INFLUENCE MINING FROM UNSTRUCTURED BIG DATA

BY

SHUBHRA KANTI KARMAKER SANTU

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2018

Urbana, Illinois

Doctoral Committee:

Professor ChengXiang Zhai, UIUC, Chair
Professor Jiawei Han, UIUC
Associate Professor Hari Sundaram, UIUC
Dr. Hao Ma, Facebook

Abstract

A crucial component of any intelligent system is to understand and predict the behavior of its users. A correct model of the user behavior enables the system to perform effectively to better serve the users need. While much work has been done on user behavior modeling based on historical activity data, little attention has been paid to how external factors influence the user behavior, which is clearly important for improving an intelligent system. The influence of external factors on user behavior are mostly reflected in two different ways: 1) Through significant growth of users' thirst about information related to the external factors (e.g., user may conduct a lot of search related to a popular event or related to some community of interest), and 2) Through the user generated contents that are directly/indirectly related to the external factors (e.g. user may tweet about a particular event). To capture these two aspects of user behavior, I introduce Influence Models for both Information Thirst as well as for Content Generation, sequentially, in this thesis. To the best of my knowledge, Influence models for Information Thirst and Content Generation have not been studied before.

The thesis starts with the introduction of a new data mining problem, i.e., how to mine the influence of real world events on users' information thirst, which is important both for social science research and for designing better search engines for users. I solve this mining problem by proposing computational measures that quantify the influence of an event on a query to identify triggered queries and then, proposing a novel extension of Hawkes process to model the evolutionary trend of the influence of an event on search queries. Evaluation results using news articles and search log data show that the proposed approach is effective for identification of queries triggered by events reported in news articles and characterization

of the influence trend over time.

However, the problem formulation in the aforementioned paragraph is based on the strong assumption that each event poses its influence independently. This assumption is unrealistic as there are many correlated events in the real world which influence each other and thus, would pose a joint influence on the user search behavior rather than posing influence independently. To relax this assumption, in the next chapter of my thesis, I propose a Joint Influence Model based on the Multivariate Hawkes Process which captures the interdependence among multiple events in terms of their influence.

The second way to observe external influence on user behavior is to analyze user generated contents that are directly/indirectly related to such external factors, which I discuss in the last chapter of the thesis. For example, user generated contents are often significantly influenced by the community to which the user belongs to. While some work has been done on mining such influence from structured information networks, little attention has been paid on how to mine community-influence from user generated unstructured data. In this chapter, I introduce the problem of mining community-influence from user generated unstructured contents, particularly in the context of text content generation. Although text generation has recently became a popular research topic after the surge of deep learning techniques, existing methods do not consider community-influence factor into the generation process and thus, the processes do not evolve over time. This clearly limits their application on text stream data as most text stream data often evolve over time showing distinct patterns corresponding to the shifting interests of the target community. Thus, it is compelling to propose an Influenced Text Generation (ITG) Process that can capture this evolution of text generation process corresponding to evolving community-influence over time. In this chapter, we propose a deep learning architecture based Influenced Text Generation Process to address this challenge. Experimental results with six independent text stream data comprised of conference paper titles show that the proposed ITG method is really effective in capturing the influences of different research communities on paper titles generated by the researchers.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Twenty first century has seen the biggest explosion with respect to the generation of data. The enormous scale of web as well as increasing number of business applications has produced all different types of data including user generated contents, activity logs, time series variables, social network graphs etc. This upsurge in the volume of generated data has resulted in the inception of a new era called *Big Data*. The huge volume of *Big Data* has opened many interesting as well as open research challenges including infrastructure for *Big Data* , Big Data Management, Big Data Search and Mining, Security and Privacy in Big Data, different applications of Big Data etc. Fortunately, twenty first century has also provided us with immense computational power as well as intelligent algorithms to utilize this *Big Data* to design more intelligent and robust systems that were not feasible before. Indeed, the combination of proliferation of big data and the significant growth computational power have enabled us to perform various interesting analysis as well as design more intelligent applications which was beyond our scope before the era of *Big Data*.

The huge scale of *Big Data* comes with the fact that most of the data are unstructured and it is almost impossible for a human to comprehend the underlying patterns associated with it. However, to build an intelligent system, it is very important to discover these underlying patterns hidden inside *Big Data* and convert them into useful knowledge. It is also recommended that the discovered knowledge be as general as possible to be applicable in a wide range of application scenarios [59]. Once such useful knowledge are extracted, computers can then simulate intelligent behavior by searching for similar patterns it has already seen and making decisions based on these patterns. This is the core philosophy of

Machine Learning which has become an eminently popular research area in the last decade. *Machine Learning* is a field of computer science that gives computers the ability to learn without being explicitly programmed. Specifically, *Machine learning* explores the study and construction of algorithms that can learn from and make predictions on data. As the scale of generated data kept growing and *Big Data* eventually became a reality, application of machine learning algorithms on *Big Data* became even more popular which resulted in the birth of a new interdisciplinary field called *Data Mining*. Now-a-days, *Data Mining* is applied in many business applications that have a large amount of data generated from the system to analyze different patterns associated with it.

Data mining is the computing process of discovering patterns in large data sets involving methods at the intersection of machine learning, statistics, and database systems. The overall goal of the data mining process is to extract information from a data set and transform it into an interpretable knowledge for further use. Data mining technology has been further empowered by the availability of Big Data. The primary objective of Big Data applications is to help an organization make more informed decisions by analyzing large volumes of data. The power of big data enables a lot of important application areas including intelligent healthcare systems, Automated Quality Control in Manufacturing, Recommender Systems, E-commerce search platforms [35] and Product Review Analysis [34], User Behavior Modeling, Cyber Security and Intelligence [31], Crime Prediction and Prevention, Acceleration of Scientific Discovery, Time series analysis [54], Stock Market prediction [69] and so on. These intelligent systems help in improving our quality of life and thus, contribute to build a better society.

A crucial component of any intelligent system is to understand and predict the behavior of its users. A correct model of the user behavior enables the system to perform effectively to better serve the users need. Data mining is the most widely adopted methodology to model the behavior of users by analyzing large amount of interaction data between the system and the users. While much work has been done on user behavior modeling based on

historical activity data, little attention has been paid to how external factors influence the user behavior, which is clearly important for improving an intelligent system. The influence of external factors on user behavior are mostly reflected in two different ways: 1) Through significant growth of users' thirst about information related to the external factors (e.g., user may conduct a lot of search related to a popular event), and 2) Through the user generated contents that are directly/indirectly related to the external factors (e.g. user may tweet about a particular event). To capture these two aspects of user behavior, I introduce Influence Models for both Information Thirst as well as for Content Generation, sequentially, in this thesis.

1.1 Power of Big Text Data

This section talks about the power of big text data in greater detail. Figure 1.1 shows the loop of how *Big Text Data* can empower intelligent systems and improve the quality of our life. The loop starts with the real world where a lot of events take place everyday. To observe the different events going on around the world, humans have developed many automatic sensors/ devices to monitor these events as well as to report abnormalities associated with many real world applications. For example, in stock market, a large number of transactions happen everyday and automatic bots can monitor these transactions and also report back the stock prices on a daily/ hourly basis. Another example is the closed-circuit television, also known as video surveillance, which captures day-to-day happenings at a particular place and generates data in the video format. Other examples are thermometer/ humidity meters which report the temperature / humidity at a particular instant of time. All these are examples of physical sensors which observe the real world and generate data in some structured format upon which data mining models can be trained to learn interesting patterns and then perform predictive analysis on unseen data. Such predictive capability of Big Data makes Intelligent systems extremely useful.

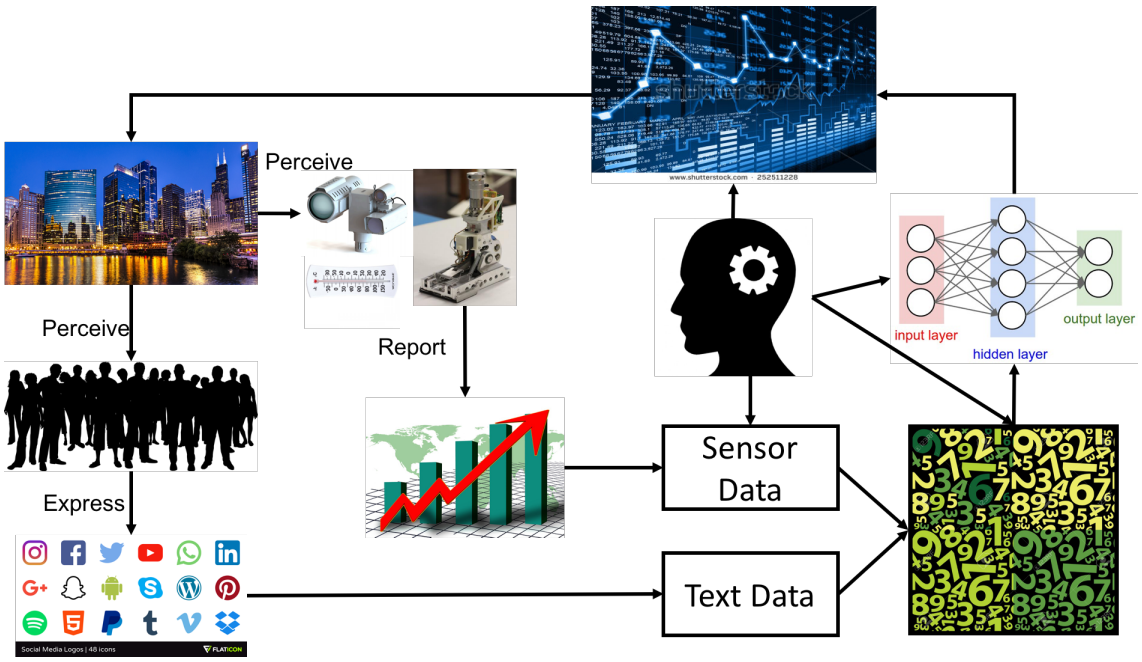


Figure 1.1: The Power of Predictive Modeling

Apart from the physical sensors mentioned above, humans themselves are another type of important sensors who observe the real world and report data in the text format. Indeed, after the proliferation of social media, blogs and forums, people talk about many different topics and events in social media and express various opinions through them. The primary format of the data generated by humans is natural language text. These user generated contents, mostly in the format of text, are equally important besides physical sensor data as they often reflect sentiment of the general mass, rapid spread of some diseases, occurrence of natural calamities etc. For example, people often tweet about US presidential campaign supporting different candidates which gives an indication about the relative popularity of different candidates. Thus, to build any intelligent system, it is crucial to leverage these text data generated by humans besides the data automatically reported by physical sensors. This means, both text data and non-text data can be used to extract features and one can then train a machine learning model to identify patterns from the features to discover new knowledge which otherwise is difficult to observe from raw data. Such new knowledge can greatly help in the decision making process tied to some particular business applications.

This in turn, results in building more intelligent systems which improve our quality of life and thus, changes the real world significantly. This new real world then generates new types of events/phenomena which the physical sensors as well as we humans observe and report to create new data, which again enable new applications and change the real world further. This loop of iterative improvement thus continues which keeps improving our quality of life.

The central focus of this thesis is to study how intelligent systems can model user behavior by mining big corpuses of text data. More specifically, this thesis presents how to model user behavior in the context of external influencing factors like popular events, shift in community interest etc. I primarily focus on two different aspects of user behavior analysis throughout this thesis including modeling of user's information thirst and user generated contents.

1.2 Challenges with Big Text Data

Text Data is an important type of data which is generated in massive amount primarily in Social Networks, Online Blogs/Forums, Online News Portals, User Search Logs etc. While structured data like information networks, connectivity graphs etc. have been vastly studied in the literature for user behavior modeling in the context of influence analysis, there has been less attention towards exploiting unstructured text data for the same. However, text data often contains interesting signals that has the potential to infer influence posed by external factors on user behavior. In spite of that, the lack of study in this domain may be attributed to the unique challenges associated with the nature of text data itself. Below, I highlight some of the challenges associated with Text Data in general.

- **Lack of Structure:** The main difference between Text data and other data in general is the lack of structure. For example, numeric datasets often contain row-column format, network data often contains specific graph format etc. However, Text data consists of natural human language which has no predefined structure. Thus, they are not directly usable as features in a predictive model and requires some non-trivial

transformations to impose some structure on them.

- **Noise:** Text data often contains a lot of noise which are irrelevant with respect to the information we seek for some particular goal task. Think about the location tagging problem, where given a sentence, the primary goal is to tag the sentence with the location it is talking about. Now, when we see the following sentence: "I went to Chicago last week and it was cold like hell". Here, the only word relevant to the tagging problem is "Chicago", while the rest are irrelevant to the given task. Filtering these noisy words is an important challenge that needs to be solved in order to use text data for knowledge mining.

- **Subjectivity:** Another challenge associated with text data is the subjectivity in human interpretation. For example, think about online customer review for some particular product like iPhone. Some customers may think the sound quality of the phone is great, while others may feel its not standard. Some may argue that a small sized screen is better while some may prefer large sized screens. Thus, the sentiment associated with human reported text data have a significant Subjectivity factor which is not observed in numerical data collected from physical sensors.

- **Ambiguity:** Ambiguity is a type of uncertainty of meaning in which several interpretations are plausible. The lexical ambiguity of a word or phrase pertains to its having more than one meaning in the language to which the word belongs. For instance, the word "bank" has several distinct lexical definitions, including "financial" institution" and "edge of a river". Context may play a role in resolving ambiguity. For example, the same piece of information may be ambiguous in one context and unambiguous in another. Thus, using context to resolve ambiguity is another challenge associated with text data.

- **Humor:** Human language often contains humors which is often very hard to model computationally. For example, think about the following sentence from a customer review about a mobile phone handset: "This phone can break down bricks!". Here the reviewer is actually making the point that the phone is very sturdy by posting a sarcastic comment. Brick is only used in the metaphorical sense here and has nothing to do with the actual phone. Understanding such subtlety associated with humorous language is another hard challenge for utilizing text data in predictive modeling.

Due to these challenges and open problems, Natural Language Understanding and Text Mining are still active research area growing significantly.

1.3 Influence Mining from Text Data

A crucial component of any intelligent system is to understand and predict the behavior of its users. A correct model of the user behavior enables the system to perform effectively to better serve the users need. While much work has been done on user behavior modeling based on historical activity data, little attention has been paid to how external factors influence the user behavior, which is clearly important for improving an intelligent system. The influence of external factors on user behavior are mostly reflected in two different ways: 1) Through significant growth of users' thirst about information related to the external factors (e.g., user may conduct a lot of search related to a popular event or related to some community of interest), and 2) Through the user generated contents that are directly/indirectly related to the external factors (e.g. user may tweet about a particular event) [refer to Figure 1.2]. To capture these two aspects of user behavior, I introduce Influence Models for both Information Thirst as well as for Content Generation, sequentially, in this thesis. To the best of my knowledge, Influence models for Information Thirst and Content Generation have not been studied before.

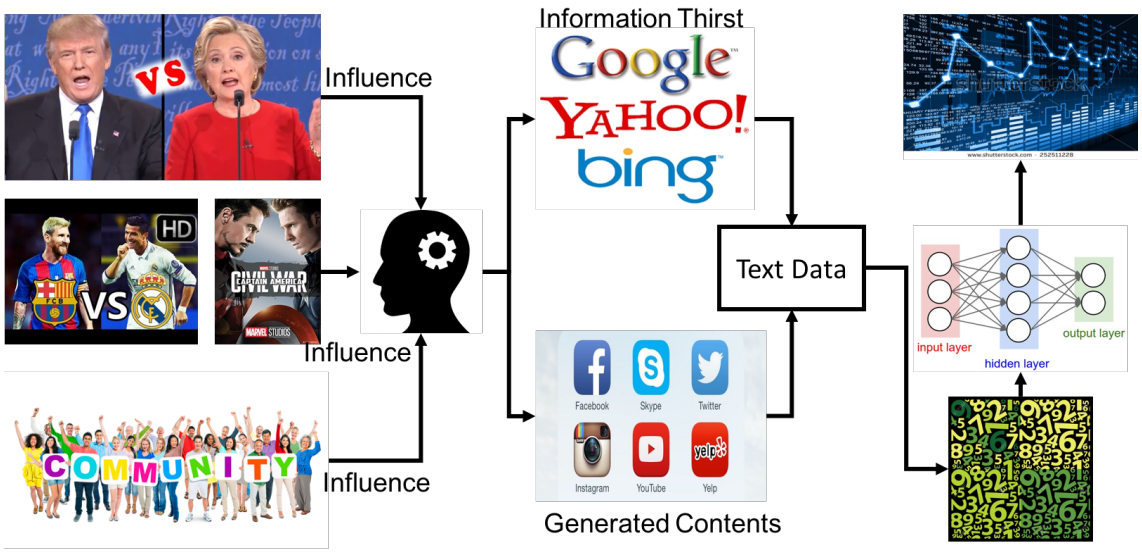


Figure 1.2: The Power of Predictive Modeling

1.3.1 Influence Models for Information Thirst

The thesis starts with the introduction of a new data mining problem, i.e., how to mine the influence of real world events on users' information thirst, which is important both for social science research and for designing better search engines for users. I solve this mining problem by proposing computational measures that quantify the influence of an event on a query to identify triggered queries and then, proposing a novel extension of Hawkes process to model the evolutionary trend of the influence of an event on search queries. Evaluation results using news articles and search log data show that the proposed approach is effective for identification of queries triggered by events reported in news articles and characterization of the influence trend over time.

However, the problem formulation in the aforementioned paragraph is based on the strong assumption that each event poses its influence independently. This assumption is unrealistic as there are many correlated events in the real world which influence each other and thus, would pose a joint influence on the user search behavior rather than posing influence independently. To relax this assumption, in the next chapter of my thesis, I propose a Joint Influence Model based on the Multivariate Hawkes Process which captures the interdepen-

dence among multiple events in terms of their influence.

1.3.2 Influence Models for Content Generation

The second way to observe external influence on user behavior is to analyze user generated contents that are directly/indirectly related to such external factors, which I discuss in the last chapter of the thesis. For example, user generated contents are often significantly influenced by the community to which the user belongs to. While some work has been done on mining such influence from structured information networks, little attention has been paid on how to mine community-influence from user generated unstructured data. In this chapter, I introduce the problem of mining community-influence from user generated unstructured contents, particularly in the context of text content generation. Although text generation has recently became a popular research topic after the surge of deep learning techniques, existing methods do not consider community-influence factor into the generation process and thus, the processes do not evolve over time. This clearly limits their application on text stream data as most text stream data often evolve over time showing distinct patterns corresponding to the shifting interests of the target community. Thus, it is compelling to propose an Influenced Text Generation (ITG) Process that can capture this evolution of text generation process corresponding to evolving community-influence over time. In this chapter, we propose a deep learning architecture based Influenced Text Generation Process to address this challenge. Experimental results with six independent text stream data comprised of conference paper titles show that the proposed ITG method is really effective in capturing the influences of different research communities on paper titles generated by the researchers.

Chapter 2

Influence Model for Information Thirst

In this chapter, I focus on modeling user's *Information Thirst* in the context of external influencing factors which impact user's information seeking behavior significantly. The text data I consider in this case are textual descriptions of external influencing factors as well as user's search activity log consisting of specific queries, timestamp of each query and the url clicked by the user after posing the query. To make the problem formulation more concrete, I choose a particular instance of *Influence Modeling* task where the goal is to model the influence of popular trending events on user search behavior by analyzing user's search query log. Search logs often contain informative signals to infer these influences indirectly by exploiting the correlation between event information and user activity information.

In summary, this chapter presents the study of how to model the influence of external events on user queries by framing it as a novel data mining problem. Specifically, given a text description of an event, I mine the search log data to predict queries that are triggered by it and further characterize the temporal trend of influence created by the same event on user queries. I solve this mining problem by proposing computational measures that quantify the influence of an event on a query to identify triggered queries and then, proposing a novel extension of Hawkes process to model the evolutionary trend of the influence of an event on search queries. Evaluation results using news articles and search log data show that the proposed approach is effective for prediction of queries triggered by events reported in news articles and characterization of the influence trend over time.

2.1 Mining the Influence of Popular Trending Events on User Search Behavior

While much work has been done on improving a search engine, little attention has been paid to how external factors influence the user search behavior, which is clearly important for improving a search engine. One important type of external factor is the trending events that "significantly" attract the general mass. Consider the following example. The hollywood movie "Captain America : Civil war" was released on May 6, 2016 and NYTimes published a review article [1] about the movie on the same day. To analyze how users search for this trending event, we collected two months (April and May, 2016) query log data from a well-known commercial search engine (https://search.yahoo.com/) and retrieved the top 500 unique queries relevant to the published NYTimes article using the BM25 [57], a state-of-the-art retrieval function. For these top 500 relevant queries, we plot their frequency distributions within the two months (April and May, 2016) in Figure 2.1. Here, the vertical red line indicates the release time of the movie (as well as the publication time of the NYTimes article about it). The x-axis represents "time in days" where the movie release time is set to be zero; the preceding and following days were set accordingly. The y-axis represents the corresponding frequency of the top 500 unique queries retrieved using BM25. From Figure 2.1, two things are evident. First, the user search activity suddenly increases near the time when the event occurred and second, the activity exponentially goes down as we move away from the origin. This confirms the fact that the "release" of the movie triggered a lot of user queries asking for relevant information, thus influenced user search behavior "significantly".

How can we computationally model and analyze the influence of such trending events on user search behavior? What kind of queries are triggered by what kind of events? What kind of events tend to be most influential? How long does the influence last? Can we predict whether a user's query was triggered by a particular event? Besides being interesting

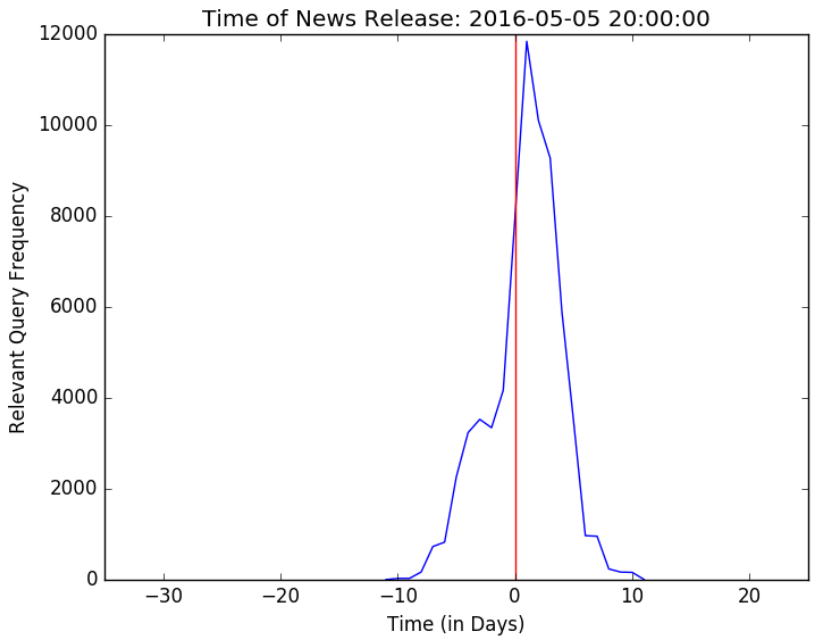


Figure 2.1: User search activity related to the release (May 6, 2016) of the Movie "Captain America Civil war".

social science research questions, these questions are also interesting from the perspective of improving the utility of a search engine. For example, if we can detect when a user's query is triggered by a particular event, it would help improve query auto-completion (by leveraging the terms occurring in news articles about the triggering event), improve the search results (by recommending relevant topics to the event), improve future query volume prediction, and detect influential events (which can then be recommended to users that might be searching for related topics).

To the best of our knowledge, the questions mentioned above have not been addressed in the existing work. In this chapter, we conduct the first study of the problem of modeling the influence of trending events on users' search behavior from the perspective of data mining. Specifically, we frame the problem as a novel data mining problem where, given a text description of an event, we mine the search log data to identify queries that are triggered by it and further characterize the temporal trend of influence created by the same event on user queries. From data mining perspective, such a joint mining problem is novel and presents

interesting challenges. First, “Influence" is an abstract concept and thus, it is not straightforward how to measure influence. Second, how can we fully characterize the influence of an event on search queries? Finally, how should we formally define this data mining problem?

