy, nose, people, fever, disease, health	4.11
#9	hour, night, sleep, time, day, bed, minute, holmes, morning, schedule	4.00
#10	study, percent, datum, research, rate, poverty, researcher, income, factor, effect	3.96

Table 3.3: Top 10 evergreen topics

topics are consistently of interest to the people and show a higher percentage of evergreen articles. In addition, other top ranked categories include *social issues*, *environment* and *science*, indicating people also pay continuous attention to society and living environment. In Figure 3.6, we show monthly median page views of top ranked categories and some most popular categories, like *politics* and *sports*. Note that categories with higher evergreen ratios tend to attract more traffics in long term, which validates the impact of articles' categories on their long-term popularity.

3.4.0.2 Topic

Besides considering manually assigned categories, in addition, we leverage on topic modeling techniques to extract more fine-gained topics based on the co-occurrence of words. To be more specific, we train a 1,000-topic noun only topic model using the LightLDA [62, 63], and compare topic distributions between evergreen and non-evergreen articles. Sorted by the *evergreen topic ratio* (i.e., $\frac{\text{topic proportion in evergreens}}{\text{topic proportion in non-evergreens}}$) of each topic's proportion in evergreen articles to its in non-evergreen articles, the top ranked evergreen topics are shown in Table 3.3. Consistent with manually assigned categories, categories such as *housing*, *health*, *education and research studies* are the most sustainable and lasting topics. Since news articles with more words co-occurred in evergreen topics are more likely to be evergreens, news articles' content information can be a good feature in evergreen detection.



(a) (b)

Figure 3.7: (a) fraction of articles published in each hour of the day; (b) fraction of articles published on each day of the week.

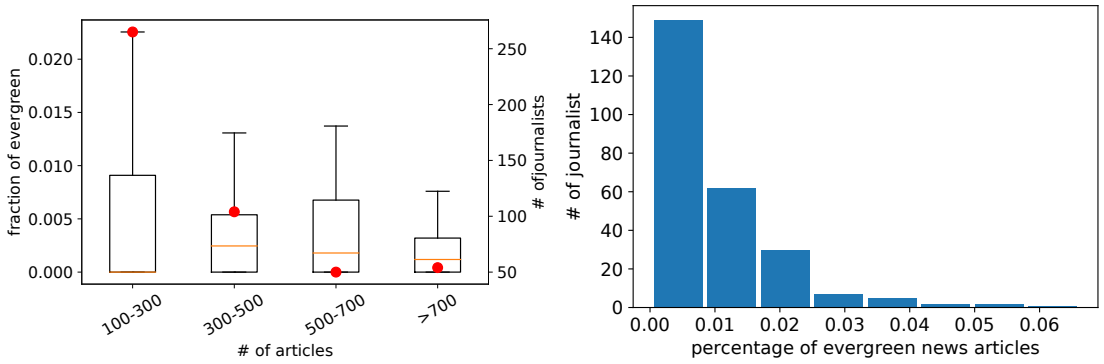
3.4.0.3 Publication Time

Next, we explore the relationship between news articles' long-term popularity and their publication time. Figure 3.7 shows the fraction of articles published in each hour of the day and on each day of the week. Interestingly, trending articles are more evenly distributed across the hours and days, even well represented at odd times such as those from midnight to 8am or over weekends. Although most news articles are published in the afternoon, evergreen news articles are mostly published in the middle of a day. More interestingly, few evergreen news articles are published on weekends, while Monday conveys the most evergreen articles. Since the distribution of evergreen news articles' publication times does show distinctly different patterns from regular articles, we include publication times of articles as an important feature in learning.

3.4.0.4 Journalist

"Who writes an article" is another important meta data to consider. In this part, we examine the role of journalists on articles' long-term popularity in two aspects–(1) how many articles a journalist has written, and (2) how many evergreen articles a journalist has written. That is, we ask if a journalist has written more (evergreen) articles in past, is her new article more likely to be evergreen?

There are around 12,000 journalists in total in our dataset. In order to mitigate the sampling bias, we first remove the journalists who has written less than 100



(a) (b)

Figure 3.8: (a) Distribution of journalists' publication number and fraction of evergreen articles in each group. The red dot presents the number of journalists in each group, while the boxplot describes distribution of fraction of evergreen articles per journalist; (b) Distribution of fraction of evergreen news articles in each journalist's publication.

articles. As a result, we have 473 journalists who have written a total of 215,948 articles. Consistent with the ratio of the entire dataset, it turns out that 854 articles among 215,948 articles (i.e., 0.4%) are (α, β, γ)-evergreen articles.

Figure 3.8 (a) presents the distribution of journalists' article numbers and the fraction of evergreens in each group. The result shows that the fractions of evergreen articles for most journalists are very small, varying from 0.1% to 1%. Spearman's rank correlation coefficient between journalists' publication count and their evergreen fractions is 0.032, indicating that there is no obvious monotonic relationship between these two variables. Thus, journalists' possibility to publish evergreen articles has no simple relationship with their total publication counts. That is, an experienced journalist does not necessarily produce an evergreen article.

Around 50% journalists published at least one evergreen news article. In Figure 3.8 (b), we present the distribution of fraction of evergreen news articles for those journalists who have at least one evergreen news article. This figure illustrates that, although most journalists have written a small proportion of evergreen news articles, there are indeed a few journalists good at publishing long-term popular news articles. Therefore, although experience in writing does not help produce evergreen articles, expertise in evergreen writing does. As such, journalist information can give some hints on early detection of evergreens.

3.4.0.5 Sentiment

We utilize Vader sentiment analyzer [64] to examine articles' sentiment and present cumulative distribution function (CDF) of the compound sentiment scores of both full content and title in Figure 3.9. For each compound sentiment score X, CDF shows the proportion of news articles with sentiment scores less than X, ranging from -1 to +1, indicating the most extreme negative to the most extreme positive. These figures reveal that most news articles from WaPo carry neutral titles and positive contents, while evergreen news articles show more clear polarity in contents. Interestingly, trending articles present much more distinctive patterns than evergreen articles in sentiment, where they convey slightly more negative titles and contain higher percentage of negative articles. One possible explanation is that breaking and thus viral news may include reports about accidents, disasters, or surprising events, that often have negative tones in their coverage. Although our finding is contrary to the previous study [65], that discovers that positive articles are more likely to be viral than negative ones, sentiment features still matter a lot in identifying trending articles. However, in the case of identifying evergreens, the importance of sentiment-based features extracted from article contents seems less significant.

3.5 RQ3: Predicting Evergreens Early

Previous section has shown that evergreen articles exhibit different characteristics from non-evergreen articles in various aspects, especially in genre and journalist information. Therefore, using these insights, we intend to answer *RQ3: can we predict evergreen articles at an early stage?* and attempt to build an accurate machine learning model to unearth early a small fraction of evergreen articles among many non-evergreen articles. To better validate our learned model, we conduct the following experiments:

E1. We first show the results of 10-fold cross validation on the historical dataset (Jan. 2012–Dec. 2015).

E2. Using the best setting from 10-fold cross validation, then, we train a model on the whole historical dataset (Jan. 2012–Dec. 2015) and test the model on the new articles published later (Jan. 2016–Apr. 2016).

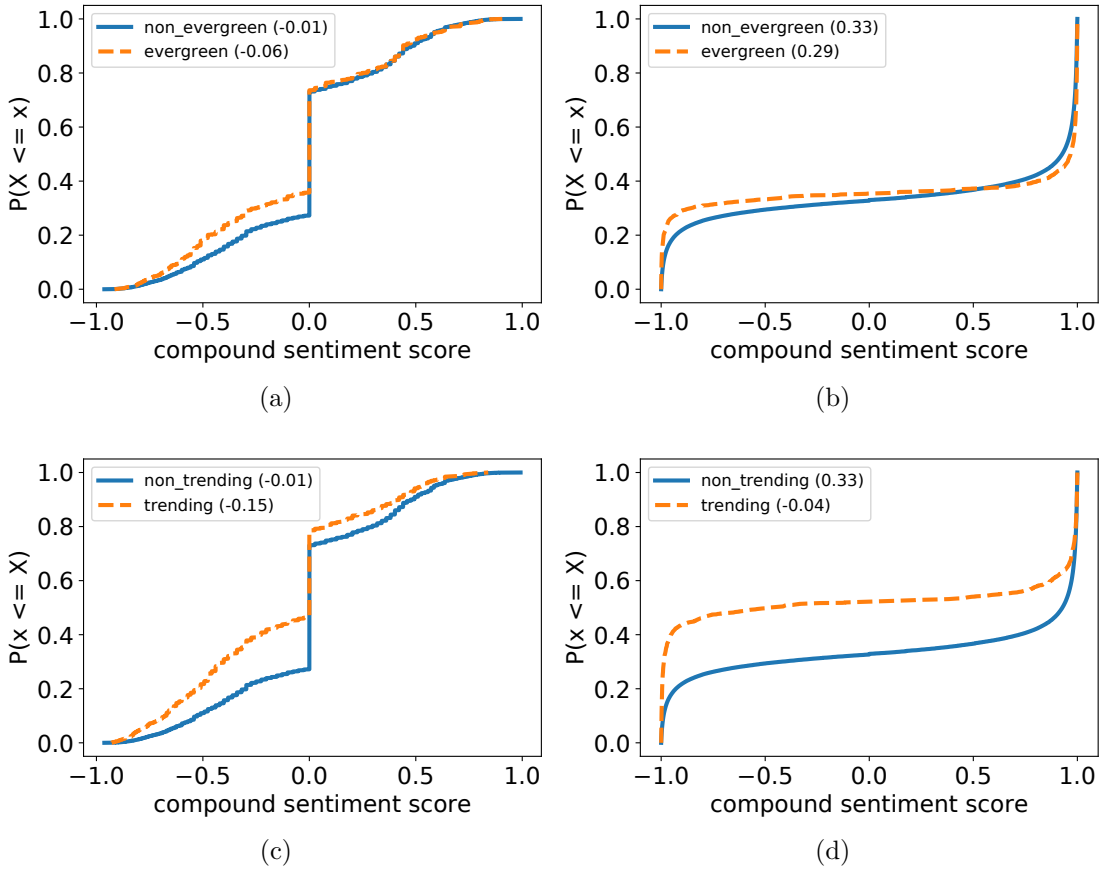


Figure 3.9: Both (a) and (b) show cumulative distribution function (CDF) of evergreen articles' compound sentiment scores on title and full content individually; Both (c) and (d) present CDF of trending articles' title and full content compound sentiment scores correspondingly. Average compound sentiment scores are included in the parenthesis. The cliffs in (a) and (c) indicate most titles are neutral.

E3. In addition to the cross validation and time-split classification, we present temporal page view patterns of predicted evergreen news articles published in subsequent months (Jan. 2016–Dec. 2016).

E4. For the most recently published articles, newsroom editors in WaPo manually check and validate the quality of predicted evergreen articles.

3.5.1 Set-Up

For a given news article, after observing its initial traffic in the first three months, we predict whether it will be an evergreen or ephemeral story. Based on the analysis in the last section, we propose to exploit three feature sets in our model as follows:

1. **Traffic features**: As indicated in prior works, the degree of popularity at early stages are strongly related with that of future popularity. Thus, we extract the traffic features from first 3-month traffic data, including the number of monthly page views, difference of page views of two consecutive months, and decreasing rate between two consecutive months. Note that, when defining evergreen news articles, we ignored their first 3-month page views and labeled articles only by their later popularity patterns.

2. **Content features**: Topic analysis demonstrates that content features, including keywords and topics, may be also important clues in detecting evergreens. Therefore, we exploit word embedding and bag-of-words trick to extract content features from news articles. More specifically, we train an unsupervised FastText [66] model on all news articles with 200-dimension feature vectors.

3. **Meta features**: Based on the observations in the last section, we selectively include meta features such as news articles' category, publication time, and journalist information. Due to the noises in manually assigned categories and a large number of journalists, *one-hot encoding* (i.e., encoding each category or journalist as an isolate feature dimension) may cause overfitting. Therefore, to obtain more compact features for each news article, we propose to use the average FastText embedding of all prior published articles under its category as its category feature. As a result, similar categories share similar feature vectors in our category feature space. For example, *opinion* is close to *perspective*, while *lifestyle* is close to *food & drink* and *entertainment*. Likewise, each article's journalist feature is extracted in a similar way. Finally, we include publication hour of the day and day of the week as categorical features.

Table 3.4 shows the precision at top 30, Receiver Operating Characteristic Curve (ROC) Area-Under-Curve (AUC) and Precision-Recall (PR) AUC scores

Classifiers	Logistic Regression	Random Forest	GBDT
Prec@30	16.33%	22.67%	**32.33%**
ROC AUC	93.52%	94.18%	**96.09%**
P-R AUC	11.10%	11.58%	**17.31%**

Table 3.4: Comparison with different classification models

Metric	initial traffic	content feature	traffic + content	content + meta data	traffic + content + meta data
Precision@10	10.00%	15.00%	36.00%	26.00%	**43.00%**
Precision@20	11.50%	14.50%	34.00%	24.00%	**34.00%**
Precision@30	13.67%	14.00%	30.33%	21.00%	**32.33%**
ROC AUC (50.00%)	92.77%	88.18%	95.98%	89.21%	**96.09%**
P-R AUC (∼0.50%)	7.03%	5.50%	16.54%	7.62%	**17.31%**

Table 3.5: 10-fold cross validation

of three learning models–i.e., Logistic Regression, Random Forest, and Gradient Boost Decision Tree (GBDT)–using all features via 10-fold cross validation. As the model learned using LightGBM package [67] consistently performed the best, in the following experiments, we only report the prediction results using the GBDT as the main learning model.

3.5.1.1 Evaluation Metrics

As described in previous section, we use (α=500, β=0.6, γ=5) as criterion that yield 1,293 articles out of the total 242,429 news articles as evergreen news articles. Note that this is a binary classification problem with significantly skewed class distribution ratio of 0.5 : 99.5. Because of the highly imbalanced dataset, we choose to compare models with both Area Under Receiver Operating Characteristic Curve (ROC AUC) and Area Under Precision Recall Curve (P-R AUC), where random baselines are 50% and 0.5% respectively.

Moreover, in real settings, newsroom editors expect top-K potential evergreen candidates and want to manually review them to determine true positives. Therefore Precision@K is also provided to show the performance of top-K predictions. Per each fold in the 10-fold cross validation, since there exist around 130 evergreen articles out of ∼24,000, Precision@K is expected to be 100% when K ranges from 10 to 30. The results of different feature combinations are conducted on the exactly same 10 folds and given in Table 3.5.

3.5.2 E1. Cross Validation

The 10-fold cross validation results show that first, using only first 3-month page view series, it is difficult to identify evergreen news articles. However, adding content and meta features gives a significant improvement on all metrics. When utilizing all features, we achieve the best performance, where the P-R AUC improves from 7.03% to 17.31% and Precision@30 increases from 13.67% to 32.33%. More interestingly, pre-publication prediction, which only considers content and meta features, achieves very promising results of having 21% accuracy in top-30 prediction. Moreover, when adding traffic features, we observe the boosted improvements in P-R AUC and Precision@K, implying that news article's initial traffics and contents contribute to its long-term popularity in different aspects and both of them are indispensable in the early prediction of evergreens. A possible explanation is that news articles with high initial traffics enjoy high visibility, thus evergreen articles in this group are more likely to be shared or archived by users. For evergreen articles with limited initial traffics, though lacking enough visibility to users in the beginning, because of their timeless quality and interesting topics, they still gain high traffics in long term via other channels like search engine or re-promotion.

3.5.3 E2. Time-Split Classification

In real applications, early detection models aim to predict evergreen articles from newly published news articles. Therefore, this time-split experiment is designed to examine how general a learned model is across time. Using the best setting from 10-fold cross validation, we *train* a model on the articles published between January 2012 and December 2015. To make sure each news article have at least 2-year traffic data, then, we choose the articles published from January 2016 to April 2016 as a *test* set. In the test set, 254 articles ($\sim 0.7\%$) out of 34,509 are identified as *(500, 0.6, 5)-evergreen*. The result is presented in Table 3.6. Comparable with 10-fold cross-validation, our model with all of traffic, content, and meta features achieves 30% top-10 accuracy and 13.88% P-R AUC. Note that, since data distribution often varies over time, a time-split experiment is a more challenging setting than cross-validation. Even for evergreen news articles, popularity still varies over time. Thus, the performance gap between cross validation and time-split experiment is due to the changes in data distributions.

Prec@10	Prec@20	Prec@30	ROC AUC	P-R AUC
30.00%	25.00%	26.67%	94.46%	13.88%

Table 3.6: Time-Split experiment

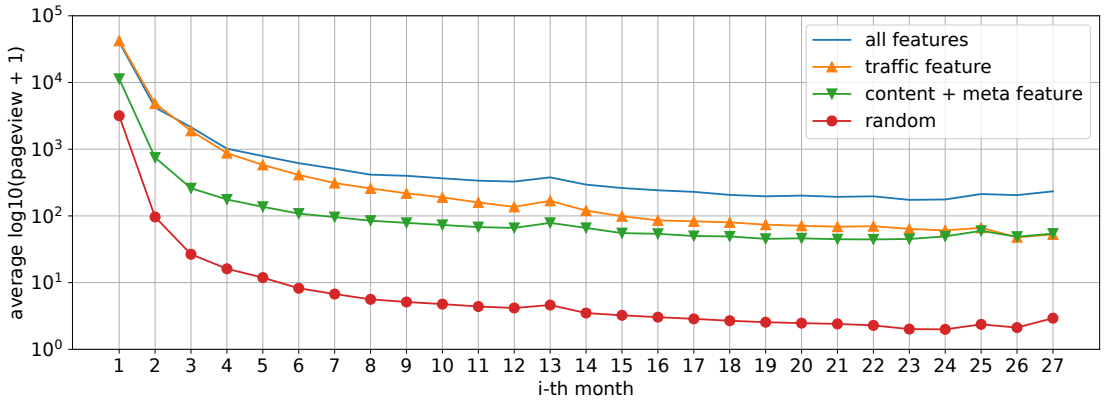


Figure 3.10: Page view patterns of top predicted evergreen articles under different feature groups

3.5.4 E3. Time-Split Traffic Evaluation

Because classification measurements are too strict to distinguish news articles with slightly different temporal patterns, we propose to examine temporal page view trajectories of predicted evergreen news articles, which gives more intuition on articles' long-term popularity and serves as more valuable indicators in production. In this part, using the best setting from 10-fold cross validation, we train a model on the whole historical dataset from January 2012 to December 2015, predict evergreen news articles in each month from January 2016 to December 2016 and monitor their page view trajectories from the publication month till March 2018.

Aimed at comparing different feature groups, we present the traffic patterns of top predicted evergreen articles with different feature groups in Figure 3.10. More specifically, for each model, news articles are ranked by the predicted probabilities among their publication month, and we choose the top 100 ranked articles in each month as potential evergreen articles. Consequently, 1,200 articles out of $\sim$100,000 in total are selected in each group. Overall, news articles selected from all feature groups exhibit a long-term popularity pattern, while combining all features obtains the highest monthly page views in the long run. For each month starting from 4th month, we conduct t-test and find that articles predicted with

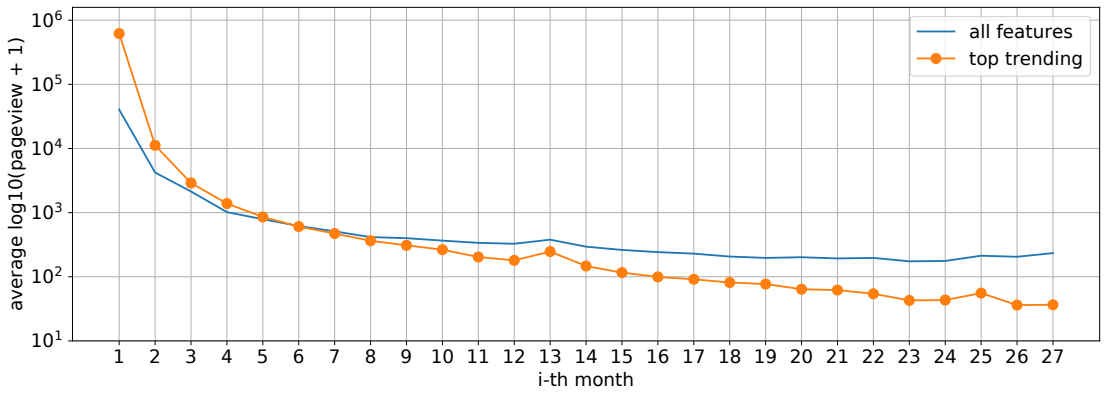


Figure 3.11: Page view pattern comparison between top predicted evergreen articles with top trending articles

all features significantly attract higher page views than other article groups at a p-value < 0.0001. Additionally, although articles predicted with traffic feature show dramatically higher initial traffics, they display similar page view trajectories to articles predicted with pre-publication features long after publication.