To tackle these challenges, we focus on studying how an event might have triggered queries from users. This restrictive perspective allows us to quantitatively measure “influence" based on the number of queries triggered by the event. To determine whether a query is triggered by an event, we propose to make the decision based on both textual similarity and temporal proximity between the clicked web documents by users filing the query and the text description of the event. To capture how the influence of an event on user queries evolve over time, we propose a formal model based on the Hawkes process to characterize the trend of influence.

We evaluate the proposed influence modeling methods using two sets of data-sets: 1) NYtimes articles: we collected the most read articles from NYtimes during two months span of April and May, 2016 using the NYtimes developers API to use them as trending events and 2) Query-log data: We also collected two months contemporary query log data from a widely used popular search engine (https://search. yahoo.com).

Evaluation results using these data sets show that the proposed approach is effective for identification of queries triggered by events reported in news articles and characterization of the influence trend over time. We further show that the proposed extended Hawkes model is useful in many ways, including improving the accuracy of predicting whether a newly entered query by a user was triggered by an event (which further enables a search engine to optimize its response to the query accordingly) and answering many interesting questions related to understanding the influence of events on queries.

In Summary, we make the following contributions in this chapter: (1) We conduct the first study of modeling the influence of trending events on search queries and frame the problem as a new data mining problem. (2) We propose a computational method for measuring the influence of an event on a query and discovering triggered queries by an event. (3) We

propose a novel extension of Hawkes process to model the influence trend of an event on user queries over time. (4) We propose a way to quantitatively evaluate an influence model using the task of predicting whether a newly entered query by a user has been triggered by some event, and show that both the method for measuring influence and the extended Hawkes process are useful for this prediction task, and they can be used immediately in a search engine to potentially customize the response of the search engine to a user's event-triggered query based on the event.

2.2 Related Work

Search query logs have been extensively studied to understand user search behavior and provide better search experience [28, 42, 72]. Existing work mostly focused on the inference of users' search intent based on their own search habit and search history. On the other hand, this chapter tries to model how user behavior on a search engine is influenced by external factors such as trending events.

Temporal Information Retrieval and Event Detection are two areas closely related to our work. While Event Detection has been studied vastly in the literature (see [4] for a recent survey), research interest on Temporal Information Retrieval has grown recently [11]. However, we want to emphasize that, neither of these is the intended goal of this study and our primary motivation is somewhat orthogonal. To be more specific, our work does not intend to study how time-sensitive information needs can be addressed [16, 7] or how users' information need change over time [41] or how to detect some events from social networks/news media [4]. Rather, given that some event has already been reported, we go one step further to investigate how the event may impact/influence the search behavior of the users.

The notion of event-based retrieval was introduced by Strötgen and Gertz [63] by returning events instead of documents. Zhang et al. [75] addressed the detection of recurrent event

queries. Ghoreishi and Sun [22] introduced a binary classifier for detecting queries related to popular events. Kanhabua [30] extended the work [22] by enabling the classifier to detect less popular queries beside popular ones. However, all these approaches are supervised classification methods and largely depend on the quality of training labels provided by humans, whereas our approach is completely unsupervised.

Kairam et. al. [29] investigated the online information dynamics surrounding trending events, by performing joint analysis of large-scale search and social media activity. Matsubara et. al. [48] presented a new model for mining large scale co-evolving online activities. Pekhimenko et al. [53] designed a system named “PocketTrend" that automatically detects trending topics in real time, identified the search content associated to the topics, and then intelligently pushed this content to users' local machine in a timely manner. However, none of these studies provide answer to the question: how to model the temporal trend of influence created by an event on user queries, which is one of the primary motivations of our work.

Another important topic related to this chapter is point process, which has been used to model social networks [8] and natural events [78]. People find self-exciting point processes naturally suitable to model continuous-time events where the occurrence of one event can affect the likelihood of subsequent events in the future. One important self-exciting process is Hawkes process, which was first used to analyze earthquakes [78], and then widely applied to many different areas, such as market modeling [20], crime modeling [62], conflict [73], viral videos on the Web [15] etc. In this work, we extend the original Hawkes process to propose a new model that can capture the dynamics of influence by trending events on user search behavior.

2.3 Problem Formulation

We solve the problem of modeling the influence of an event on search queries by framing it as a novel data mining problem where we would jointly mine two different types of data,

i.e., text data describing many events and search log data that contains user queries and clickthrough records. We assume that each event, E, is represented as a tuple $<W_E,t_E>$, where, W_E is some natural text description of that event (e.g., some news article about that event) and t_E is the publication timestamp of W_E. A query is represented as a tuple with three attributes, i.e., $<W_q,t_q,U_q>$. Here, W_q is the set of keywords that query q contains, t_q is the timestamp of the query submission and U_q is the URL that the user clicked after posing the query. The desired output includes the following three elements:

1. Influential Events (or "Trending Events"): An influential event is any event E that attracts the interest of the general mass significantly and has triggered queries from many users. We want to discover a set of most influential events from the data sets.

2. Triggered Queries: A triggered query (denoted by q) by event E is a query entered by a user due to knowing information about event E. In other words, had the user not heard about E, he/she would not have entered query q. For each influential event E, we want to discover all the triggered queries by the event.

3. Influence Trend Model (or just Influence Model): An influence trend model for event E is a parameterized process that can model the temporal trend of influence by event E on user search queries. The parameter values of the model should be interpretable for characterizing how the intensity of the influence evolves over time.

The rationale of requiring the model to be parameterized with interpretable parameters is so that we can use the parameter values to obtain a concise quantitative summary of the dynamics of the influence from an event, which is essential for enabling many interesting applications of influence analysis (e.g., answering questions such as "how quickly does the influence intensity grow over time?" and "how quickly the influence disappears?")

Our problem formulation would enable many interesting applications, particularly for understanding what kind of events tend to have more significant influence on user queries, what kind of queries were triggered by a particular event, and how the influence evolves over time. Such analysis can be potentially configured to compare different kinds of events,

similar events reported in different time periods or by different sources, and to compare different user groups to understand how different groups of users may have been influenced by the same event in different ways.

2.4 Methods for Influence Discovery and Characterization

To solve the proposed influence mining problem, we need to complete three subtasks: 1) Discovery of events that have significantly influenced user queries, which we will refer to as influential events. 2) Discovery of queries influenced by any influential event, which we will refer to as event-triggered queries, or simply triggered queries. 3) Characterization of the temporal trend of the influence of an influential event on user queries over time. All these tasks are new tasks that are challenging due to the lack of labeled data for supervised learning. They have not been studied in the previous work, thus we do not have natural baseline methods to start with either. Below we present our proposed unsupervised method for solving all the three problems.

A careful analysis of these tasks suggests that subtask one and subtask two both rely on solving the basic problem of determining whether one event has influenced a query. Once we can do that, we would be able to quantify the influence of an event by counting how many queries are influenced by the event, and also easily obtain which queries are influenced by which event.

To solve subtask three, we propose to model the frequency of triggered queries over time with a temporal process model based on the Hawkes process, which assumes that the frequency of triggered queries at time t is a function of a base frequency λ_0 capturing the general popularity of this kind of queries, how quickly the influence decreases over time (captured by a parameter β), and the homogeneity of user search behavior over time (captured by a parameter α). The advantage of such a model is that by fitting the model

to our observed frequency of triggered queries, we can obtain these meaningful parameters that are directly useful for characterizing the trend of influence.

2.4.1 Discovery of influential events and triggered queries

Our basic problem is the following: Given a query and an event, how can we know whether the event has influenced the query? Due to the complexity of the notion of influence, a completely rigorous definition of influence is nearly impossible. To make the problem more tractable, we propose three reasonable heuristics to guide us in designing a computational measure of influence. Specifically, given an event $E =< W_E, t_E >$ and a particular query submission $q =< W_q, t_q, U_q >$, we can reasonably make the following three assumptions about influence, which would help us design a function to computationally measure the influence of E on q. Here, we denote the content of the clicked-URL, i.e., $content(U_q)$ simply by W_U.

Assumption 2.4.1 (Query-Textual-Similarity)**.** *The higher the textual-similarity between W_E and W_q, the higher the chances that q is triggered/influenced by E. This assumption allows us to prune cases where the query is completely irrelevant to the event.*

Assumption 2.4.2 (Temporal-Similarity)**.** *The higher the temporal-similarity between t_q and t_E, the higher the chances that q is triggered/influenced by E. This assumption allows us to distinguish queries triggered by similar events in the past from those triggered by a current event.*

The Temporal-Similarity is important because Query-Textual Similarity alone is insufficient. For example, consider the two trending events “US election 2012" and “US election 2016". Now, if a user poses a query “US election", it is hard to tell which event actually triggered the query submission. However, if we know the timestamp of the query submission, we can better predict the triggering event. For example, if the query was posed in the year 2016, then with high probability, it was triggered by the event “US election 2016" as it was trending at that moment. On the other hand, if it was posed in the year 2012, probably the

triggering event was "US election 2012". Thus, besides textual similarity, temporal similarity also plays an important role in predicting the influence.

Another useful piece of information that helps to verify whether some event E indeed influenced the submission of query q is the content of the URL which the user clicked after posing the query. If the content of the clicked-URL, i.e., U_q, is highly similar to the text description of event E, that means the user was actually looking for news about the same event. This, in turn, means that query q was influenced/triggered by event E. This gives us our third heuristic:

Assumption 2.4.3 (ClickedURL-Textual-Similarity)**.** *The higher the textual-similarity between W_E and W_U (W_U is the text of the clicked documents by users who entered query q), the higher the chances that E triggered/influenced q.*

Intuitively, we would like to design a measure that combines all the three heuristics so that a query-event pair would be scored high if (1) the query text is similar to the event text description, (2) the clicked documents are similar to the event text description, and (3) the time stamp of the query and that of the event are close. One way to combine them is to design a similarity/distance function for each of these three dimensions and combine the three functions into one single scoring function. Specifically, we use the the following scoring function to measure the influence:

$$F(E, q) = TxtSim(W_E, W_q) \cdot TmpSim(t_E, t_q) \cdot TxtSim(W_E, W_U) \tag{2.1}$$

We discuss these components in more detail below:

$\boldsymbol{TxtSim(W_E, W_q)}$: Similarity between query-document pair has been studied in the literature for a long time. One popular function from the literature is the "Okapi BM25" ranking function [57]. However, we could not use "Okapi BM25" directly for the major limitations discussed below.

First, each "event-text" (description of the event) usually contains a title and a body. The title often contains more important words pertaining to the event, while the body contains verbose details. It is thus necessary to put more emphasis on matching the title keywords first and then match the body details. The original "BM25" similarity function unfortunately does not provide such customizations. "BM25F" [74], an extension of BM25, handles this relative term weighting scenario, although "BM25F" is also not directly applicable to our problem setting for the following reason: To be able to compare the influence across different trending events, it is necessary that the "BM25F" score computed for different pairs of "event-text" and "query-text" be comparable. However, this is not the case because there is a large variance in the length of both "event-text" and "query-text". One might argue that, "BM25F" provides "Document Length Normalization" and "Relative Term Weighting", which should resolve the problem. But ones careful attention would reveal that "BM25F" is designed for a setting where the information need, i.e., the query is constant and only the document is varied to compute the similarity. But, in our case, both the document and query are variable and thus, we need both "document length normalization" and some kind of "query length normalization".

To address the two issues mentioned above, we use the following *modified* version of BM25 as the TxtSim function to fit our problem setting. Let, $W_E =< W_{E_1}, W_{E_2}, ..., W_{E_n} >$ be the "event-text" and $W_q =< W_{q_1}, W_{q_2}, ..., W_{q_n} >$ be the "query-text".

$$TxtSim(W_E, W_q) = \sum_{i=1}^{|W_E|} \frac{\omega(W_{E_i}).IDF(W_{E_i}).TF(W_{E_i}, W_q).(k_1+1)}{TF(W_{E_i}, W_q) + k_1.(1 - b + b.\frac{|W_q|}{avgql})}$$

$$\text{subject to} \sum_{i=1}^{|W_E|} \omega(W_{E_i}) = 1 \tag{2.2}$$

Note that, equation 3.1 is similar to the original "BM25" with the exception of the new term $\omega(W_{E_i})$ and the constraint that the weights must sum to 1. $\omega(W_{E_i})$ is essentially the

weight of each n-gram in the "event-text" which reflects the importance of that particular n-gram with respect to the "event-text". $\omega(W_{E_i})$ allows the $TxtSim(W_E, W_q)$ to be comparable across different W_E and W_q as we enforce the constraint $\sum_{i=1}^{|W_E|} \omega(W_{E_i}) = 1$. Furthermore, one can easily set $\omega(W_{E_i})$ in such a way so that title n-grams get more weight than body n-grams as well as bigrams get more weight over unigrams and vice-versa. The specific weights we used for our experiments are mentioned in section 3.6. The IDF (inverse document frequency) and TF (term frequency) bear the usual meaning as in the original BM25 function.

$\boldsymbol{TxtSim(W_E, W_U)}$: $TxtSim(W_E, W_U)$ is basically the similarity between a pair of documents, in contrast with $TxtSim(W_E, W_q)$, which is the similarity between a query and document pair. $TxtSim(W_E, W_U)$ is almost similar to $TxtSim(W_E, W_q)$ with the exception that $TxtSim(W_E, W_q)$ contains only one ω (for W_E), while $TxtSim(W_E, W_U)$ contains two, i.e., ω_1 and ω_2, where ω_1 and ω_2 are weight distributions for the event-text (W_E) and clicked-url-content (W_U) respectively.

$$TxtSim(W_E, W_U) = \sum_{i=1}^{|W_E|} \omega_1(W_{E_i}) \cdot \omega_2(W_{E_i}) \cdot IDF(W_{E_i}) \cdot TF(W_{E_i}, W_U)$$

subject to,

$$\sum_{i=1}^{|W_E|} \omega_1(W_{E_i}) = 1 \ , \sum_{i=1}^{|W_U|} \omega_2(W_{U_i}) = 1 \ , \ \omega_2(W_{E_i}) = 0 \text{ if } E_i \notin W_U \tag{2.3}$$

The essence of equation 2.3 is that matching an n-gram which has high weights for both W_E and W_U contributes more to the similarity between W_E and W_U, whereas, n-grams having low weights for one/both of the articles contribute less to the similarity.

$\boldsymbol{TmpSim(t_E, t_q)}$: The $TmpSim$ function is expected to behave in the following way: if two events are far distant in time, their temporal similarity should be low; whereas, if they

are close in time, the temporal similarity should be high. We also assume that the temporal similarity decreases exponentially as the distance in time increases. Below is an example of a function with such desired properties, where δ is the decaying parameter:

$$TmpSim(t_E, t_q) = e^{-\delta.|t_E - t_q|} \tag{2.4}$$

2.4.2 Influence Trend Modeling

Once we can measure the influence an event E has over different user queries, the next task is to model the trend of such influence over time. Such modeling would allow us to study the characteristics of influence in a systematic way and enable many interesting applications like predicting future volume of queries, optimizing search recommendations etc. To accomplish this, we propose function $Trend(E, t)$, which takes as input an event E and timestamp t and returns the popularity/trendiness of event E at timestamp t. There are many ways one can define how to measure the popularity/trendiness of an event. For example, the number of tweets related to the event, number of views for news articles relating to the event, click counts for the event webpage, number of social media posts sharing the event etc. In this work, we define popularity/trendiness of an event by the users tendency to pose queries that are relevant to the event. We choose this definition because we are specifically interested in modeling the influence of trending events on user search behavior.

Defining the $Trend(E, t)$ is not trivial. First, we introduce a set of assumptions that will help us formalize the notion of "trendiness".

Assumption 2.4.4 (Influence Growth)**.** *Each query submitted to search engine τ that is relevant to event E increases the chance of subsequent submission(s) of relevant queries to τ, thus, grows the trendiness/influence of event E.*

Assumption 2.4.4 simply says that each relevant query submission from one user indicates an increase in the tendency of other users to pose similar relevant queries. In other words, the trendiness of an event is directly/indirectly influenced by the previous query submissions

relevant to the same event, which in turn, reflects the tendency to receive new queries relevant to the event. To see the rationale of Assumption 2.4.4, consider a scenario when a user is exposed to an event that he/she feels interested about, the user may use multiple queries to find out more about the event and then share the event details on some social media platform or talk to some friends about the event. These friends, being interested in the event after hearing about it, may do further search. Thus, the influence of the event propagates from one user to another and reflects in their search activity. As another example scenario, say some popular news portal publishes a featured article about some event. Many people would then read that article to know about the event/incident and start searching for more details about it. In this case, new incoming queries searching about a particular event gives useful indication about further submission of similar relevant queries. Thus, a significant number of relevant queries actually indicate the growth of the trendiness/influence of the event.

Given that the "trendiness" of an event, E, at moment t is dependent on all the relevant (w.r.t. E) queries posed before timestamp t, the next obvious question is: to what extent each of the previous queries contribute to the current "trendiness"? To answer this, we make two further assumptions as mentioned below.

Assumption 2.4.5 (Query Relevance)**.** *The contribution of a query q (submitted at time t_q) to the "trendiness" of an event E at moment t, where $t > t_q$, is proportional to the textual similarity between the "event-text", W_E and the "query-text", W_q, i.e., $TxtSim(W_E, W_q)$.*

Assumption 2.4.6 (Query Timestamp)**.** *The contribution of a query q (submitted at time t_q) to the "trendiness" of an event E at moment t, where $t > t_q$, exponentially decays as the difference between t and t_q increases.*

Assumption 2.4.5 and 2.4.6 are very reasonable. Assumption 2.4.5 basically says that, highly relevant queries grow the trendiness of an event, thus, indicates the growth in the volume of future relevant queries; at the same time, Assumption 2.4.6 says that the contribution of a past query to the current "trendiness" of the event decays exponentially with time.

A parametric model for influence

Incorporating these two assumptions, we introduce Equation 2.5 presented below to compute the "trendiness" of an event E at time t.

$$Trend(E,t) = \lambda_0 + \sum_{i=1}^{n} \alpha \cdot TxtSim(W_E, W_i) \cdot e^{-\beta(t-t_i)} \tag{2.5}$$

Equation 2.5 contains three parameters, i.e., λ_0, α and β. λ_0 is a constant which reflects the base trendiness that is assumed to be always present. α is a scaling factor to control the contribution of $TxtSim(W_E, W_i)$ on the current "trendiness" and β is the scaling factor to control the exponential decay in time. W_1, W_2, ..., W_n represents all the queries relevant to event E that were posed before timestamp t.

Equation 2.5 is not entirely new; it is similar to the self-exciting point processes [17] e.g. Hawkes Process [25]. However, the point processes models the recurring events of the same type, whereas our task is to model influences of one type of event (e.g. Trending articles) on other type of events (e.g. user search behavior). Thus, we include the textual-similarity between the past queries and event-text, i.e., $TxtSim(W_E, W_i)$, into the basic Hawkes process to fit our problem scenario.

Figure 2.2 shows a hypothetical simulation of how equation 2.5 works. Without loss of generality, we assume that $TxtSim(W_E, W_i) = 1.0$ for any choice of W_E and W_i. However, this choice does not affect our attempt to present the spirit of equation 2.5. The x-axis in Figure 2.2 represents time and the y-axis represents the corresponding "trendiness" of some hypothetical event E. The "Blue" dots represent each query submission that is relevant to E. These "Blue" dots were generated by simulating the Hawkes Process (see [51] for details). For this particular simulation, λ_0, α and β were set to 0.5, 2.5 and 3.0 respectively. This means, there is always a base trendiness of λ_0 equal to 0.5. The "trendiness" goes up as people start querying about the event E (note the first blue dot), which, increases the chance of generating further queries. Thus, the volume of queries influenced by E and

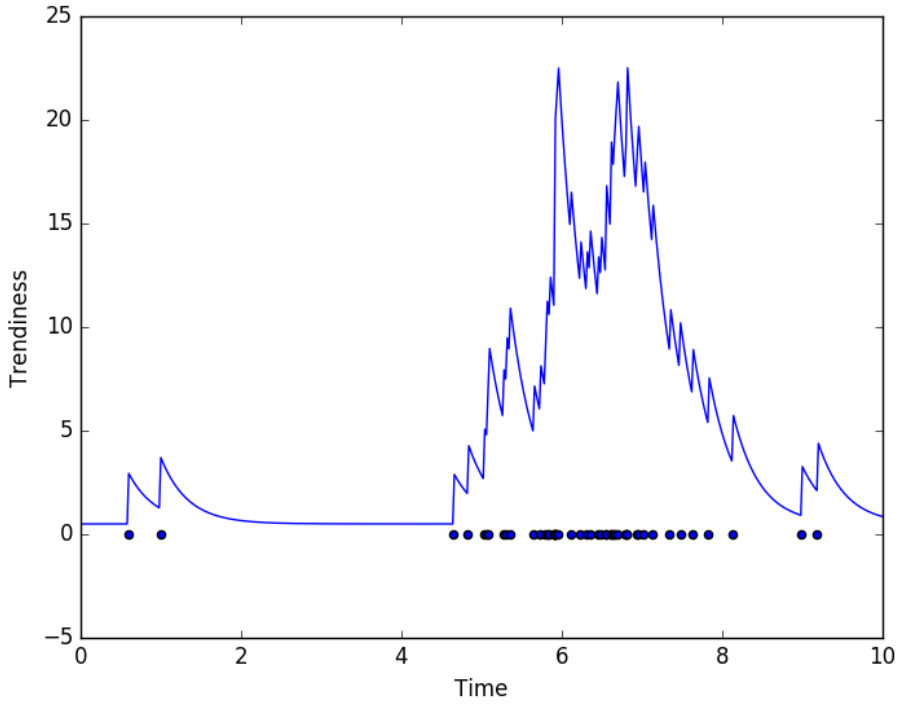


Figure 2.2: Simulating the trend of some hypothetical event.

"trendiness" of E grows mutually by enhancing each other. For example, at the vicinity of time zone $t = 6$, a lot of queries were posed which resulted in further query submissions and the "trendiness" rises up significantly. The "trendiness" goes down exponentially with time if no further queries are posed. This is signified by the exponential decay near time zone $t = 8$. Note that, the current estimate of "trendiness" is directly correlated to the expected volume of relevant queries in near future. The higher the current "trendiness" is, the more the probability of observing high volume of relevant queries in future. However, the estimate of "trendiness" is incrementally updated as we move forward in time and observe (do not observe) new user queries.

Estimation of the parameters

Let the set of parameters be $\Lambda = \{\lambda_0, \alpha, \beta\}$. We adopt maximum likelihood estimation technique to find the optimal parameter values for equation 2.5. First, we show how to compute the log-likelihood for a single event E and then we extend it to multiple events

case. Considering the query submission sequence q_1,q_2, ..., q_n (all related to event E) as a simple point process, the likelihood for a single trending event can be written as follows (see [51] for background):

$$\log L = \int_0^{t_{q_n}} (1 - Trend(E,t))dt + \sum_{i=1}^{n} \log(Trend(E, t_{q_i})) \tag{2.6}$$

After some simple mathematical operations, equation 2.6 boils down to the following form:

$$\begin{aligned} \log L = &- \left(\frac{\alpha}{\beta}\right) \cdot TxtSim(W_E, W_{q_i}) \cdot \left\{1 - e^{-\beta \cdot (t_{q_n} - t_{q_i})}\right\} \\ &+ t_{q_n} - \lambda_0 \cdot t_{q_n} + \sum_{i=1}^{n} \log(Trend(E, t_{q_i})) \end{aligned} \tag{2.7}$$

Given the close form of the log likelihood function (equation 2.7), the optimization problem to find the optimal parameter set Λ^* is written as follows:

$$\Lambda^* = \arg\max_{\Lambda} L(\{q_1, .., q_n\}|E, \Lambda) \tag{2.8}$$

For multiple events $E_1, E_2, ...E_m$, the optimization problem is extended in the following way:

$$\Lambda^* = \arg\max_{\Lambda} \sum_{j=1}^{m} L(\{q_{j1}, .., q_{jn}\}|E_j, \Lambda) \tag{2.9}$$

One can use any non-linear optimization method to solve this maximization problem. Nelder-Mead Simplex Method [23] is one such popular optimization technique. Another useful approach is the Sequential Least SQuares Programming (SLSQP) [9].

Category	# of articles	Avg. Title Length	Avg. Body Length
Movies	25	18.88	458.08
Sports	15	19.53	508.4
US	18	20.38	487.77
World	11	18.18	438.81
Total	69	19.30	473.69

Table 2.1: Description of Trending-Event data set

2.5 Experimental Design

Data Sets: We collected two sets of data sets: one for trending events and one for user query history. We call these two data sets *Trending-Event* dataset and *Query-Log* dataset respectively. The following two paragraphs provide details about these two data-sets:

Trending-Event data-set: An obvious choice for a text data set describing events is news articles (though other data such as social media might also be applicable). The NYTimes Developers Network (thanks to them) provides a very useful api called *"The Most Popular API"* [2], which automatically provides the *url's* of the most e-mailed, most shared and most viewed articles from NYTimes.com during the last month from the date of the issue of the query. We chose to use this API because of two major benefits: 1) it automatically removes duplicate articles, thus we don't need to deal with cases where multiple articles are related to the same event. 2) it only provides the most popular articles from NYTimes, thus the quality/accuracy of the events represented by these articles is very high. Using this API, we collected the most e-mailed, most shared and most viewed articles from the two months span: April and May, 2016. Each article consists of a tuple $<$*title-text*, *body-text*, *timestamp*$>$. Among different categories of news, we used only four categories for our experiments: *US (National Affairs)*, *Movies*, *Sports* and *World (International Affairs)*. Table 2.1 shows some details about the data-set.

Category	Total query	% Pos. instance	% Neg. instance	Avg. txt-sim
Movies	193,282	16.24	83.75	2.49
Sports	616,449	0.84	99.15	2.48
US	204,926	33.72	66.27	1.99
World	22,197	7.68	92.3	1.96
Total	1,036,854	10.35	89.64	2.38

Table 2.2: Description of Query-Log data set

Query-Log data-set: To analyze the user queries contemporary to the articles in *Trending-Event* data-set, we use the two-months (April and May, 2016) user query log data from the widely used search engine at https://search.yahoo.com. Each query submission q is represented as a tuple $<$*query-text*, *timestamp*, *clickedURL*$>$. The two-months query log data contains $105,925,732$ query submissions in total. To keep the computation feasible, for each article E in the *Trending-Event* data-set, we retrieved top 500 unique (in terms of text) queries that has at least a similarity score of 1.5 (with respect to E) according the textual similarity function in equation 3.1 and discarded the rest. This filtering step is reasonable because if the textual similarity is very low (less than 1.5), we assume that the influence prediction problem becomes trivial, i.e., there is no influence of E on the query. Thus, textual similarity itself is sufficient in this case to decide whether there is an influence or not. However, more challenging cases are when the query shares a high degree of textual similarity to the event E, but still is not influenced by the event. In this chapter, we focus on these type of queries with significant textual similarity to the event and assume that the other queries are not influenced by any event in our data set. The summary of this data-set is presented in Table 2.2.

Predictive modeling for quantitative evaluation: Quantitative evaluation of the mining results pose challenges because there is a lack of gold standard for what events are influential and the ground truth for the true influence trend of an event. We overcome this

difficulty by proposing a way to perform indirect quantitative evaluation based on the task of predicting whether a user's newly input query is triggered by an event. The prediction setup is intended to simulate a real application scenario when a search engine receives a query from a user. In such a scenario, it would be beneficial for the search engine to "know" whether this query was triggered by a particular event since if it was, then the search engine would be able to leverage this knowledge to optimize the search results to be presented to the user (e.g., recommending content related to this event).

With such a setup, we can use the component techniques in our proposed mining approach, including text similarity functions, temporal similarity function, and the extended Hawkes model, to construct a prediction model to attempt to predict whether a "new" query in a separate held-out search log data set is influenced by any event based solely on the query without using any clickthrough information. The clickthrough information, however, is only used to create the gold standard labels for the evaluation purposes, i.e., whether such a query is indeed triggered by an event (we used equation 2.1 from section 2.4.1 along with threshold 0.01). One can argue that a better way to create the gold standard labels is to involve human judgments. However, for our data-set, this means the human annotators would have to go through $1,318,359$ <event, query, url-content> triplets, which is practically infeasible. So, we had to opt for some automated techniques for annotating the gold standard labels.

To evaluate the quality of the gold standard labels created by our automatic approach, we randomly sampled 200 positive and 200 negative examples labeled by the automatic process. Then, we asked three volunteers to independently go through these 400 <event, query, url-content> triplets and manually label each of them with 1 if, after reading the event description and contents of "url-content", the annotator thinks the query was indeed influenced by the event or 0, otherwise. We computed Cohen's kappa coefficient [14] to measure the inter-rater-agreement which was found to be reasonably high, i.e., 0.835. Thus, we conclude that the gold labels created by our automatic approach is reliable.

The labeled data-set created in the way previously described is highly imbalanced as most of the queries are not influenced with respect to some particular event E. To make the data-sets balanced, we randomly under-sampled from the pool of the negative samples to match the size of the positive examples for reporting the results in section 3.6. Concretely, we can use the following equation (eqn. 2.10) to compute the influence relation between event E and query q without using the clickthrough information and then pick a reasonable threshold to separate the influenced queries from the rest. We did threshold analysis which showed that the prediction model remains very stable for wide range of threshold value, i.e., $[0.8, 5]$ and we chose 1.0 for our experiments (the details are omitted due to lack of space). We call this model the "IP" (Influence Prediction) model.

$$F(E, q) = TxtSim(W_E, W_q) \cdot TmpSim(t_E, t_q) \cdot Trend(E, t_q)) \tag{2.10}$$

Performance Metric: To evaluate the performance of the proposed predictive model, i.e., IP, we use the four popular measures available in the literature: *precision*, *recall*, *specificity* and *F-measure* (see [24] for details). We also present results for the recently introduced K-measure [60] to show that "IP" model achieves better performance in terms of this new measure too.

2.6 Results

In this section, we report our experimental findings including both qualitative and quantitative evaluation results. We first start with some implementation details, then describe the qualitative and quantitative evaluation sequentially.

2.6.1 Implementation Detail

For the weight distribution ω in equation 3.1, we followed the weighting scheme presented in Table 2.3. This significance of Table 2.3 is that it puts more weight on the title-text matching

	Bigram	**Unigram**	**Sum**
Title	0.49	0.21	0.70
Body	0.21	0.09	0.30
Sum	0.70	0.30	1.00

Table 2.3: Weight allocation for Title vs Body and Unigram vs Bigram in equation 3.1.

(0.7) in comparison to the body-text matching. Similarly, it puts more weight on bigram-matching (0.7) in comparison to unigram-matching (0.3). An immediate consequence is that, bigram-matching in the title-text gets the highest reward (0.49), whereas, unigram-matching in the body-text gets the least reward (0.09). In other words, the weights for all bi-grams in the title of the event-text summed up to 0.49 and the weight of each individual bi-gram is proportional to its term frequency in the title-text. Similarly, the weights for all unigrams in the body of the event-text summed up to 0.09 and the weight of each individual unigram is proportional to its term frequency in the body-text. The same weighting scheme was used for $\omega_1(W_{E_i})$ and $\omega_2(W_{E_i})$ in equation 2.3.

The "trendiness" parameters, i.e, $\Lambda = \{\lambda_0, \alpha, \beta\}$ are learnt automatically using equation 3.13 (see table 2.6 for the exact values). Parameter δ (equation 2.4) was heuristically set to 0.8. This heuristic value is not an issue because for all the variants of the "IP" model (mentioned in table 2.7), we use the same δ value, thus, the optimal value is irrelevant for comparative analysis.

2.6.2 Qualitative evaluation

We show some sample data mining results to analyze their quality. First, we show the top four most influential events for each category in our data set as measured by the overall/total number of triggered queries in Table 2.4. They are all intuitively influential events. For example, the top one for category "Movies" is the release of "Caption America : Civil War", while the top one for category "World" is the "Panama Papers leaked".

Second, we show a sample of triggered queries with highest frequency for the event "curt

#	**Movies**	**Sports**	**US**	**World**
1	“captain America: Civil War" released	san antonio spur vs oklahoma city thunder basketball	harriet tubman ousts andrew jackson	panama papers leaked
2	alden ehrenreich Cast “hail caesar"	Rise of leicester city in premiere league	donald trump comments on transgenders toilet use	sadiq khan elected in london
3	gosling and Crow star “nice guys"	curt schilling fired from espn	donald trumps running mate	philippine presidential election
4	ken loach wins Palme dor	western conference finals	indiana primary elections	brazil president impeachment

Table 2.4: Top influential events for different categories.

#	**curt schilling fired from espn**	**panama papers leaked**
1	curt schilling espn	panama paper leak
2	espn curt schilling suspension	celebrity involved in panama offshore account
3	curt schilling facebook post	panama paper politicians
4	curt schilling comment	panama paper american
5	curt schilling blog	panama paper law firm

Table 2.5: Popular queries triggered by influential events.

schilling fired from espn" and event “panama papers leaked" in Table 2.5. We see that these queries are indeed well associated with these events.

Finally, we examine the optimal parameters learned by fitting the modified Hawkes model in Table 2.6.

Interpreting Model Parameters: We focus on interpreting the optimal values of the modeling parameters, $\Lambda = \{\lambda_0, \alpha, \beta\}$, which are automatically learnt by the estimation technique introduced in section 3.4.2. Table 2.6 shows these values. Indeed, these values have intuitive interpretation that matches our real-life expectation. For example, λ_0 essentially reflects the general interest in posing queries related to some trending event. Table 2.6 shows that people usually have the most interest ($\lambda_0 = 0.0656$) in the US category, i.e.,

Category	λ_0	α	β
Movies	0.0420	0.9082	2.6539
Sports	0.0146	0.4810	1.0208
US	0.0656	1.0892	2.3727
World	0.0117	0.1464	0.3292

Table 2.6: Trendiness parameter for different types of events.

events related to the national affairs. While the international affairs, i.e., "World" category generally draws the least attention ($\lambda_0 = 0.0117$). Next, α models the degree of homogeneity in user search behavior. So, high value of α means high degree of similarity in the intra-community search pattern. For example, in case of the category "US" ($\alpha = 1.0892$) and "Movies" ($\alpha = 0.9082$), we found the homogeneity to be significantly higher than for the category "Sports" ($\alpha = 0.4810$) or "World" ($\alpha = 0.1464$), indicating the search behavior is more diverse and discrete for the "Sports" and "World" category. Finally, β models the decay in user interest with time; thus, high value of β indicates a quick drop of interest among the general mass. As expected, β obtained for the "Movies" category ($\beta = 2.6539$) was found to be the highest, as people usually talks a lot about movies when they get released and the topic disappears quickly in few days. However, to our surprise, we also obtained high value of β for the "US" category ($\beta = 2.3727$). One plausible explanation for this fact may be that there are too many national news to follow and people switch their interest from time to time following different national news. On the other hand, value of β for the "Sports" ($\beta = 1.0208$) and "World" ($\beta = 0.3292$) category was found to be smaller, indicating the general interest is somewhat more prevailing in these cases. All these results show some qualitative analysis about how effective our proposed mining model is.

Capturing Trend: To verify how well our proposed extended Hawkes model can capture the trend of influence by some event on user queries, we generated the simulated "trendiness" plot (Figure 2.3) for the event "Captain America : Civil War" (The real data is plotted in Figure 2.1). To generate Figure 2.3, we used the learnt optimal parameters from table 2.6,

i.e., $\{\lambda_0, \alpha, \beta\} = \{0.0420, 0.9082, 2.6539\}$. We also assumed that at a certain moment t, we know all the past queries that were influenced by the event along with their textual similarity to the event-text. Then, we used equation 2.5 to compute $Trend(E, t)$ at different t and plotted that in Figure 2.3. Cross examination of Figure 2.3 and 2.1 reveals that, the extended Hawkes model can, in fact, capture the real trend quite reasonably.

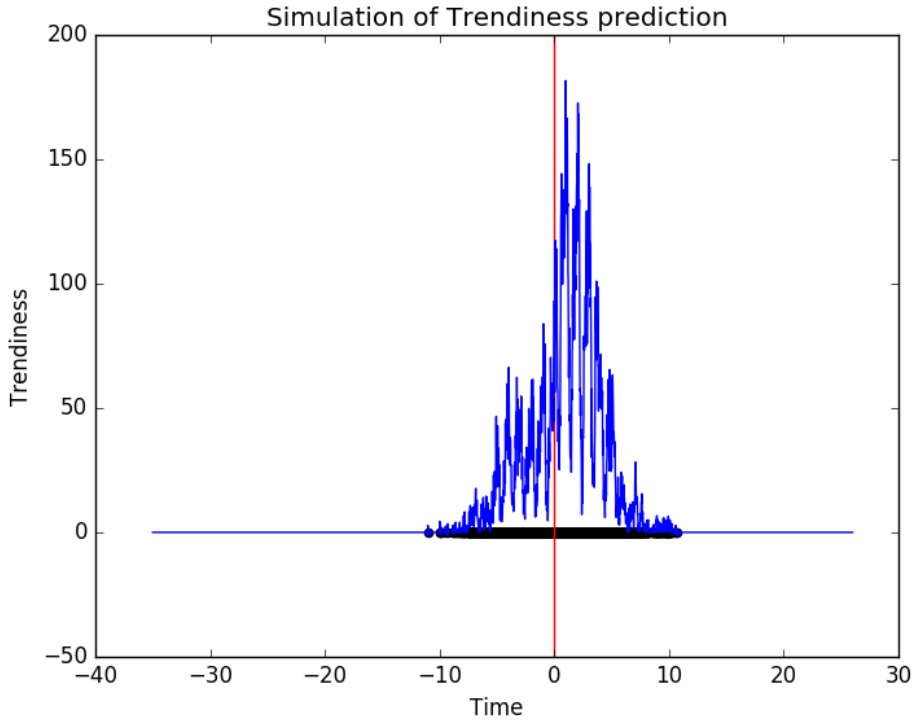


Figure 2.3: Simulation of Trendiness for the event: release of “Captain America : Civil War" Movie

Our qualitative analysis thus shows that overall the proposed approach is able to generate meaningful and interesting knowledge that can help better understand the influence of news events on user queries.

2.6.3 Quantitative evaluation with predictive modeling

We now turn to quantitative evaluation of the proposed approach using the predictive modeling task for predicting whether a user's newly input query has been influenced by an event. To better understand the role of each three basic components of the “IP" model (see

Method	TxtSim	TmpSim	Trend
txt	Yes	No	No
txt-time	Yes	Yes	No
txt-trnd	Yes	No	Yes
txt-time-trnd	Yes	Yes	Yes

Table 2.7: Summary of different versions of the “IP" model.

Category	Method	F-measure	K-measure	Precision	Recall-f	Recall-K	Specificity
Movies	txt	0.6667	0.2366	0.5282	0.9035	0.5684	0.6682
	txt-time	0.7089	0.2717	0.5505	0.9952	0.8656	0.4061
	txt-trnd	0.8084	0.5526	0.6978	0.9606	0.9068	0.6457
	txt-time-trnd	**0.8087**	**0.5532**	0.6987	0.9598	0.9045	0.6488
Sports	txt	0.6667	0.1482	0.5000	1.0000	0.7626	0.3855
	txt-time	0.6783	0.2584	0.5132	1.0000	0.5634	0.6950
	txt-trnd	**0.8239**	0.5990	0.7345	0.9382	0.9382	0.6609
	txt-time-trnd	**0.8239**	**0.5992**	0.7348	0.9378	0.9378	0.6615
US	txt	0.6710	0.0372	0.5063	0.9945	0.0952	0.9420
	txt-time	0.7441	0.5234	0.6444	0.8804	0.6806	0.8428
	txt-trnd	0.8120	0.5758	0.7291	0.9161	0.9161	0.6597
	txt-time-trnd	**0.8155**	**0.5878**	0.7380	0.9112	0.9112	0.6766
World	txt	0.6680	0.0205	0.5018	0.9988	0.0393	0.9812
	txt-time	0.6759	0.4560	0.5111	0.9977	0.5252	0.9308
	txt-trnd	**0.8198**	**0.6020**	0.7489	0.9056	0.9056	0.6964
	txt-time-trnd	**0.8193**	0.6014	0.7493	0.9039	0.9039	0.6975

Table 2.8: Prediction Results for different IP models

equation 2.10), we create four different versions of our model by throwing out one or more components at a time and using the rest of the components to predict the influence of events on user queries. Table 2.7 shows these different versions of IP model along with the components it contains. For example, the “txt" method only contains the “TxtSim" component, while “txt-trend" contains both “TxtSim" and “Trend" components, but does not incorporate the “TmpSim" component. From now onwards, we refer to all the methods compared in this chapter by the terminology introduced in table 2.7 to report the experimental results. All results reported in this section used equation 3.1 as the “TxtSim" component. We also experimented with other text-similarity functions, e.g., TF-IDF cosine similarity; however, the results turned out to be significantly poor (more than 10% relative difference in F-measure)

as compared to using equation 3.1 [the details are omitted due to lack of space].

Table 2.8 shows the summary of the performance obtained by the four different versions of "IP" model on the *Trending event* and *Query-Log* Dataset. For each method and event-category, the table reports the F-measure (with corresponding Precision and Recall, i.e., Recall-f) and and K-measure (with corresponding Specificity and Recall, i.e., Recall-K). Each result reported in Table 2.8 is the average of 25 runs using five-iterated-five-fold cross validation, each time with a random initialization of the parameter set $\{\lambda_0, \alpha, \beta\}$. It is evident that for all the categories of events, the "txt-time-trnd" method performs the best in terms of both F-measure and K-measure. For example, in case of the category "Movies", the "txt-time-trnd" method obtains a F-measure and K-measure value of 0.8087 and 0.5532 respectively, while the only textual similarity based method, i.e., "txt" achieves a F-measure and K-measure value of 0.6667 and 0.2366 respectively.

Close observation of Table 2.8 reveals that, only textual similarity is not sufficient for the influence prediction task as demonstrated by the relative poor performance of the "txt" method. Adding the "TmpSim" component, i.e., "txt-time" improves the prediction accuracy, although not to a significant degree. However, adding the "Trend" component to the "TxtSim" component results in a significant jump in the prediction accuracy ("txt-trnd" method) which verifies that the "Trend" component is very important to detect the influence of events on user query submissions. For example, for the "US" category, "txt" obtains a F-measure value of 0.6710 and "txt-time" obtains 0.7441, whereas, "txt-trnd" obtains a F-measure value of 0.8120. Finally, combining all the three components, i.e., "txt-time-trnd" achieves slightly better performance (F-measure 0.8155) than "txt-trnd" (F-measure 0.8120) signifying the fact that, once we have incorporated the "Trend" component, there is little room for "TmpSim" to further improve the prediction performance. This verifies our assumption that, "Trend" is an essential component for this kind of influence prediction task.

Overall, these quantitative evaluation results show that the basic component techniques

we proposed for modeling influence, i.e., text similarity, temporal similarity, and extended Hawkes model, are all useful for the prediction task, suggesting that they indeed capture useful signals for modeling the influence relation of events on user queries. It is especially interesting to note that the modified Hawkes model provides a "trendiness" score that is shown to be beneficial for the prediction task, suggesting that the model has indeed captured the trend of influence well.

2.7 Conclusion and Future Work

In this chapter, we conducted the first study of how trending events influence search queries, where we frame the problem as a novel data mining problem of joint mining of trending event news data and search log data. We proposed a computational method to quantitatively measure the influence of an event on a query and to discover queries triggered by the event. Specifically, we proposed a novel extension of Hawkes process to model the evolutionary trend of the influence of an event on search queries. Evaluation results show that our proposed approach effectively identifies queries triggered by events and characterizes the influence trend of different types of events.

Although we mainly applied the proposed model to the problem where we predict if a query was triggered by an event, our model can be applied to many other problems. For example, it can help query auto-completion by leveraging terms related to the triggering event, and it can also improve search results by boosting documents that are relevant to the event. In addition, analysis on different characteristics of the events can enable us accurately detect more influential events. All of these interesting directions are left as future work.

Chapter 3

Joint Influence Model for Information Thirst

Previous chapter has shown that popular trending events are important external factors which pose significant influence on user search behavior and also provided a way to computationally model this influence. However, the problem formulation in the previous chapter was based on the strong assumption that each event poses its influence independently. This assumption is unrealistic as there are many correlated events in the real world which influence each other and thus, would pose a joint influence on the user search behavior rather than posing influence independently. In this chapter, I study this novel problem of Modeling the Joint Influences posed by multiple correlated events on user's information seeking behavior. I propose a *Joint Influence Model* based on the Multivariate Hawkes Process which captures the inter-dependency among multiple events in terms of their influence upon user search behavior. I evaluate the proposed *Joint Influence Model* using two months query-log data from https://search.yahoo.com/. Experimental results show that the model can indeed capture the temporal dynamics of the joint influence over time and also achieves superior performance over different baseline methods when applied to various interesting application tasks as well as real-word application scenarios, e.g., *query auto-completion*.

3.1 Introduction

Search Engine optimization has been a vastly studied research area in the past decade. One key component of search engine optimization is analyzing the user search behavior in order to better understand their information need. User search behavior has been studied from

multiple perspectives, e.g., user's own browsing history, click log analysis etc. Recently, how various external factors influence the user search behavior has attracted increasing attention [33]. One important type of external factor is the external events that "significantly" attract the general mass. They trigger user's thirst for information related to the event and thus, pose influence on how the users search to fulfill their information need. How to model the influence of such external events on user search behavior is the high level research question we study in this chapter.

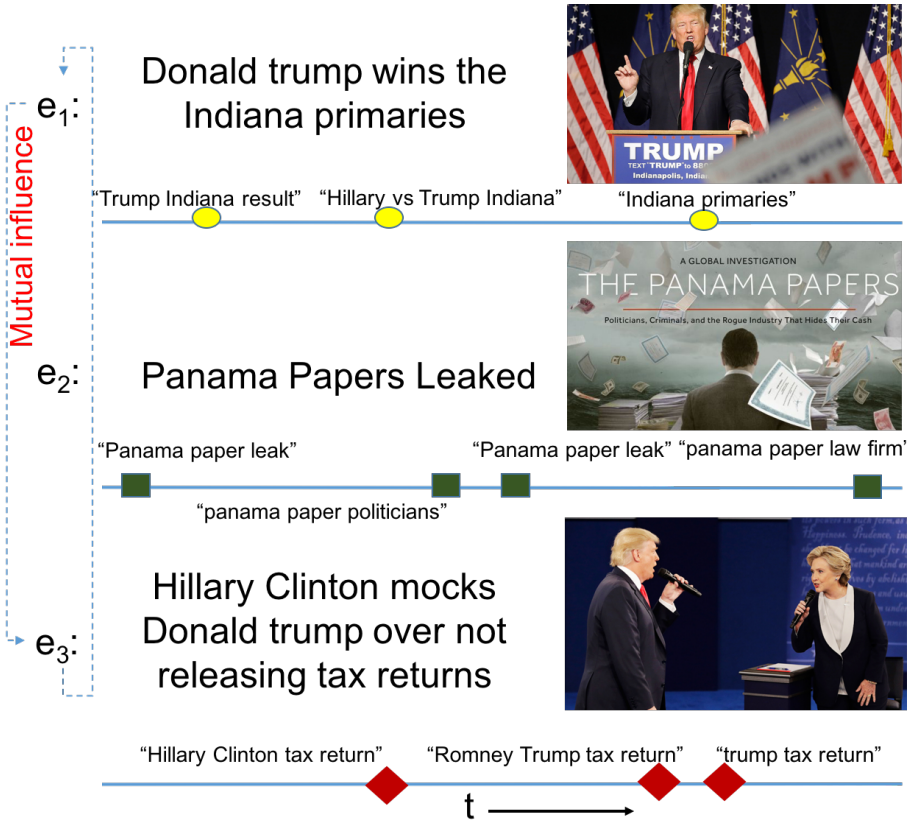


Figure 3.1: A toy example with three events e_1, e_2, e_3. The circles, squares and dices represent queries generated by the influence of event e_1, e_2 and e_3 respectively.