As shown in Figure 3.5, trending news articles, which enjoy significantly higher visibility when just published, display much higher popularity than most news articles, and thereby should be a strong baseline in long-term popularity prediction. Therefore, in Figure 3.11, we present page view pattern comparison between top predicted evergreen news articles and top trending news articles. Top trending articles consist of top 100 articles with the highest initial traffics in each month. As expected, top trending articles attract much more traffics in the first a few months, while top predicted evergreen articles demonstrate a more stable traffic pattern and consistently gain more page views one year later. For each month starting from 9th month, articles predicted with all features attract significantly higher page views than trending articles at a p-value < 0.0001 under t-test.

3.5.5 E4. Newsroom Editor's Evaluation

In order to help newsroom editors at WaPo, we plan to deploy the early detection model and recommend recently published potential evergreen news articles to them. Although we have presented promising performances in 10-fold cross validation and indeed observed long-term popularity patterns of predicted evergreens in the time-split experiments, quality check by journalists themselves is indispensable

before actual deployment. Therefore, we include newsroom editors' evaluation in this section, presenting both pre-deployment evaluation and production evaluation.

Identifying evergreen news articles at early stages is challenging to both machine as well as journalists. Predicting whether a news article will be long-term popular requires domain expertise in several areas, including news editing, social media promotion, and search engine optimization (SEO). As only a few domain experts are qualified for this task, we consult an audience development team in WaPo, who is responsible for optimizing news articles and conducting social promotions, to manually check the quality of evergreen predictions in recent months.

There are three major use cases for evergreen news articles in the newsroom:

1. Regularly re-promote evergreens on social media.

2. Update content with latest information. For example, people keep searching for mass shooting related articles, and journalists should update those evergreen articles with the newest events.

3. Embedded as linking URLs to related context and references when editing new articles.

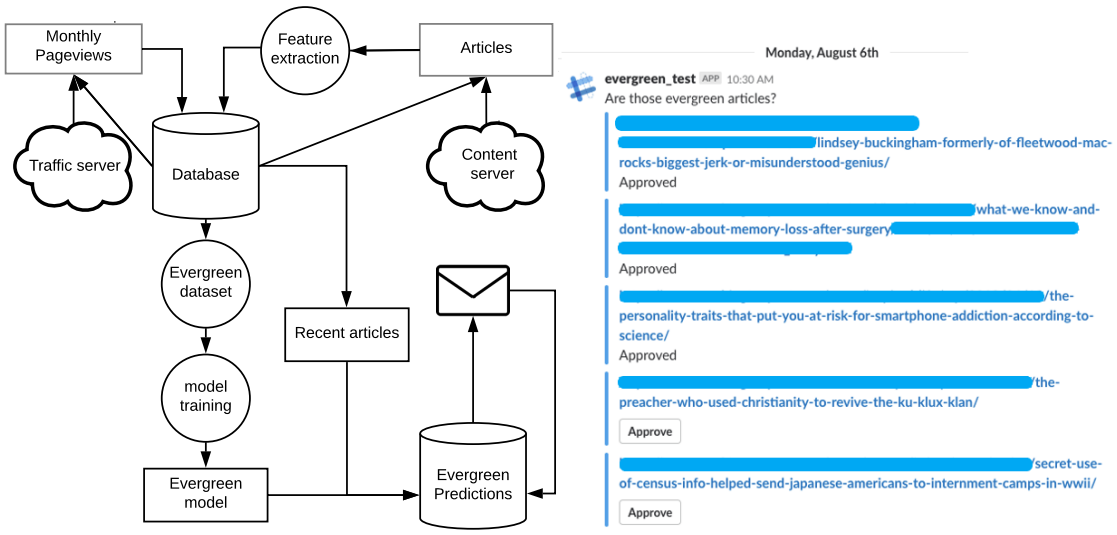
Based on these three use cases, for each news article in the evaluation, if its value is confirmed in any use case, it will be labeled as a true evergreen.

3.5.5.1 Pre-deployment Evaluation.

In the evaluation, for each month from January 2017 to June 2017, we recommended top 10 potential evergreen news articles predicted by our model to the team at WaPo. Through manually reviewing these 60 news articles, 65% of predictions, 39 in total, were labeled as true evergreens. Considering that only tens of news articles out of thousands published in each month could be evergreen articles, 65% top-10 accuracy in monthly prediction is encouraging and indicates that our model is practical enough to be deployed in production.

3.5.5.2 Production Evaluation.

Based on the promising result from pre-deployment evaluation, we deployed the evergreen prediction system with the architecture presented in Figure 3.12(a) at



(a) (b)

Figure 3.12: (a) Architecture of evergreen prediction; (b) An example of weekly recommendation (identifying information is masked due to double-blind review.)

WaPo. The system regularly updates evergreen prediction model and sends out weekly recommendations to the newsroom. On every Monday, editors receive top 5 potential evergreen articles and provide feedback in Slack. An example of weekly recommendation is shown in Figure 3.12 (b), where 3 out of 5 articles are verified as true evergreens. The system was deployed at WaPo in July 2018 and achieved $\sim 43\%$ accuracy in the first 5 weeks.

More importantly, Both 65% of top-10 accuracy in monthly prediction and 43% of top-5 accuracy in weekly prediction are achieved without any human labels, validating that our parametized definition of evergreen is reasonable and can produce highly qualified evergreen datasets.

3.6 Discussion

Despite encouraging results, there are several areas to be improved. First, there is a room for improvement in the early prediction of evergreen articles. For example, this work focused on building prediction models mainly using article features. In future, in addition to article features, we will incorporate users' feedback such as users' dwell time and sentiment left in comments. Second, although our proposed definition can produce highly qualified evergreen datasets, a small number of false positives

may still exist. Moreover, there are chances that some true evergreen articles may not get enough attentions by both journalists and readers, which will lead to a few false negatives in the evergreen datasets. To overcome this issue, instead of formulating evergreen article identification as a rare class classification problem, learning classifiers from only positive and unlabeled data [68] or noisy labels [69] may help. However, besides taking time to monitor articles' long-term traffic trajectories, how to evaluate predictions from these techniques both automatically and efficiently still requires further exploration. Finally, as shown in the case of time-split classification, long-term popularity prediction faces a challenge that data distribution varies with time. To tackle this challenge, we plan to explore time evolving models in future work.

3.7 Summary

This work presents a study on characterizing and early detecting evergreen news articles. Firstly, taking temporal dynamics into consideration, we proposed a parameterized definition to capture evergreen news articles, which is validated by journalists (**RQ1**). Next, driven by the research question why evergreen news articles show such different traffic patterns, we conducted a quantitative analysis to shed light on evergreen news articles' long-term popularity (**RQ2**). Finally, based on the insights from our analysis, we extracted related features from news articles to build early prediction models on evergreen identification (**RQ3**). Throughout extensive experiments, we have validated that our models gain promising results in early detection of evergreen news articles, and shown that the predicted evergreen news articles indeed exhibit a long-term high traffic pattern. More importantly, verified by journalists, our proposed model achieves encouraging performance in production.

Chapter 4 | Understanding News Evolution

This section is based on our collaboration with the Washington Post.

4.1 Introduction

Along with the rapid development of web services, an increasing number of news articles are published daily, describing both major and minor events worldwide. Due to the tremendous amount of news articles being produced every day, readers easily get lost in this information flood. Fortunately, *news timeline*, which summarizes each event with primary messages in chronological order, makes it easy for readers to gain key insights and understand the evolution of news events. As such, many major news media has adopted the idea and have frequently produced news timelines of major news events. For example, Figure 4.1 (b) describes a news timeline summarizing how the United States and China went through the economic conflict in 2018. Note that as the example illustrates, creating a news timeline requires the resolution of two sub-problems: (1) choosing an ideal number of days among hundreds or thousands of candidate days, and (2) generating succinct text summaries per days.

4.1.1 Industrial Use Case

Combined with visual or interactive interfaces, news timelines can provide a convenient way to compress overloaded news to the audience. Figure 4.1 illustrates two real timeline examples from two major US newspapers. Figure 4.1 (a) is an

[1] `https://www.washingtonpost.com/graphics/2018/national/trump-kim-jong-un-timeline/`

Feb. 25, 2018 North Korea is “willing to have talks” with the United States, South Korea says, as the PyeongChang Winter Olympics close in a burst of fireworks and diplomacy.
Mar. 8, 2018 Trump agrees to meet Kim for talks, an extraordinary development after months of heightened tension. Kim commits to stopping nuclear and missile testing.
...
Jun. 1, 2018 Trump says the summit will take place June 12 as planned. During a visit to the White House, a North Korean hands Trump a large letter from Kim.

Table 4.1: An example timeline by The Washington Post about the summit between United States and North Korea.[1]

(a) Trump-Russia investigation (The Washington Post)

(b) China-US Trade War (The New York Times)

Figure 4.1: News timeline summarization examples on major news media

interactive timeline summarization about Trump-Russia investigation from The Washington Post, while Figure 4.1 (b) is a text-based timeline summarization about China-US Trade War from The New York Times. To help readers better understand the evolution of each news event, journalists take time to collect and organize related news articles, figure out major events and story-lines, and “manually" summarize them in chronological order. As events such as natural disasters and political issues can span from several months to multiple years and involve thousands of news articles, such a manual process cannot scale well. As this process is both time-consuming and labor-intensive, currently, despite the popularity of the concept, not all newspapers are able to quickly produce such news timelines.

To address this challenge, several automatic news timeline summarization methods have emerged in recent years [1,70–75]. By and large, there are mainly two

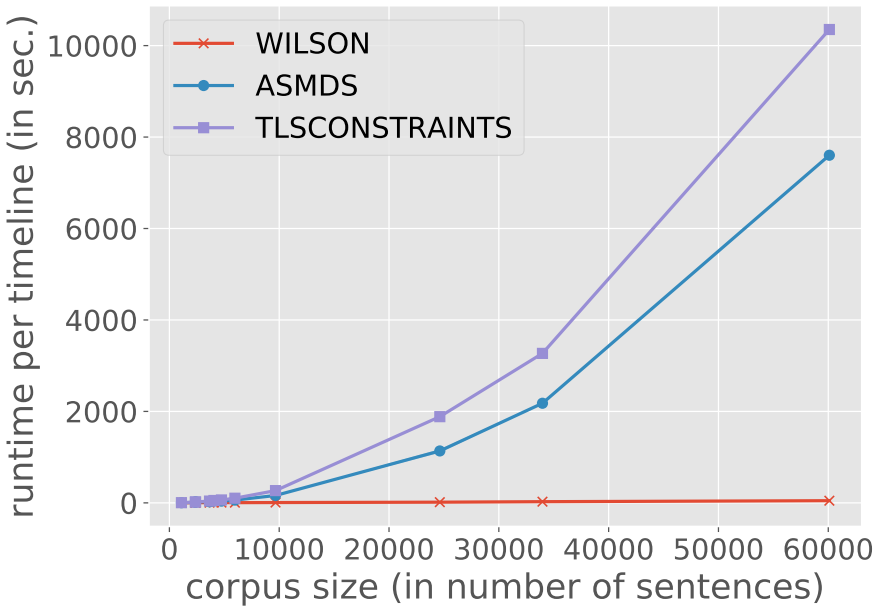


Figure 4.2: Running times over varying corpus sizes using *Timeline*17 and *Crisis*. Note that, as both ASMDS and TLSCONSTRAINTS are not sufficiently scalable to the large corpus, we followed [1] to use a reduced corpus here.

categories of news timeline generation methods. One is aimed at separating different stories from a whole news corpus, such as using variants of topic modeling [76, 77] and neural networks [78]. Another category focuses on generating a series of chronological summaries for one specific event from only relevant news articles [1, 74, 79], where the first categories can serve as pre-processing to find relevant news articles for each event. In this paper, our focus is on the latter category in an unsupervised manner.

However, the majority of existing methods focus only on the quality of generated timelines and neglect the generation speed. For example, the state-of-the-art unsupervised approach adopts the submodular framework [1] and requires the pairwise similarities for all tokenized sentences, which could be over 100,000 per timeline. This yields an extremely slow running times, as clearly demonstrated in the comparison of running times in Figure 4.2.

As the compression rates of timeline summarization vary with events and journalists may not know the exact value beforehand, iterative trials with different values are necessary, which makes faster timeline generation even more important. Therefore, in this work, we are greatly motivated by real industrial use cases to

speed up the timeline generation by dividing the whole summarization tasks into multiple small summarization tasks by date separation.

News timelines are composed of both salient dates and daily summaries, but previous studies mainly focus on modeling relationships among article contexts while paying less attention to date selection. For example, some models [75, 80, 81] just treat date information the same as text information and include it as one of the features, while others [71, 82] simply use date frequency to resolve events. Although simply modeling text correlation shows good performance on both timeline summarization and date selection [1], it is not clear how date selection will contribute to news timeline summarization. In addition, existing state-of-the-art unsupervised approaches mostly include global optimization, which helps daily summaries to be relevant to the topic. However, using global optimization also makes daily summaries less specific per each day and very time-intensive to generate timelines. Therefore, considering both the quality and speed of news timeline summarization, this paper makes the following main contributions:

1. We re-examine the role of date selection in timeline summarization and show that, even without considering contextual correlation across different dates, accurate date selection is sufficient to generate high-quality news timelines. More importantly, although ignoring contextual correlation across dates leads to a lower oracle upper bound than other models, all of the previous approaches still fail to reach this lower upper bound, not even close.

2. Leveraging on the explicit date selection, we propose a simple but fast and effective unsupervised news timeline summarization method, named as WILSON. Experimented on two widely used timeline summarization datasets, WILSON outperforms state-of-the-art approaches in both ROUGE scores and speed, significantly improving ROUGE-2 F1 score by 9.5%∼17.7% and reducing generation time by two orders of magnitude.

3. To our best knowledge, WILSON is the first work to include an evaluation by professional journalists in news timeline summarization. Through manually reviewing the machine-generated news timelines with corresponding human-generated ones, journalists confirm that our approach produces better timelines than competing methods.

4. Based on the proposed WILSON, we build a real-time news timeline summarization system on an industrial-level news corpus.

4.2 Related Works

Existing works in summarizing timelines for a specific topic from relevant news articles include both supervised and unsupervised approaches. As representatives of supervised approaches, [73] leverages learning to rank techniques based on sentence features, while [75] proposes a matrix factorization framework to predict importance scores of sentences. Unsupervised approaches usually optimize task-specific heuristic object functions, which measure relevance, coverage and diversity of daily summaries. For example, [79] solves the optimization problem by iteratively substituting sentences in summaries, while the state-of-the-art framework TILSE adapts the sub-modularity framework from multi-document summarization domains to optimize timeline summarization [1].

In addition to extractive methods, some recent works also utilize abstractive summarization methods to generate more compact sentences as daily summaries [82]. Although the generated sentences are empirically proved to be readable, the reliability of generated summaries is not guaranteed, probably leading to false information. Extractive summarization methods, however, directly borrow sentences from original news articles and do not encounter the reliability issue. Thus, in this paper, we utilize extractive summarization for both readability and reliability of generated timelines.

Besides ROUGE scores, [82] is the only existing work to include humans in the evaluation, but they just assess the readability of daily summaries as they utilize the abstractive summarization. Since none of the previous studies utilize user study to measure the generation quality of the whole news timelines, we are the first work to include user study in timeline evaluation and consult journalists to assess the generation quality of the entire news timelines.

4.3 The Proposed Method: WILSON

In this section, we introduce our proposed method, named as WILSON (neWs tImeLine SummarizatiON), also illustrated in Figure 4.3. Besides pre-processing

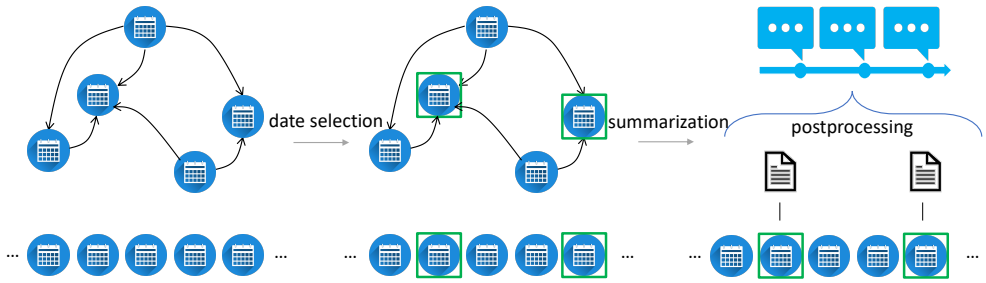


Figure 4.3: Workflow of our proposed method–WILSON.

such as temporal tagging and search engine indexing, WILSON mainly consists of two components – explicit date selection and text summarization for each selected date.

4.3.1 Problem Formulation

A news timeline can be viewed as a series of chronologically ordered daily summaries over main events, denoted by (d_i, S_i), where d_i and S_i stand for i_{th} date and i_{th} summary. For both readability and reliability of generated timelines, we utilize extractive summarization, which selects sentences from the corpus as summaries:

Definition 4.1 (News Timeline Summarization) *Given a corpus of articles C_q, which is associated with a topic query q and a time window $[t1, t2]$, the corpus is first tokenized to dated sentences $\{(date_i, sentence_i) | date_i \in [t1, t2], sentence_i \in C_q\}$ by a date expression in the sentence or by the publication date, then the timeline generation is to produce a series of daily summaries $(d_1, S_1), ..., (d_T, S_T)$, where $t1 \leq d_i \leq t2$ and $S_i = (sentence_{i1}, ..., sentence_{iN})$.*

The number of selected dates T and sentences N are hyper-parameters and chosen by users to control the compression rate of the generated timelines. Date selection is evaluated by f1 scores and summaries are evaluated by ROUGE [83].

4.3.2 Date Selection

We use HeidelTime [84] to tag temporal expressions in sentences during pre-processing stage and start with an unsupervised date selection algorithm [85] to select the most salient dates: (1) we build a date reference graph with dates as nodes

and reference relationships as edges; (2) then, we run the PageRank algorithm [86] on the graph and select the top T ranked nodes as the most salient dates. Date references refer to the sentences s_{ij} that are published on $date_i$ while mentioning $date_j$. We experiment with four types of edge weights as follows:

- W1: the number of reference sentences $|s_{ij}|$, assuming that more frequently reported events are more important.
- W2: temporal distance $|date_j - date_i|$ in days, supposing that only influential events are referred to over long-term period and assigning more weights to longer-term references.
- W3: W1 $*$ W2, which considers both frequency and temporal distance.
- W4: we adopt BM25 [87] to estimate the relevance of sentences to the query, and use $\max BM25(s_{ij}, q)$ as edge weight for each date reference.

For example, considering $date_i$=2018-06-01, $date_j$=2018-06-12, and s_{ij} composed of only two sentences, i.e. *Trump says summit with North Korea will take place on June 12* and *The summit will take place on June 12*. Then, W1 is the number of sentences and equals 2, while W2 is the difference between 2018-06-01 and 2018-06-12 in days and equals 11. Accordingly, W3 equals W1 $*$ W2 and is 22. For W4, we treat each sentence as a document, use topic query q to score each document with BM25, and take the maximum BM25 score as W4.

As Table 4.2 shows that all four edge weights yield comparable results, date reference relationship alone can extract as accurate date selections as topical information. Since constructing topical relationships across dates takes extra time, we finally adopt W3 as the edge weight to select the most salient dates in the rest of this paper.

Although the occurrence of an event signals its importance within the news timeline [71] and is well leveraged in existing timeline summarization algorithms, we note that the occurrence of events is also correlated with the recency of events, where past events occur earlier and are more heavily reported than recent events. Consequently, existing approaches may suffer from this issue. For example, approaches that optimize the summaries to recover the whole corpus, such as ETS [72] and TILSE [1], will generate more summaries on past events.