The problem of modeling the influence of popular trending events on user search behavior is not entirely new, specifically, this problem was introduced by Karmaker et.al. [33]. However, the problem definition provided in [33] was based on the strong assumption that the influence posed by each event is independent of the other events, which clearly limits the applicability of such solution to cases where there are multiple correlated events and these

events pose a joint influence on individual user's search pattern. To clearly motivate the problem, let us start with the example in Figure 3.1, where we show three popular events from the month of April, 2016. The first event (denoted by e_1) is Donald Trump's win in the Indiana Primaries. The *blue* line below the event description represents the time dimension and the "yellow" dots represent queries related to/triggered by the event e_1. For example, the query "Trump Indiana result" is clearly seeking information about Trump's election results for Indiana Primaries. Note that, the same query can be posed by multiple users at different instants of time. Here, e_1 is an influential event that has triggered a lot of user queries related to that event. We call these triggered queries as *Influenced queries*. Similarly, event e_2, i.e., "Panama Papers Leaked" and event e_3, i.e., "Hillary Clinton mocks Donald Trump over not releasing tax returns" also trigger numerous queries from users asking for relevant information about the respective event. A deeper thought would also reveal that some of these events may be correlated and they may have a joint-influence on the generation of some queries. For example, people searching for "Hillary Clintons mocking about Donald Trump" might also be interested in information about Trump's Indiana Primary Results and vice-versa. Thus, mutual influence exist among events that jointly affect user search behavior and this joint influence also evolves over time causing corresponding change in the user search pattern. In this chapter, we model this evolution of joint influence posed by multiple external events on the search behavior of users.

As mentioned in the previous paragraph, the major limitation of the previous work by Karmaker et.al. [33] is the assumption that influence posed by one event is independent of the other events. In this chapter, we relax this assumption by providing a new problem formulation, i.e., modeling the joint influence posed by multiple events on user search behavior. Specifically, we introduce a new data mining problem, where, given a search query log and a set of (correlated) events, the task is to mine both these datasets to infer the joint influence posed by the provided set of (correlated) events on triggering queries from users. This specifically means, beside measuring the influence of the primary event that triggered

the query (lets call it *Direct Influence*), the task also requires to measure the influence of secondary (correlated) events for the same (lets call it *Indirect Influence*). This is a new problem because besides computing the degree of influence posed by each event, we also need to come up with a way to compute how their influences are temporally correlated to each other. The joint influence mining task naturally raises many associated interesting research questions, including, how to come up with a numerical formula for measuring influence that is comparable across multiple events (note that, the influence scores computed by Karmaker et.al. [33] are not directly comparable across multiple events), how influence of multiple events jointly evolve over time and how they correlate in the temporal dimension etc. (see section 3.3 and section 3.5.2 for a detailed list of questions).

To solve the joint influence modeling task, we propose a novel mining algorithm based on Multivariate Hawkes Process [45], which is a mutually exciting point process suitable for modeling the frequencies of random events. The joint influence modeling approach proposed by us has several benefits over the independent influence model proposed in [33]; first, it relaxes the assumption that each event poses an influence that is independent of the other events and thus can model real word scenarios better; second, it can capture the temporal correlation of influences posed by two correlated events providing a way to categorize direct influence versus indirect influence and thus can leverage this correlation to better model the evolution of joint influence over time; third, it provides a formal way to measure the influence of multiple events in a comparable numerical scale. Another beneficial feature of the proposed method, as demonstrated by the experimental results (section 3.6), is that the proposed joint influence model is fairly general and is widely applicable on various interesting prediction tasks and search intent related applications (e.g., query suggestion, query auto-completion) and obtains superior results in comparison to multiple baseline methods. The core contributions of this chapter are listed below:

1. We introduce the novel problem of modeling the (temporal) dependency across multiple events in terms of the influences posed by them on user search behavior. To the best of our

knowledge, this problem has not been studied before.

2. We propose a *Joint Influence Model* based on *Multivariate Hawkes Process* which captures the joint-influence posed by multiple events on user search behavior as well as models how this joint influence evolves over time.

3. We present efficient numerical techniques to compute the likelihood of any query log data w.r.t. the proposed *Joint Influence Model*; which provides us with a way to estimate the optimal parameters of the model by maximizing the likelihood.

4. We evaluate the proposed *Joint Influence Model* using two months query-log data from https://search.yahoo.com/. Experimental results show that the model can indeed capture the temporal dynamics of the joint influences over time and can be applied to solve various interesting prediction problems as well as real-word application scenarios, e.g., *query auto-completion*.

3.2 Related Work

Search query logs have been extensively studied to understand user search behavior and provide better search experience [28, 42, 72]. Existing work mostly focused on the inference of users' search intent based on their own search habit and search history. On the other hand, our chapter tries to model how user behavior on a search engine is influenced by external factors such as trending events.

Temporal Information Retrieval [11, 16, 7, 41] and Event Detection [4, 77, 67, 19, 3] are two areas closely related to our work. While Event Detection has been studied vastly in the literature (see [4] for a recent survey), research interest on Temporal Information Retrieval has grown recently [11]. However, we emphasize that, neither of these is the intended goal of this study and our primary motivation is somewhat orthogonal, i.e., given that some (possibly multiple) events have already been reported, we go one step further to investigate

how these events may jointly impact/influence the search behavior of the users.

The notion of event-based retrieval was introduced by Strötgen and Gertz [63] by returning events instead of documents. Zhang et al. [75] addressed the detection of recurrent event queries. Ghoreishi and Sun [22] introduced a binary classifier for detecting queries related to popular events. Kanhabua [30] extended the work [22] by enabling the classifier to detect less popular queries beside popular ones. However, all these approaches are supervised classification methods and largely depend on the quality of training labels provided by humans, whereas our approach is unsupervised.

Kairam et. al. [29] investigated the online information dynamics surrounding trending events, by performing joint analysis of large-scale search and social media activity. Matsubara et. al. [48] presented a new model for mining large scale co-evolving online activities. Pekhimenko et al. [53] designed a system named "PocketTrend" that automatically detects trending topics in real time, identified the search content associated to the topics, and then intelligently pushed this content to users' local machine in a timely manner. However, none of these studies provide answer to the question: how to model the evolution of joint influence posed by multiple events on user search behavior, which is one of the primary motivations of our work. The closest match to this chapter is the work by Karmaker et.al. [33] where they first introduce the problem of modeling the influence of popular trending events on user search behavior. However, as mentioned in section 3.1, their problem definition was based on the unrealistic assumption that only one event can influence the triggering of a particular query and the influences posed by multiple events are independent of each other. In this chapter, our primary focus is to relax these assumptions and propose a more realistic model to capture the joint influence of multiple events.

Another important topic related to this chapter is point process, which has been used to model social networks [8] and natural events [78]. People find self-exciting point processes naturally suitable to model continuous-time events where the occurrence of one event can affect the likelihood of subsequent events in the future. One important self-exciting process

is Hawkes process, which was first used to analyze earthquakes [78], and then widely applied to many different areas, such as market modeling [20], crime modeling [62], conflict [73], viral videos on the Web [15] etc. In this work, we propose a novel Joint Influence Model based on multivariate Hawkes process [45] that can capture the dynamics of simultaneous influence by multiple events on user search behavior.

3.3 Problem Formulation

Let, $E = \{e_1, e_2, ..., e_k\}$ be the set of all events for which we want to analyze their influence on the user search behavior, where k is the total number of events under consideration and each event e_j is represented in terms of natural text (for details on the representation of an event, refer to the work by Karmaker et.al. [33]). Also assume that, each e_j is associated with a set of queries that were generated from influence ("to some extent") by the same event. Let this set be denoted by $Q_j = \{q_{j1}, q_{j2}, ...\}$. Each q_{ji} consists of a tuple $<w_{ji}, t_{ji}, x_{ji}>$, where, w_{ji} is the query-text, t_{ji} is the timestamp of receiving the query and x_{ji} is a textual-similarity score between event-text e_j and query-text w_{ji}. The higher the similarity between e_j and w_{ji}, the higher the x_{ji} score is. For details on how we can get the query set Q_j associated with each event e_j and how to compute x_{ji} for an event-query pair, please see [33]. We omit the details here due to lack of space.

Given the input data mentioned above, our goal is to model the temporal dynamics of the joint influence posed by different events in E on user search behavior. Specifically, we seek answers to the following questions which were never investigated before: 1) Is there a way to computationally model the dependency among different events in terms of the influences posed by them on user search behavior? 2) How these (correlated) influences of multiple events jointly evolve over time? 3) Given that we have seen a query which is triggered by some event e_j, how does that change the future influence of some event *other* than e_j? 4) Can we use the correlation among multiple events to distinguish between *Direct Influence*

and *Indirect Influence* (defined in Section 3.1)? We also ask the same questions raised by Karmaker et.al. [33], e.g., 5) How the textual similarity between an influential event and an influenced query affects the influence trend of that event? 6) How long the influence of different events last? To provide answers to these questions, we formally introduce a novel *Joint Influence Model* based on Multivariate Hawkes Process, in the following section.

3.4 Joint Influence Model

We model the joint influence of multiple events on user search behavior through a generative multivariate point process where each point corresponds to the submission of a new query influenced by some event $e_j \in E$. To be more specific, we propose a new generative model based on Multivariate Hawkes Process (a specific mutually exciting point process) to describe the generation of the influenced queries. This way of modeling query generation is beneficial because this would also allow us to quantify the influence of different events on this generation process at any instant of time. Multivariate Hawkes process is naturally suitable to our problem scenario because it can model the frequencies of occurrences of multiple events in the continuous time domain. For a detailed background on Multivariate Hawkes Process and for further justification on why it is helpful, please refer to [45].

Let, $Q = Q_1 \cup Q_2 \cup .... \cup Q_k$, be the set of all query submissions which were influenced by some event $e_j \in E$. Additionally, let Q_j be the set of all queries that were triggered by the direct influence of event e_j. One naive way to collect Q_j corresponding to event e_j is to retrieve queries from the search log that are textually similar to event-text. For further details on how to retrieve a good quality Q_j for event e_j, please refer to [33]. For modeling the joint influence, we consider the union set, i.e., Q, where each query $q_i \in Q$ corresponds to one point in the multivariate point process and is represented by the tuple $<t_i,d_i,x_i>$. Here, t_i is the timestamp of receiving the the query and thus, always $t_i > 0$; d_i is the event which influenced the generation of q_i and thus, d_i can be any event e_j, i.e., $d_i \in E$; x_i is the

textual-similarity score between event-text, $text(d_i)$ and query-text, $text(q_i)$.

Given this setup, the core technical challenge in designing the *Joint Influence Model* boils down to the problem of how we can formally define the multi-event influenced query generation process; in other words, how to fully characterize the multivariate point process? This is not trivial due to the abstractness in the concept of influence. We address this challenge by introducing the notion of *Influence Function*, which we will discuss in detail in the following section[1]:

3.4.1 The Influence Function

We characterize the multivariate point process by defining a set of continuous functions λ_j for $j = \{1, 2, ..., k\}$, we call them *Influence Functions*, which represent the influence of each event $e_j \in E$ on the generation of the queries in Q at any instant of time. Designing a suitable λ function is the main challenge towards building a reasonable Joint Influence Model. However, defining influence is more of a philosophical question rather than a mathematical one. With this constraint in mind, we adopt to define influence through different components that the final influence function should accommodate and eventually, combine all these components into a single influence function. We first start with various components of the influence function λ_j.

Base Influence: We assume that there is always a non-negative influence posed by each event $e_j \in E$ on the generation of the queries in Q_j. Thus, each event e_j is associated with a constant η_j which governs the rate at which we expect to observe new queries influenced by event e_j. This gives our first set of parameters for the influence function, i.e., $\eta_j \geq 0$ for $j = \{1, 2, ..., k\}$. In contrast, the independent influence model proposed by Karmaker et.al. [33] (let's call it *IIM*), has only one parameter η for all events.

[1]All the codes and evaluation scripts for experimentation can be found at the following link: (**https://bitbucket.org/karmake2/influencemodeling/src/master/**)

Decay Functions: The decay functions characterize how the influence of each event diminishes over time. Thus, each event e_j is associated with a decay function w_j which decides how fast the influence of the same event decays with time. Without loss of generality, we use Exponential Decay functions for our *Joint Influence Model*. While other forms of the decay function are certainly possible, the investigation of the choosing the right decay function is orthogonal to the goal of this research. Mathematically, Exponential Decay Functions are represented as the following:

$$w_j(t) = \alpha_j \exp(-\alpha_j t)$$

The corresponding cumulative decay function is the following (we will need this later):

$$\bar{w}_j(t) = 1 - \exp(-\alpha_j t)$$

In contrast, *IIM* [33] has one decay parameter w for all events.

Impact Functions: Whenever the search engine receives a query triggered by some event e_j, we assume that this newly received query increases the influence of all events (not only e_j), which in turn, increases the probability of receiving further queries influenced by different events. The more we receive new (influenced) queries, the higher the influence of different events become; yielding a higher probability of receiving more influenced queries in the future. Thus, the influence of different events as well as frequency of influenced queries we receive mutually grow together, which is similar to the idea of mutually exciting multivariate point processes [25]. Note that, for some event (mostly uncorrelated events), the increment of its influence can be zero which is also expected.

Given that we have received a new query q_i ($<t_i,d_i,x_i>$) triggered by event d_i, the amount by which the influences of different events increase depends on the textual-similarity score,

i.e. x_i, between the event-text and the query-text. This is intuitive because, highly relevant queries are expected to have more impact on the change of influence than less relevant queries. To capture this, we introduce a set of *Impact Functions* which govern how the influence of all events change depending on the textual-similarity score between the newly received query and its triggering event. Let us denote these *Impact Functions* by the notation $g_{d_i}(x_i)$. The interpretation of $g_{d_i}(x_i)$ is as follows: assuming that the newly received query q_i was triggered by event d_i and the textual similarity between $text(d_i)$ and $text(q_i)$ is x_i, the influences of all the events are then increased in proportion to $g_{d_i}(x_i)$.

Note that, reception of query q_i increases the influences of all events by the same amount, i.e., by $g_{d_i}(x_i)$: which is not desirable. To address this issue, we have a whole new set of parameters, namely "Mutual-Influence Co-efficients" which we will discuss shortly after this. However, the purpose of "Impact Functions" is solely to define how the influence of an event changes based on the textual-similarity score between the newly received query q_i and its triggering event d_i.

Impact functions take the textual similarity score x_i as an input parameter. The exact form of Impact Function we choose would thus depend on the distribution of x_i, let us call it *Intent-Match Distribution*. Below we discuss the *Intent-Match Distribution* briefly, choose a reasonable function for it and then choose the corresponding suitable *Impact Function*

.

Intent-Match Distribution: Intent-Match Distribution is essentially the distribution of the textual-similarity score between the triggering event and the influenced query. For textual-similarity score, we choose the following modified version of the $BM25$ introduced in the work [33] (The details of this function and rationale behind choosing it can be found in the paper [33]). Let, $W_E =< W_{E_1}, W_{E_2}, ..., W_{E_n} >$ be the "event-text" and $W_q =< W_{q_1}, W_{q_2}, ..., W_{q_n} >$ be the "query-text". Then,

$$x_i(W_E, W_q) = \sum_{i=1}^{|W_E|} \frac{\omega(W_{E_i}).IDF(W_{E_i}).TF(W_{E_i}, W_q).(k_1+1)}{TF(W_{E_i}, W_q) + k_1.(1 - b + b.\frac{|W_q|}{avgql})}$$

$$\text{subject to} \sum_{i=1}^{|W_E|} \omega(W_{E_i}) = 1 \tag{3.1}$$

Note that, the textual-similarity score x_i is independent of the past history of received queries and solely depends on the similarity between the "event-text" and the "query-text". Further, $x_i \geq 0$.

To specify the *Intent-Match Distribution*, we hypothesize that a power law probability distribution is the most suitable for our case because of the following reasoning: among the set of all queries that are influenced by some event e_j, very few queries would exactly match with the details in the event-text, while a lot of queries intent would match the details only partially or marginally (these are general exploratory queries). The higher the intent-match, the rarer the frequency becomes; in fact, the frequency decreases exponentially with the increase in textual-similarity. Our empirical evaluation also supports this hypothesis (details in section 3.6.1).

Based on the argument presented above and without loss of generality, we select "Pareto distribution" as our *Intent-Match Distribution*, which is a popular power law probability distribution. "Pareto distribution" is defined on the half line $[0, \infty)$ and has two parameters $\mu > 0$ and $\rho > 0$. Each event e_j is associated with a Intent-Match Distribution f_j ("Pareto distribution" in this case).

$$f_j(x) = \frac{\rho_j \mu_j^{\rho_j}}{(x + \mu_j)^{\rho_j + 1}} \tag{3.2}$$

Under the restriction that $\rho_j > 2$, a suitable impact function is the following with parameters $\rho_j \geq 0$, $\mu_j \geq 0$, $\phi_j \geq 0$, $\psi_j \geq 0$ (Please see [45] for details and rationale) :

$$g_j(x) = \frac{(\rho_j - 1)(\rho_j - 2)}{\phi_j(\rho_j - 1)(\rho_j - 2) + \psi_j \mu_j(\rho_j - 2)}(\phi_j + \psi_j x) \tag{3.3}$$

Thus, each event e_j is associated with a Intent-Match Distribution f_j as well as an impact function g_j. In contrast, the *IIM* [33] has only one impact function $g(x)$ for all events, which was defined as $g(x) = x$; whereas, $g_{d_i}(x_i)$ is a generalization of that with more flexibility to capture the impact.

Mutual-Influence Co-efficients: While the impact function captures the relationship between the textual-similarity of an "influenced query-triggering event pair" and the corresponding change in the influence of an event, it fails to distinguish the different impacts the same query might pose for different events. For example, the submission of query "Trump Indiana Results" should directly indicate an increasing influence of the event "Donald Trump wins Indiana primaries" (This is the *Direct Influence*); however, the same query might have little/no indication about the increasing influence of the event "Messi scores a hat-trick against real madrid" (lets call it *No Influence*). At the same time, query "Trump Indiana Results" might have an indirect indication about the increasing influence of the correlated event "Hillary Clinton results for Iowa Primaries" (The is the *Indirect Influence*). Modeling these inter-dependencies among multiple events in terms of the influence posed by them is one of the central key questions we investigate in this chapter. Our proposed *Joint Influence Model* addresses this question by introducing a new set of Co-efficients, we call them Mutual-Influence Co-efficients, which is a unique component of our proposed model.

To capture the three types of influences, i.e., *Direct Influence, Indirect Influence, No Influence* as mentioned above; we introduce a $k \times k$ matrix of coefficients which we call the Mutual-Influence Co-efficients.

$$MIC = \begin{bmatrix} \nu_{11} & \nu_{12} & \nu_{13} & \dots & \nu_{1k} \\ \nu_{21} & \nu_{22} & \nu_{23} & \dots & \nu_{2k} \\ & & \dots\dots\dots & & \\ \nu_{k1} & \nu_{k2} & \nu_{k3} & \dots & \nu_{kk} \end{bmatrix}$$

The diagonal elements of the matrix represent *Direct Influence*, while non-diagonal elements represent *Indirect Influence*. We also impose the constraint, $\nu_{ji} \geq 0$ for $i, j = \{1, ..., k\}$. A zero value for any element in the MIC matrix represents *No Influence*, while higher non-zero values indicate *Significant Influence*. Thus, the MIC matrix contains valuable information about the inter-dependencies among multiple external events in terms of their influence on user search behavior.

Influence Function and Query Generation Process: So far, we have discussed all the components we needed to define our influence function. Below we present the actual definition of the influence function by combining all these components:

$$\lambda_j(t) = \eta_j + \sum_{j=1}^{k} \nu_{ji} \int_{(-\infty,t)\times R} w_j(t-s) g_j(x) e_j(ds \times dx) \tag{3.4}$$

Now, we define the mutually-exciting query generation process:

Definition 3.4.1 (Mutually-Exciting Query Generation Process)**.** *Let us assume that, we observe queries in the form of triples* $<t_i,d_i,x_i>$ *for* $1 \leq i \leq n$*, where* $t_i \in [T_*, T^*]$ *and* $t_i > t_{i-1}$*,* $d_i \in \{1, 2, ..., k\}$ *and* $x_i \in R^+$*. For the i-th query, it occurs at timestamp* t_i*, the triggering event is* d_i *and the corresponding textual-similarity is* x_i*. At any instant of time t, each event* e_j *for* $j = \{1, 2, ..., k\}$ *has an influence* λ_j *defined by equation 3.4. This constitutes our Generative Multivariate Hawkes Process.*

For a Multivariate Hawkes Process to be well defined, we need the following two conditions to be satisfied:

1. The maximum of the Eigen Values of the MIC matrix is defined as the spectral radius of MIC, i.e,
   $Spr(MIC) = max(eigenValues(MIC))$. Multivariate Hawkes Process requires the following condition to satisfy:
   $$Spr(MIC) < 1$$

2. The decay functions must satisfy the following constraints:
   $$\int_0^\infty t w_j(t) dt < \infty$$

Finally, for computational feasibility, we present the numerical version of the continuous influence function in equation 3.4 below. Let us assume that we have observed queries at points $\{t_i\}$, for $1 \leq i \leq n$. Then, for any timestamp t_i, the influence of event j, $\lambda_j(t_i)$ is defined as:

$$\hat{\lambda}_j(t_i) = \eta_j + \sum_{m=1}^{i-1} \nu_{j,d_m} w(t_i - t_m) g_{d_m}(x_m) \tag{3.5}$$

3.4.2 Estimation of the Optimal Parameters

This section presents the estimation techniques for the optimal parameter values of the influence function. For this purpose, we define the likelihood function for any observed sequence of queries with respect to the proposed mutually exciting multivariate point process. We find the optimal parameters by maximizing the likelihood of the observed query data. Specifically, the log-likelihood function corresponding to the *Mutually-Exciting Query Generation Process* (see Definition 3.4.1) looks like the following:

$$\log L = \sum_{j=1}^{d} \int_{[T_*, T^*] \times R} \log \lambda_j(t) e_j(dt \times dx) + \sum_{j=1}^{k} \int_{[T_*, T^*] \times R} \log f_j(x) e_j(dt \times dx) - \sum_{j=1}^{k} \Lambda_j(T^*) \tag{3.6}$$

Here, T^* is the upper bound of the observation period and $\hat{\Lambda}_j(T^*)$ is called the compensator function and is defined as follows:

$$\Lambda_j(t) = \eta_j(t - T_*) + \sum_{m=1}^{k} \nu_{jm} \int_{(-\infty, t) \times R} [\hat{w}_j(t-u) - \hat{w}_j(T_* - u)] g_m(x) e_m(du \times dx) \tag{3.7}$$

Numerical Computation: For computational feasibility, we now present the way to numerically compute the log-likelihood function defined in Eqn (3.6). Specifically, the numerical version of the log-likelihood function takes the following form:

$$\log \hat{L} = \sum_{i=1}^{n} \log \hat{\lambda}_{d_i}(t_i) + \sum_{i=1}^{n} \log f_{d_i}(x_i) - \sum_{j=1}^{k} \hat{\Lambda}_j(T^*) \tag{3.8}$$

While computation of $f_{d_i}(x_i)$ is straight-forward from equation 3.2, computation of $\hat{\lambda}_{d_i}(t_i)$ and $\hat{\Lambda}_j(T^*)$ are more involved. Below we present the exact formulas to compute $\hat{\lambda}_j(t_i)$ and $\hat{\Lambda}_j(T^*)$ omitting the derivation details due to lack of space. We assume exponential decay function, i.e., $w_j = \alpha_j \exp(-\alpha_j t)$, for the exact computational formula, while other forms of decay functions are certainly possible.

$$\hat{\lambda}_j(t_i) = \eta_j + [\lambda_j(t_{i-1}) - \eta_j] \exp[-\alpha_j(t_i - t_{i-1})] + \nu_{j,d_{i-1}} g_{d_{i-1}}(x_{i-1}) \alpha_j \exp[-\alpha_j(t_i - t_{i-1})] \tag{3.9}$$

$$\hat{\Lambda}_j(T^*) = \eta_j(T^* - T_*) + \sum_{i=1}^{n} \nu_{j,d_i} \bar{w}_j (t^* - t_i) g_{d_i}(x_i) \tag{3.10}$$

By plugging in equation 3.2, 3.9 and 3.10, we obtain the complete numerical version of the log-likelihood function as follows:

$$\begin{aligned}\log \hat{L} = &\sum_{i=1}^{n} \log \left\{ \eta_j + [\lambda_j(t_{i-1} - \eta_j)] \exp[-\alpha_j(t_i - t_{i-1})] + \nu_{j,d_{i-1}} g_{d_{i-1}}(x_{i-1}) \alpha_j \exp[-\alpha_j(t_i - t_{i-1})] \right\} \\ &+ \sum_{i=1}^{n} \log \left(\frac{\rho_{d_i} \mu_{d_i}^{\rho_{d_i}}}{(x + \mu_{d_i})^{\rho_{d_i}+1}} \right) - \sum_{j=1}^{k} \left\{ \eta_j(T^* - T_*) + \sum_{i=1}^{n} \nu_{j,d_i} \bar{w}_j (t^* - t_i) g_{d_i}(x_i) \right\}\end{aligned} \tag{3.11}$$

Here, $g_{d_i}(x_i)$ is defined by as:

$$g_{d_i}(x) = \frac{(\rho_{d_i} - 1)(\rho_{d_i} - 2)}{\phi_j(\rho_{d_i} - 1)(\rho_{d_i} - 2) + \psi_{d_i} \mu_{d_i}(\rho_{d_i} - 2)} (\phi_{d_i} + \psi_{d_i} x)$$

Given the log-likelihood function in equation 3.11, the set of parameters associated with it is the following:

$$\Theta = \{\eta_j, \alpha_j, \nu_{ji}, \rho_j, \mu_j, \phi_j, \psi_j\}, \text{ where } (1 \leq i, j \leq k) \tag{3.12}$$

Incorporating L2 regularization, the optimization problem to find the optimal parameter set Θ^* is written as follows:

$$\Theta^* = \arg\max_{\Theta} \left(\log \hat{L}(\Theta) - ||\Theta|| \right) \tag{3.13}$$

Here, $||\Theta||$ is the L-2 norm of the parameter vector Θ. One can use any non-linear optimization method to solve this maximization problem. Nelder-Mead Simplex Method [23] is one such popular optimization technique. Another useful approach is the Sequential Least SQuares Programming (SLSQP) [9].

3.5 Experimental Design

3.5.1 Data-set

Due to the absence of any readily available joint event-query dataset, we decided to create one from two sets of available data-sets: one for popular events and one for user query history. We call these two data sets *Event* dataset and *Query-Log* dataset respectively. The following two paragraphs provide details about these two data-sets.

Event data-set: An obvious choice for a text data set describing events is news articles (though other data such as social media might also be applicable). The NYTimes Developers Network (thanks to them) provides a very useful api called *"The Most Popular API"* [2], which automatically provides the *url's* of the most e-mailed, most shared and most viewed articles from NYTimes.com during the last month from the date of the issue of the query. We chose to use this API because of two major benefits: 1) it automatically removes duplicate articles, thus we don't need to deal with cases where multiple articles are related to the same event. 2) it only provides the most popular articles from NYTimes, thus the quality/accuracy of the events represented by these articles is very high. Using this API, we collected the most e-mailed, most shared and most viewed articles for the month: April, 2016. Each article consists of a tuple $<$*title-text*, *body-text*, *timestamp*$>$. Among different categories of news, we used four categories for our experiments: *US (National Affairs)*, *Movies*, *Sports* and *World (International Affairs)*.

Query-Log data-set: To analyze the user queries contemporary to the articles in *Event* data-set, we use the two-months (April and May, 2016) user query log data from the widely used search engine at https://search.yahoo.com/. Each query submission q is represented as a tuple $<$*query-text*, *timestamp*$>$. The two-months query log data contains $105,925,732$ query submissions in total.

Query-Event Joint data-set: To create the Query-Event Joint data-set, for each article e_j in the *Event* data-set, we retrieved top relevant queries that have at least a similarity

Section	Total # of events	Avg. Title Length	Avg. Body Length	Total # of queries	Avg. Textual Sim.
Movies	25	18.88	458.08	193,282	2.49
Sports	15	19.53	508.4	616,449	2.48
US	18	20.38	487.77	204,926	1.99
World	11	18.18	438.81	22,197	1.96

Table 3.1: Description of Event-Query Joint Dataset

score of 1.25 (with respect to e_j) according the textual similarity function in equation 3.1 and discarded the rest. This filtering step is reasonable because if the textual similarity is very low (less than 1.25), we assume that there is no influence of e_j on the query. This process provides us with a set of influenced queries triggered by each event from the *Event* data-set. The summary of this data-set is presented in Table 3.1.

3.5.2 Qualitative Evaluation of the Model

It is not possible to do a direct quantitative evaluation of the influence model due to the lack of ground truth information. Thus, to evaluate the quality of the proposed *Joint Influence Model*, we do a formal investigation of the optimal parameters learnt through the optimization process as described in section 3.4.2. Below, we present the specific research questions we ask to evaluate the model quality and provide the roadmap of how we can answer each question.

Research Questions:

1. **Is the "Query Generation Process" well-defined ?**

The "Query Generation Process" is well defined only if the Spectral Radius of the Mutual-Influence Coefficient Matrix is less than 1, i.e, $Spr(MIC) < 1$. [see section 3.4.1 for more details]

2. **How to compare influences posed by different events?**

We can answer this question by computing average influence posed by each event and then

compare them. The average influence vector where each element is the average influence of the corresponding event can be obtained using the following formula: $(\mathbb{1}_k - MIC)^{-1}\boldsymbol{\eta}$, where, $\mathbb{1}_k$ is a $k \times k$ identity matrix.

3. **How to compare *Direct* Vs *Indirect* influence ?**

The diagonal elements of matrix MIC represent the *Direct Influence*, whereas, the non-diagonal elements present *Indirect Influence*. We can do direct numeric comparison here.

4. **How to measure the influence longevity of an event?**

The $\boldsymbol{\alpha}$ parameter defines how fast the influence of any event decays over time. Higher values of α denotes a faster decay.

5. **Is "Pareto Dist." suitable for "Intent Match Dist."?**

To answer this question, we look at the empirical distribution of "Intent Match" score between event-text and query text and verify whether "Pareto Distribution" is a good match for it.

6. **How well the Model fit the original data?**

This question can be answered by jointly plotting the simulated influence of an event and the actual frequency of queries generated by that event over the same period of time and see if the trend of the simulated influence is similar to the trend of the actual frequency of generated queries.

3.5.3 Applications and Quantitative Evaluation

In this section, we demonstrate the wide applicability of the proposed "Joint Influence Model" by demonstrating how the model could be used to solve various interesting prediction problems as well as real-world problems associated with search engine systems. Another benefit of these experiments is to conduct indirect quantitative evaluation of the *Joint Influence Model* as direct evaluation is impossible due to the lack of ground truth data for influence

which is an abstract concept. The primary purpose of these experiments is to see if modeling the influence inter-dependencies among multiple events actually help us achieve better performance in real life application scenarios. To achieve these goals, we present a set of prediction tasks / application scenarios and provide a roadmap on how we can adopt the "Joint Influence Model' to solve these tasks.

Application Tasks:

1. **Predict the most influential event in the future:**

We assume the influence of an event in the current hour is proportional to the frequency of queries generated by it in the next hour. Thus, the event with the highest influence score in the current hour is predicted to be the event that generates highest number queries in the next hour. We then compare this predicted most influential event with the actual most influential event (computed from the original query log) and based on that, we can report the accuracy of the prediction for a separate held-out testing set.

2. **Rank multiple events based on their future influences:**

This prediction problem is similar to previous prediction problem, except that, now we want to predict the ranking of events in terms of their future influence instead of just predicting the future top influential event. Again, we use the current hour influence scores to predict the next hour's generated query frequencies and rank the events accordingly. To evaluate the quality of the ranking, we compare the predicted ranking against the actual ranking obtained from the query log and compute two different popular ranking evaluation metrics: i.e, NDCG [70] and Rank Biased Overlap (RBO) [71].

3. **Predict the most frequent query in the future:**

This prediction problem is the same as the prediction problem in (1) except that now we

want to predict the most frequent query in the future instead of the most influential event. For this prediction task, we use a slightly modified version of the original "Joint Influence Model" where apart from computing the evolving influence at the event level, we also compute the evolving influence at the query level. The basic idea is to break each event-level influence into smaller units where each unit would correspond to the query level. We omit the full details of process due to lack of space.

4. **Rank queries based on their future frequencies:**
This prediction problem is similar to the prediction problem in (2) except that now we want to rank queries instead of events. Again we report NDCG [70] and Rank Biased Overlap(RBO) [71] to evaluate the quality of the predicted ranking.

5. **Solve a real world application problem, e.g., query auto completion task:**
Finally, we select *Query Auto Completion* as a goal task and use our proposed "Joint Influence Model" to solve it. Specifically, for a new query from the testing set, we look at the first word and try to predict the exact query based on the latest available influence scores of all the queries starting with the first word. Based on these influence scores, we rank the potential queries and then, compute the reciprocal rank of the actual query in the predicted ranked list. We repeat the whole process for all the queries in a separate held-out testing set and report the mean reciprocal rank (MRR) [66], which is the most popular evaluation metric used in measuring the performance of query auto completion tasks.

Baseline Methods: For all the quantitative evaluation tasks, we compare the proposed Joint Influence Model against the obvious baseline method, i.e., *Independent Influence Model* (We call it "IIM") introduced in [33]. If the Joint Influence Model(*JIM*) performs better than *IIM*, we can conclude that capturing inter-dependencies is indeed useful and can help us

Acronym	Method
NF	Naive Frequency
AR	Auto Regression [12]
ARD	Auto Regression with difference [12]
VAR	Vector Auto Regression [10]
IIM	Independent Influence Model [33]
JIM	Joint Influence Model
JIM-G	Joint Influence Model-Generalized

Table 3.2: Methods Compared for Quantitative Evaluation

achieve superior performance in real life applications. Additionally, as all these quantitative evaluation tasks are some kind of forecasting problems, we also use some popular time series prediction methods as the baselines including Autoregressive Models (AR), Vector Auto Regression (VAR) etc. Note that, our primary focus is not the quantitative evaluation, rather demonstrating the usefulness of capturing influence inter-dependencies among different events. Thus, experimenting with many different forecasting methods is an orthogonal direction with respect to our focus which we do not explore in this work. We also include the simplest baseline method *Naive Frequency* (NF), where the current hour's frequency is used to predict the next hour's frequency. Table 3.2 lists down all the methods we experimented and also provides with an acronym for each method for notational convenience. *JIM* is the "Joint Influence Model" proposed in this chapter, whereas, "JIM-G" is a minor variation of "JIM" with the constraint that events share the same α, i.e., the decay parameter.

3.6 Results

3.6.1 Qualitative Evaluation of the Model

First, we do a qualitative investigation of the optimal parameters learnt through the optimization process as described in section 3.4.2. Table 3.3 presents these learnt parameters. While the individual numbers in Table 3.3 are not very meaningful, the comparison across

parameter	η	α	ρ	μ	ϕ	ψ
Movies	0.1961	0.8697	4.9706	3.0197	0.4542	0.1644
Sports	0.317	1.1999	6.2745	4.2272	1.1608	0.5304
US	0.2328	1.0999	6.3056	1.777	0.6962	0.508
World	0.074	0.677	3.9747	1.5226	0.2465	0.1685

Table 3.3: Parameters learnt for different categories of events

Movies (0.9319)	**Sports** (0.9649)	**US** (0.9192)	**World** (0.9213)

Table 3.4: Spectral Radius of MIC Mat. for different categories

different categories of events is quite interesting. For example, η for "Sports" category (0.3170) is generally much higher than that for "World" category (0.0740), suggesting that the general interest in "Sports" events is much higher than "World" events among the common mass. Another interesting parameter is α, which indicates the longevity of influence for different categories of events. According to Table 3.3, "World" events ($\alpha = 0.6770$) usually have a longer lasting influence compared to "Sports" events ($\alpha = 1.1999$). Next, we move onto providing answers to the specific research questions asked in section 3.5.2, sequentially one at a time.

Is the "Query Generation Process" well defined?

Table 3.4 shows the spectral radius of Mutual-Influence Co-efficient Matrix obtained for different categories of events. It is evident that, all the numbers are less than 1. Thus, we conclude that, the "Query generation process" is indeed well defined.

How to compare influences posed by different events?

Table 3.5 reports the top 2 influential events from each category along with their average influence score computed by the formula presented in section 3.5.2. For example, the movie "Captain America: Civil War" was found to be the most influential event in the "Movies"

Events	Sections	
	Movies	**Sports**
1	Movie: "Captain America: Civil War" (11.5514)	Horse-Racing: Kentucky Derby (13.5346)
2	Movie: "X-men: Apocalypse" (2.0532)	Basketball: Stephen Curry (6.6432)
	Sections	
	US	**World**
1	Donald trump Vs Hillary Clinton (14.0117)	Panama Papers Released (0.8179)
2	Las Vegas Squatters Housing Collapse (9.6340)	Philippine Presidential Race (0.5821)

Table 3.5: Top two most influential events from four different Categories

Category with an average influence score of 11.5514, while "Donald trump Vs Hillary Clinton" was found to be the most influential event (average score 14.0117) in the "US" category. Manual inspection reveals that all these reported influential events are indeed popular events which match with our intuition.

How to compare *Direct* Vs *Indirect* influence ?

Table 3.6 reports the average of the diagonal elements (*Direct Influence*) as well as the non-diagonal elements (*Indirect Influence*) of the MIC matrix for each category of events. It is evident that the influence posed by the triggering event, i.e., *Direct Influence* is significantly larger than that of a non-triggering event, i.e., *Indirect Influence* which also concur with our expectation. For example, *Direct Influence* (0.6342) of events in the "US" category is much higher than the *Indirect Influence* (0.0201) in the same category. In fact, this observation holds for any category.

How to measure the influence longevity of an event?

Direct inspection of α values from Table 3.3 can provide answer to this question. For example, Table 3.3 suggests that "Sports" events generally have short term influence ($\alpha = 1.1999$

Influence	Movies	Sports	US	World
Direct	0.5285	0.6495	0.6342	0.5798
Indirect	0.0255	0.0165	0.0201	0.0166

Table 3.6: *Direct Influence* Vs *Indirect Influence*

), while “World" events have comparatively long lasting influence ($\alpha = 0.6770$).

Is “Pareto Dist." suitable for “Intent Match Dist."?

To answer this question, we show the plot for the empirical distribution of “Intent Match" score between event-text and query-text for the events of “Sports" category in Figure 3.2. This figure demonstrates that as the “Intent Match" score goes high, the number of queries with corresponding score becomes exponentially smaller, suggesting that, indeed “Pareto Distribution" is a reasonable candidate for the “Intent Match Distribution".

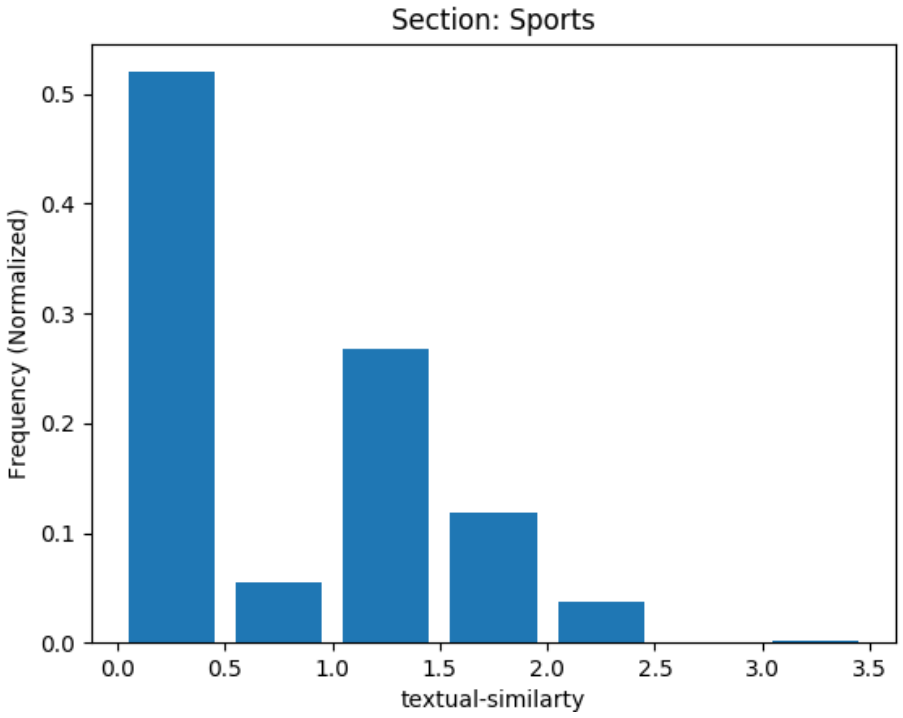


Figure 3.2: Intent Match Distribution for category “Sports”

How well the Model fits the original data?

We plot the the simulated influence of the event “release of movie Captain America: Civil

War" from the “Movies" category along with the actual frequency of queries generated by that event during the same span of time (hour 1500 to hour 1700) in Figure 3.3. It is clearly evident that the simulated influence can indeed capture the trend of the actual frequency of generated queries and thus, we conclude that the model can indeed capture the influence trend with a decent accuracy.

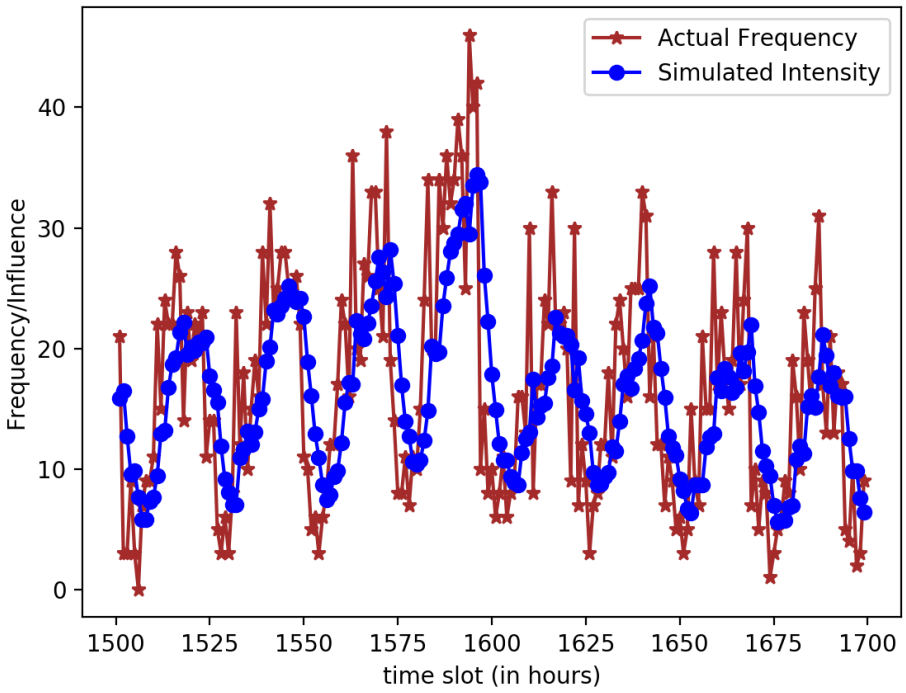


Figure 3.3: Demonstration of the goodness of fit for the event “release of movie Captain America: Civil War"

3.6.2 Applications and Quantitative Evaluation

This section presents the quantitative evaluation results for the five different application tasks presented in section 3.5.3, namely, Predict the most influential event in the future [Table 3.7], Rank multiple events based on their future influences [Table 3.8], Predict the most frequent query in the future [Table 3.9], Rank queries based on their future frequencies [Table 3.10], Solve a real world application problem, e.g., query auto completion task [Table 3.11]. General inspection of Table[3.7-3.11] reveals that, “JIM-G" is found to be the most robust method for

all these different application tasks by obtaining the highest number for performance metrics most of the time. For example, for the task “Predict the most influential event in the future" [Table 3.7], “JIM-G" is found to achieve the highest accuracy for all four categories of events. For the “Query auto completion task", the mean reciprocal rank for “JIM-G" is found to be the highest for all categories except the category “World", for which “IIM" obtains a slightly better number.

In case of event level predictions (Table 3.7 and 3.8), JIM turns out to be the second best performing method. This suggests that the Joint Influence Model indeed captures useful information which results in its superior performance over other baseline methods. The superiority of “JIM-G" over “JIM" may be explained by the fact that, while “JIM" has more parameters for α (i.e., one α for each single event) than “JIM-G" (i.e., single α for all events), “JIM" might be suffering from over-fitting the training data while “JIM-G" would learn a more general model suitable across multiple events. This over-fitting problem seems more prominent for query level predictions (Table 3.9 and 3.10), especially for category “World" where the number of queries in the dataset is comparatively very small (Table 3.1). Here, “JIM" cannot even achieve the second best performance. We believe this is due to the sparsity of query level data. Interestingly, the simple baseline “NF", achieves quite good result at the query level prediction problems, while “VAR" suffers severely from overfitting. However, “JIM-G" still performs the best for most of the cases in query level predictions.

In summary, Table[3.7-3.11] suggest that the Joint Influence Model is quite robust and useful in many different applications with superior performance over a number of reasonable baseline methods.

3.7 Conclusion

The assumption that each popular event poses influence upon user search behavior independently is unrealistic as many real world events are closely related to each other. The primary

Metric	Methods	Movies	Sports	US	World
Accuracy	NF	0.6638	0.6647	0.9302	0.5073
	AR	0.7256	0.6818	0.8959	0.3934
	ARD	0.7445	0.4249	0.9388	0.0609
	VAR	0.7399	0.4997	0.5105	0.1237
	IIM	0.7193	0.6162	0.9376	0.5671^{2}
	JIM	0.7520^{2}	0.6938^{2}	0.9491^{2}	0.5307
	JIM-G	$\mathbf{0.7531^{1}}$	$\mathbf{0.6967^{1}}$	$\mathbf{0.9542^{1}}$	$\mathbf{0.5905^{1}}$

Table 3.7: Predicting the most influential event in future

Metric	Methods	Movies	Sports	US	World
NDCG	NF	0.9074	0.9105	0.9792	0.7798
	AR	0.9370	0.9168	0.9460	0.6951
	ARD	0.8604	0.7458	0.9529	0.4358
	VAR	0.8831	0.7914	0.8950	0.5175
	IIM	0.9348	0.8975	0.9831	0.8393^{2}
	JIM	0.9485^{2}	0.9278^{2}	0.9870^{2}	0.8275
	JIM-G	$\mathbf{0.9508^{1}}$	$\mathbf{0.9322^{1}}$	$\mathbf{0.9879^{1}}$	$\mathbf{0.8517^{1}}$
RBO	NF	0.6596	0.6800	0.8573	0.5140
	AR	0.7052	0.6821	0.7695	0.3967
	ARD	0.5320	0.4122	0.7267	0.0942
	VAR	0.5752	0.4808	0.6331	0.1647
	IIM	0.6961	0.6479	0.8597	0.5992^{2}
	JIM	0.7194^{2}	0.6980^{2}	0.8672^{2}	0.5623
	JIM-G	$\mathbf{0.7252^{1}}$	$\mathbf{0.7069^{1}}$	$\mathbf{0.8705^{1}}$	$\mathbf{0.6087^{1}}$

Table 3.8: Predicting future influences of multiple events (Wilcoxon's signed rank test at level 0.05)

Metric	Methods	Movies	Sports	US	World
Accuracy	NF	0.3281	0.4894^{2}	0.5717^{2}	0.3879
	AR	$\mathbf{0.3879^{1}}$	0.4794	0.5400	0.4504
	ARD	0.2424	0.1965	0.4410	0.0443
	VAR	0.0023	0.0007	0.0029	0.0001
	IIM	0.3413	0.3660	0.5408	$\mathbf{0.4710^{1}}$
	JIM	0.3642	0.4688	0.5563	0.3035
	JIM-G	0.3820^{2}	$\mathbf{0.5134^{1}}$	$\mathbf{0.5843^{1}}$	0.4544^{2}

Table 3.9: Predicting the most frequent query in future

Metric	Method	Movies	Sports	US	World
NDCG	NF	0.5914	0.6693	0.8060	0.4465
	AR	0.6713^{2}	0.7440^{2}	0.7789	0.5200
	ARD	0.2642	0.2977	0.4717	0.0827
	VAR	0.0087	0.0052	0.0136	0.0015
	IIM	0.6355	0.6976	0.8121^{2}	$\mathbf{0.6555^{1}}$
	JIM	0.6484	0.7204	0.8022	0.4809
	JIM-G	$\mathbf{0.6870^{1}}$	$\mathbf{0.7650^{1}}$	$\mathbf{0.8430^{1}}$	0.6062^{2}
RBO	NF	0.4349	0.5707	0.6491	0.3665
	AR	0.4947^{2}	0.5908^{2}	0.6102	0.4130
	ARD	0.1803	0.2191	0.3237	0.0538
	VAR	0.0042	0.0019	0.0045	0.0001
	IIM	0.4562	0.5174	0.6509^{2}	$\mathbf{0.4676^{1}}$
	JIM	0.4782	0.5724	0.6436	0.3048
	JIM-G	$\mathbf{0.5059^{1}}$	$\mathbf{0.6172^{1}}$	$\mathbf{0.6764^{1}}$	0.4332^{2}

Table 3.10: Predicting future frequencies for multiple queries. (Wilcoxon's signed rank test at level 0.05)

Metric	Methods	Movies	Sports	US	World
MRR	NF	0.6427	0.8427	0.8489	0.6899
	AR	0.7382	0.9129	0.8339	0.7471
	ARD	0.2842	0.4077	0.5238	0.2754
	VAR	0.1911	0.1722	0.1186	0.3696
	IIM	0.7839	0.9171	0.8896	$\mathbf{0.9262^{1}}$
	JIM	0.8082^{2}	0.9509^{2}	0.8903^{2}	0.8999
	JIM-G	$\mathbf{0.8226^{1}}$	$\mathbf{0.9556^{1}}$	$\mathbf{0.8988^{1}}$	0.9193^{2}

Table 3.11: Query Auto-Completion Results: MRR reported. (Wilcoxon's signed rank test at level 0.05)

contribution of this chapter is to relax this unrealistic assumption made in the previous work by proposing a Joint Influence Model based on multivariate Hawkes Process that captures the inter-dependency of multiple events in terms of the influence posed by them upon user search behavior. Experimental results demonstrate that the proposed method not only effectively capture the temporal dynamics of joint influences by multiple events, but also when applied to various application tasks, achieves superior performance most of the time over different baseline methods that do not consider this mutual-influence among multiple events. This signifies that the mutual influence which exists among multiple correlated events is an important factor which should be considered while designing such influence models.