Edge Weight	Date F1	Rouge-1 F1	Rouge-2 F1
timeline17			
W1	0.5512	0.3905	0.0969
W2	0.5528	0.4029	0.1002
W3	0.5628	0.4009	0.0995
W4	0.5068	0.3934	0.0934
crisis			
W1	0.3022	0.3476	0.0715
W2	0.2838	0.3604	0.0715
W3	0.2710	0.3575	0.0738
W4	0.2925	0.3509	0.0726

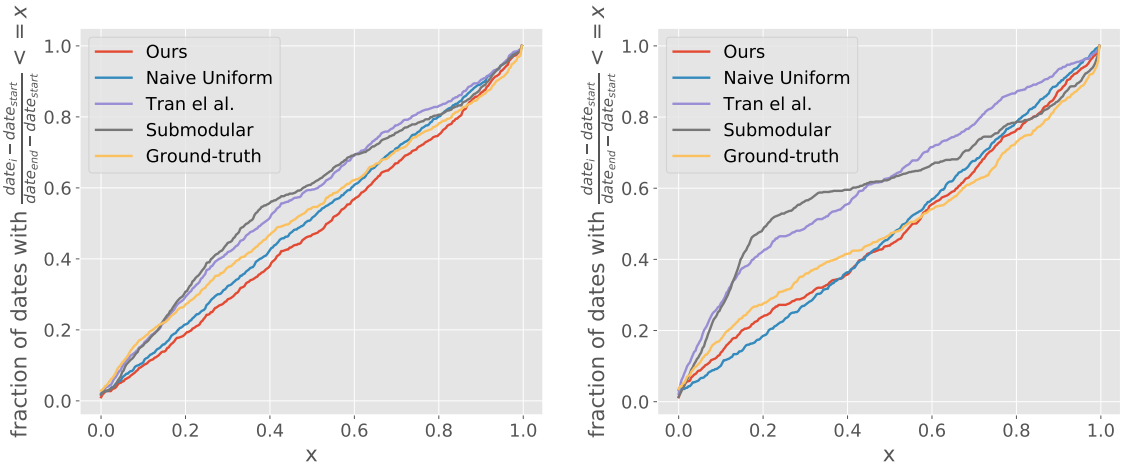
Table 4.2: Performance of different edge weights

Date Selection	Date Coverage (±3)	Date F1	ROUGE-1	ROUGE-2	ROUGE-S*
Timeline17					
Uniform	0.8398	0.4475	0.3896	0.0917	0.1598
W3	0.7828	0.5668	0.4000	0.0995	0.1676
W3 + Recency	0.8111	0.5542	0.4036	0.1005	0.1702
Crisis					
Uniform	0.5932	0.1325	0.3387	0.0570	0.1138
W3	0.5459	0.2726	0.3573	0.0738	0.1246
W3 + Recency	0.5885	0.2748	0.3597	0.0760	0.1270

Table 4.3: Performance on date coverage

In addition, as most references in articles refer to past events, the current date selection algorithm tends to give too much weight on old dates and will also result in timelines that lack recent dates. For a better illustration, we present the Cumulative Distribution Function (CDF) of the date duration between selected dates and the start date in Figure 4.4. As expected, both TILSE and date selection via PageRank tends to select more old dates, while the date distribution of ground-truth timelines is generally more uniform. Thus, we use the standard deviation of differences between consecutive dates to measure the uniformity of date distribution:

Definition 4.2 (Uniformity of Date Selection) *Given a series of selected dates* $\{d_1, d_2, ..., d_T\}$ *in chronological order, we regard the differences between two consecutive dates as* $\{diff_i = d_{i+1} - d_i\}$*, then we define its standard deviation* $\sigma = \sqrt{\frac{1}{T}\sum_{i=1}^{T}(diff_i - \bar{diff})^2}$*, as the uniformity of date selection.*



(a) Timeline17 (b) Crisis

Figure 4.4: Distribution of selected dates among different approaches.

4.3.2.1 Recency Adjustment

To add more weights on recent dates, we leverage the Personalized PageRank algorithm [88], where the restart distribution is not uniform. More specifically, we weight each date node $date_i$ by $W_i = \alpha^{-d_i}$, where $d_i = |date_i - date_{start}|$. $\alpha \in (0, 1)$ and is used to control the restart distribution. In practice, we use a grid search to find the α that gives the best uniformity score in the date selection, then use the chosen dates for news timeline generation.

4.3.2.2 Date Coverage

To better check the coverage of generated timelines, besides f1 score on date selection, we also measure the date coverage, e.g., if any day of ground-truth date $g_i \pm 3$ days lies in the generated timeline, g_i will be considered to be covered and we will measure what percentage of ground-truth dates are covered per timeline. For comparison, we also generate news timelines on truly uniformly distributed dates and present the results in Table 4.3. As we can see, although truly uniformly distributed dates cover the most ground-truth dates, due to the low accuracy in the date selection, the generated daily summaries are poor. However, adding recency adjustment with uniformity contributes to date selection in coverage, thus yields better timeline summarization.

4.3.3 Daily Summarization

Having selected the most salient dates, next, we divide timeline summarization into sub-summarization tasks. Although daily summarization tasks can be accomplished by any supervised or unsupervised document summarization algorithms, we utilize TextRank [89] to generate daily summaries for its efficiency. Similar to the task of date selection, TextRank constructs a sentence graph with sentences as nodes and similarity scores as edge weights. In particular, we use BM25 [87] to compute edge weights [90] and run PageRank on the directed graph to select the most important sentences as daily summaries.

4.3.3.1 Post-processing

Dividing large text summarization tasks into smaller ones greatly speeds up timeline generation, and these sub-tasks can naturally be further accelerated through parallel processing. Conducting text summarization on a daily basis rather than on the whole corpus, however, ignores temporal correlation and thus introduces redundancy in summarization. To remove redundancy across dates, therefore, we incorporate post-processing to re-rank daily summaries based on the whole summarization. Similar to MMR [91], instead of directly using daily summaries, we add sentences into timeline summarization by their daily ranks and only accept sentences whose maximum similarity with selected one (e.g., < 0.5).

4.3.4 Time Complexity Analysis

In this section, we briefly provide a time complexity analysis of our approach with a comparison to the submodular framework. Let T as the total number of dates, N as the average number of sentences per date, t as the desired number of dates, and n as the desired number of sentences per date in the summarized timeline. According to PageRank on the dense graph, date selection takes $O(T^2)$ while t daily summarization tasks take $O(t * N^2)$. Thus the time complexity of WILSON is $O(T^2 + t * N^2)$.

For submodular framework [1], which conducts global summarization, it takes $O((TN)^2)$ to obtain pair-wise similarities for all sentences and takes $O(t * n * T * N)$ to iterate $t*n$ times to select each individual sentence in a greedy manner. Therefore,

Dataset	# of topics	# of timelines	average per timeline		
			# of doc	# of sents	duration days
Timeline17	9	19	739	36,915	242
Crisis	4	22	5,130	173,761	388

Table 4.4: Dataset overview

the total time complexity is $O((TN)^2 + t * n * N * T)$. In Figure 4.2, the corpus size is defined as the total number of sentences (i.e. $T * N$). As expected, the submodular frameworks show quadratic growth with $O((TN)^2)$ time complexity, while our approach is almost linear to the corpus size with $O(T^2 + t * N^2)$ time complexity.

Given the approximation that T and N are in the same order of magnitude (based on Table 4.4), WILSON runs faster than the submodular framework in $O(\frac{T^2}{t})$. Given around 10% date compression rate ($\frac{t}{T}$) and T in hundreds, theoretically, our approach could gain over three orders of magnitude improvement in generation speed. Note that, due to the scalability issue of the submodular framework, [1] filtered sentences with predefined keywords to reduce N by over one order of magnitude, reducing the time complexity in practice. Given ∼10% filtering rate, our approach could still gain about two orders of magnitude in generation speed, which is consistent with experiments in Table 4.7.

4.4 Empirical Validation

4.4.1 Set-Up

4.4.1.1 Datasets

We run experiments on *timeline17* [73, 80] and *crisis* [74]. Both datasets[2] consist of journalist generated timelines from major news media such as CNN, BBC and Reuters, and a corresponding corpus of articles per topic (e.g. H1N1 flu and Egypt war). More specifically, *timeline17* contains 19 timelines from 9 topics, while *crisis* involves 22 timelines from 4 topics. An overview of the two datasets is shown at Table 4.4.

[2] `http://l3s.de/~gtran/timeline/`

4.4.1.2 Competing methods

- **Random**: The system generates daily summaries by randomly selecting sentences from the corpus.
- **MEAD** [92]: a classic centroid-based multi-document summarization system.
- **Chieu et al.** [71]: a multi-document summarization system that uses date related TFIDF scores to measure sentence importance among corpus.
- **ETS** [72]: an unsupervised timeline summarization algorithm via simultaneously optimizing multiple heuristic metrics, including relevance, coverage, coherence, and diversity.
- **Tran et al.** [73]: a supervised timeline summarization algorithm, which extracts various features from sentences and leverages learning to rank techniques.
- **Regression** [81]: a supervised approach that formulates sentence selection as a linear regression problem.
- **Wang et al.** [75]: a supervised approach that formulates sentence selection as a matrix factorization problem.
- **Liang et al.** [93]: a dynamic evolutionary framework leveraging distributed representation for timeline summarization.
- **ASMDS (TILSE)** [1]: TILSE is a state-of-the-art unsupervised timeline summarization approach, which incorporates submodularity-based multi-document summarization framework with temporal criteria.
- **TLSCONSTRAINTS (TILSE)** [1]: as a variant of TILSE, this method uses the same objective function with ASMDS but adopts different temporal constraints.

4.4.1.3 Measurement

Among all the baselines, TILSE is the only one with source code available. Consequently, for all the other baselines, we follow the existing works [73, 75, 93], which

Methods	ROUGE-1	ROUGE-2	ROUGE-S*
Random	0.128	0.021	0.026
Chieu et al.	0.202	0.037	0.041
MEAD	0.208	0.049	0.039
ETS	0.207	0.047	0.042
Tran et al.	0.230	0.053	0.050
Regression	0.303	0.078	0.081
Wang et al. (Text)	0.312	0.089	0.112
Wang et al. (Text + Vision)	0.331	0.091	0.115
Liang et al.	0.334	**0.105**	0.103
WILSON **(Ours)**	**0.370**	0.083	**0.141**

Table 4.5: Results on Timeline17

conduct experiments on *timeline*17 with settings mentioned at the beginning of Section 5.2 in [73] and directly report the baseline results from previous papers. More specifically, in the generated timeline, the number of selected dates T is set to the number of dates in each ground-truth timeline, while the number of sentences per day N is forced to be the rounded value of the average number of sentences per date from the ground-truth timeline.

To fairly compare with TILSE, we re-run the their code, follow all their pre-processing, including text cleaning and keywords filtering, and conduct experiments on exactly the same sentence corpus per timeline generation[3]. Wall time is measured on a 24-core 128GB machine.

4.4.1.4 Evaluation Metrics

The commonly used summarization metrics, ROUGE scores [83], including ROUGE-1, ROUGE-2 and ROUGE-S* F1 scores, are adopted to evaluate the agreement between machine-generated and journalist generated timelines. Moreover, to be consistent with TILSE comparison, we also include time-sensitive ROUGE scores as additional measurements [94].

[3]Different from previous papers, for Timeline17 dataset, [1] mixed articles of the same topic from different news agencies together, which yields a bit higher ROUGE scores in timeline generation. In addition, it suffers from the scalability issue and thereby uses filtered sentence corpus for both datasets. Thus, we followed their settings for a fair comparison with TILSE in Table 4.7.

Methods	ROUGE-1	ROUGE-2	ROUGE-S*
Regression	0.207	0.045	0.039
Wang et al. (Text)	0.211	0.046	0.040
Wang et al. (Text + Vision)	0.232	0.052	0.044
Liang et al.	0.268	0.057	0.054
WILSON **(Ours)**	**0.352**	**0.074**	**0.123**

Table 4.6: Results on Crisis

4.4.2 Performance Comparison

Table 4.5 and 4.6 shows that our unsupervised approach WILSON outperforms all baselines in ROUGE-1 and ROUGE-S* f1 scores by a significant margin, and is only second to [75], a supervised approach, and [93] in ROUGE-2 f1 score on Timeline17 dataset.

In addition, Table 4.7 illustrates that WILSON outperforms the state-of-the-art unsupervised framework TILSE in all ROUGE metrics. Averagely, our method outperforms the submodular approaches by 12.9% in concatenate ROUGE-2 scores, by 58.3% in agreement ROUGE-2 scores, and by 40.1% in alignment ROUGE-2 scores. More importantly, our method also gains two orders of magnitude improvement in generation speed, making it possible to generate news timelines in a real-time manner.

In Table 4.7, We also include multiple variants of WILSON for ablation analysis. WILSON-uniform simply adopts uniform date selection, while WILSON-Tran directly uses W3 as edge weight without recency adjustment. As expected, selecting uniformly distributed dates results in the worst summarization, while including recency adjustment improves time-sensitive ROUGE-2 scores by 9.0%$\sim$21.6%.

Overall, comparing with all competing approaches, the performance improvement of our method is higher in *Crisis* dataset. One explanation is that *Crisis* dataset contains more articles and spans a longer period, making it difficult for those competing approaches to correctly identify the long-term event dependencies, while our method mainly focuses on local dependencies.

	concat			agreement			align+ m:1			Date	Running Time
Model	Rouge 1	Rouge 2	our impr.	Rouge 1	Rouge 2	our impr.	Rouge 1	Rouge 2	our impr.	F1	Per timeline (sec.)
Timeline17											
ASMDS	0.3452	0.0890	13.8%	0.0913	0.0270	20.0%	0.1047	0.0299	17.1%	0.5437	338.68
TLSCONSTRAINTS	0.3685	0.0916	10.6%	0.0912	0.0242	33.9%	0.1049	0.0270	29.6%	0.5127	560.24
WILSON-uniform	0.3659	0.0848	19.5%	0.0754	0.0191	69.6%	0.0924	0.0218	60.6%	0.4366	**1.97**
WILSON-Tran	0.4007	0.0993	2.0%	0.1035	0.0293	10.6%	0.1181	0.0321	9.0%	**0.5668**	2.12
WILSON w/o Post	0.4036	0.1005	0.8%	0.1057	0.0318	1.9%	0.1202	0.0344	1.7%	0.5542	5.63
WILSON	**0.4075**	**0.1013**		**0.1065**	**0.0324**		**0.1211**	**0.0350**		0.5542	7.59
Crisis											
ASMDS	0.3066	0.0645	17.7%	0.0415	0.0091	123.1%	0.0658	0.0135	71.9%	0.2435	3055.96
TLSCONSTRAINTS	0.3307	0.0693	9.5%	0.0564	0.0130	56.2%	0.0764	0.0166	39.8%	0.2739	4098.07
WILSON-uniform	0.3314	0.0551	37.7%	0.0235	0.0059	244.1%	0.0392	0.0080	190.0%	0.1251	**4.68**
WILSON-Tran	0.3575	0.0739	2.7%	0.0621	0.0167	21.6%	0.0798	0.0202	14.9%	0.2726	5.69
WILSON w/o Post	0.3600	0.0756	0.4%	0.0677	0.0201	1.0%	0.0843	0.0230	0.9%	**0.2748**	22.95
WILSON	**0.3605**	**0.0759**		**0.0679**	**0.0203**		**0.0846**	**0.0232**		**0.2748**	30.14

Table 4.7: Comparison with TILSE. We also indicate our improvement on Rouge 2 f1 score for different metrics.

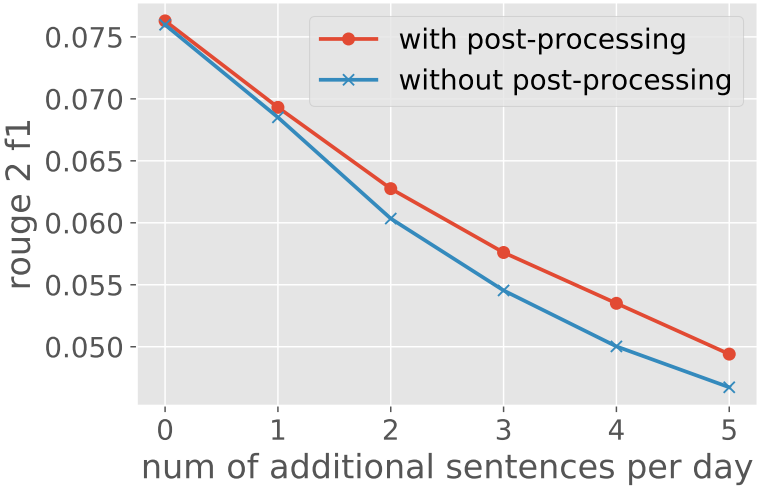


Figure 4.5: Concatenate Rouge 2 f1 scores when adding more sentences on each date on Crisis.

4.4.2.1 Effectiveness of Post-processing

In Table 4.7, we observed that considering correlation across different dates and reducing redundant daily summaries are seemingly minor, especially on *Crisis* datasets. Different from *Timeline17* datasets, *Crisis* datasets consist of more compact daily summaries, where more than 90% dates contain only 1 sentence. Although reducing redundancy across dates is not necessary for timelines with compact daily summaries, we intend to verify the effectiveness of post-processing for timelines with abundant daily summaries. Instead of using the exact number of sentences per date in the ground-truth timelines, we generate timelines with more sentences per date, which is more practical as the true numbers are unknown. As demonstrated in Figure 4.5, simply adding daily summaries together suffers from the redundancy issue and using post-processing indeed helps. Note that we use the ROUGE f1 score, so the overall scores going down with more sentences is because more generated texts lead to lower ROUGE accuracy.

4.4.2.2 Empirical Bounds

Empirical bounds of our two-stage method are given in Table 4.8, where we use ground-truth dates as date selections for daily summarization. Note that, besides using ground-truth dates, oracle bounds [1] also employ ground-truth summaries

Method	ROUGE-1	ROUGE-2
timeline17		
Oracle [1]	0.50	0.18
Ground-truth date + Daily summary	0.41	0.11
Crisis		
Oracle [1]	0.49	0.16
Ground-truth date + Daily summary	0.42	0.10

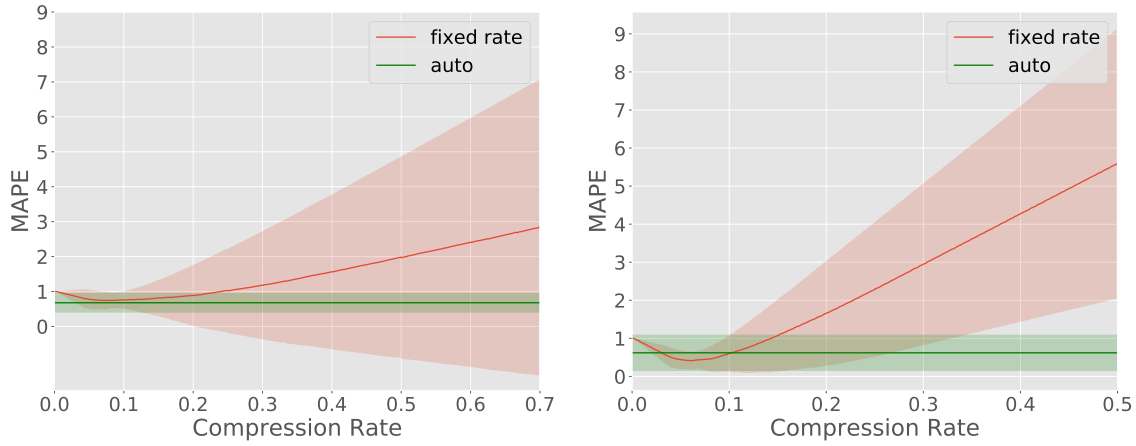
Table 4.8: Empirical bound of our two-stage method

and are calculated by directly optimizing ROUGE f1 scores in a supervised way, but we only use ground-truth dates and never touch ground-truth summaries, making us aware of how date selection will contribute to news timeline summarization. As demonstrated, even without considering contextual correlation across different dates in text summarization, it is still possible to generate reasonable news timelines with accurate date selection. Although the upper bound of our two-stage framework is much lower than that of the submodular framework, it is worth mentioning that all existing approaches fail to reach this upper bound, not even close on the *Crisis* dataset.

4.4.2.3 Automatic Date Compression

As defined in Section 4.3.1, existing news timeline summarization works only use a preset number of dates and length of daily summaries to generate news timelines. Unlike the length of daily summaries, which only implies the compression rate for a single day and is usually set to 2 or 3 sentences, determining the number of dates requires understanding for the whole corpus, making it difficult to select. To solve this issue, we intend to automatically detect the number of dates for news timelines. Motivated by the fact that news timelines consist of major events within the duration, we propose to consider major event coverage to determine the number of dates. Specifically, we use the daily summarization to generate major events for each date and encode daily summaries with BERT [95]. Then, we use Affinity Propagation [96] to cluster daily summaries into event clusters, and use the detected cluster number as date selection number.