Chapter 4

Influence Models for User Generated Contents

User generated contents are often significantly influenced by the community to which the user belongs to. While some work has been done on mining such influence from structured information networks, little attention has been paid on how to mine community-influence from user generated unstructured data. In this paper, we introduce and study the problem of modeling community-influence on user generated unstructured contents, particularly in the context of text content generation. Although text generation has recently became a popular research topic after the surge of deep learning techniques, existing methods do not consider community-influence factor into the generation process and thus, the process does not evolve over time. This clearly limits their application on text stream data as most text stream data often evolve over time showing distinct patterns corresponding to the shifting interests of the target community. Thus, it is compelling to propose an Influenced Text Generation (ITG) Process that can capture this evolution of text generation process corresponding to evolving community-influence over time. In this paper, we propose a deep learning architecture based Influenced Text Generation Process to address this challenge. Experimental results with six different text stream data comprised of conference paper titles show that the proposed ITG method is really effective in capturing the influences posed by different research communities on paper titles generated by the researchers.

4.1 Introduction and Motivation

A crucial component of any intelligent system is to understand and predict the behavior of its users. A correct model of the user behavior enables the system to perform effectively to better serve the users need. User's behavior is often significantly influenced by the community to which they belong to. Community-Influence on user behavior are mostly reflected in two different ways: 1) Through significant growth of users' thirst about information related to the community and 2) Through the user generated contents that are directly/indirectly related to the community. While some work have been done on modeling user's information thirst that are influenced by their community of interest [33, 32] , little attention has been paid to how related communities influence their users in terms of generating contents. In this paper, we introduce and study the problem of modeling community-influence on user generated unstructured contents, particularly in the context of text content generation.

Automated text generation has become a popular research topic recently after the upsurge of deep learning techniques [18, 39, 36, 56]. Recurrent neural networks, specially, Long-Short-Term-Memory (LSTM) has shown to be promising to solve various text generation tasks [47]. However, one common limitation with the existing text generation techniques is that most of them are static generation processes, i.e., they assume that the generated text data has no notion of time. Existing methods also do not consider community-influence factor into the generation process and thus, the process does not evolve over time. This clearly limits their application on text stream data as most text stream data often evolve over time showing distinct patterns corresponding to the shifting interests of the target community. For example, think about the titles of research papers published by a popular conference (e.g. KDD) over many years. The paper titles that appeared in KDD 2001 are significantly different from the titles that appeared in KDD 2015. This is due to the evolution of research interests that took place during these years within the research community. The same is true with user generated contents across different social media, e.g., Twitter, where many

users share and talk about different popular stories related to their community of interest as time passes by. Existing text generation techniques are unable to take such evolving community-influence into account and thus, cannot capture these evolution of patterns in text stream data. This evolving nature of the text stream data thus demands for a more dynamic text generation process.

In this paper, we propose a novel Influenced Text Generation (ITG) process built upon recurrent neural network based deep learning architecture that incorporates community-influence to model the generation of dynamically evolving text stream data. The difference between normal text data and text stream data is that every piece of generated text in the text stream data is associated with a timestamp. Thus, one input to the ITG process is the timestamp for which we want to generate text. We also assume that the distributions of words corresponding to text data from two different timestamps are somewhat different which explain the evolution of topics in the stream data. Modeling the generation of text stream data is a hard problem due to the following three challenges that need to be addressed simultaneously: 1) The model should learn to generate syntactically correctly sentences, 2) The model should generate semantically coherent sentences and 3) More importantly, the model should generate sentences which is time-sensitive and aligned with the evolution of the text stream and reflect the dynamics of community-influence on the generated text data. To demonstrate the three challenges more clearly, let us think about the task of generating a KDD-2016 like paper title. First, the sentence must be syntactically correct: “Privacy preserving Class Ratio Estimation" is a good example of syntactically correct sentence, while “Preserving Ratio Class Estimation Privacy" is not. Second, the sentence should be semantically coherent: “Hidden Markov Models for sequence Modeling" is a good example of semantically coherent sentence, however, “Hidden Markov Models for Polygon cutting" may not be so semantically coherent because Polygon cutting is neither related to Hidden Markov Models nor Data mining in general. Finally, the model needs to generate sentences relevant to the input timestamp: for example, while “Solving regression problems with rule-

based ensemble classifiers" is both syntactically correct and semantically coherent, regression problems and rule-based classifiers are somewhat old research topics mostly popular during the 90's. A more relevant title to KDD-2016 would be "Deep Visual-Semantic Hashing for Cross-Modal Retrieval" as deep learning gained much popularity among the mining community recently.

To address the three challenges mentioned above, our proposed ITG process is comprised of three basic components that interacts with each other to model the generation of text stream data. The first component is the *Sequence Generator* which ensures that ITG generates syntactically correct sentences. The second component is the *Topic Generator* which captures the trends of different topics of interest within the community. Finally, the third component is the *Influence Generator* which ensures that ITG can compute the community-influence corresponding to the input timestamp and incorporate it into the generation process to ensure that the generated text is consistent with that particular timestamp. ITG process combines these three essential components into a single unified model and learns all their optimal parameter values by mining a training corpus of the target text stream data.

The proposed ITG method has three major advantages over the existing static text generation methods available in the literature. They are briefly mentioned below:

- **Time sensitive Text Generation:** ITG understands the notion of time and can generate text which is relevant to a particular timestamp.

- **Capturing Community-Influence:** ITG can capture the evolution of interests happening inside the target community and adapt the text generation process accordingly.

- **Chronological Summary Generation of Past Text Stream:** ITG provides us with a way to generate a chronological summary of the past text stream, i.e., it can summarize how sentences from a particular text stream evolved over time in the past.

We conducted comprehensive experiments with six different sets of publication stream datasets to demonstrate the power of ITG. These datasets contain titles of the papers pub-

lished in different machine learning theory and applied machine learning conferences over 20 years, which were collected from the Open Academic Graph. For quantitative evaluation, we compared ITG against multiple baseline methods for Chronological Summary Generation task. Experimental results show that ITG achieves superior performance over the other baseline methods by a clear margin in almost all cases. This confirms that ITG process is fairly general and can effectively capture the evolution of community-influence while generating text. We also found that incorporating influences from external correlated communities can further enhance ITG's performance. In summary, we make the following contributions in this paper:

1. We study the problem of modeling community-influence on user generated text contents. To the best of our knowledge, this problem has not been studied before.

2. We propose a Influenced Text Generation (ITG) process built upon recurrent neural network based architecture which can capture evolving community-influence to explain the generation of text stream data.

3. We demonstrate how ITG can be applied to an interesting text mining application, i.e., Chronological Summary Generation of Past Text Stream. Through comprehensive experiments with six different text stream datasets, we show that ITG achieves superior performance over multiple baseline methods for this task. We also show that incorporating influences from external correlated communities can further enhance the performance of ITG.

4.2 ITG - Influenced Text Generation

In this section, we present the details about how Influenced Text Generation (ITG) Process models the generation of text stream data. We assume that each piece of generated text in the stream is associated with a discrete timestamp, where, the definition of timestamp

varies across different types of data. For example, for conference paper titles, a reasonable discrete timestamp is the year in which the paper is published, whereas, for a news headline, a more reasonable discrete timestamp is the actual date when the news was broadcast. We also assume that the text data across different timestamps are generated by sampling words from different distributions, i.e., each timestamp is associated with a unique distribution of n-grams, which is essentially the key to capture the evolution of community-influence on text generation. ITG explains this evolution over a time period through the variance in the word distributions across different timestamps.

Below, we first discuss briefly the three major components that are the building blocks of ITG and then present how ITG combines them into a single unified model.

4.2.1 Sequence Generator

Sequence Generator is the central component of ITG which generates the next word x_t in the sentence given $t-1$ previous words. Thus, *Sequence Generator* is essentially a language model which provides a probability distribution over a sequence of words that can be used to predict the next word in the sequence. Any sequence modeling framework, e.g., Hidden Markov Models, Recurrent Neural Networks etc. can work as a sequence generator. For ITG, we chose recurrent neural network with LSTM cells as the *Sequence Generator* due to its recent promising results obtained for language modeling tasks [39, 37]. Given the previous t words, i.e., $x_{1:t}$, the recurrent neural network based language models compute the conditional probability for the next word $y_t = i$ for $i \in V$, the vocabulary set, by computing a hidden state h_t and passing it through a Softmax function:

$$P(y_t = i|x_{1:t}) \equiv P(y_t = i|h_t) \tag{4.1}$$

$$P(y_t|h_t) \propto \exp(W^{\Omega}h_t + B^{\Omega}) \tag{4.2}$$

$$h_t = \Omega(h_{t-1}, x_t) \tag{4.3}$$

Here, Ω can be a standard RNN cell or more complicated cell like LSTM, GRU etc. Output at timestamp t, i.e., y_t is fed as input for timestamp $t+1$, thus, $x_{t+1} = y_t$. Note that, the small t notation means the timestamp associated with the word positions. It has nothing to do with the evolution of text stream data over a time period, for which, we use the big T notation (see section 4.2.2).

4.2.2 Topic Generator

The next component of ITG is the *Topic Generator*. The primary purpose of this component is to analyze different topics across the text stream data and compute the evolution of topic distributions within the community over time. It takes all past text stream data as input and applies a probabilistic topic model to infer n (a user defined parameter) different topics, each represented with a unique distribution over the entire vocabulary. For ITG, we chose LDA as the *Topic Generator* since it has been the most popular topic modeling technique for more than a decade. Once n topics are identified, the *Topic Generator* computes the distribution of n topics over different timestamps observed in the training data. For this notion of timestamp that corresponds to the physical generation time of text, we use the big T notation which is completely different from the small t notation in the *Sequence Generator*. We distinguish big T notation by naming it global timestamp as opposite to the local timestamp t. For each global timestamp T, the *Topic Generator* computes a topic distribution θ_T.

The *Topic Generator* also provides a sub-component, i.e., *History Extractor*, which, given a particular global timestamp T as input, retrieves the topic distributions of previous r (a user defined parameter) timestamps computed by LDA. We mathematically denote the output of *History Extractor* by $\theta_{T-r:T-1}$, where, $\theta_{i:j}$ denotes topic distributions from timestamp i to j augmented into a single vector. This means the cardinality of vector $\theta_{T-r:T-1}$ is $r \times n$.

4.2.3 Influence Generator

The main function of the *Influence Generator* is to compute the community-influence during the text generation process. Given a particular global timestamp T, we represent community-influence through a real valued vector (γ_T) of dimension K (another user defined parameter), which is essentially the output of *Influence Generator*. The input to the *Influence Generator* is the $r \times n$ dimensional vector of topic distributions from previous r timestamp, i.e., $\theta_{T-r:T-1}$ (assuming T as the current global timestamp). Thus, *Influence Generator* essentially maps a $r \times n$ dimensional topic vector to a K dimensional influence vector. Here, we assume that the influence vector γ_T corresponding to current timestamp T, can be approximated from the historical topic distribution $\theta_{T-r:T-1}$. This assumption is reasonable, because most text stream data do not evolve dramatically over night, rather their topical shift happens quite gradually. Think about some particular research community like SIGKDD. The topic distribution in papers published in a particular year is not dramatically different from previous two years, rather they are somewhat correlated.

ITG maps topic-distribution-history vector (described in the previous paragraph) to an influence vector through a feed-forward neural network. Although any function that can perform this mapping can resemble as Influence Generator, we chose a feed-forward neural network for ITG due to its capability of approximating a wide family of functions. Without loss of generality, we used ReLU activation units in the hidden layers. Once the influence vector (γ_T) is computed, it is then injected as a bias into the *Sequence Generator* when generating the next word (x_t) in the sequence (More details in section 4.2.4). *Influence Generator* is the component which enables ITG to generate text tied to a particular global timestamp. Thus, *Influence Generator* is a pivotal component in ITG which makes influence-aware text generation possible.

Mathematically, let θ_T denote the topic distribution for the generated text at timestamp T, then the function of *Influence Generator* is expressed as follows ($\|$ means the concatena-

tion operation):

$$\theta_{T-r:T-1} = \theta_{T-r} \| \theta_{T-r+1} \| .... \| \theta_{T-1} \tag{4.4}$$

$$\gamma_T = \Gamma(\theta_{T-r:T-1}) = W_2^{\Gamma} \cdot \left[ReLU(W_1^{\Gamma} \cdot \theta_{T-r:T-1} + B_1^{\Gamma}) \right] + B_2^{\Gamma} \tag{4.5}$$

4.2.4 ITG as a Unified Model

Now that we have presented the three building blocks of ITG, this section presents how these different components interact with each other and work as a unified model to explain the evolution of text stream data corresponding to the fluctuation of community-influence. ITG process can be thought of as generating text corresponding to a particular timestamp T. Thus, the whole process starts with a timestamp T as input and the start-of-sentence marker (let's call it #) as the sequence generated so far. The next task is to generate one word at a time iteratively until the end-of-sentence marker is generated (let's call it *). The exact process of generating the next word y_t in the sequence is demonstrated in Figure 4.1.

The first step of the generation process is to infer n different topics from the historical text stream and compute the topic distribution for each unique timestamp observed in the training data. Next, given a particular global timestamp T as input, the *History Extractor* Module extracts the historical topic distributions corresponding to previous r global timestamps and concatenates them to generate a vector representation of the history, i.e., $\theta_{T-r:T-1}$ of dimension $r \times n$ (please see the previous subsection for details). $\theta_{T-r:T-1}$ is then passed through *Influence Generator* Γ which outputs the K dimensional influence vector γ_T. For a particular global timestamp T, γ_T is fixed and can be re-used for any text generation task tied to the global timestamp T. The bottom middle section (Green Color) of Figure 4.1 shows the feed-forward neural network architecture of *Influence Generator*.

The next trick in ITG is to concatenate the influence vector (γ_T) to the vector representation of each word in the sequence generated so far. This means, for each word in

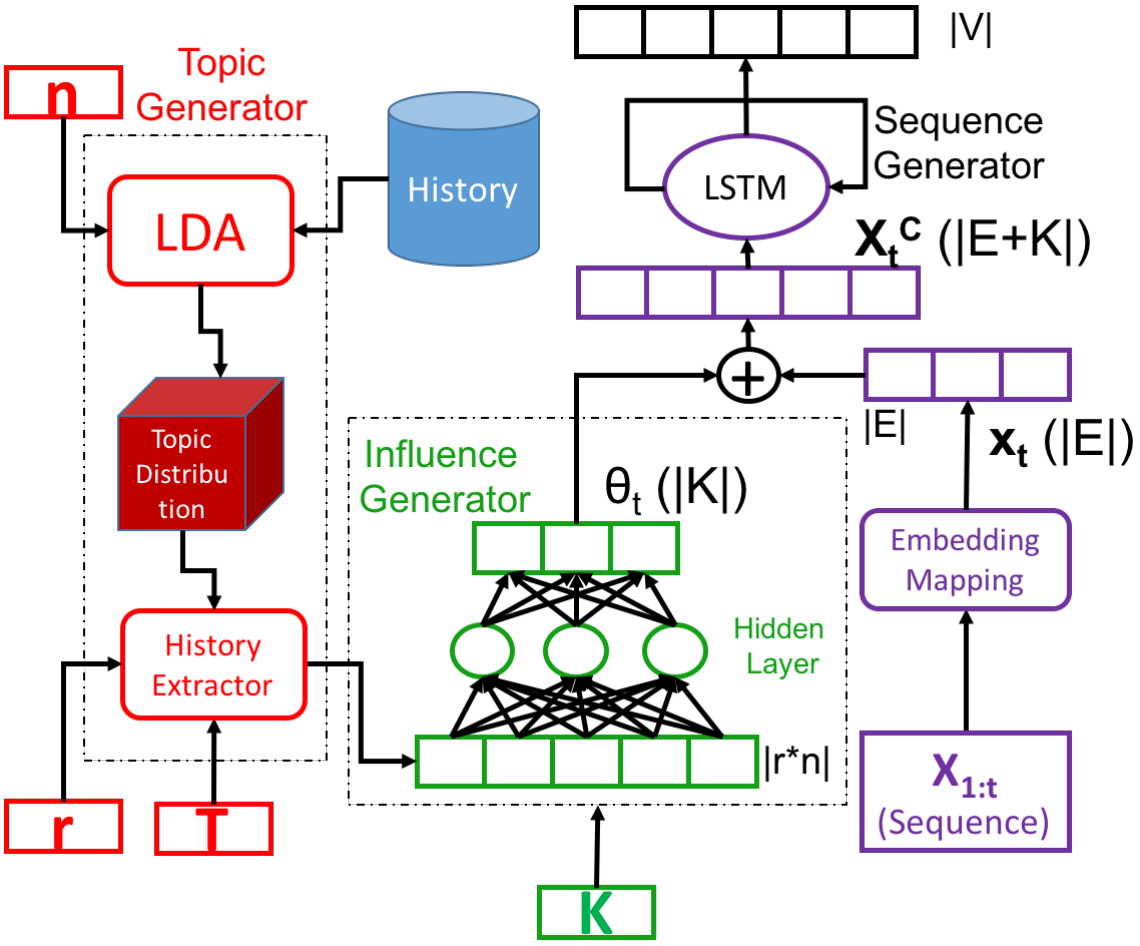


Figure 4.1: Influenced Text Generation (ITG): Compact Form

$\{x_1, x_2, ..., x_t\}$, γ_T is concatenated to each of their vector representations to get the augmented representation $\{x_1^C, x_2^C, ..., x_t^C\}$, i.e., while generating the next word $x_{t+1} = y_t$ in the sequence, all the previous words in the sequence share the same community-influence represented by vector γ_T. This augmented representation essentially allows ITG to capture the dynamic nature of text stream data as the influence vector injects evolving community-influence into the generation process. Finally, the augmented representations $\{x_1^C, x_2^C, ..., x_t^C\}$ are fed into the recurrent neural network model to compute a hidden state h_t . The final output vector Y is computed by applying a linear transformation on h_t. Note that, vector Y is a real-valued vector. We apply a Softmax function on Y to convert it into a valid probability distribution, sampling from which, the next word in the sequence is generated.

The mathematical formulas behind the entire generation process is summarized below:

$$Y = W^{\Omega} \cdot h_t + B^{\Omega}, \tag{4.6}$$

$$h_t = \Omega(h_{t-1}, x_t^C), \tag{4.7}$$

$$x_t^C = x_t \| \gamma_T. \tag{4.8}$$

Thus, Y can be written as follows:

$$Y = W^{\Omega} \cdot \Omega(h_{t-1}, x_t \| \gamma_T) + B^{\Omega}. \tag{4.9}$$

Here, γ_T is obtained as follows:

$$\gamma_T = W_2^{\Gamma} \cdot \left[ReLU(W_1^{\Gamma} \cdot \theta_{T-r:T-1} + B_1^{\Gamma}) \right] + B_2^{\Gamma}. \tag{4.10}$$

Here, $\theta_{T-r:T-1}$ is the concatenation of topic distribution vector from previous r timestamps.

Finally, we apply a Softmax function on the output vector Y to convert it into a valid probability distribution $P(y_t|h_t)$, as follows:

$$P(y_t = i|h_t) = \frac{\exp(Y_i)}{\sum_{j=1}^{|V|} \exp(Y_j)}. \tag{4.11}$$

The next word $y_t = x_{t+1}$ in the sequence is generated by sampling from this conditional distribution. The ITG repeats this whole process multiple times to generate new words in the sentence until a end-of-sentence marker is generated.

We graphically demonstrate how ITG generates sentences by iteratively generating one word at a time in Figure 4.2, which also shows the unrolled architecture of ITG. Before generating text, ITG infers n topics from the historical stream data and computes topic

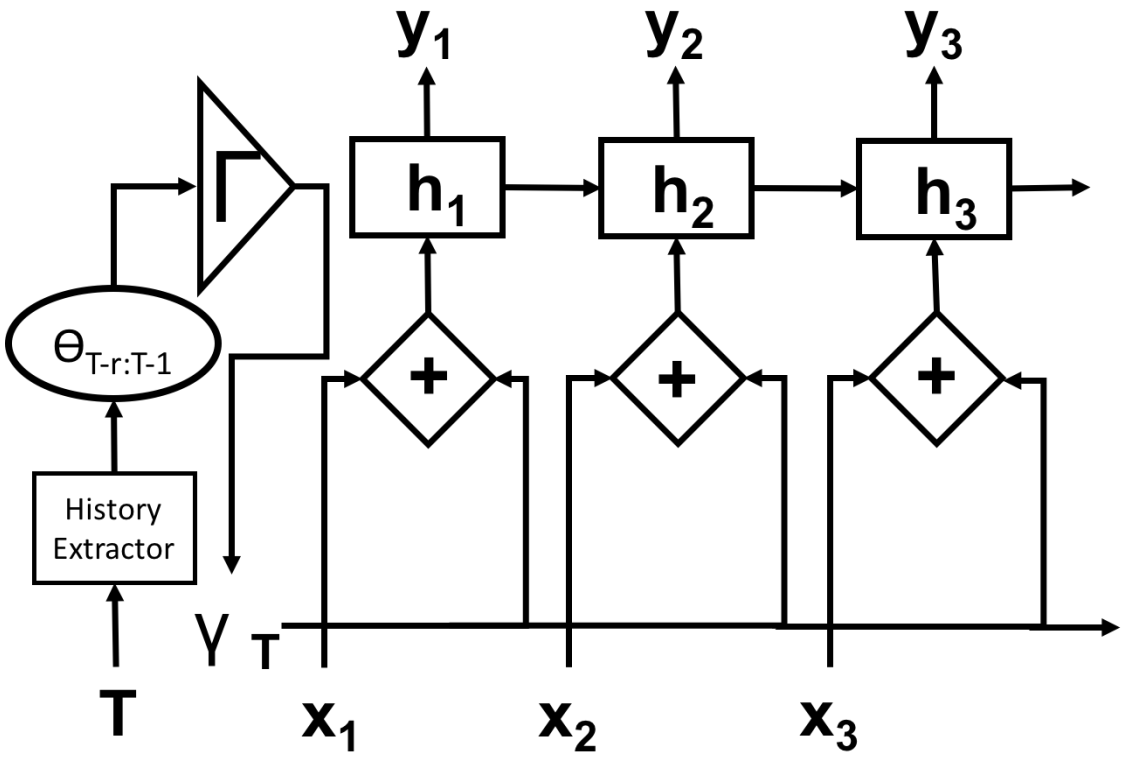


Figure 4.2: ITG: Unrolled Architecture

distribution for each timestamp which are stored in the *History Extractor* Module. The actual generation process starts with a timestamp T as input and the start-of-sentence marker (#) as the first generated word, i.e., $x_1 = \#$. Given input T, the *History Extractor* Module generates the concatenated topic distribution vector $\theta_{T-r:T-1}$, which in turn, is fed as an input to the *Influence Generator*, Γ. Γ generates the influence vector γ_T, which is augmented with the vector representation of x_1 as a bias, to form x_1^C. Then, x_1^C is passed through *Sequence Generator* which generates a hidden state h_1 as well as outputs the next word in the sequence, i.e., y_1. For generating the next word, y_1 becomes x_2 and goes through the same steps as before and generates hidden state h_2 as well as outputs y_2. This iterative process continues until $y_t = *$, the end-of-sentence marker. At this moment, the current sentence generation is complete and the next sentence can be generated by again starting with the start-of-sentence marker (#) as the first word in the sequence. Finally, the whole

process is summarized in Algorithm 1 which describes the generation of a single sentence by ITG for a particular global timestamp T.

Algorithm 1: Influenced Text Generation (ITG) Process

Process ITG $(T, \Theta, \Gamma, \Omega, r, n, K, E)$;
Input : T: discrete global timestamp
Θ: Topic Generator (n: number of topics)
Γ: Influence Generator (K: cardinality of influence vector)
Ω: Sequence Generator (E: cardinality of word vector)
r: History Window
Output: Generated sentence X corresponding to T
$x \leftarrow \{\text{start of sentence marker}\}$
$\theta_{T-r:T-1} \leftarrow \Theta(\theta_{T-r}) \| \Theta(\theta_{T-r+1}) \| ... \| \Theta(\theta_{T-1})$, Topic History
$\gamma_T \leftarrow \Gamma(\theta_{T-r:T-1})$, Generate Influence Vector for timestamp T
$t = 1$
repeat
  for $i \leftarrow 1$ **to** t **do**
    $x_i^C \leftarrow x_i \| \gamma_T$, where, $\|$ is concatenation operation
  end
  compute $h_t \leftarrow \Omega(x_{1:t-1}^C)$ by applying Ω recursively
  Draw word $y_t \sim P(y_t|h_t)$, where $P(y_t|h_t) \propto \exp(W^{\Omega} h_t + B^{\Omega})$
  $x \leftarrow x \cup \{y_t\}$
until *end of sentence marker is generated*;
return x

4.2.5 Estimation of ITG model parameters

In this section, we present the estimation techniques for the optimal values of ITG model parameters. Close observation through Equation 4.6-4.10 reveals that ITG contains the following set of parameters:

$$\boldsymbol{W} = \left\{W^{\Omega}, B^{\Omega}, W_1^{\Gamma}, B_1^{\Gamma}, W_2^{\Gamma}, B_2^{\Gamma}\right\} \tag{4.12}$$

We find the optimal values for the parameter set $\boldsymbol{W}$ by maximizing the log-likelihood of the training text stream data. The optimization problem thus can be written as follows:

$$\boldsymbol{W^*} = \underset{W}{argmax} \quad \log L(x_1 x_2 ... x_n | W) \tag{4.13}$$

As maximizing the log-likelihood is the same as minimizing the negative of the log-likelihood function and as we know the exact word which comes next in the sequence during the training process, our optimization problem boils down to minimizing the softmax cross entropy with logits between the conditional distribution $P(y_t|h_t)$ and the actual word that appears next in the training data. Softmax Cross Entropy with logits essentially measures the probability error in discrete classification tasks in which the classes are mutually exclusive. Thus,

$$\boldsymbol{W^*} = \underset{W}{argmin} \{ -\log L(x_1 x_2 ... x_n | W) \} \tag{4.14}$$

$$= \underset{W}{argmin} \quad \left\{ -\sum_{j=1}^{N} \sum_{i=1}^{|V|} I(x_j, i) \cdot \log P(y_t = i | h_t(W)) \right\} \tag{4.15}$$

Here, N is the total number of words in the training data. $I(x, y)$ is an indicator function that returns 1 if $x = y$ and 0 otherwise.

We use back-propagation to learn the weights of the network connection edges of ITG. Specifically, we use Adaptive Moment Estimation which is a popular stochastic gradient descent technique and commonly known as Adam Optimizer to compute the gradient for minimizing our objective function in Equation 4.15. Adam Optimizer is an update to the RMSProp [26], which is another popular optimizer. In Adam Optimizer, running averages of both the gradients and the second moments of the gradients are used. More more details, refer to [38].

4.3 Experimental Design

In this section, we describe our experimental setup including datasets used for experiments, baseline algorithms for comparison and the evaluation roadmap to measure the performance of ITG.

4.3.1 Dataset

We experimented with six different sets of publication title stream data to evaluate the performance of ITG. These datasets were collected from the Open Academic Graph[1][65, 61]. Here, we focused on studying how the paper titles published by different machine learning related conferences evolved over time. As community, we considered both the core machine learning community, e.g., NIPS, ICML as well as research communities that applies a fair share of machine learning, e.g, KDD, SIGIR. Specifically, we considered all the titles of papers published during the year span 1996-2015 by the following six conference venues: NIPS, CVPR, ICML, KDD, SIGIR and WWW. For these datasets, the discrete timestamp corresponds to a year, i.e., all papers published in the same year share the same discrete timestamp. Each row in these datasets consists of a tuple <timestamp,paper title> and altogether they contain $35,650$ paper titles in total. Again, each paper title can contribute multiple instances for predicting the next word which resulted in $309,966$ total instances. More details about the publication dataset is presented in Table 4.1.

4.3.2 Evaluation Roadmap

It is not possible to do a direct quantitative evaluation of the community-influence on text generation due to the lack of ground truth information. Thus, to evaluate the performance of the proposed ITG process, we rely on two different strategies: 1) Conduct qualitative evaluation and 2) Conduct indirect quantitative evaluation by applying ITG on some ap-

[1] `https://www.openacademic.ai/oag/`

Conf.	# of Titles	Title/Year	Total Words	Words/Title
KDD	5,499	274.95	49,980	9.08
NIPS	6,229	311.45	49,792	7.99
SIGIR	3,994	199.7	34,892	8.73
ICML	4,106	205.3	34,159	8.32
WWW	5,701	285.05	50,253	8.82
CVPR	10,121	506.05	90,890	8.98

Table 4.1: Dataset Summary

plication task. The primary purpose of indirect quantitative evaluation is to see whether capturing the evolution of community-influence actually helps us achieve better performance in some text generation application task. However, the goal task must involve the notion of evolving text stream data over time. For this reason, we consider Chronological Summary Generation as our goal task. The benefit of this task is that it provides a way to conduct indirect quantitative evaluation of ITG, as direct evaluation is impossible due to the lack of ground truth data for community-influence which is an abstract concept.

We first partition each publication dataset into 20 different timestamps where each partition corresponds to a year of publication and consists of the papers published in that particular year. Note that, this timestamp corresponds to the big T notation from the model description in section 4.2. This allows us to view the publication title datasets as text stream data.

While training of ITG is straightforward, testing of ITG is challenging due to the following two reasons: first, there is no standard testing metric to evaluate Chronological Summary Generation task and second, each timestamp T is associated with multiple independent publication titles; thus, after ITG generates a piece of title, it is unclear what is the golden publication title in the testing set against which the generated text should be compared.

Due to the difficulties associated with evaluating the performance of ITG, we propose a new way to evaluate the Chronological Summary Generation task. Two popular evaluation

Algorithm 2: Time aware BLEU score computation

Time aware BLEU (T, G, R);
Input : T: discrete timestamp
G: Generated Text set
R: Reference Text set
Output: Time sensitive BLEU
$score \leftarrow 0$
$|G| \leftarrow$ number of sentences in G
for *each sentence* g *in* G **do**
  for *each sentence* r *in* R **do**
    compute $BLEU(g, r)$
  end
  $r^* = \underset{r}{argmax} \quad BLEU(g, r)$
  $score \leftarrow score + BLEU(g, r^*)$
  $R \leftarrow R - \{r^*\}$
end
return $\frac{score}{|G|}$

metrics from the literature for text summarization are BLEU [52] and ROUGE [43] where a score is generated by comparing the automatically generated text against some reference text generated by humans. However, both BLEU and ROUGE do not consider the notion of time, thus we need a time-sensitive customization of both BLEU and ROUGE. The simplest way to do this is to compare a ITG generated text for timestamp T against the partition corresponding to the same timestamp T. This means, when ITG is asked to generate a text relevant to timestamp T, the ground truth text data against which the generated text is compared must also correspond to the timestamp T. To tackle the second challenge of matching against multiple independent publication titles, we adopt a simple greedy approach where the ITG generated text is matched against each ground truth sentence in the partition corresponding to the timestamp T and paired with the most similar one in terms of BLEU or ROUGE score. The matched ground truth sentence is then removed from the partition so that the next generated sentence cannot match with the previously matched sentence again.

This ensures that ITG is generating diverse set of sentences rather than just memorizing one single sentence from each timestamp T. This way, we can use ITG to generate multiple sentences for a particular timestamp T and then average the scores of all generated sentences to get an evaluation score corresponding to timestamp T. This whole computation process is presented in Algorithm 2, where we demonstrate the case for time-sensitive BLEU score. The case for ROUGE is exactly similar. Finally, these average scores across different timestamps can be further averaged to provide a unified Chronological Summarization score[2].

4.3.3 Baseline Methods

As the notions of influenced text generation and chronological summary generation of text stream data are new, there is no existing baseline we could readily use. Therefore, we had to resort to existing static text generation processes as our baseline methods to compare against. Two such baselines are simple bigram language models and Long-Short Term Memory (LSTM) [27, 21]. For a fair comparison, we also created two artificial baselines where text generation can evolve over time. The first one is called *RHLSTM* which is identical to ITG except the fact that the influence vector of RHLSTM is generated randomly as opposed to generating it by the *Influence Generator* of ITG. The second baseline is called *IILSTM* where we do not inject the influence vector as a bias into the vector representation of words, rather, the *Influence Generator* directly computes a probability distribution for sampling the next word and this probability is multiplied with the probability computed independently by LSTM. Table 4.2 contains the summary of these baseline algorithms along with ITG.

[2]All the codes and evaluation scripts for experimentation can be found at the following link: (**https://bitbucket.org/karmake2/itg/src/master/**)

Acronym	Details	Nature
Bigram	Bigram Language Model	static
LSTM	Long short-term memory	static
RILSTM	LSTM with Random Influence	dynamic
IILSTM	LSTM with Independent Influence	dynamic
ITG	Influenced Text Generation	dynamic
ITG-EI	Externally Influenced Text Generation	dynamic
ITG-CI	Combined Influenced Text Generation	dynamic

Table 4.2: Methods for Quantitative Comparison. The details of ITG-EI and ITG-CI are provided in section 4.4.1

4.4 Results

This section presents both quantitative and qualitative evaluation results for ITG. For quantitative evaluation, we compare the performances of ITG against multiple baseline methods for the Chronological Summary Generation task (section 4.4.1). For qualitative evaluation, we demonstrate how ITG captures the evolution of community-influence while generating text corresponding to a particular time-stamp (section 4.4.2). For all the results reported in this section, ITG used the following parameter settings: r was set to 3, for both *Sequence Generator* and *Influence Generator*, the number of hidden units was empirically set to 256, K (dimension of influence vector) was set to 15, batch size was set to 2000 instances and the learning rate was set to 0.01. For the Topic Generator, n (number of topics) was set to 15 and LDA was run using $\alpha = 0.1$ and $\beta = 0.05$. Note that, a comprehensive study of different parameter settings for ITG is beyond the scope of this paper.

4.4.1 Quantitative Evaluation

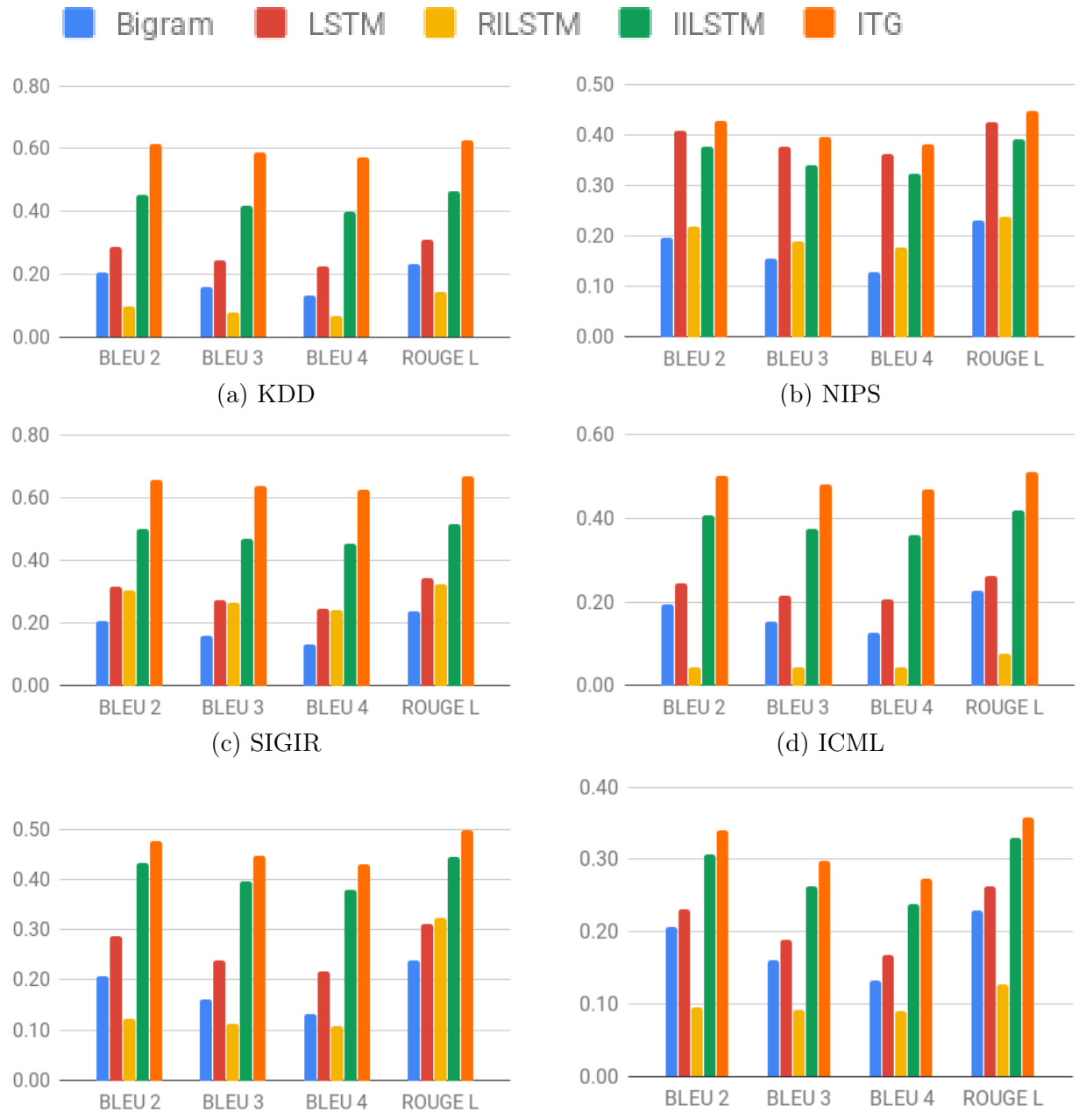
Figure 4.3 provides the summary of results for the Chronological Summary Generation task. Close examination of Figure 4.3 reveals that ITG outperforms all other baselines by a clear margin for all six datasets. For example, BLEU-4 score obtained by ITG on KDD Dataset is

0.57, while LSTM obtained only a score of 0.22. ROUGE-L score obtained by ITG is 0.63, while it is 0.31 for LSTM. This clearly indicates that ITG can indeed capture the temporal evolution of KDD paper titles over time and given a input timestamp T, can generate text relevant to T. Also note that, RILSTM performs significantly worse compared to LSTM for most datasets which implies that the influence vector plays the key role in helping ITG capture the evolution of the text stream. It is also noteworthy that IILSTM is the second best performing method which confirms that injecting influence vector as a bias into the word representation works better than using influence vector independently to compute a probability distribution and then multiply it with LSTM probabilities.

To get more insight into the performance of ITG, we plot the timestamp-wise performance of all compared methods for KDD Dataset (BLEU 4) and WWW Dataset (ROUGE L) in Figure 4.4 [other similar plots omitted due to lack of space]. A general inspection of Figure 4.4 also demonstrates the superiority of ITG for Chronological Summary Generation task, where, for any performance metric, ITG obtains the best score across different timestamps for most of the cases.

External Influence:

So far, we have only considered the influence of the community for which the text generation is targeted towards. However, we argue that, other related communities also pose indirect influence on the text generated within the target community. For example, a shift in the interest of theoretical machine learning conferences like ICML often influence the research directions pursued by the more applied conferences like KDD or WWW. To test this hypothesis, we conducted a series of experiments where instead of using the influence of the target community (e.g., KDD, WWW), we computed the influence vector from the historical topic distribution of a core machine learning community (e.g. ICML, NIPS). We call this approach ITG-EI where EI means external influence. We conducted another set of experiments where we computed the influence vectors from both the target community and a related external

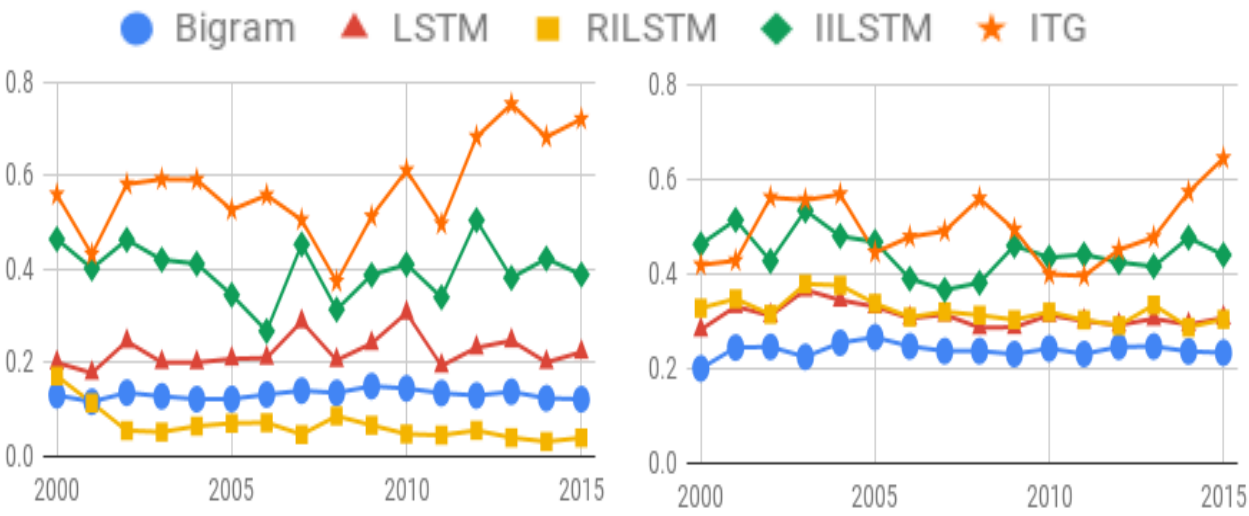


(a) KDD (b) NIPS

(c) SIGIR (d) ICML

(e) WWW (f) CVPR

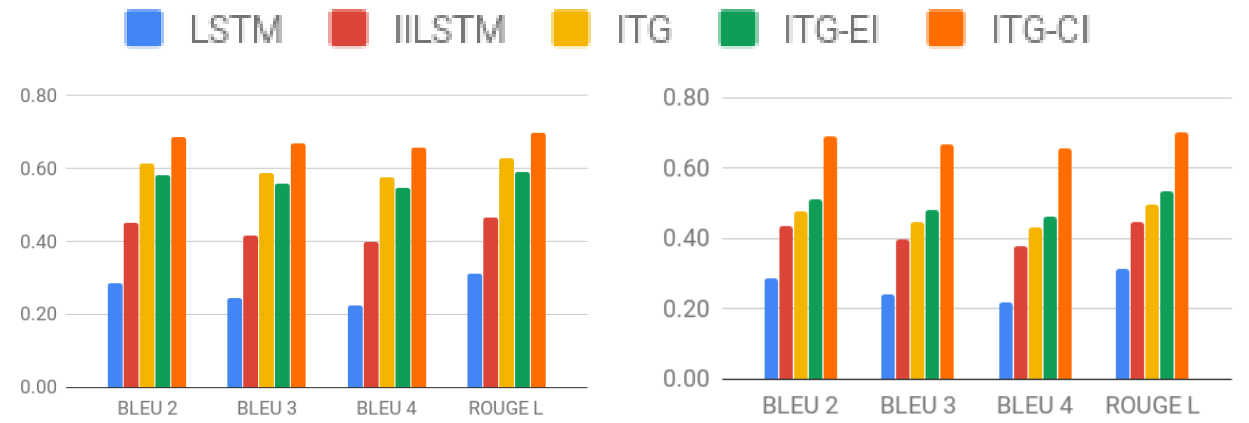
Figure 4.3: Comparison of ITG against baseline text generation techniques for Chronological Summarization task



(a) KDD: BLEU 4 (b) WWW: ROUGE L

Figure 4.4: Year-Wise Performance distribution of ITG against different baseline text generation techniques

community and injected both influence vectors into the ITG process. We call this approach ITG-CI where CI stands for combined influence. Two sample results from these experiments are shown in Figure 4.5. Experiments results suggest that, although ITG-EI is not always better than ITG itself, however, ITG-CI outperforms basic ITG as ITG-CI combines both internal and external influence.



(a) KDD Influenced by ICML (b) WWW Influenced by ICML

Figure 4.5: Results of adding external influence while generating text for Chronological Summarization task

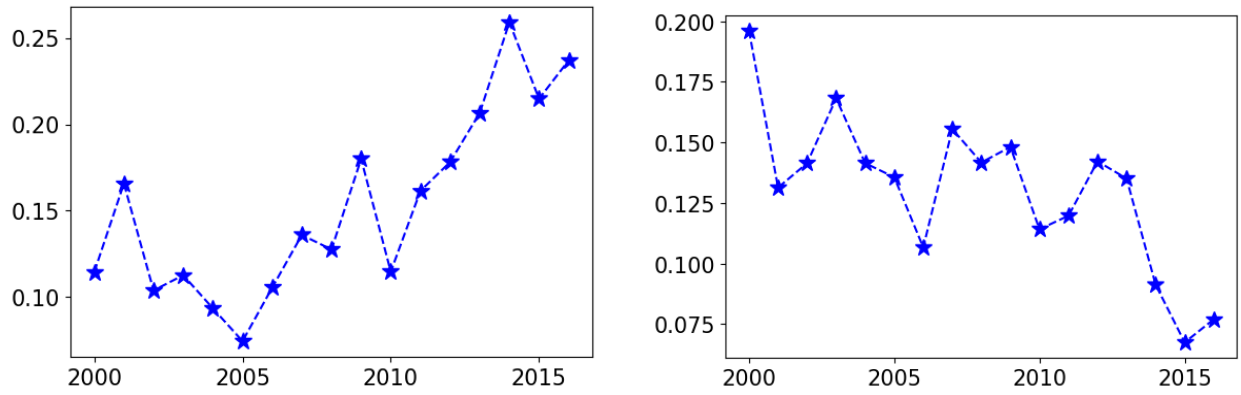
Topic	Top Keywords
Optimization	matrix, gradient, sparse, convex, stochastic
Search Relevance	information, retrieval, search, index, document
Rule Mining	rule, discovery, association, pattern, mine
Social Networks	social, network, recommender, community, topic
SVM Classifiers	supervised, learning, support, vector, machine
Behavior Modeling	log, behavior, personalization, click, feedback

Table 4.3: Samples Topics Extracted from KDD, SIGIR and ICML paper titles for year range [1995-2015] Using LDA

4.4.2 Qualitative Evaluation

In this section, we present qualitative results to show the great potentials of ITG. As mentioned in section 4.1, ITG should be able to generate syntactically correct, semantically coherent and time-sensitive evolving text. To demonstrate that ITG can generate sentences aligned with the evolution of the text stream corresponding to evolving community-influence, we first ran LDA on paper titles from KDD, ICML and SIGIR over the year range 2000-2015. Number of topics was set to 15. Table 4.3 shows six example topics along with top 5 keywords for each topic. For two topics, we also show how their distributions evolved over time within ICML community in Figure 4.6. For example, Figure 4.6a (4.6b) shows how the proportion of topic *Optimization* (*SVM*) within the paper titles published by ICML evolved over the year range [2000-2015]. We can clearly see from Figure 4.6 that, Optimization became more and more popular over the years in ICML (from $\sim$ 10% to $\sim$ 25%), while research on Support Vector Machine Theory matured during this time signified by the decay in proportion from $\sim$ 20% to $\sim$ 7%.