We compared our methods with fixed compression rates for date selection and presented the results in 4.6. As shown, our automatic date compression method



(a) Timeline17 (b) Crisis

Figure 4.6: Mean Absolute Percentage Error (MAPE) of predicted number of date selection

generally performs well on both datasets.

4.4.3 Evaluation by Journalists

In addition to ROUGE scores, we also consult two professional journalists at one of the leading daily American newspapers, the Washington Post, to manually evaluate the quality of machine-generated news timelines. Among 41 timelines from the two datasets, we sample 10 timelines (20%) from 6 topics, including H1N1 flu, BP oil spill, Egypt crisis, Libra war, Yemen war, and Syria war. For each sampled timeline, we present the human-generated ground-truth timeline and three machine-generated timelines from ASMDS, TLSCONSTRAINTS, and WILSON (Ours) to journalists. The ground-truth timeline is labeled as a reference, while the other three are given in random order and the order is independent for each evaluation.

For each evaluation, the two journalists are asked to review $\sim$ 81 daily summaries from $\sim$ 51 distinct dates, which adds up to $\sim$ 810 daily summaries from $\sim$ 510 distinct dates in total, and work together to rank the three machine-generated timelines. To measure the ranking performance of each method, we adopt two common rank-aware measurements, Mean Reciprocal Rank (MRR) and Discounted Cumulative Gain (DCG), and present the results in Table 4.9. As shown, when evaluated by professional journalists, our method outperforms the state-of-the-art

Method	rank 1st	2nd	3rd	MRR	DCG
ASMDS	4	3	3	0.72	7.39
TLSCONSTRAINTS	1	6	3	0.56	6.29
WILSON (Ours)	**5**	1	4	**0.76**	**7.63**

Table 4.9: Results of journalist evaluation on the quality of machine-generated timelines

unsupervised framework TILSE again. Interestingly, although TLSCONSTRAINS generally achieves higher ROUGE scores than ASMDS in table 4.7, TLSCONSTRAINS receives unexpectedly lower rank scores than ASMDS in this evaluation by journalists. This may imply a warning that automated measures may not be enough for news timeline summarization and human evaluation could be beneficial at times.

4.4.4 A Case Study

In this section, we perform a qualitative analysis of the generated timelines of our approach. Since TILSE [1] is the only baseline with source code available, we also include its output in comparison. Table 4.11 presents a subset of timelines about the lawsuit of Micheal Jackson's death in the *Timeline17* dataset, where the manually generated timeline was collected from BBC[4]. As different approaches generate timelines with different date selections, we only consider the dates that appear in all 4 timelines and show the first a few dates and their summaries in chronological order. We highlight the overlaps between manually generated and automatic generated timelines in colors and observe that the output of our approach is aligned better with the handcrafted one.

Interestingly, more summaries of our outputs are closer to the main events on each date than those of TILSE's, though they are all relevant to this topic. We think it may be because more important daily events are reported more heavily on each date, while existing models try but fail to effectively capture the evolution clues across dates, thus simple daily summarization can work well. Apparently, how to balance local and global summarization and effectively capture event evolution

[4] `https://www.bbc.com/news/entertainment-arts-15060651`

2018-03-08 Jason Aldag The Post reports : North Korea 's belligerent leader , Kim Jong Un , has asked President Trump for talks and Trump has agreed to meet him " by May , ...	2018-04-01 CIA Director Mike Pompeo , left , and North Korean leader Kim Jong Un shake hands during a meeting in in Pyongyang , North Korea on Easter Weekend .
2018-04-16 The only way the United States can persuade North Korea to peacefully give up its pursuit of these weapons is if Kim believes Trump 's threat of military force is credible .	2018-04-20 7:30 a.m. Friday North Korea 's state media reports that leader Kim Jong Un has left Pyongyang for the North - South summit meeting with South Korean President Moon Jae - in .
2018-04-27 ... North Korean leader Kim Jong Un on Friday morning ... walking into South Korea for a historic summit with President Moon Jae - in that will lay the groundwork for a meeting between Kim and President Trump .	2018-05-09 Their release came as Secretary of State Mike Pompeo visited North Korea on Wednesday to finalize plans for a historic summit meeting between Trump and the North 's leader , Kim Jong Un .
2018-05-16 North Korea has taken repeated ... and threatening to scrap next month 's planned summit between Kim and U.S. President Donald Trump , saying it won't be unilaterally pressured into relinquishing its nuclear weapons .	2018-05-24 After weeks of receiving and even appearing to encourage chants of " Nobel " ahead of a planned historic meeting with North Korea dictator Kim Jong Un , President Trump on Thursday abruptly canceled the June 12 summit .
2018-06-05 ... Donald Trump cast his Tuesday summit with North Korea 's Kim Jong Un as a " one - time shot " for the autocratic leader to ditch his nuclear weapons and enter the community of nations ...	2018-06-12 President Trump said the U.S. will end its " war games " with South Korea after the historic summit with North Korean leader Kim Jong Un on June 12 .

Table 4.10: WILSON generated output of the timeline about how the U.S. and North Korea finally had the summit. Main coverage with journalist generated timeline is colored blue, and some trivial contexts are omitted by ellipsis for space.

could be one potential direction for news timeline summarization.

4.5 System Framework

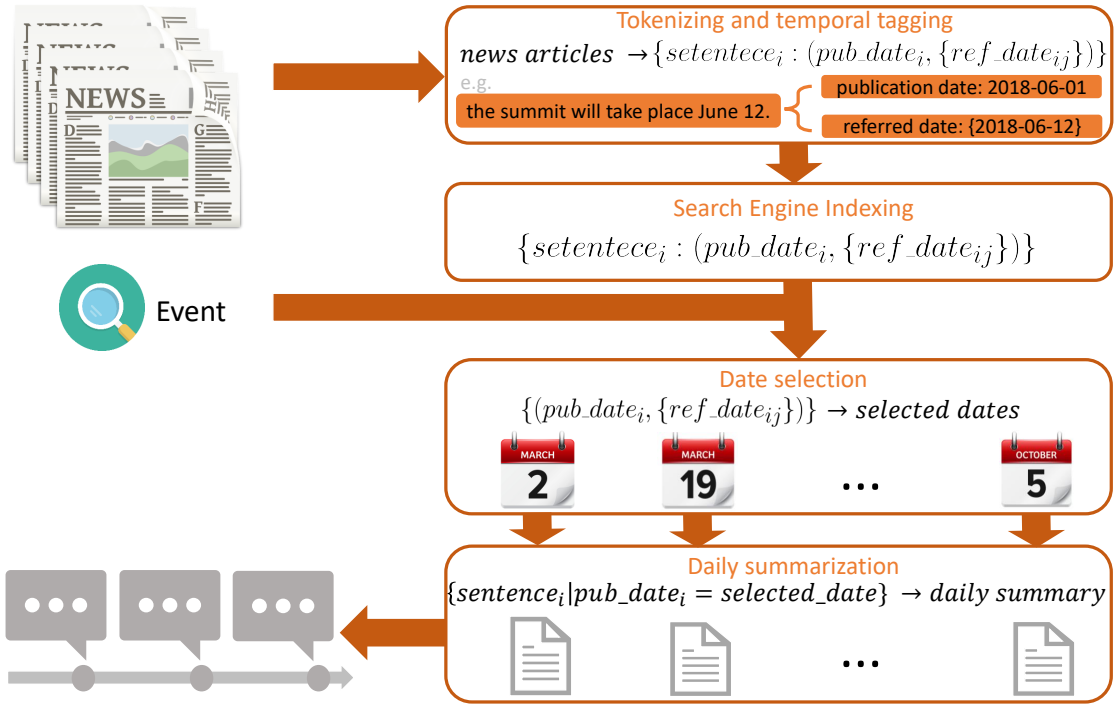


Figure 4.7: Framework for Real-Time News Timeline Summarization

To obtain a real-time news timeline summarization system, we build a search engine for tokenized news articles. Our system framework is demonstrated in Figure 4.7. In particular, we apply WILSON to 4-year news corpus of over 1 million articles[5] from the Washington Post, use ElasticSearch [97] as the backend search engine and build a real-time news timeline summarization system, which can generate timelines by event keywords in seconds. For example, we generate a timeline about how the United States and North Korea reached the summit by setting query keywords to "trump, north korea, kim, summit, united states" and time duration between 2018-01-02 to 2018-06-12. We set the timeline length to 10 and present the output in Table 4.10. Taking journalist generated timeline from the URL in Table 4.1 as a reference, note that our output is aligned well with journalist-generated version, demonstrating the effectiveness of WILSON.

[5]We excluded all news articles containing the keyword "timeline".

Groundtruth (From BBC)	TILSE (TLSCONSTRAINTS)	TILSE (ASMDS)	WILSON (Ours)
2009-06-25 Dr Murray finds Jackson unconscious in the bedroom of his Los Angeles mansion . Paramedics are called to the house while Dr Murray is performing CPR , according to a recording of the 911 emergency call . He travels with the singer in an ambulance to UCLA medical center where Jackson later dies .	2009-06-25 Jackson died at his Los Angeles home on 25 June aged 50 . Jackson died at his home on 25 June last year at the age of 50 .	2009-06-25 Michael Jackson died on 25 June 2009 from an overdose of the powerful anesthetic propofol .	2009-06-25 Same drug class as morphine Given by tablets or injection Used post-surgery or for childbirth High doses can stop breathing or lead to delirium and seizures Jackson , who had a history of health problems , collapsed at his Los Angeles home around midday on Thursday . Mr Martinez , who interviewed Dr Murray two days after Jackson 's death on 25 June 2009 , said the doctor told him the singer had stopped breathing shortly after 1100 .
2009-06-28 Los Angeles police interview Dr Murray for three hours . His spokeswoman insists he is " not a suspect " .	2009-06-28 Jackson 's body was released to the family on Friday night . Jackson 's body was released to the family on Friday night .	2009-06-28 Jackson family left ' speechless and devastated ' by star 's death Relatives of Michael Jackson will seek a second autopsy on the star because they still have unanswered questions about his death , family friends say .	2009-06-28 Michael Jackson 's family are said to be seeking a second autopsy because they still have questions about his death . Earlier , veteran politician Rev Jesse Jackson , who has been counselling the family , said they had a flurry of questions of their own for Dr Murray .
2009-07-28 Dr Murray 's home is also raided . The search warrant allows " authorised investigators to look for medical records relating to Michael Jackson and all of his reported aliases " . A computer hard drive and mobile phones are seized , and a pharmacy in Las Vegas is later raided in connection with the case .	2009-07-28 Dr Conrad Murray , who police say is not a suspect , was at Jackson 's mansion and tried to revive him before he died . Police raid Jackson doctor 's home Drug police are searching the Las Vegas home of Michael Jackson 's doctor as part of a manslaughter investigation into the singer 's death .	2009-07-28 On Tuesday , police searched the Las Vegas home and offices of Jackson 's doctor , Conrad Murray , as part of a manslaughter investigation into the singer 's death .	2009-07-28 Police raid Jackson doctor 's home Drug police are searching the Las Vegas home of Michael Jackson 's doctor as part of a manslaughter investigation into the singer 's death . On Tuesday , police searched the Las Vegas home and offices of Jackson 's doctor , Conrad Murray , as part of a manslaughter investigation into the singer 's death .
2010-06-25 Michael Jackson 's father , Joseph , files a wrongful death lawsuit against the physician .	2010-06-25 Randy Jackson recently succeeded in stopping an unapproved tribute show to his brother Michael in Rome , which had been scheduled for 25 June , the anniversary of his death . The suit was filed as fans around the world marked the first anniversary of Jackson 's death at the age of 50 .	2010-06-25 Jackson died of a cardiac arrest at his home on 25 June last year .	2010-06-25 25 June 2010 Michael Jackson 's father , Joseph , files a wrongful death lawsuit against the physician . Fans sing outside the Jackson family home .
2011-07-25 Rehearsal footage from Michael Jackson 's This Is It tour can not be used as evidence , the judge rules .	2011-07-25 Judge Michael Pastor concluded on Monday that it would not help the defense and that " it was a waste of my time . " 25 July 2011 Rehearsal footage from Michael Jackson 's	2011-07-25 But Judge Michael Pastor ruled on Monday that the film would not help the defense team and was a waste of his time .	2011-07-25 Judge Michael Pastor concluded on Monday that it would not help the defense and that " it was a waste of my time . 25 July 2011 Rehearsal footage from Michael Jackson 's
2011-08-30 Michael Jackson 's dermatologist is barred from giving evidence at the trial . Dr Murray 's lawyers had planned to argue that Arnold Klein had administered the singer with painkillers for " no valid reason " but prosecutors said they were attempting to transfer responsibility for his death away from Dr Murray . Testimony from five other doctors who treated Jackson is also disallowed .	2011-08-30 Janet Jackson to miss concert Janet Jackson said she would find it " difficult " to attend the tribute concert in Cardiff Janet Jackson will not be attending her brother Michael Jackson 's tribute concert in Cardiff . Because of the trial , the timing of this tribute to our brother would be too difficult for me , " Ms Jackson said in a statement .	2011-08-30 Janet Jackson to miss concert Janet Jackson said she would find it " difficult " to attend the tribute concert in Cardiff Janet Jackson will not be attending her brother Michael Jackson 's tribute concert in Cardiff .	2011-08-30 But Superior Court Judge Michael Pastor ruled that Arnold Klein would not be called to testify after prosecution lawyers said the defense wanted to transfer responsibility for Jackson 's death to the dermatologist . Because of the trial , the timing of this tribute to our brother would be too difficult for me , " Ms Jackson said in a statement .
2011-09-29 Jackson 's bodyguard , Alberto Alvarez , testifies that on the night Jackson died , Dr Murray ordered him to pick up vials of medicine before phoning for an ambulance . " In my personal experience , I believed Dr Murray had the best intentions for Mr Jackson , " Mr Alvarez said .	2011-09-29 29 September 2011 Last updated at 15:44 GMT Help Live coverage of the trial of Michael Jackson 's personal physician , Dr Conrad Murray , who is charged with involuntary manslaughter of the singer . 29 September 2011 Last updated at 04:16 GMT Help A key aide and a security guard have told the manslaughter trial of Michael Jackson 's doctor of events on the day the superstar died .	2011-09-29 However , Jermaine and Randy Jackson said it should not go ahead because it would clash with the trial of Conrad Murray , the singer 's former doctor accused of his involuntary manslaughter .	2011-09-29 Jackson 's bodyguard Alberto Alvarez claims Dr Murray " grabbed a handful of vials " and told him to put them in a bag Michael Jackson 's doctor told the performer 's bodyguard to pick up vials of medicine before phoning for help on the day he died , his trial has heard . 29 September 2011 Last updated at 04:16 GMT Help A key aide and a security guard have told the manslaughter trial of Michael Jackson 's doctor of events on the day the superstar died .

Table 4.11: Summary examples on the Death of Micheal Jackson. To save space, we only select the first a few dates in chronological order, which appear in all 4 timelines. Main coverage with groundtruth is colored: red texts highlight the overlaps between groundtruth and TILSE/ours while blue texts highlight the distinct overlaps between groundtruth and ours. Note that all summarization approaches use exactly the same sentence candidates pool.

4.6 Summary

This paper shows that, with accurate date selection, we can generate high-quality news timelines without considering the temporal correlation of text summarization. Leveraging the explicit date selection, we propose a fast and effective unsupervised timeline summarization method named WILSON. Specifically, WILSON outperforms state-of-the-art approaches in both ROUGE scores and speed, significantly improving concatenate ROUGE-2 F1 scores by 9.5%∼17.7%, time-sensitive ROUGE-2 F1 scores by 17.1%∼123.1% and reducing generation time by two orders of magnitude. More importantly, a user study with professional journalists also confirms that the outputs of WILSON are closer to human-generated ones than outputs of other methods. Last but not least, this work also suggests two potential directions for future works, i.e. considering both occurrence and recency of events for better salient date selection and reducing contextual correlation across dates by balancing local and global summarization to improve daily summarization.

Chapter 5 | Predicting Graph Evolution

5.1 Introduction

Graph structured data are ubiquitous in a wide range of domains, such as social networks [98], biology [99], and recommender systems [100]. Graph embedding, which aims to learn low dimensional vector representations of nodes that capture the network topological structures, has shown to be very effective for various downstream graph mining tasks such as link prediction [101, 102], community detection [103, 104], node classification [105, 106], and visualization [107, 108]. Due to its superior performance and wide applications, graph embedding has attracted increasing attention and many efforts have been taken [101, 105, 109–111]. For example, DeepWalk [105] introduces the idea of Skip-gram to learn node representations from random-walk sequences. Node2vec [101] extends DeepWalk with biased random walk in sampling sequences.

Many existing works on graph embedding mainly focus on static graphs. In real applications, however, graphs are naturally evolving as they are often generated by a series of temporal interactions between nodes. Therefore, ignoring such temporal information, including the regularities and abnormalities across the time dimension, is likely to yield imperfect models that may suffer severely from the generalization errors in prediction. For example, in a user-item graph, if a model does not incorporate users' changing interests and behaviors over time [100], it can introduce inaccurate recommendations. Similarly, if a model ignores the fact that items' semantic meaning could drift over time [112], it can result in false inferences. Therefore, to mitigate these issues, several dynamic graph embedding

algorithms have been proposed to preserve temporal evolution of *each individual node* [100, 113, 113–117]. These works can be mainly grouped into two categories: (i) for each timestamp t, for each node, learn an embedding and add a smoothness constraint between two consecutive embeddings of the same node [113, 114, 117–119]; and (ii) for each node, maintain one embedding, but update the embedding when new links are added using complex updating rules [100, 115, 116].

Despite the initial success of these dynamic graph embedding algorithms, they often suffer from the scalability issue in dealing with large-scale dynamic graphs. This is because existing embedding algorithms either require many parameters (i.e., one set of embedding for each graph snapshot) or have complex updating rules (i.e., update the embedding of each node when new links are added), which leads to large space and time complexity. However, large-scale temporal graphs are common and pervasive in real applications. For instance, the Facebook graph consists of over two billion user nodes and one trillion edges [120], while the Alibaba graph contains over one billion users and two billion items [121]. More importantly, these graphs are growing dramatically. For example, Facebook has over 2 billion monthly active users, with an average user clicking 13 adverts and making 6 comments per month [122]. Therefore, we need **scalable** embedding methods for **dynamic** graphs *with small space and time complexities*.

In many real-world dynamic graphs such as social networks, user preferences/representations can be divided into two parts, i.e., static preferences and temporal preferences. Static preferences come from a user's culture or identity which will not change. For example, food preference and dining habits are discovered to be formed at home and resistant to change once established [123]. Though temporal preferences change over time, however, user's temporal preference shift usually follow similar patterns. For example, when it is winter to spring, people usually go from skiing to a picnic. On Facebook, a seasonal event such as Thanksgiving and the Super Bowl can also trigger user behaviors in sharing and commenting relevant articles at a specific period. This observation suggests that we can learn one vector representation for each user as the static preference and capture the global temporal shift of the graph at each timestamp with another vector representation. An individual's temporal representation can then be projected to temporal space using the global temporal embedding. Exploring the temporal evolution of graphs can significantly reduce space and time complexity while simultaneously capturing

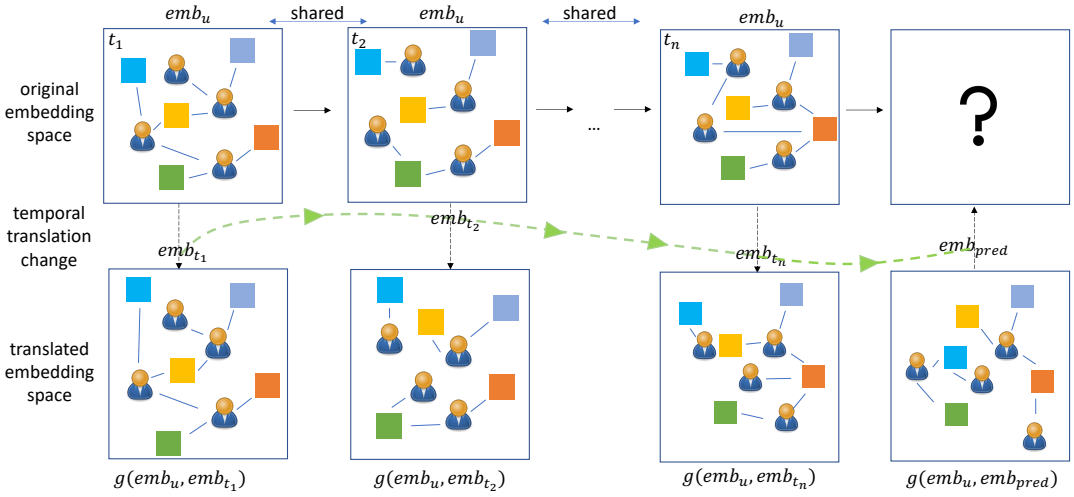


Figure 5.1: Upper: shared base embedding; Lower: time embeddings are introduced to translate node embeddings into corresponding temporal space. Network dynamics are captured by learning the temporal space change.

the temporal dynamics. However, to our best knowledge, no existing work uses this approach.