Next, we did the same analysis for KDD and SIGIR in Figure 4.7 with respect to two different topics for both KDD and SIGIR. For KDD, we considered topics *Social Networks* [Figure 4.7a, 4.7b] and Rule Mining [Figure 4.7c, 4.7d] and for SIGIR, we considered topics Search and Relevance [Figure 4.7e, 4.7f] and User Behavior Modeling [Figure 4.7g, 4.7h].



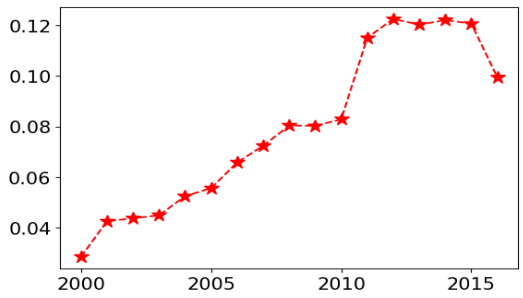
(a) ICML: Topic Optimization

(b) ICML: Topic SVM

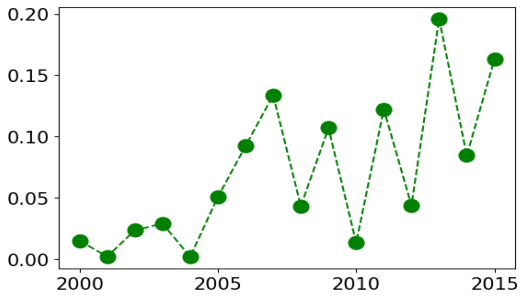
Figure 4.6: Topic Distribution in ICML paper titles over the year range: 2000-2015. Only 2 topics shown for lack of space.

However, beside plotting the original topic-distribution trend from the real conference proceedings (Figures with α marker and red color), we also plot the simulated topic-distribution trend computed from the text generated by ITG (Figures with β marker and green color). Close observation of Figure 4.7 confirms that ITG can indeed generate sentences aligned with the evolution of the text stream corresponding to evolving community-influence. For example, Figure 4.7g shows that research interest towards user behavior modeling grew significantly within SIGIR community in the past ten years, which is also nicely reflected in the text generated by ITG [Figure 4.7h]. On the other hand, research on association rule mining almost matured after 2008 within the KDD community [Figure 4.7c], which has also been captured effectively by ITG which is apparent from the decaying trend of Figure 4.7d.

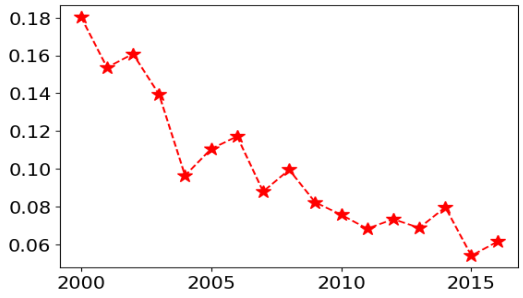
Finally, Table 4.4 presents some sample paper titles generated by ITG for different time ranges targeted towards KDD community. Given a year as input, ITG was invoked to generate a title for that particular year. A closer look into Table 4.4 reveals that ITG can indeed generate syntactically correct, semantically coherent and time-sensitive evolving text. It is also worth mentioning that, ITG did not store any paper to year mapping information. Table 4.4 also nicely captures the interest shift within KDD community over the years. For example, paper titles generated for year 2000-2002 includes topics like rule mining and tree



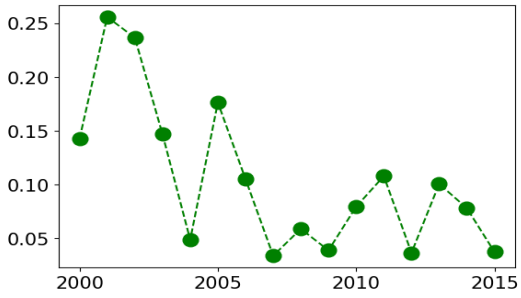
(a) KDD^{α}: Social Network



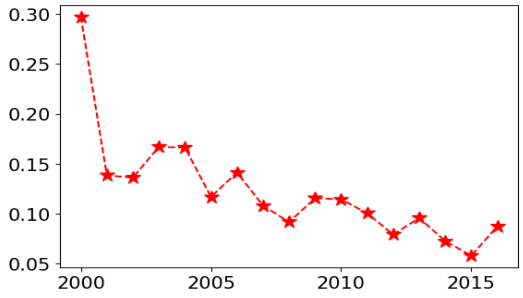
(b) KDD^{β}: Social Network



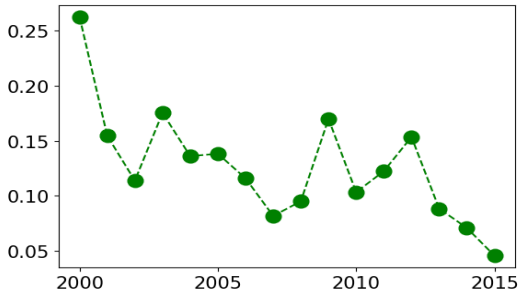
(c) KDD^{α}: Rule Mining



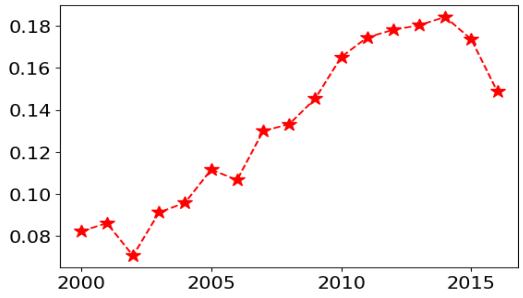
(d) KDD^{β}: Rule Mining



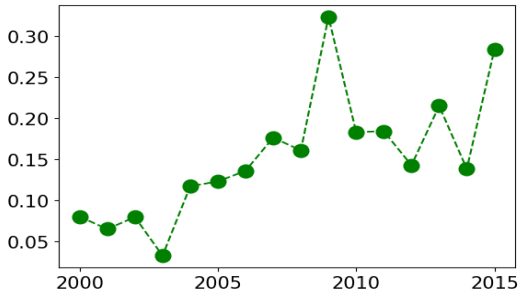
(e) $SIGIR^{\alpha}$: Search & Relevance



(f) $SIGIR^{\beta}$: Search & Relevance



(g) $SIGIR^{\alpha}$: Behavior Modeling



(h) $SIGIR^{\beta}$: Behavior Modeling

Figure 4.7: Topic distribution trend analysis to demonstrate how ITG captures community-influence while generating text. Captions of each subfigure is written in the following form: $x^{\alpha|\beta} : y$, where, x denotes the conference name, y denotes the topic being analyzed and α means the original topic distributions from real conference proceedings and β means topic distributions in text generated by ITG.

year	#	Sample Generated Title
2000 - 2002	1	discovering in hierarchical rules using lexical knowledge
	2	data mining criteria for tree based regression and classification
	3	mining frequent class sets in spatial databases
2007 - 2009	1	a framework for community identification in dynamic social networks
	2	learning preferences of new users in recommender systems
	3	data mining for intrusion detection from outliers
2012 - 2015	1	deep model based transfer and multi task learning for biological image analysis
	2	a bayesian framework for estimating properties of information network
	3	active learning for sparse bayesian classification

Table 4.4: Sample titles generated by ITG for conference KDD across different year ranges

based classifications, while paper titles generated for year 2012-2015 includes topics like deep learning and active learning, which is really interesting.

4.5 Related Works

There has been a surge of research interest in the use of neural network (NN) models for automatic text generation in recent years [18, 39, 36, 56]. The first NN-based text generator was proposed by Kukich [40], although generation was done only at the phrase level. Recent advances in recurrent neural network-based language models (RNN-LM) have demonstrated the value of distributed representations and its power to model arbitrarily long dependencies [49, 50]. Sutskever et al. [64] introduced a simple variant of the RNN that can generate meaningful sentences by learning from a character-level corpus. Mao et.al. have demonstrated how Recurrent Neural Networks, specially, Long-Short-Term-Memory (LSTM) is effective in solving various text generation tasks [47]. TopicRNN proposed by Dieng [18] integrated the merits of RNNs and latent topic models to capture long-range semantic de-

pendency. Recently, Generative Adversarial Nets (GANs) that use a discriminative model to guide the training of the generative model has shown promising results in automated text generation [55, 44, 13, 76]. However, all these existing methods from the literature are static text generation processes with no notion of time and thus, can not model the community-influence associated with dynamically evolving text stream data. Whereas, our aim was to develop a more dynamic text generation process that can capture this community-influence associated with the generation of stream text data.

Evolution of text stream data has primarily been studied from the perspective of topic modeling techniques [68, 46, 6], whereas our work is focused on the generation of exact sentences. Influence-Based Community Detection is another related area to our work (see [5] for a comprehensive survey), however, our goal is completely different as we primarily study Influence-Based Text Generation.

4.6 Conclusion

In this paper, we introduced and studied the challenging problem of modeling community-influence on the generation of dynamically evolving text stream data. We proposed an influenced text generation (ITG) process built upon a recurrent neural network based architecture which consists of three major components: 1) *Sequence Generator* for generating syntactically correct sentences 2) *Topic Generator* for generating historical topic distributions within the target and other related communities and finally 3) *Influence Generator* for capturing evolving community-influence on the text generation process. We quantitatively evaluated ITG on chronological summarization task through comprehensive experiments with six publication stream datasets. We demonstrated that ITG outperforms multiple baseline methods by a significant margin on the goal task and also presented a handful number of qualitative results to verify that ITG can indeed generate syntactically correct, semantically coherent and time-sensitive evolving text. Although we described ITG specifically in the context of

text generation, however, the model is quite general and can be applied to any sequence data which evolves over time. An important future direction is to apply ITG on different types of sequence data to verify the generality of the model.

Chapter 5

Conclusion

Understanding the influence of external factors on the behavior of users is an important research challenge which has not received significant attention so far in the literature. This thesis presents the first study of modeling external influence on user's information seeking and content generation behavior, which are two prominent ways to observe user interaction behavior.

The thesis started with demonstrating the value of big text data for mining the external influence on user interaction behavior. Next, I developed a new model for mining the influence of popular trending events on user search behavior. One particular limitation of this method was the assumption that each event poses its influence on the user search behavior which is unrealistic as many real word events are correlated and would pose a joint influence on user search behavior. To relax this assumption, I proposed a joint influence model based on Multivariate Hawkes Process which can model the inter-dependencies among different events in terms of their influence. Experimental results verified that the joint influence model can effectively capture the trend of the influence.

The last part of this thesis focuses on modeling the influence of external factors on user generated contents. I particularly focused on how evolving community-interest influence the content generation process by its users. I proposed a new deep-learning architecture called ITG which computes an influence embedding that is injected as a bias in the text generation process. Experimental results indicate that the influence embedding indeed captures evolving community-interest in a meaningful way which enables time-aware text generation possible.

All the problems studied in this thesis are entirely new which have never been studied

before. Thus, there are still many open challenges in this area which need to be addressed before this research direction can mature. However, as a first step towards mining influence from unstructured data, this thesis can provide a roadmap for following research in this direction. One particular future direction is to incorporate the influence models into the SOFSAT framework proposed in [58]. I hope this thesis will encourage my fellow researchers to work more in this area and thus, contribute towards building more comprehensive intelligent systems in the future.

References

[1] Review: In 'captain america: Civil war', super-bro against super-bro. http://www.nytimes.com/2016/05/06/movies/captain-america-civil-war-review-chris-evans.html?_r=0. Accessed: 2016-07-30.

[2] The most popular api: Nytimes developers network. https://developer.nytimes.com/most_popular_api_v2.json\#/README, 2016. Accessed: 2016-07-30.

[3] H. Abdelhaq, C. Sengstock, and M. Gertz. Eventweet: Online localized event detection from twitter. *Proceedings of the VLDB Endowment*, 6(12):1326–1329, 2013.

[4] F. Atefeh and W. Khreich. A survey of techniques for event detection in twitter. *Computational Intelligence*, 31(1):132–164, 2015.

[5] N. Barbieri, F. Bonchi, and G. Manco. Efficient methods for influence-based network-oblivious community detection. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(2):32, 2017.

[6] R. C. Barranco, A. P. Boedihardjo, and M. S. Hossain. Analyzing evolving stories in news articles. *arXiv preprint arXiv:1703.08593*, 2017.

[7] K. Berberich, S. Bedathur, O. Alonso, and G. Weikum. A language modeling approach for temporal information needs. In *European Conference on Information Retrieval*, pages 13–25. Springer, 2010.

[8] C. Blundell, K. A. Heller, and J. M. Beck. Modelling reciprocating relationships with hawkes processes. *NIPS*, 2012.

[9] P. T. Boggs and J. W. Tolle. Sequential quadratic programming. *Acta numerica*, 4:1–51, 1995.

[10] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[11] R. Campos, G. Dias, A. M. Jorge, and A. Jatowt. Survey of temporal information retrieval and related applications. *ACM Computing Surveys (CSUR)*, 47(2):15, 2015.

[12] C. Chatfield. *The analysis of time series: an introduction*. CRC press, 2016.

[13] T. Che, Y. Li, R. Zhang, R. D. Hjelm, W. Li, Y. Song, and Y. Bengio. Maximum-likelihood augmented discrete generative adversarial networks. *CoRR*, abs/1702.07983, 2017.

[14] J. Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960.

[15] R. Crane and D. Sornette. Robust dynamic classes revealed by measuring the response function of a social system. *Proceedings of the National Academy of Sciences of the United States of America*, 105(41):15649–15653, 2008.

[16] W. Dakka, L. Gravano, and P. Ipeirotis. Answering general time-sensitive queries. *IEEE Transactions on Knowledge and Data Engineering*, 24(2):220–235, 2012.

[17] D. J. Daley and D. Vere-Jones. *An introduction to the theory of point processes: volume II: general theory and structure*. Springer Science & Business Media, 2007.

[18] A. B. Dieng, C. Wang, J. Gao, and J. Paisley. Topicrnn: A recurrent neural network with long-range semantic dependency. *arXiv preprint arXiv:1611.01702*, 2016.

[19] X. Dong, D. Mavroeidis, F. Calabrese, and P. Frossard. Multiscale event detection in social media. *Data Mining and Knowledge Discovery*, 29(5):1374–1405, 2015.

[20] E. Errais, K. Giesecke, and L. R. Goldberg. Affine point processes and portfolio credit risk. *SIAM J. Fin. Math.*, 1(1):642–665, Sep 2010.

[21] F. A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: Continual prediction with lstm. 1999.

[22] S. N. Ghoreishi and A. Sun. Predicting event-relatedness of popular queries. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 1193–1196. ACM, 2013.

[23] R. Glaudell, R. T. Garcia, and J. B. Garcia. Nelder-mead simplex method. *Computer Journal*, 7:308–313, 1965.

[24] C. Goutte and E. Gaussier. A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. In *ECIR*, pages 345–359. Springer, 2005.

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

[26] G. Hinton, N. Srivastava, and K. Swersky. Rmsprop: Divide the gradient by a running average of its recent magnitude. *Neural networks for machine learning, Coursera lecture 6e*, 2012.

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

[28] J. Jiang, D. He, and J. Allan. Searching, browsing, and clicking in a search session: changes in user behavior by task and over time. In *ACM SIGIR*, 2014.

[29] S. Kairam, M. Morris, J. Teevan, D. Liebling, and S. Dumais. Towards supporting search over trending events with social media. In *International AAAI Conference on Web and Social Media*, 2013.

[30] N. Kanhabua, T. Ngoc Nguyen, and W. Nejdl. Learning to detect event-related queries for web search. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1339–1344. ACM, 2015.

[31] S. K. Karmaker Santu, V. Bindschadler, C. Zhai, and C. A. Gunter. Nrf: A naive re-identification framework. In *Proceedings of the 2018 Workshop on Privacy in the Electronic Society*, pages 121–132. ACM, 2018.

[32] S. K. Karmaker Santu, L. Li, Y. Chang, and C. Zhai. Jim: Joint influence modeling for collective search behavior. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 637–646. ACM, 2018.

[33] S. K. Karmaker Santu, L. Li, D. H. Park, Y. Chang, and C. Zhai. Modeling the influence of popular trending events on user search behavior. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 535–544. International World Wide Web Conferences Steering Committee, 2017.

[34] S. K. Karmaker Santu, P. Sondhi, and C. Zhai. Generative feature language models for mining implicit features from customer reviews. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 929–938. ACM, 2016.

[35] S. K. Karmaker Santu, P. Sondhi, and C. Zhai. On application of learning to rank for e-commerce search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 475–484. ACM, 2017.

[36] C. Kiddon, L. Zettlemoyer, and Y. Choi. Globally coherent text generation with neural checklist models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 329–339, 2016.

[37] Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush. Character-aware neural language models. In *AAAI*, pages 2741–2749, 2016.

[38] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[39] R. Kiros, R. Salakhutdinov, and R. Zemel. Multimodal neural language models. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 595–603, 2014.

[40] K. Kukich. *Where do Phrases Come from: Some Preliminary Experiments in Connectionist Phrase Generation*, pages 405–421. Springer Netherlands, Dordrecht, 1987.

[41] A. Kulkarni, J. Teevan, K. M. Svore, and S. T. Dumais. Understanding temporal query dynamics. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 167–176. ACM, 2011.

[42] L. Li, H. Deng, A. Dong, Y. Chang, and H. Zha. Identifying and labeling search tasks via query-based hawkes processes. In *ACM SIGKDD*, 2014.

[43] C.-Y. Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8. Barcelona, Spain, 2004.

[44] K. Lin, D. Li, X. He, Z. Zhang, and M. Sun. Adversarial ranking for language generation. *CoRR*, abs/1705.11001, 2017.

[45] T. J. Liniger. *Multivariate hawkes processes*. PhD thesis, SWISS FEDERAL INSTITUTE OF TECHNOLOGY ZURICH, 2009.

[46] Z. Lu, N. Mamoulis, and D. W. Cheung. A collective topic model for milestone paper discovery. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 1019–1022. ACM, 2014.

[47] J. Mao, W. Xu, Y. Yang, J. Wang, Z. Huang, and A. Yuille. Deep captioning with multimodal recurrent neural networks (m-rnn). *arXiv preprint arXiv:1412.6632*, 2014.

[48] Y. Matsubara, Y. Sakurai, and C. Faloutsos. The web as a jungle: Non-linear dynamical systems for co-evolving online activities. In *Proceedings of the 24th International Conference on World Wide Web*, pages 721–731. ACM, 2015.

[49] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048, 2010.

[50] T. Mikolov, S. Kombrink, L. Burget, J. Cernocký, and S. Khudanpur. Extensions of recurrent neural network language model. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2011, May 22-27, 2011, Prague Congress Center, Prague, Czech Republic*, pages 5528–5531, 2011.

[51] T. Ozaki. Maximum likelihood estimation of hawkes' self-exciting point processes. *Annals of the Institute of Statistical Mathematics*, 31(1):145–155, 1979.

[52] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

[53] G. Pekhimenko, D. Lymberopoulos, O. Riva, K. Strauss, and D. Burger. Pockettrend: Timely identification and delivery of trending search content to mobile users. In *WWW*, pages 842–852. ACM, 2015.

[54] M. M. Rahman, S. K. K. Santu, M. M. Islam, and K. Murase. Forecasting time series?a layered ensemble architecture. In *Neural Networks (IJCNN), 2014 International Joint Conference on*, pages 210–217. IEEE, 2014.

[55] S. Rajeswar, S. Subramanian, F. Dutil, C. J. Pal, and A. C. Courville. Adversarial generation of natural language. *CoRR*, abs/1705.10929, 2017.

[56] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*, 2015.

[57] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, M. Gatford, et al. Okapi at trec-3. *NIST SPECIAL PUBLICATION SP*, 109:109, 1995.

[58] S. K. K. Santu, C. Geigle, D. Ferguson, W. Cope, M. Kalantzis, D. Searsmith, and C. Zhai. Sofsat: Towards a set-like operator based framework for semantic analysis of text.

[59] S. K. K. Santu, M. M. Rahman, M. M. Islam, and K. Murase. Towards better generalization in pittsburgh learning classifier systems. In *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pages 1666–1673. IEEE, 2014.

[60] F. Sebastiani. An axiomatically derived measure for the evaluation of classification algorithms. In *SIGIR ICTIR, 2015*, pages 11–20. ACM, 2015.

[61] A. Sinha, Z. Shen, Y. Song, H. Ma, D. Eide, B.-j. P. Hsu, and K. Wang. An overview of microsoft academic service (mas) and applications. In *Proceedings of the 24th international conference on world wide web*, pages 243–246. ACM, 2015.

[62] A. Stomakhin, M. B. Short, and A. L. Bertozzi. Reconstruction of missing data in social networks based on temporal patterns of interactions. *Inverse Problems.*, 27(11), Nov 2011.

[63] J. Strötgen and M. Gertz. Event-centric search and exploration in document collections. In *Proceedings of the 12th ACM/IEEE-CS joint conference on Digital Libraries*, pages 223–232. ACM, 2012.

[64] I. Sutskever, J. Martens, and G. E. Hinton. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 1017–1024, 2011.

[65] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su. Arnetminer: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 990–998. ACM, 2008.

[66] E. M. Voorhees et al. The trec-8 question answering track report. In *Trec*, volume 99, pages 77–82, 1999.

[67] M. Walther and M. Kaisser. Geo-spatial event detection in the twitter stream. In *ECIR*, pages 356–367. Springer, 2013.

[68] X. Wang, C. Zhai, and D. Roth. Understanding evolution of research themes: a probabilistic generative model for citations. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1115–1123. ACM, 2013.

[69] Y. Wang, D. Seyler, S. K. K. Santu, and C. Zhai. A study of feature construction for text-based forecasting of time series variables. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 2347–2350. ACM, 2017.

[70] Y. Wang, L. Wang, Y. Li, D. He, W. Chen, and T.-Y. Liu. A theoretical analysis of ndcg ranking measures. In *Proceedings of the 26th Annual Conference on Learning Theory (COLT 2013)*, 2013.

[71] W. Webber, A. Moffat, and J. Zobel. A similarity measure for indefinite rankings. *ACM Transactions on Information Systems (TOIS)*, 28(4):20, 2010.

[72] R. W. White, W. Chu, A. Hassan, X. He, Y. Song, and H. Wang. Enhancing personalized search by mining and modeling task behavior. In *WWW*, 2013.

[73] A. Z.-Mangion, M. Dewarc, V. Kadirkamanathand, and G. Sanguinetti. Point process modelling of the afghan war diary. *PNAS*, 109(31):12414–12419, July 2012.

[74] H. Zaragoza, N. Craswell, M. J. Taylor, S. Saria, and S. E. Robertson. Microsoft cambridge at trec 13: Web and hard tracks. In *TREC*, volume 4, pages 1–1, 2004.

[75] R. Zhang, Y. Konda, A. Dong, P. Kolari, Y. Chang, and Z. Zheng. Learning recurrent event queries for web search. In *Proceedings of the EMNLP 2010*, pages 1129–1139. Association for Computational Linguistics, 2010.

[76] Y. Zhang, Z. Gan, K. Fan, Z. Chen, R. Henao, D. Shen, and L. Carin. Adversarial feature matching for text generation. *arXiv preprint arXiv:1706.03850*, 2017.

[77] D. Zhou, L. Chen, and Y. He. An unsupervised framework of exploring events on twitter: Filtering, extraction and categorization. In *AAAI*, pages 2468–2475, 2015.

[78] J. Zhuang, Y. Ogata, and D. V. Jones. Stochastic declustering of space-time earthquake occurrences. *Journal of the American Statistical Association.*, 97(458):369–380, 2002.