Therefore, in this chapter, we study a new problem of scalable temporal graph embedding by exploring static preferences and global evolution of graphs. In essence, we are faced with two challenges: (i) how to effectively capture static preferences of each individual and the global temporal dynamics of the graph; and (ii) how to make the algorithm scalable. In an attempt to solve these challenges, we propose a temporal translation-based model named MAGI to learn temporal graph embedding. An illustration of MAGI is shown in Figure 5.1. Instead of learning embedding for graphs at each timestamp or introducing complex updating rules for new links, to scalably encode time information and reduce parameters, we use shared base embedding for all the graph snapshots. For each graph snapshot, we introduce a time embedding (one vector) to translate node embeddings into corresponding temporal space. Meanwhile, we model the dependency of time embedding to capture the temporal dynamics. In addition, we capture the graph dynamics by incorporating a sequential model to learn the time embedding change and utilize the inferred time embedding for future prediction. The main contributions of the paper are:

- We study a new problem of scalable temporal graph embedding by exploring

the global temporal evolution of graphs;

- We propose a novel translation-based temporal graph embedding framework MAGI, which can effectively capture graph evolution and handle scalability simultaneously; and
- We conduct extensive experiments on real-world datasets to evaluate the effectiveness of MAGI in learning representations that capture temporal information and its scalability.

5.2 Related Works

In this section, we review related works, i.e., (i) static graph embedding, (ii) dynamic graph embedding, and (iii) translation-based model in the knowledge graph.

Static Graph Embedding. Early works of unsupervised graph representation learning exploits graph factorization [124, 125]. Inspired by the success of skip-gram model [126] in learning word embedding in Natural Language Processing (NLP), various approaches [101,105,107] have been proposed to treat graphs as "documents", use the random walk to sample node sequences as "sentences", and learn node embeddings with the skip-gram model. In addition, Graph Attention Network (GAT) [111] also introduces Attention [127] mechanism in graph embedding. Since Convolution Neural Networks (CNN) [128] are good at capturing local information and achieve huge success in Computer Vision, several convolutional neural network architectures for learning over graphs have been proposed [109, 129–132]. Graph Convolutional Network (GCN) [109] and GraphSAGE [110] take advantage of node attribute features and leverage attribute features of node neighbors to handle both transductive and inductive prediction. Besides accuracy, scalability is another challenge for graph embedding, thus PyTorch BigGraph [133] is presented as an industrial-level graph embedding system.

Dynamic Graph Embedding. Since many real-world graphs are dynamically changing, dynamic graph embedding approaches are proposed to capture the temporal information. The first a few works employ temporal regularization to smooth graph embeddings across different time snapshots [118, 119]. For example, The dynamic triad algorithm [119] considered triadic closure (i.e. the addition of a third edge among three nodes in the graph) as a reference to capture network

temporal dynamics, though it does not work on bipartite networks as it requires the presence of triads. Similar to static graph embedding, neural network techniques have been applied in dynamic graph embedding. DynGEM [114] uses autoencoder to connect two consecutive graph snapshots and learns incremental node embeddings through initialization from the previous time steps, but it may fail to capture long-term graph dependency. Then DyAERNN [113] is proposed to include a look back parameter to control the length of previous time steps in learning but increases training speed heavily. To model sequential change, JODIE [100] uses RNNs to model user and item embeddings, but it scales poorly and is only applied on bipartite networks. Both DySAT [117] and TGAT [134] leverages Graph Attention Network (GAT) and replace positional embedding with time embedding to encode time information. Besides structural attention, DySAT also applied attention to the temporal dimension, but it uses embeddings in the final snapshot for prediction and fails to predict future embeddings. The problem studied by GCN and GAT like approaches is orthogonal with ours, as those approaches mainly utilize node features while we only target at learning embeddings solely from graph structures. HTNE [115] models the neighborhood formation sequence of each node via multivariate Hawkes Process [135], while CTDNE [116] is the state-of-the-art random walk based graph embedding algorithm and uses temporally constrained random walks to generate one final node embeddings for prediction. Despite their STOA performance, many of them cannot deal with large-scale dynamic graphs. Thus, we investigate scalable temporal graph embedding.

Translation-Based Model for Knowledge Graph. To model multi-relational data, Translation-based models [136–139] in knowledge graphs embed triplets with different relationships into different affine spaces. For example, TransE [138] model uses addition translation to make the head entity and tail entity close in corresponding relation space. As time is intrinsically correlated across different graph snapshots, we introduce a translation-based dynamic graph embedding framework to translate different graph snapshots into different embedding space. But different from relations in knowledge graphs, where time is continuous, we propose to model the change of time embedding space and predict the next time embedding space for future tasks.

5.3 Problem Definition

We define the dynamic graph and dynamic graph embedding problem as follows:

Dynamic Graph: A dynamic graph is defined as a series of observed static graph snapshots, i.e. $G = (V, \mathscr{E}_T)$, where $\mathscr{E}_T = (E_1, ...E_T)$ and T is the total number of time snapshots. V is a set of vertices, and each temporal edge $(u, v, t) \in E_t$ is connection between vertice u and vertice v at timestamp t. With a focus on graph structural information without node features, we only examine the representation for nodes that have been observed in training and assume the node set V stay the same.

Dynamic Graph Embedding: Given a temporal network $G = (V, \mathscr{E}_{T<=t})$, the goal is to learn a function $f : V \to \mathbb{R}^D$ that maps vertices in G to D-dimensional feature representations, which can capture temporal evolution of networks and are suitable for temporal link prediction at $G = (V, \mathscr{E}_{T>t})$.

Our goal is to develop a scalable dynamic graph embedding framework that can handle large-scale temporal networks.

5.4 Proposed Method - MAGI

An illustration of the proposed framework MAGI is shown in Figure 5.2. It is composed of three parts: (i) the basic shared embedding of all graph snapshots to capture the intrinsic stationary properties of nodes in the graph; (ii) time embedding which captures the global temporal graph evolution to translate each graph snapshot into its corresponding time space; and (iii) sequential modeling to capture model the dependencies of time embedding. Next, we give the details of each component.

5.4.1 Basic Graph Embedding

Though the graph is dynamically changing, so as the nodes latent properties, however, there are some intrinsic properties of the nodes that will not change over time. For example, compared with users' interest in movies, their demographics such as gender and nationality are more stable. Thus, we assume that all the graph snapshots share the basic stationary graph embedding, which captures the

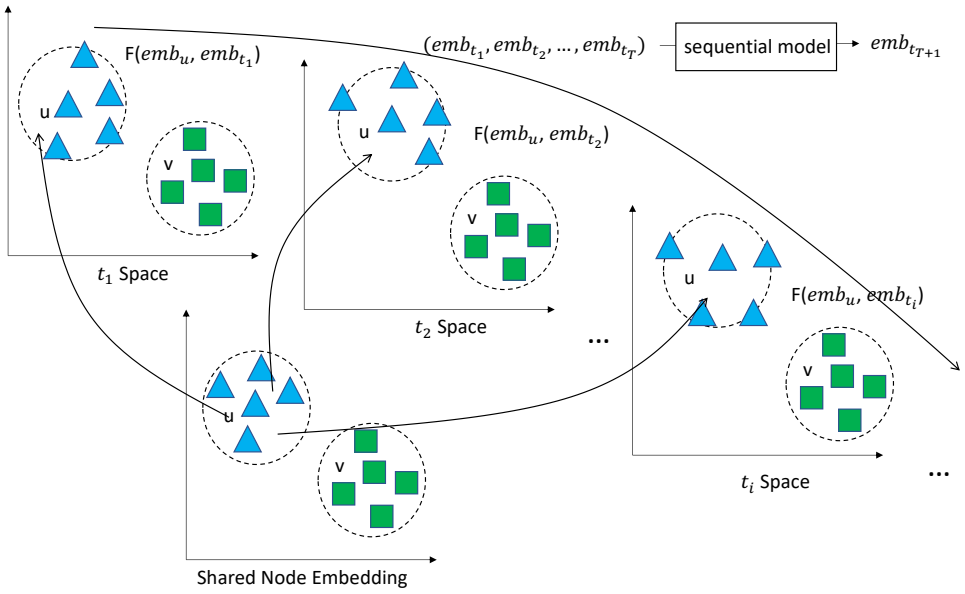


Figure 5.2: Framework

static preference of nodes over time. Since our goal is to develop scalable temporal graph embedding, we adopt PBG [133] as our basic graph representation learning approach as it is demonstrated to be very effective and scalable to large-scale static graphs. Specifically, for each edge $e = (u, v)$ in the graph, it samples a set of negative edges e' obtained by corrupting edges with either a sampled source or destination node, i.e., $S_e = \{(u, v') \notin \mathcal{E} | v' \in V\} \cup \{(u', v) \notin \mathcal{E} | u' \in V\}$. It then enforces a node more similar to its positive samples than negative samples with margin-based ranking loss as

$$\sum_{e \in G} \sum_{e' \in S_e} max(\lambda - f(e) + f(e'), 0) \tag{5.1}$$

where $f(e) = f(emb_u, emb_v)$ with f being the similarity measurement such as cosine similarity and emb_u and emb_v as the node embeddings of node u and v. G is the original graph, V is the node set and $\lambda > 0$ is the margin hyperparameter.

5.4.2 Time Embedding

The basic graph embedding captures the static preference of nodes. However, in dynamic graphs, node preferences are also changing. Instead of learning updating

rules/representations for each individual, which is very inefficient in space and time complexity to encode large-scale temporal graph, for each snapshot G_t, we introduce a time embedding vector emb_t to project the node embedding to the time space. The time embedding emb_t can be treated as capturing the global preference shift of nodes in G_t. For example, suppose the season shift from Autumn to Winter, user preferences shift globally to winter related activities such as skiing or Christmas movies. Thus, the time embedding is a reasonable assumption.

Specifically, for G_t, instead of directly using node embeddings, emb_u and emb_v, to measure the similarity between nodes, emb_t is included to encode emb_u and emb_v, such that u and v are more close to each other in the temporal embedding space t rather than in any other spaces. Node embeddings are shared across temporal embedding spaces and serve as alignments, while time embedding captures the node characteristic change over time, allowing us to obtain the most recent node features in the most recent temporal space. Besides, through translating node embeddings from each graph snapshot into corresponding temporal space, we can process all graph snapshots simultaneously while still preserving the temporal constraint in learning graph evolution, which largely reduces training time in encoding temporal information.

The interaction between node embedding and time embedding can be denoted as $g(emb_u, emb_t)$. As the additive model is proven to be effective in other embedding problems, such as TransE knowledge graph [138] and Positional Embedding in attention models [127], we also start with an additive formula – MAGI-add:

$$g(emb_u, emb_t) = emb_u + emb_t \tag{5.2}$$

Motivated by the fact that embeddings are usually normalized for the training stability and only direction matters in this situation, we propose a novel direction based time encoding formula to ignore the effect of the magnitude of embeddings and only tweak the direction of node embedding in each temporal space – MAGI-dir:

$$g(emb_u, emb_t) = \frac{emb_u}{\|emb_u\|_2} + \alpha \cdot \frac{emb_t}{\|emb_t\|_2}) \tag{5.3}$$

α is expected to be larger than 1, allowing an arbitrary change in the direction for different temporal spaces. In this work, we set α to 2.0 as a hyperparameter. We

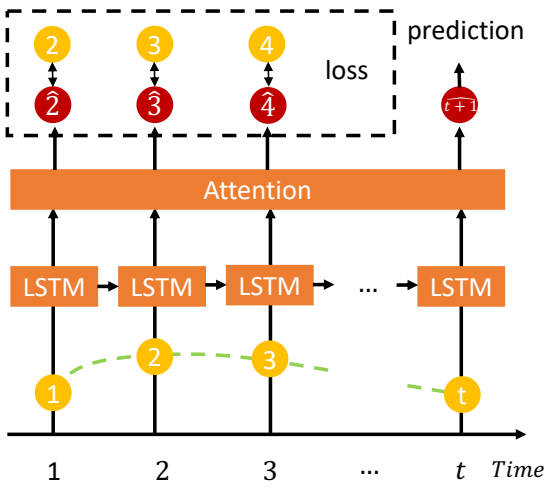


Figure 5.3: Sequential modeling of temporal embedding space

also experiment with various α values for parameter sensitivity analysis in Section 5.5.1.7.

For directed graphs, we adopt *asymmetric* temporal translation for each edge by only incorporating time embedding in one side. For example, to measure the similarity between two nodes in each edge, either $f(g(emb_u, emb_t), emb_v))$ or $f(emb_u, g(emb_v, emb_t))$ is used. This setting is generalized to undirected graphs by viewing each undirected edge as two directed edges. With time embedding, the loss function in Eq.(5.1) is rewritten as:

$$loss_{graph} = \sum_{(u,v,t)\in G} \sum_{v'\in S_u} max(\lambda - f(u,v,t) + f(u,v',t), 0) \tag{5.4}$$

where $f(u,v,t) = \cos(g(emb_u, emb_t), emb_v))$ and S_u is the set of sampled negative nodes.

5.4.3 Sequential Modeling of Time Embedding

In a temporal graph, due to the temporal dependencies between interactions across different nodes, it is necessary to maintain temporal order of edges during training, such that all edges are processed in chronological order, making sequential modeling of individual node embedding change very expensive, even if optimizing training batches with topological sorting [100]. To preserve this temporal constraint but also tackle scalability issues in training, we propose to model the global

temporal evolution of graphs as a whole rather than modeling each individual node embedding change separately. In this way, we are able to train all node embeddings simultaneously by embedding each graph snapshot into corresponding temporal space, and then align all temporal spaces together via learning the evolution pattern of the time embeddings.

Since time embedding captures the temporal information, we use a Long Short-Term Memory (LSTM) [140] autoencoder with a temporal attention mechanism to model the evolution of the temporal embedding spaces. An illustration of the sequential modeling framework is shown in Figure 5.3. Specifically, at time step i, we use emb_i as input to predict the next time embedding $\widehat{emb}_{i+1}$ as:

$$\begin{aligned} \widehat{emb}_{i+1} &= W_e \cdot [c_i || h_i], \\ c_i &= \sum_{j=2}^{T} att_{ij} \cdot h_j, \\ h_i &= LSTM(emb_i, h_{i-1}), \\ att_{ij} &= \frac{exp(h_i^T W_a h_j)}{\sum_k exp(h_i^T W_a h_k)} \end{aligned} \tag{5.5}$$

where $||$ denotes concatenation operation and att_{ij} are the attention weights. The context vector c_i is a weighted sum of the hidden states. Instead of directly using context vector c_i, we concatenate the context vector with the corresponding hidden state h_i to do prediction at each snapshot. Since each hidden state h_i has already encoded neighboring information before each snapshot by LSTM, we expect this concatenation to encode information beyond neighboring snapshots, and the findings are in Section 5.5.3.2. The loss function is given as

$$loss_{time} = -\frac{1}{T-1} \sum_{i=2}^{T} \cos(\widehat{emb}_i, emb_i) \tag{5.6}$$

5.4.4 Objective Function of MAGI

With the basic graph representation capturing the static node properties, time embedding projecting the static node embedding to temporal space and LSTM for capturing the temporal smoothness, the final objective function of MAGI is given as

$$\arg\min_{\theta} loss_{graph} + \beta \; loss_{time} \tag{5.7}$$

Algorithm 5.1: Training Algorithm for MAGI

Input : the temporal network $G = (V, \mathscr{E}_{T<=t})$
Output : $\theta = \{emb_{t\in\{1..T\}}, emb_{v\in V}, W_a, W_e, \theta_{LSTM}\}$

Initialize parameters θ ;
Randomly shuffle edges (u, v, t) and prepare batches ;
repeat
  for *each batch* **do**
    Sample a set of negative nodes S_u for each edge ;
    Use Eq. 5.4 and Eq. 5.6 for *forward propagation* ;
    Optimize Eq. 5.7 in *backpropagation* to get updated parameters θ ;
  end for
  Use Eq. 5.5 with $emb_{t\in\{1..T\}}$ to obtain the predicted time embedding $\widehat{emb}_{T+1}$;
  Based on $\widehat{emb}_{T+1}$ and $emb_{v\in V}$, apply Eq. 5.2 or Eq. 5.3 to get translated node embedding $\widetilde{emb}_{v\in V}$;
  Evaluate $\widetilde{emb}_{v\in V}$ on validation set.
until *iteration limit or early stop criterion is reached*;
return θ

where β is the hyperparameter that balances the graph reconstruction and temporal space modeling, and θ is the parameter set $\{emb_{t\in\{1..T\}}, emb_{v\in V}, W_a, W_e, \theta_{LSTM}\}$.

Existing dynamic graph embedding algorithms, such as CTDNE [116] and DySAT [117], usually use node embeddings of the final snapshot for prediction. However, with the help of temporal space modeling, we are able to predict the time space for the next graph snapshot, and demonstrate that the predicted time space is better at representing the graph in the newest state, signaling the success in capturing the temporal evolution patterns of graphs.

5.4.5 Training Algorithm

A training algorithm is summarized in Algorithm 5.1. First, we initialize parameters. Specifically, node embeddings $emb_{v\in V}$ are initialized by tensors filled with random numbers from standard normal distribution $\mathcal{N}(0, 1)$, and time embeddings $emb_{t\in\{1..T\}}$ are initialized by tensors filled with 0 in MAGI-add and 0.1 in MAGI-dir respectively. Then, we randomly shuffle edges and process edges in batches. For each batch, we optimize the overall objective function as in Eq. 5.7 to update parameters. After each iteration, we predict the time embedding in future graph

snapshot with Eq. 5.5 and use it to obtain translated node embedding for validation and test.

5.4.6 Space and Time Complexity Analysis

In this subsection, we analyze space and time complexity of MAGI.

Space Complexity. Let the embedding dimension be D, the number of graph snapshots be T and the total number of nodes be N, our MAGI model introduces an extra space of $O(T \cdot D)$ time embeddings, $O(4(D \cdot D + D \cdot D + D))$ LSTM and $O(D \cdot D + 2D \cdot D)$ attention module, which is $O((T + 11D) \cdot D)$ in total. Considering that there are at least $O(N \cdot D)$ node embeddings and $N >> T$, the extra space is negligible.

Time Complexity. The extra time complexity comes from both temporal space translation and sequential modeling. The time complexity of LSTM per weight and time step is $O(1)$ [140] and the attention computation takes $O(T \cdot D \cdot D + 2D \cdot D)$ per time step. Therefore, the total time complexity of introducing the sequential model is $O(T \cdot (4(D \cdot D + D \cdot D + D) + T \cdot D \cdot D + 2D \cdot D)) = O((T \cdot D)^2)$, which is only relevant to the number of graph snapshots and invariant to the graph size. Since space translation only involves in edge computation and each edge computation $f(u, v, t)$ takes at least $O(D)$ time, space translation only introduces constant extra complexity in edge computation with $O(D)$ time in each edge computation. Thus, theoretically, our MAGI model remains the same time complexity with the static graph embedding model, and we also empirically validate this in Section 5.5.2.

5.5 Experiments

In this section, we conduct experiments to validate both the performance and scalability of our proposed model. Specifically, we intend to answer the following evaluation questions:

- **EQ1**: Is MAGI able to capture the graph evolution by introducing temporal translation and modeling the evolution of temporal spaces?
- **EQ2**: How efficient is MAGI in learning node representations for large temporal graphs?

Dataset	#users	#item	#edges	duration
lastfm	961	79,733	2,424,043	2005-02 to 2009-02
movielens	16,835	5,090	2,951,543	2004-02 to 2014-02
yelp	95,038	53,137	2,226,369	2010-01 to 2018-01

Table 5.1: Bipartite network datasets

- **EQ3**: What information can we gain from the temporal space translation through the learned time embedding?

5.5.1 Link Prediction

To validate the effectiveness of temporal translation (**EQ1**), we experiment with 3 large bipartite temporal graphs, which contain up to 140K nodes and over 2 million edges per network, and set the duration of each graph snapshot to a month. Targeting at industrial-level graphs, instead of using a downstream link prediction setting, we directly use the similarity scores between node embeddings as metrics for link prediction, which can be easily tackled by Approximate Nearest Neighbor approaches in practice.

5.5.1.1 Dataset

We select the following large temporal graph datasets in link prediction experiments:

1. **lastfm-1k** [141]: this dataset contains 4-year of who-listens-to-which song information from 961 users, yielding a temporal graph with $\sim 80,000$ nodes and about 2.5 million edges.

2. **movielens-20m** [142]: we only keep positive ratings (≥ 4), and use the data from active users and popular movies in a 10-year duration, leading to a temporal graph with $\sim 22,000$ nodes and ~ 3 million edges.

3. **yelp** [143]: we extract an 8-year dataset from Yelp Challenge Round 13, and construct a temporal graph with $\sim 150,000$ nodes and around 2.2 million edges.

We use 10-core setting for *lastfm* and *yelp*, and 100-core for *movielens*. Using monthly graph snapshots, our datasets contain 49 to 121 time steps per graph. The detailed statistics are provided in Table 5.1.

5.5.1.2 Competing Methods

We include state-of-the-art graph embedding methods that solely relies on graph structural information in both static and dynamic settings as baselines in our experiments:

Static embedding methods:

- DeepWalk [105]: This method is the first to apply random walks in generating node sequences from the network and then uses them as input to a skip-gram model to learn node representations.
- Node2Vec [101]: This work extends DeepWalk by balancing local and global properties in the random walk to sample node sequences.
- LINE [107]: LINE optimizes node representations by preserving both first-order and second-order proximities for a network.
- PBG [133]: PyTorch-BigGraph utilizes the efficient negative sampling strategies and distributed processing techniques to learn graph embedding on a large scale.

Dynamic embedding methods:

- HTNE [115]: this method uses multivariate Hawkes processes [135] to model the neighborhood formation sequence of each node, where each edge is weighted by a time decay factor. We use suggested parameters in the original paper and adopt the implementation from the authors[1].
- DynAERNN [113]: this model is based on a fully connected auto-encoders to incrementally generate the embedding of the current graph with the initialization from the previous graph and feeds the hidden states into LSTMs to learn network dynamics. We use suggested parameters in the original paper and adopt the implementation from the authors[2].

Other recent dynamic embedding methods include CTDNE [116], tNodeEmbed [144] and DySAT [117]. However, the existing implementation from the authors of

[1] http://zuoyuan.github.io/files/htne.zip
[2] https://github.com/palash1992/DynamicGEM

tNodeEmbed[3] is not reproducible, while DySAT[4] scales poorly and requires more than 256GB memory on our datasets, so we exclude them. And we also remove CTDNE[5], as it is not scalable in time and takes more than 1 day in sampling temporal walks on large graphs.

5.5.1.3 Set-Up

We use the temporal link prediction task to evaluate the performance of our proposed method. We reserve the latest 12 months of each dataset for test on a monthly basis and thereby have 12 tests for each dataset. Particularly, for each test, we use i_{th} month as test set, $i-1_{th}$ month as validation set, and all data before the $i-1_{th}$ month as training set. For validation and test, we only consider link prediction on the observed nodes and remove edges that contain unseen nodes in the training set.

Evaluation Metrics. We measure the performance of the methods in terms of the Mean Reciprocal Rank (MRR) and Recall@K (K=1, 10, 50). For each method, we evaluate the MRR on the validation set after each epoch and save the best model in validation for test performance. In addition, we report raw ranking metrics, which use all candidate nodes for ranking, and do not filter out existing edges in the training, as dynamic graph embedding models are expected to distinguish the time of edges by themselves. For PBG and MAGI, we use cosine similarity between node embeddings as ranking scores. As for other methods, we experiment with both cosine similarity and negative euclidean distance in ranking and report the best scores.

5.5.1.4 Implementation Details

For fair comparisons, we set the node embedding dimension to 128 for all approaches. And we use suggested parameters in original papers for both HTNE [115] and DynAERNN [113]. Observing that HTNE takes 5 to 15 minutes per epoch in training, we set the maximum epoch number to 20. However, since DynAERNN takes about 3 hours per epoch in training large datasets like *lastfm* and we have 12 tests for each dataset, we only train DynAERNN for 1 epoch. In addition, we ignore

[3]https://github.com/urielsinger/tNodeEmbed
[4]https://github.com/aravindsankar28/DySAT
[5]https://github.com/urielsinger/CTDNE

the performance of DynAERNN on *yelp*, as it requires more than 256GB memory in training *yelp* dataset. For the sake of speed, we adopt GraphVite [145], a high-speed graph embedding engine, to generate embeddings for DeepWalk, Node2Vec, and LINE. Node2Vec is trained under p = 0.8 and q = 1. As for PBG and MAGI, we set margin hyperparameter $\lambda = 0.2$, use single side negative sampling (on item side), and set in-batch negative sample and uniform negative sample number to 50 and 1000 respectively.

5.5.1.5 Results

Results are shown in Table 5.2, demonstrating the state-of-the-art performances of our approach on all three datasets. As shown, all static embedding methods gain comparable performance in temporal link prediction. Using temporal information, HTNE outperforms static embedding methods on *lastfm*. As DynAERNN processes graph snapshots in order and only looks at a very small number of graph snapshots at each time, it could suffer from catastrophic forgetting issue on capturing node preference and thus perform poorly on large temporal graphs such as *movielens*, which contains over 100 graph snapshots. Due to the time cost, DynAERNN and HTNE are trained by limited epochs, which might explain their worse performance than static embedding methods on *movielens* and *yelp*. This also suggests the importance of developing scalable and efficient dynamic graph embedding methods for large temporal graphs. Besides MAGI full model (MAGI-dir + sequential modeling), we also include MAGI-dir and MAGI-add for ablation analysis, which learn time embedding in temporal translation without sequential modeling. By introducing the time embedding to capture graph evolution and handle scalability simultaneously, all translation-based methods achieve much higher MRR scores than baseline methods across all graphs. Notably, direction based translation model MAGI-dir outperforms additive translation model MAGI-add on *lastfm* and *yelp*, and gains comparable results with it on *movielens*. More importantly, using the sequential model to encode the evolution of temporal spaces significantly improves MAGI-dir by 5% on average, indicating the success in predicting graph evolution.

Methods	**MRR**	Our impr.(%)	**Recall@1**	Our impr.(%)	**Recall@10**	Our impr.(%)	**Recall@50**	Our impr.(%)
				lastfm				
DeepWalk	0.0005464***	97.1%	0.0000182***	831%	0.0003786***	295%	0.0028599***	130%
LINE	0.0005369***	101%	0.0000127***	1235%	0.0003501***	327%	0.0027881***	136%
Node2Vec	0.0005744***	87.5%	0.0000264***	541%	0.0003958***	278%	0.0030993***	112%
PBG	0.0006288***	71.2%	0.0000560***	202%	0.0006448***	132%	0.0037417***	75.6%
DynAERNN	0.0005224***	106%	0.0000636***	166%	0.0006457***	132%	0.0030372***	116%
HTNE	0.0006738***	59.8%	0.0000694***	144%	0.0008082***	85.1%	0.0038969***	68.6%
MAGI-add	0.0007491***	43.7%	0.0000832**	104%	0.0008646***	73.0%	0.0043727***	50.3%
MAGI-dir	0.0010178***	5.79%	0.0001210**	39.9%	0.0013913*	7.51%	0.0062903*	4.47%
MAGI-dir + seq.	**0.0010767**		**0.0001694**		**0.0014958**		**0.0065715**	
				movielens				
DeepWalk	0.0070316***	237%	0.0010634***	464%	0.0111902***	335%	0.0522821***	198%
LINE	0.0072236***	228%	0.0012460***	381%	0.0112955***	331%	0.0528325***	195%
Node2Vec	0.0071524***	231%	0.0011680***	414%	0.0117020***	316%	0.0530705***	194%
PBG	0.0073374***	223%	0.0008356***	618%	0.0097438***	399%	0.0535727***	191%
DynAERNN	0.0011938***	1885%	0.0001443***	4059%	0.0014725***	3203%	0.0060582***	2473%
HTNE	0.0059516***	298%	0.0006660***	801%	0.0089292***	445%	0.0450091***	246%
MAGI-add	0.0228770	3.58%	**0.0063108**	-4.93%	0.0464062	4.87%	0.1462500*	6.57%
MAGI-dir	0.0222721	6.39%	0.0052925	13.4%	0.0454575*	7.06%	0.1529562	1.90%
MAGI-dir + seq.	**0.0236956**		0.0059995		**0.0486661**		**0.1558553**	
				yelp				
DeepWalk	0.0065413***	102%	0.0004109***	579%	0.0103344***	144%	0.0632411***	54.9%
LINE	0.0065578***	101%	0.0004556***	513%	0.0102442***	146%	0.0632691***	54.8%
Node2Vec	0.0065303***	102%	0.0004694***	495%	0.0102463***	146%	0.0625283***	56.6%
PBG	0.0058132***	127%	0.0007676***	264%	0.0092574***	172%	0.0489771***	100%
DynAERNN	-	-	-	-	-	-	-	-
HTNE	0.0057244***	131%	0.0009791***	185%	0.0099769***	152%	0.0416965***	135%
MAGI-add	0.0075502***	74.8%	0.0010566***	164%	0.0129932***	93.7%	0.0624995***	56.7%
MAGI-dir	0.0127802***	3.25%	0.0024665**	13.2%	0.0245117*	2.68%	0.0973539	0.60%
MAGI-dir + seq.	**0.0131950**		**0.0027912**		**0.0251682**		**0.0979411**	

Table 5.2: Performance of different approaches on three large bipartite temporal graphs. We do paired t-test between each approach with MAGI full model (MAGI-dir + sequential modeling) and indicate the significance by *, ** and *** for p-value < 0.05, 0.01 and 0.001 respectively.

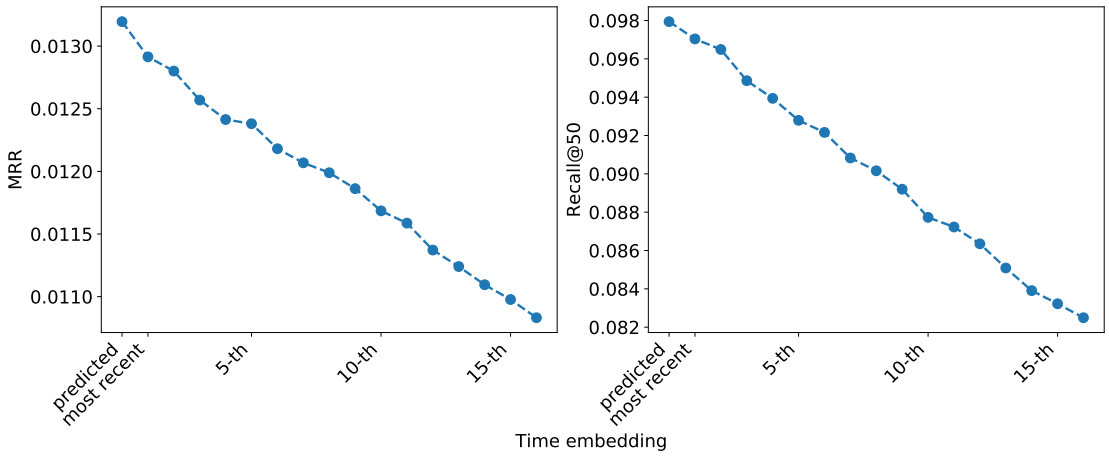


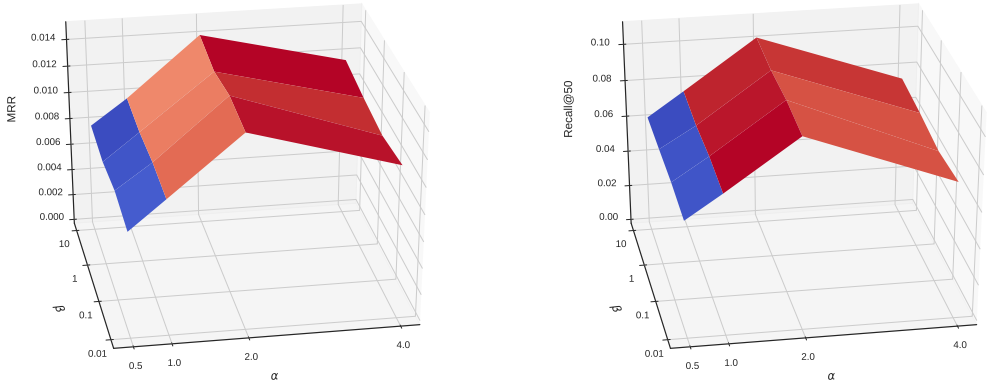
Figure 5.4: Experiments with time embeddings from different graph snapshots on *yelp* dataset. The predicted time embedding achieves the best performance.

5.5.1.6 Time Embedding

Time embedding translates node embeddings in each graph snapshot into corresponding temporal space. Besides using the predicted time embedding to project the translation space in the future graph snapshot for link prediction, we also experiment with using the existing temporal spaces, where we directly employ the time embeddings of the most recent a few graph snapshots to translate node embeddings to predict future graph snapshot. The results of *yelp* dataset is shown in Figure 5.4. As expected, prediction performance is higher when the time of the translation space is closer to the prediction time, and the predicted time embedding achieves the best performance, indicating that our model successfully captures the evolution of the translation spaces. Note that, using the most recent embedding space is commonly used in existing works such as CTDNE [116] and dySAT [117], however, they fail to go one step further to predict the future embeddings, which are proven to be more effective in our work.

5.5.1.7 Parameter Sensitivity

As shown in section 4, α controls the effect of time embedding in direction based translation, while β balances the graph reconstruction and temporal space modeling. To test the impact of α and β, we vary α as $\{0.5, 1.0, 2.0, 4.0\}$ and β as $\{0.01, 0.1, 1.0, 10.0\}$ respectively, and report MRR and Recall@50 in link prediction on *yelp* dataset. In Figure 5.5, we note that: 1) with the increase of *alpha*, the



(a) MRR (b) Recall@50

Figure 5.5: The impact of α and β for link prediction

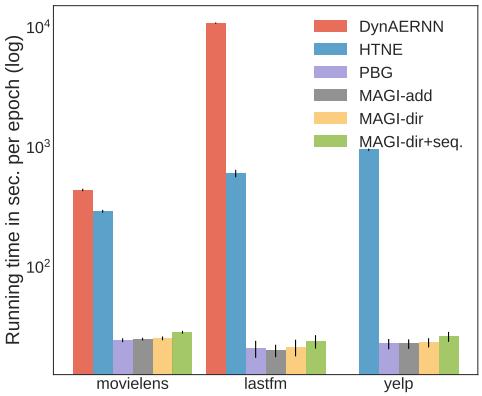


Figure 5.6: Running time comparison

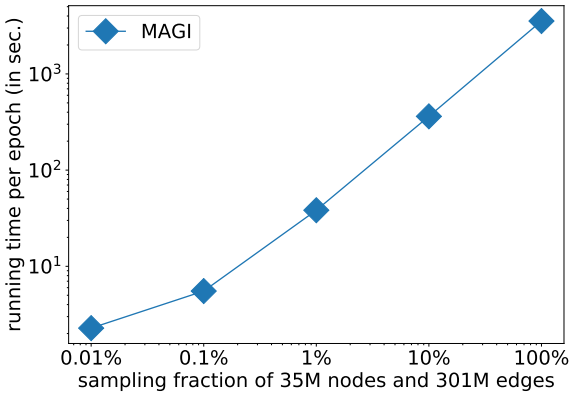


Figure 5.7: Running time of MAGI over various graph sizes

performance increases first and then decreases, and $\alpha = 2$ gives the best performance. This is because time embedding is not able to tweak the direction of node embedding when α is small, while large α makes all translated embeddings in each translation space become similar and loses node discrepancies; 2) the performance remains stable with various β, signaling the robustness to various choices of β.

5.5.2 Running Time Analysis

To answer the **EQ2**, we first compare the running time of MAGI with other dynamic graph embedding baseline algorithms. Based on the implementation, HTNE runs on one NVIDIA Tesla P100 GPU, while other approaches use CPU on a 24 Intel

Xeon cores machine. As shown in Figure 5.6, HTNE and DynAERNN scale poorly on large graphs, and DynAERNN even fails to train *yelp* dataset due to the RAM issue. Since introducing time embedding and modeling the time embedding change do not add much computation, MAGI shares a similar training time with static embedding method PBG, indicating that using time embedding can improve the performance in future link prediction without sacrificing training speed.

5.5.3 Discussion

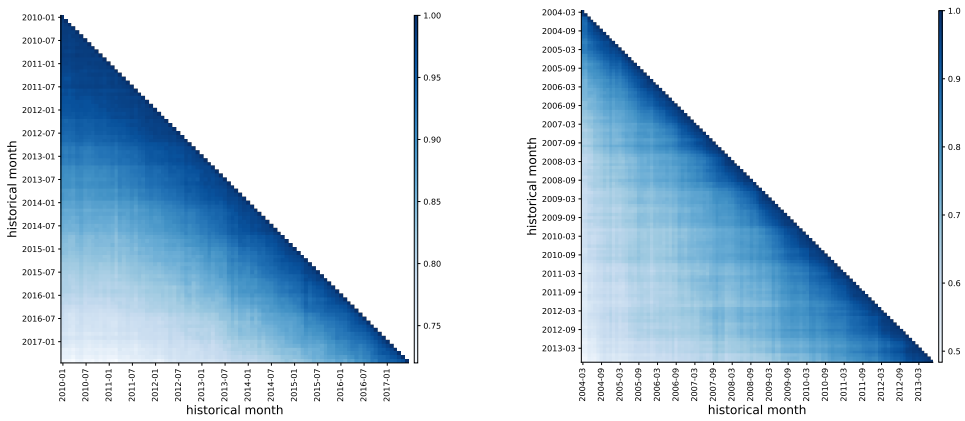
In addition, to further validate the scalability of our MAGI full model, we use the largest network dataset with timestamps from the Koblenz Network Collection (KONECT) [146] – *Delicious user–URL*, which contains 35 million nodes and 301 million edges and spans 5 years. Still using monthly graph snapshots, we randomly sample a few sub-networks to measure MAGI's running time over different graph sizes. The running time is measured by an average of 10 epochs and presented in Figure 5.7. As expected, because MAGI samples a fixed number of negatives edges for each positive edge and trains all graph snapshots simultaneously, the running time grows linearly over graph size, suggesting the scalability of MAGI in training large temporal graphs.

In this section, we conduct a visualization analysis on time embedding to answer **EQ3: what can we learn from the temporal space translation?**

5.5.3.1 Time Embedding

First, we analyze the characteristics of time embeddings in translation spaces. In Figure 5.8, We demonstrate two heatmaps of cosine similarities between time embeddings on *yelp* and *movielens* datasets. As shown, the closer the graph snapshots, the similar their time embeddings, which implies similar translation spaces. Unlike relationships in knowledge graphs, time is intrinsically correlated across different graph snapshots, where node features such as users' behaviors and items' semantic meanings are gradually changing over time, thus translation spaces are expected to shift gradually as well, leading to positively correlating with time differences.

In addition, as time embedding spaces encode the temporal dynamics of node features, we are not only able to predict user interest drifting, but also can discover



(a) yelp (b) movielens

Figure 5.8: Time embedding similarity

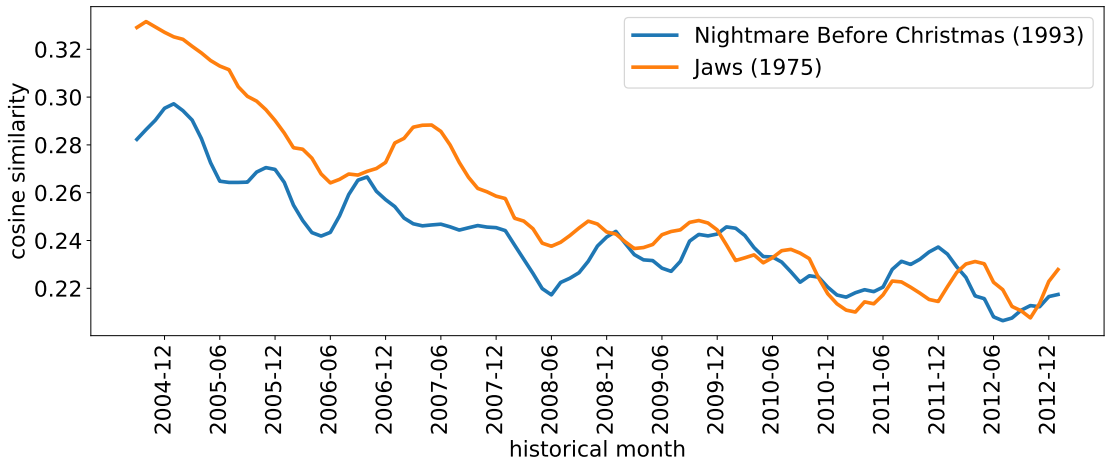


Figure 5.9: Two classic seasonal movies.

some temporal patterns of items. Taking *Movielens* dataset for example, we select two classic seasonal movies, *Jaws* and *Nightmare Before Christmas*. *Jaws* is the first summer blockbuster and tells a story on the ocean, making it popular for summer watching, while *Nightmare Before Christmas* is a beloved musical animation about Christmas. We calculate the cosine similarity between the movies' embedding with each time embedding, and plot their temporal similarity trajectories in figure 5.9. To make it more clear, we smooth the trajectories by a Savitzky-Golay filter [147]. Interestingly, most similarity peaks of *Jaws* occur in the summertime, while *Nightmare Before Christmas* reaches most similarity peaks in December. This is because people interact with certain items at some specific time, and proves that

we can encode temporal patterns in temporal translation spaces and our model succeeds in capturing these temporal patterns.

5.5.3.2 Attention Scores

Then, we start a qualitative analysis of the distribution of temporal attention weights learned by MAGI. For each dataset, as we conduct a monthly graph snapshot prediction of one consecutive year, we examine the temporal attention coefficients learned at each prediction month, which indicates the relative importance of each historical graph snapshot in link prediction. Since we model the dynamics of the temporal translation spaces, the attention scores signal the correlation between different translation spaces. Figure 5.10 visualizes heatmaps of the learned temporal attention weights of the 12 prediction months with all historical months for *movielens* dataset. Note that, we concatenate the context vector with the corresponding hidden state to do prediction at each snapshot, thus the attention scores should reveal the importance of historical graph snapshots besides neighboring graph snapshots. Interestingly, the predicted translation spaces are not only correlated with recent graph snapshots but also strongly connected with very old graph snapshots, signaling all historical graph snapshots are equally important. It is worthwhile to mention that, the connection between two translation space does not necessarily correspond to similar translation spaces, where time embeddings are alike, but only indicate the ability to predict the translation space. For example, their time embeddings can be negatively correlated.

5.5.3.3 Visualization of Embedding Change

Last but not least, we demonstrate the temporal change of user embedding learned by our MAGI model. We apply t-SNE [148] to project user embeddings at each graph snapshot to a 2-dimensional space. Figure 5.11 (a) shows the embedding change of 10 randomly sampled users, where we observe a clear pattern of user embedding change over graph snapshots, corresponding to our temporal space translation. In Figure 5.11 (b), we show one sampled user embedding change and her interacted items at each graph snapshot. As shown, user embedding at each graph snapshot is close to the embeddings of the items she interacted with in neighboring graph snapshots, indicating that our model successfully encodes user

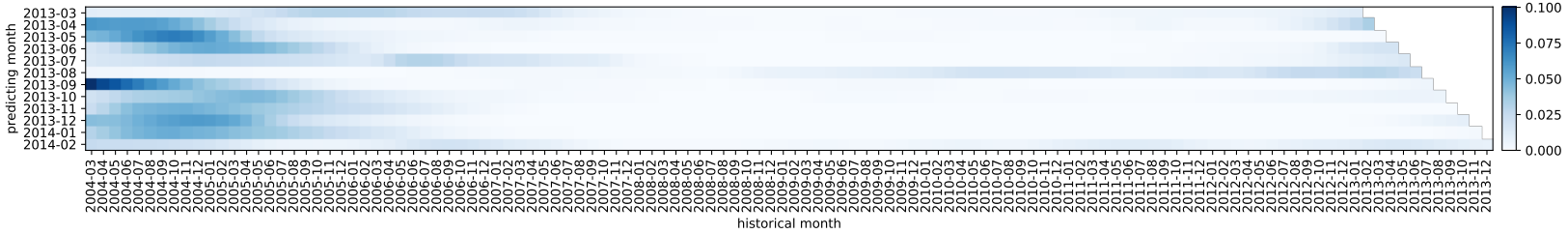
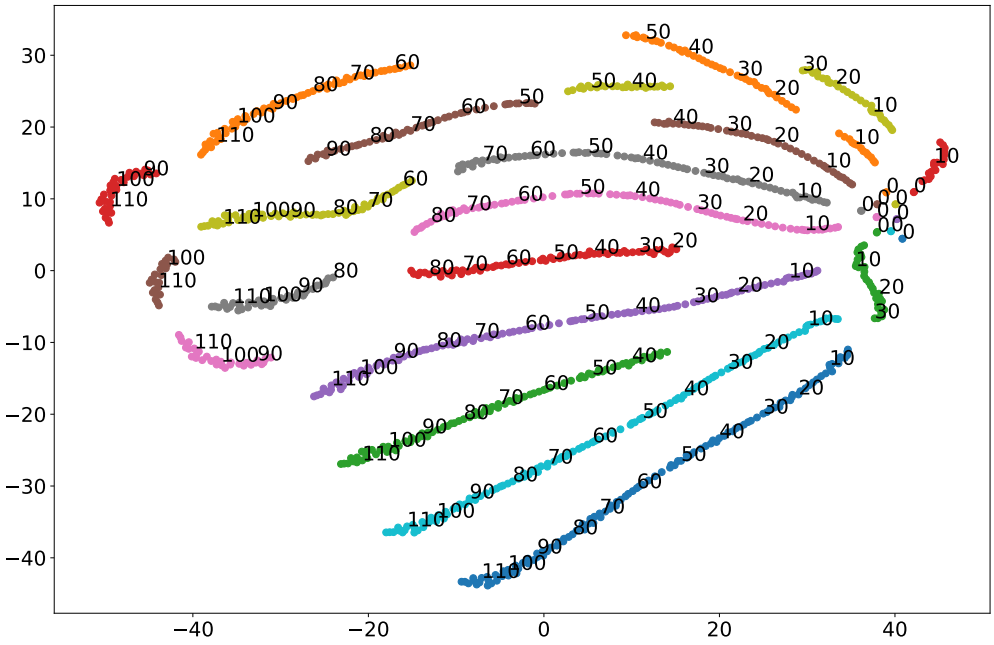
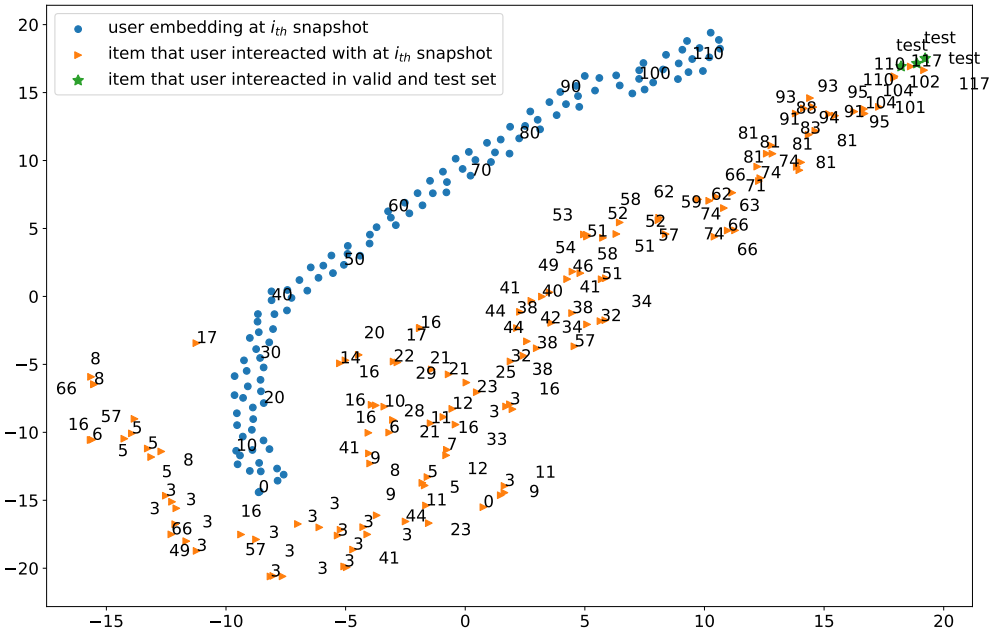


Figure 5.10: Attention scores for *movielens* dataset



(a)



(b)

Figure 5.11: (a) The embedding dynamics of 10 randomly sampled users. Each user is signaled by one color while the numbers indicate the order of embedding change over graph snapshots. (b) The user embedding change of one sampled user and her interacted items at each graph snapshot. Note that orange triangles denote items from the training set while green stars denote items in the test set.

interests in each graph snapshot. More importantly, user interacted items in the future appear to be close to the most recent embeddings, proving that our model can capture the user interest change by modeling the temporal space change.

5.6 Summary

In this chapter, we introduce a scalable framework to model node embedding change on temporal graphs. Specifically, we propose a novel translation-based framework named as MAGI to project node embedding in each graph snapshot into corresponding temporal space. Then we design a sequential model to better align temporal spaces and capture the global evolution of graphs. Modeling the temporal space change not only helps us capture the graph evolution effectively, but also improves the performance in link prediction with the predicted translation space. MAGI allows us to train all graph snapshots simultaneously while still preserving the temporal constraint in learning, making it scalable to large temporal graphs. In the future, we would like to include node features and extend this paradigm to graph convolutional networks.

Chapter 6 | Conclusions and Future Work

6.1 Conclusions

In this dissertation, we introduce our path along with characterization, understanding, and prediction of the temporal dynamics of web content. First, we demonstrate two examples on how to discover and characterize temporal patterns of web content. Second, to directly help users better understand how events evolve over time, we propose a fast and effective news timeline summarization to deliver real-time timelines. Third, we focus on predicting content evolution, such as users' interest change and items' semantic drifting, by learning dynamic graph embedding and propose a scalable approach to encode large temporal graphs.

In the problem of characterizing temporal patterns, we study two types of web content, user behavioral change, and content popularity pattern. In the first study, we collect large-scale datasets from two of the most popular crowdfunding platforms and uncover four distinct temporal backing patterns on both platforms. With quantitative analysis, we reveal possible factors affecting users' backing behaviors. For example, both the outcomes of backers' first a few projects and their connections with project creators are important in maintaining a favorable relationship between backers and the platforms. The second study characterizes a small fraction of news articles that consistently gain traffic over time. We conduct a quantitative analysis on more than 400K news articles in a 5-year duration and shed light on news articles' long-term popularity. Based on the insights from our analysis, we build early prediction models on evergreen article identification, work with journalists to manually validate our recommendations, and succeed in deploying our model at

the Washington Post with encouraging performance in production.

To prevent users from getting lost in information floods and better understand news event evolution, we move to investigate news timeline summarization. Observing that all existing works only focus on the generation quality but ignore the generation speed, we propose to speed up news timeline generation by dividing the whole summarization tasks into multiple sub summarization tasks. In our framework, we divide the giant timeline summarization task into multiple small daily summarization tasks by explicitly selecting salient dates at first. As expected, ignoring the temporal correlation of daily summaries greatly speeds up the generation but suffers from the redundancy issue, especially for timelines with abundant daily summaries. Thus, we mitigate this issue with an efficient post-processing process. Surprisingly, although the empirical upper bound of our method by ignoring the temporal correlation of daily summaries is much lower than other fancy frameworks like submodular formulation, all other frameworks fail to reach this lower upper bound, not even close, and our approach reaches state-of-the-art performance in both generation quality and speed. It is also worth noting that, using our approach, we succeed in building a real-time news timeline summarization system an industrial-level news corpus and achieve promising results.

Furthermore, we take a step to predict content evolution by learning dynamic graph embedding. As graphs are generated by a series of temporal interactions between nodes, node attributes are evolving with time and encoded in the graph topological structure changes. To maintain the temporal evolution, processing edges in chronological order is necessary. Due to temporal topological graph structure, however, modeling sequential changes of individual nodes can be expensive, making it difficult to extend to large graphs. Thus, we propose a temporal translation-based model to embed different graph snapshots simultaneously, and encode the graph topological structure evolution by aligning the temporal translation spaces with a sequential model. We experimented on two types of temporal graphs, including user-user communication graphs and user-item bipartite graphs, and show superior performance on all datasets. Notably, on three large bipartite graphs, which consist of over 20 thousand nodes and 2 million edges, our approach outperforms the baseline method by a significant margin without sacrificing training speed. More interestingly, using temporal space translation, our model is able to encode and discover temporal popularity patterns of nodes.

6.2 Future Work

Many directions are promising for future research. For temporal pattern discovery, using machine learning and statistical modelings are only the first step, while domain knowledge is required to further validate and utilize the findings. Although it is necessary to include domain experts in the loop, it is not an easy task to collaborate with the human. How to efficiently collect feedback from domain experts needs further investigation, which can involve Human-Computer Interaction and Reinforcement Learning. In addition, evaluating temporal patterns in practice can also be non-trivial, as some long-term temporal pattern requires an extended observation period for verification. One possible direction could be developing a short-term proximity to approximate the long-term reward. Adding time information not only introduces opportunities to enhance interpretability and predictivity of machine learning models but also includes extra difficulties to process data. For example, computational complexity can be significantly increased with time information, making it difficult to be applied to industrial big data. Thus, examining the balance between the expressive power and computational complexity of the temporal models is critical, and we plan to continue working on deployable and scalable algorithms for temporal data.

Bibliography

[1] Martschat, S. and K. Markert (2018) "A Temporally Sensitive Submodularity Framework for Timeline Summarization," in *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pp. 230–240.

[2] Moscato, V., A. Picariello, and A. M. Rinaldi (2010) "A recommendation strategy based on user behavior in digital ecosystems," in *Proceedings of the International Conference on Management of Emergent Digital EcoSystems*, ACM, pp. 25–32.

[3] Keneshloo, Y., S. Wang, E.-H. Han, and N. Ramakrishnan (2016) "Predicting the popularity of news articles," in *SDM*, SIAM, pp. 441–449.

[4] Wu, Q., T. Wang, Y. Cai, H. Tian, and Y. Chen (2017) "Rumor restraining based on propagation prediction with limited observations in large-scale social networks," in *Proceedings of the Australasian Computer Science Week Multiconference*, ACM, p. 1.

[5] Allan, J., R. Papka, and V. Lavrenko (1998) "On-line new event detection and tracking," in *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, ACM, pp. 37–45.

[6] Catledge, L. D. and J. E. Pitkow (1995) *Characterizing browsing behaviors on the world-wide web*, *Tech. rep.*, Georgia Institute of Technology.

[7] Radinsky, K., K. Svore, S. Dumais, J. Teevan, A. Bocharov, and E. Horvitz (2012) "Modeling and predicting behavioral dynamics on the web," in *Proceedings of the 21st international conference on World Wide Web*, ACM, pp. 599–608.

[8] Lo, C., D. Frankowski, and J. Leskovec (2016) "Understanding behaviors that lead to purchasing: A case study of pinterest," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, ACM, pp. 531–540.

[9] Adar, E., J. Teevan, S. T. Dumais, and J. L. Elsas (2009) "The web changes everything: understanding the dynamics of web content," in *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, ACM, pp. 282–291.

[10] Ye, S. and S. F. Wu (2010) "Measuring message propagation and social influence on Twitter. com," in *International Conference on Social Informatics*, Springer, pp. 216–231.

[11] Szabo, G. and B. A. Huberman (2010) "Predicting the popularity of online content," *Communications of the ACM*, **53**(8), pp. 80–88.

[12] An, J., D. Quercia, and J. Crowcroft (2014) "Recommending investors for crowdfunding projects," in *Proceedings of the 23rd international conference on World wide web*, ACM, pp. 261–270.

[13] Tsagkias, M., W. Weerkamp, and M. De Rijke (2009) "Predicting the volume of comments on online news stories," in *CIKM*, ACM, pp. 1765–1768.

[14] Rizos, G., S. Papadopoulos, and Y. Kompatsiaris (2016) "Predicting news popularity by mining online discussions," in *WWW*, ACM, pp. 737–742.

[15] Liao, Y., T. Tran, D. Lee, and K. Lee (2017) "Understanding Temporal Backing Patterns in Online Crowdfunding Communities," in *Proceedings of the 2017 ACM on Web Science Conference*, ACM, pp. 369–378.

[16] Kuppuswamy, V. and B. L. Bayus (2015) "Crowdfunding creative ideas: The dynamics of project backers in Kickstarter," .

[17] Stanko, M. A. and D. H. Henard (2016) "How Crowdfunding Influences Innovation," *MIT Sloan Management Review*, **57**(3), p. 15.

[18] Althoff, T. and J. Leskovec (2015) "Donor retention in online crowdfunding communities: A case study of donorschoose. org," in *Proceedings of the 24th International Conference on World Wide Web*, ACM, pp. 34–44.

[19] Rakesh, V., J. Choo, and C. K. Reddy (2015) "Project Recommendation Using Heterogeneous Traits in Crowdfunding." in *ICWSM*, pp. 337–346.

[20] Belleflamme, P., T. Lambert, and A. Schwienbacher (2014) "Crowdfunding: Tapping the right crowd," *Journal of business venturing*, **29**(5), pp. 585–609.

[21] Gerber, E. M. and J. Hui (2013) "Crowdfunding: Motivations and deterrents for participation," *ACM Transactions on Computer-Human Interaction (TOCHI)*, **20**(6), p. 34.

[22] Hui, J. S., M. D. Greenberg, and E. M. Gerber (2014) "Understanding the role of community in crowdfunding work," in *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*, ACM, pp. 62–74.

[23] Gerber, E. M., J. S. Hui, and P.-Y. Kuo (2012) "Crowdfunding: Why people are motivated to post and fund projects on crowdfunding platforms," in *Proceedings of the International Workshop on Design, Influence, and Social Technologies: Techniques, Impacts and Ethics*, vol. 2, p. 11.

[24] Siering, M., J.-A. Koch, and A. V. Deokar (2016) "Detecting Fraudulent Behavior on Crowdfunding Platforms: The Role of Linguistic and Content-Based Cues in Static and Dynamic Contexts," *Journal of Management Information Systems*, **33**(2), pp. 421–455.

[25] Cumming, D. J., G. Leboeuf, and A. Schwienbacher (2014) "Crowdfunding models: Keep-it-all vs. all-or-nothing," in *Paris December 2014 finance meeting EUROFIDAI-AFFI paper*, vol. 10.

[26] Mollick, E. (2014) "The dynamics of crowdfunding: An exploratory study," *Journal of business venturing*, **29**(1), pp. 1–16.

[27] Greenberg, M. D., J. Hui, and E. Gerber (2013) "Crowdfunding: a resource exchange perspective," in *CHI'13 Extended Abstracts on Human Factors in Computing Systems*, ACM, pp. 883–888.

[28] Lu, C.-T., S. Xie, X. Kong, and P. S. Yu (2014) "Inferring the impacts of social media on crowdfunding," in *Proceedings of the 7th ACM international conference on Web search and data mining*, ACM, pp. 573–582.

[29] Muller, M., W. Geyer, T. Soule, and J. Wafer (2014) "Geographical and organizational distances in enterprise crowdfunding," in *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*, ACM, pp. 778–789.

[30] Xu, A., X. Yang, H. Rao, W.-T. Fu, S.-W. Huang, and B. P. Bailey (2014) "Show me the money!: An analysis of project updates during crowdfunding campaigns," in *Proceedings of the SIGCHI conference on human factors in computing systems*, ACM, pp. 591–600.

[31] Lin, Y., W.-C. Lee, and C.-C. H. Chang (2016) "Analysis of rewards on reward-based crowdfunding platforms," in *Advances in Social Networks Analysis and Mining (ASONAM), 2016 IEEE/ACM International Conference on*, IEEE, pp. 501–504.

[32] Greenberg, M. D., B. Pardo, K. Hariharan, and E. Gerber (2013) "Crowdfunding support tools: predicting success & failure," in *CHI'13 Extended Abstracts on Human Factors in Computing Systems*, ACM, pp. 1815–1820.

[33] Chung, J. and K. Lee (2015) "A Long-Term Study of a Crowdfunding Platform: Predicting Project Success and Fundraising Amount," in *Proceedings of the 26th ACM Conference on Hypertext & Social Media*, ACM, pp. 211–220.

[34] Etter, V., M. Grossglauser, and P. Thiran (2013) "Launch hard or go home!: predicting the success of kickstarter campaigns," in *Proceedings of the first ACM conference on Online social networks*, ACM, pp. 177–182.

[35] Mitra, T. and E. Gilbert (2014) "The language that gets people to give: Phrases that predict success on kickstarter," in *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*, ACM, pp. 49–61.

[36] Rakesh, V., W.-C. Lee, and C. K. Reddy (2016) "Probabilistic Group Recommendation Model for Crowdfunding Domains," in *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, ACM, pp. 257–266.

[37] Rakthanmanon, T., B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh (2012) "Searching and mining trillions of time series subsequences under dynamic time warping," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, pp. 262–270.

[38] Berndt, D. J. and J. Clifford (1994) "Using dynamic time warping to find patterns in time series." in *KDD workshop*, vol. 10, Seattle, WA, pp. 359–370.

[39] Petitjean, F., A. Ketterlin, and P. Gançarski (2011) "A global averaging method for dynamic time warping, with applications to clustering," *Pattern Recognition*, **44**(3), pp. 678–693.

[40] Arthur, D. and S. Vassilvitskii (2007) "k-means++: The advantages of careful seeding," in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, Society for Industrial and Applied Mathematics, pp. 1027–1035.

[41] Wilcox, R. (2005) "Kolmogorov–smirnov test," *Encyclopedia of biostatistics*.

[42] Zhu, J., H. Zou, S. Rosset, and T. Hastie (2009) "Multi-class adaboost," *Statistics and its Interface*, **2**(3), pp. 349–360.

[43] Drummond, C., R. C. Holte, et al. (2003) "C4. 5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling," in *Workshop on learning from imbalanced datasets II*, vol. 11, Citeseer.

[44] Liao, Y., S. Wang, E.-H. S. Han, J. Lee, and D. Lee (2019) "Characterization and Early Detection of Evergreen News Articles," .

[45] Tatar, A., P. Antoniadis, M. D. De Amorim, and S. Fdida (2014) "From popularity prediction to ranking online news," *Social Network Analysis and Mining*, **4**(1), p. 174.

[46] Ma, C., Z. Yan, and C. W. Chen (2017) "LARM: A Lifetime Aware Regression Model for Predicting YouTube Video Popularity," in *CIKM*, ACM, pp. 467–476.

[47] Pinto, H., J. M. Almeida, and M. A. Gonçalves (2013) "Using early view patterns to predict the popularity of youtube videos," in *WSDM*, ACM, pp. 365–374.

[48] Gelli, F., T. Uricchio, M. Bertini, A. Del Bimbo, and S.-F. Chang (2015) "Image popularity prediction in social media using sentiment and context features," in *MM*, ACM, pp. 907–910.

[49] Wu, B., T. Mei, W.-H. Cheng, Y. Zhang, et al. (2016) "Unfolding Temporal Dynamics: Predicting Social Media Popularity Using Multi-scale Temporal Decomposition." in *AAAI*, pp. 272–278.

[50] Chang, B., H. Zhu, Y. Ge, E. Chen, H. Xiong, and C. Tan (2014) "Predicting the popularity of online serials with autoregressive models," in *CIKM*, ACM, pp. 1339–1348.

[51] Leskovec, J., L. Backstrom, and J. Kleinberg (2009) "Meme-tracking and the dynamics of the news cycle," in *SIGKDD*, ACM, pp. 497–506.

[52] Marujo, L., M. Bugalho, J. P. d. S. Neto, A. Gershman, and J. Carbonell (2013) "Hourly traffic prediction of news stories," *arXiv preprint arXiv:1306.4608*.

[53] Tatar, A., J. Leguay, P. Antoniadis, A. Limbourg, M. D. de Amorim, and S. Fdida (2011) "Predicting the popularity of online articles based on user comments," in *WIMS*, ACM, p. 67.

[54] Lerman, K. and T. Hogg (2010) "Using a model of social dynamics to predict popularity of news," in *WWW*, ACM, pp. 621–630.

[55] Bandari, R., S. Asur, and B. A. Huberman (2012) "The pulse of news in social media: Forecasting popularity." *ICWSM*, **12**, pp. 26–33.

[56] Elhenfnawy, W., J. Wright, K. Kallepalli, K. Racheal, A. Gupta, R. Parimi, P. Shah, and Y. Li (2016) "What Differentiates News Articles with Short and Long Shelf Lives? A Case Study on News Articles at Bloomberg. com," in *BDCloud-SocialCom-SustainCom*, IEEE, pp. 131–136.

[57] Tan, Z., Y. Wang, Y. Zhang, and J. Zhou (2016) "A novel time series approach for predicting the long-term popularity of online videos," *IEEE Transactions on Broadcasting*, **62**(2), pp. 436–445.

[58] Shen, H.-W., D. Wang, C. Song, and A.-L. Barabási (2014) "Modeling and Predicting Popularity Dynamics via Reinforced Poisson Processes." in *AAAI*, vol. 14, pp. 291–297.

[59] Zhao, Q., M. A. Erdogdu, H. Y. He, A. Rajaraman, and J. Leskovec (2015) "Seismic: A self-exciting point process model for predicting tweet popularity," in *SIGKDD*, ACM, pp. 1513–1522.

[60] Cheng, J., L. Adamic, P. A. Dow, J. M. Kleinberg, and J. Leskovec (2014) "Can cascades be predicted?" in *WWW*, ACM, pp. 925–936.

[61] Mishra, S., M.-A. Rizoiu, and L. Xie (2016) "Feature driven and point process approaches for popularity prediction," in *CIKM*, ACM, pp. 1069–1078.

[62] Martin, F. and M. Johnson (2015) "More efficient topic modelling through a noun only approach," in *Proceedings of the Australasian Language Technology Association Workshop 2015*, pp. 111–115.

[63] Yuan, J., F. Gao, Q. Ho, W. Dai, J. Wei, X. Zheng, E. P. Xing, T.-Y. Liu, and W.-Y. Ma (2015) "LightLDA: Big Topic Models on Modest Computer Clusters," in *WWW*, ACM, pp. 1351–1361.

[64] Gilbert, C. H. E. (2014) "Vader: A parsimonious rule-based model for sentiment analysis of social media text," in *ICWSM*.

[65] Berger, J. and K. L. Milkman (2012) "What makes online content viral?" *Journal of marketing research*, **49**(2), pp. 192–205.

[66] Bojanowski, P., E. Grave, A. Joulin, and T. Mikolov (2016) "Enriching Word Vectors with Subword Information," *arXiv preprint arXiv:1607.04606*.

[67] Ke, G., Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu (2017) "LightGBM: A highly efficient gradient boosting decision tree," in *NIPS*, pp. 3149–3157.

[68] Elkan, C. and K. Noto (2008) "Learning classifiers from only positive and unlabeled data," in *SIGKDD*, ACM, pp. 213–220.

[69] Natarajan, N., I. S. Dhillon, P. K. Ravikumar, and A. Tewari (2013) "Learning with noisy labels," in *NIPS*, pp. 1196–1204.

[70] Swan, R. C. and J. Allan (2000) "TimeMine: visualizing automatically constructed timelines." in *SIGIR*, p. 393.

[71] Chieu, H. L. and Y. K. Lee (2004) "Query based event extraction along a timeline," in *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, ACM, pp. 425–432.

[72] Yan, R., X. Wan, J. Otterbacher, L. Kong, X. Li, and Y. Zhang (2011) "Evolutionary timeline summarization: a balanced optimization framework via iterative substitution," in *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, ACM, pp. 745–754.

[73] Tran, G. B., T. A. Tran, N.-K. Tran, M. Alrifai, and N. Kanhabua (2013) "Leveraging learning to rank in an optimization framework for timeline summarization," in *SIGIR 2013 Workshop on Time-aware Information Access (TAIA*.

[74] Tran, G., M. Alrifai, and E. Herder (2015) "Timeline summarization from relevant headlines," in *European Conference on Information Retrieval*, Springer, pp. 245–256.

[75] Wang, W. Y., Y. Mehdad, D. R. Radev, and A. Stent (2016) "A low-rank approximation approach to learning joint embeddings of news stories and images for timeline summarization," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 58–68.

[76] Zhou, D., H. Xu, X.-Y. Dai, and Y. He (2016) "Unsupervised Storyline Extraction from News Articles." in *IJCAI*, pp. 3014–3021.

[77] Li, J. and C. Cardie (2014) "Timeline generation: Tracking individuals on twitter," in *Proceedings of the 23rd international conference on World wide web*, ACM, pp. 643–652.

[78] Zhou, D., L. Guo, and Y. He (2018) "Neural Storyline Extraction Model for Storyline Generation from News," in *Proceedings of NAACL-HLT*, pp. 1727–1736.

[79] Yan, R., L. Kong, C. Huang, X. Wan, X. Li, and Y. Zhang (2011) "Timeline generation through evolutionary trans-temporal summarization," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, pp. 433–443.

[80] Tran, G. B., M. Alrifai, and D. Q. Nguyen (2013) "Predicting relevant news events for timeline summaries." in *WWW (Companion Volume)*, pp. 91–92.

[81] Wang, L., C. Cardie, and G. Marchetti (2015) "Socially-Informed Timeline Generation for Complex Events," in *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1055–1065.

[82] Steen, J. and K. Markert (2019) "Abstractive Timeline Summarization," in *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pp. 21–31.

[83] Lin, C.-Y. (2004) "Rouge: A package for automatic evaluation of summaries," *Text Summarization Branches Out*.

[84] Strötgen, J. and M. Gertz (2015) "A Baseline Temporal Tagger for all Languages," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Lisbon, Portugal, pp. 541–547.
URL `http://aclweb.org/anthology/D15-1063`

[85] Tran, G., E. Herder, and K. Markert (2015) "Joint graphical models for date selection in timeline summarization," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, vol. 1, Association for Computational Linguistics, pp. 1598–1607.

[86] Page, L., S. Brin, R. Motwani, and T. Winograd (1999) *The pagerank citation ranking: Bringing order to the web.*, *Tech. rep.*, Stanford InfoLab.

[87] Robertson, S., H. Zaragoza, et al. (2009) "The probabilistic relevance framework: BM25 and beyond," *Foundations and Trends® in Information Retrieval*, **3**(4), pp. 333–389.

[88] Bahmani, B., A. Chowdhury, and A. Goel (2010) "Fast incremental and personalized pagerank," *Proceedings of the VLDB Endowment*, **4**(3), pp. 173–184.

[89] Mihalcea, R. and P. Tarau (2004) "Textrank: Bringing order into text," in *Proceedings of the 2004 conference on empirical methods in natural language processing*.

[90] Barrios, F., F. López, L. Argerich, and R. Wachenchauzer (2016) "Variations of the similarity function of textrank for automated summarization," *arXiv preprint arXiv:1602.03606*.

[91] Carbonell, J. and J. Goldstein (1998) "The use of MMR, diversity-based reranking for reordering documents and producing summaries," in *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 335–336.

[92] Radev, D. R., H. Jing, M. Styś, and D. Tam (2004) "Centroid-based summarization of multiple documents," *Information Processing & Management*, **40**(6), pp. 919–938.

[93] Liang, D., G. Wang, and J. Nie (2019) "A Dynamic Evolutionary Framework for Timeline Generation based on Distributed Representations," *arXiv preprint arXiv:1905.05550*.

[94] Martschat, S. and K. Markert (2017) "Improving rouge for timeline summarization," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pp. 285–290.

[95] Devlin, J., M.-W. Chang, K. Lee, and K. Toutanova (2018) "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*.

[96] Frey, B. J. and D. Dueck (2007) "Clustering by passing messages between data points," *science*, **315**(5814), pp. 972–976.

[97] Gormley, C. and Z. Tong (2015) *Elasticsearch: the definitive guide: a distributed real-time search and analytics engine*, " O'Reilly Media, Inc.".

[98] Wang, S., J. Tang, C. Aggarwal, Y. Chang, and H. Liu (2017) "Signed network embedding in social media," in *Proceedings of the 2017 SIAM international conference on data mining*, SIAM, pp. 327–335.

[99] Nagl, M. (1976) "Graph rewriting systems and their application in biology," in *Mathematical Models in Medicine*, Springer, pp. 135–156.

[100] Kumar, S., X. Zhang, and J. Leskovec (2019) "Predicting Dynamic Embedding Trajectory in Temporal Interaction Networks," in *Proceedings of the 25th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM.

[101] Grover, A. and J. Leskovec (2016) "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, pp. 855–864.

[102] Ou, M., P. Cui, J. Pei, Z. Zhang, and W. Zhu (2016) "Asymmetric transitivity preserving graph embedding," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1105–1114.

[103] Cavallari, S., V. W. Zheng, H. Cai, K. C.-C. Chang, and E. Cambria (2017) "Learning community embedding with community detection and node embedding on graphs," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp. 377–386.

[104] Yang, L., X. Cao, D. He, C. Wang, X. Wang, and W. Zhang (2016) "Modularity Based Community Detection with Deep Learning." in *IJCAI*, vol. 16, pp. 2252–2258.

[105] Perozzi, B., R. Al-Rfou, and S. Skiena (2014) "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710.

[106] Wang, X., P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang (2017) "Community preserving network embedding," in *Thirty-first AAAI conference on artificial intelligence*.

[107] Tang, J., M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei (2015) "Line: Large-scale information network embedding," in *Proceedings of the 24th international conference on world wide web*, pp. 1067–1077.

[108] Pan, S., R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang (2018) "Adversarially regularized graph autoencoder for graph embedding," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, AAAI Press, pp. 2609–2615.

[109] Kipf, T. N. and M. Welling (2016) "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*.

[110] Hamilton, W., Z. Ying, and J. Leskovec (2017) "Inductive representation learning on large graphs," in *Advances in neural information processing systems*, pp. 1024–1034.

[111] Veličković, P., G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio (2017) "Graph attention networks," *arXiv preprint arXiv:1710.10903*.

[112] Bamler, R. and S. Mandt (2017) "Dynamic word embeddings," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, JMLR. org, pp. 380–389.

[113] Goyal, P., S. R. Chhetri, and A. Canedo (2020) "dyngraph2vec: Capturing network dynamics using dynamic graph representation learning," *Knowledge-Based Systems*, **187**, p. 104816.

[114] Goyal, P., N. Kamra, X. He, and Y. Liu (2018) "Dyngem: Deep embedding method for dynamic graphs," *arXiv preprint arXiv:1805.11273*.

[115] Zuo, Y., G. Liu, H. Lin, J. Guo, X. Hu, and J. Wu (2018) "Embedding temporal network via neighborhood formation," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2857–2866.

[116] Nguyen, G. H., J. B. Lee, R. A. Rossi, N. K. Ahmed, E. Koh, and S. Kim (2018) "Continuous-time dynamic network embeddings," in *Companion Proceedings of the The Web Conference 2018*, International World Wide Web Conferences Steering Committee, pp. 969–976.

[117] Sankar, A., Y. Wu, L. Gou, W. Zhang, and H. Yang (2020) "DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention Networks," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, pp. 519–527.

[118] Zhu, L., D. Guo, J. Yin, G. Ver Steeg, and A. Galstyan (2016) "Scalable temporal latent space inference for link prediction in dynamic social networks," *IEEE Transactions on Knowledge and Data Engineering*, **28**(10), pp. 2765–2777.

[119] Zhou, L., Y. Yang, X. Ren, F. Wu, and Y. Zhuang (2018) "Dynamic network embedding by modeling triadic closure process," in *Thirty-Second AAAI Conference on Artificial Intelligence*.

[120] Ching, A., S. Edunov, M. Kabiljo, D. Logothetis, and S. Muthukrishnan (2015) "One trillion edges: Graph processing at facebook-scale," *Proceedings of the VLDB Endowment*, **8**(12), pp. 1804–1815.

[121] Wang, J., P. Huang, H. Zhao, Z. Zhang, B. Zhao, and D. L. Lee (2018) "Billion-scale commodity embedding for e-commerce recommendation in alibaba," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 839–848.

[122] (2020), H. . W. A. S., "Digital 2020 April Global Statshot," `https://datareportal.com/reports/digital-2020-april-global-statshot`.

[123] Chang, R. C., J. Kivela, and A. H. Mak (2010) "Food preferences of Chinese tourists," *Annals of tourism research*, **37**(4), pp. 989–1011.

[124] Ahmed, A., N. Shervashidze, S. Narayanamurthy, V. Josifovski, and A. J. Smola (2013) "Distributed large-scale natural graph factorization," in *Proceedings of the 22nd international conference on World Wide Web*, pp. 37–48.

[125] Belkin, M. and P. Niyogi (2002) "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Advances in neural information processing systems*, pp. 585–591.

[126] Mikolov, T., K. Chen, G. Corrado, and J. Dean (2013) "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*.

[127] Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin (2017) "Attention is all you need," in *Advances in neural information processing systems*, pp. 5998–6008.

[128] LeCun, Y., Y. Bengio, et al. (1995) "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, **3361**(10), p. 1995.

[129] Bruna, J., W. Zaremba, A. Szlam, and Y. LeCun (2013) "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*.

[130] Defferrard, M., X. Bresson, and P. Vandergheynst (2016) "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in neural information processing systems*, pp. 3844–3852.

[131] Duvenaud, D. K., D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams (2015) "Convolutional networks on graphs for learning molecular fingerprints," in *Advances in neural information processing systems*, pp. 2224–2232.

[132] Niepert, M., M. Ahmed, and K. Kutzkov (2016) "Learning convolutional neural networks for graphs," in *International conference on machine learning*, pp. 2014–2023.

[133] Lerer, A., L. Wu, J. Shen, T. Lacroix, L. Wehrstedt, A. Bose, and A. Peysakhovich (2019) "PyTorch-BigGraph: A Large-scale Graph Embedding System," in *Proceedings of the 2nd SysML Conference*, Palo Alto, CA, USA.

[134] Xu, D., C. Ruan, E. Korpeoglu, S. Kumar, and K. Achan (2020) "Inductive Representation Learning on Temporal Graphs," *ICLR*.

[135] Hawkes, A. G. (1971) "Spectra of some self-exciting and mutually exciting point processes," *Biometrika*, **58**(1), pp. 83–90.

[136] Nickel, M., V. Tresp, and H.-P. Kriegel (2011) "A Three-Way Model for Collective Learning on Multi-Relational Data." in *ICML*, vol. 11, pp. 809–816.

[137] Yang, B., W.-t. Yih, X. He, J. Gao, and L. Deng (2014) "Embedding entities and relations for learning and inference in knowledge bases," *arXiv preprint arXiv:1412.6575*.

[138] Bordes, A., N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko (2013) "Translating embeddings for modeling multi-relational data," in *Advances in neural information processing systems*, pp. 2787–2795.

[139] Trouillon, T., J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard (2016) "Complex embeddings for simple link prediction," International Conference on Machine Learning (ICML).

[140] Hochreiter, S. and J. Schmidhuber (1997) "Long short-term memory," *Neural computation*, **9**(8), pp. 1735–1780.

[141] "Lastfm 1K Dataset," `http://ocelma.net/MusicRecommendationDataset/lastfm-1K.html`.

[142] Harper, F. M. and J. A. Konstan (2015) "The movielens datasets: History and context," *Acm transactions on interactive intelligent systems (tiis)*, **5**(4), pp. 1–19.

[143] "Yelp Dataset, Round 13, 2019," `https://www.yelp.com/dataset/challenge`.

[144] Singer, U., I. Guy, and K. Radinsky (2019) "Node embedding over temporal graphs," *arXiv preprint arXiv:1903.08889*.

[145] Zhu, Z., S. Xu, M. Qu, and J. Tang (2019) "GraphVite: A High-Performance CPU-GPU Hybrid System for Node Embedding," in *The World Wide Web Conference*, ACM, pp. 2494–2504.

[146] Kunegis, J. (2013) "Konect: the koblenz network collection," in *Proceedings of the 22nd International Conference on World Wide Web*, pp. 1343–1350.

[147] Press, W. H. and S. A. Teukolsky (1990) "Savitzky-Golay smoothing filters," *Computers in Physics*, **4**(6), pp. 669–672.

[148] Maaten, L. v. d. and G. Hinton (2008) "Visualizing data using t-SNE," *Journal of machine learning research*, **9**(Nov), pp. 2579–2605.