LEARNING TO PREDICT THE FUTURE USING WEB KNOWLEDGE AND DYNAMICS

KIRA RADINSKY

LEARNING TO PREDICT THE FUTURE USING WEB KNOWLEDGE AND DYNAMICS

RESEARCH THESIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

KIRA RADINSKY

SUBMITTED TO THE SENATE OF THE TECHNION — ISRAEL INSTITUTE OF TECHNOLOGY

KISLEV, 5773 HAIFA DECEMBER, 2012

THIS RESEARCH THESIS WAS DONE UNDER THE SUPERVISION OF
ASSOC. PROF. SHAUL MARKOVITCH AND DR. NIR AILON IN THE
DEPARTMENT OF COMPUTER SCIENCE

ACKNOWLEDGMENTS

The journey towards this thesis crossed paths with so many other people's journeys, each leaving their mark on me. I had the honor of having three great mentors each raising me to be the person and the researcher I am today:

Shaul Markovitch – for his father-like guidance and care over the past 6 years, since the first day I showed up in his office as a B.Sc student. I am most grateful for him giving me the freedom to choose my research path, and for allowing me to do mistakes and supporting me when it was time to correct them. For explaining me what a good researcher is – pushing me to focus and consolidate my work rather than researching multiple, less significant problems. But above all, I thank him for teaching me, by way of example, to be kind, less impulsive and more patient towards my work, in my decisions and in my opinions.

Eric Horvitz – for encouraging me to dream, for his passion for research, for intriguing me with thoughts and ideas that I never had before, and, eventually, for helping me reach my research goals.

Susan Dumais – for helping me to focus my research ADD into a well-polished set of ideas and not to overlook the little details.

I am also in debt to all my collaborators and co-authors around the world. This thesis would not be what it is today without them: Evgeniy Gabrilovich, Eugene Agichtein, Krysta Svore, Jaime Teevan, Milad Shokouhi, Dan Liebling, Alex Bocharov, Fernando Diaz, Maarten de Rijke, Paul Bennett, Yi Chang, Anlei Dong, Ashish Kapoor, Susan Dumais, Eric Horvitz, Shaul Markovitch and Sagie

Davidovich. In particular, I'd like to express my appreciation to Evgeniy Gabrilovich who introduced me to the Web community.

My time at the Technion gave me the opportunity to work with many amazing people who shared their wisdom, expertise and kind advice. I owe special thanks to Freddy Bruckstein, Erez Petrank and Nader Bshouty for countless advice.

I would like to thank my longtime friend and officemate Gal Lavee for helping me enjoy the moment and for adding his final touch to everything I wrote. Thanks to Lev Finkelstein for many brain storms. Thanks to Hadas Yaari, Yaniv Hemo, Dotan Elharrar, Jonathan Yaniv, Eli Nazarov and Assaf Israel for being truly happy for me when I needed them.

"And, when you want something, all the universe conspires in helping you to achieve it" (Paulo Coelho, "The Alchemist"), and in my case, I want to thank my parents who always believed in me unconditionally and my husband and dearest childhood friend, Sagie Davidovich, for always conspiring to help me, when the universe could not. This thesis is dedicated to you.

THE GENEROUS FINANCIAL HELP OF THE TECHNION IS GRATEFULLY ACKNOWLEDGED

To my husband, Sagie Davidovich,
and my mom and aunt, Natasha and Milena Radinsky

List of Publications

The work described in this thesis is partially based on the following publications:

1. K. Radinsky and E. Horvitz, Mining the Web to Predict Future Events, in proceedings of the 6th ACM International Conference on Web Search and Data Mining (WSDM '13), Rome, Italy.

2. K. Radinsky and Paul N. Bennett, Predicting Content Change on the Web, in proceedings of the the 6th ACM International Conference on Web Search and Data Mining (WSDM '13), Rome, Italy.

3. M. Shokouhi and K. Radinsky, Time-Sensitive Query Auto-Completion, in proceedings of the 35th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '12), Portland, US.

4. K. Radinsky, S. Davidovich and S. Markovitch, Learning Causality for News Events Prediction, in proceedings of the 21st International World Wide Web Conference (WWW '12), France.

5. K. Radinsky, K. Svore, S. Dumais, J. Teevan, A. Bocharov and E. Horvitz, Modeling and Predicting Behavioral Dynamics on the Web, in proceedings of the 21st International World Wide Web Conference (WWW '12), France.

6. K. Radinsky, E. Agichtein, E. Gabrilovich and S. Markovitch, A Word at a Time: Computing Word Relatedness using Temporal Semantic Analysis, in proceedings of the 20th International World Wide Web Conference (WWW '11), India.

7. K. Radinsky and N. Ailon, Ranking From Pairs and Triplets: Information Quality, Evaluation Methods and Query Complexity, in proceedings of the fourth ACM International Conference on Web Search and Data Mining (WSDM '11), Hong Kong.

8. K. Radinsky, S. Davidovich and S. Markovitch, Predicting the news of tomorrow using patterns in web search queries, in proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (WI'08), Sydney, Australia.

9. K. Radinsky, S. Davidovich, S. Markovitch, Learning Causality from Textual Data, in proceedings of the IJCAI Workshop on Learning by Reading and its Applications in Intelligent Question-Answering (IJCAI '11 Workshops), Spain.

10. K. Radinsky, A. Kapoor, A. Oron, K. Master, Brain-Computer Interfaces for Music Recommendation, in proceedings of the Neural Information Processing Systems Foundations (NIPS '11 Workshops), Spain.

11. K. Radinsky, F. Diaz, S. T. Dumais, M. Shokouhi, A. Dong, Y. Chang, Temporal web dynamics and its application to information retrieval, in proceedings of the the 6th ACM International Conference on Web Search and Data Mining (WSDM '13), Rome, Italy.

12. K. Radinsky, S. Davidovich and S. Markovitch, Learning to Predict from Textual Data. Journal of Artificial Intelligence Research (JAIR) 45: 641-684 (2012).

13. K. Radinsky, K. Svore, S. Dumais, M. Shokouhi , J. Teevan, A. Bocharov and E. Horvitz, Behavioral Dynamics on the Web: Learning, Modeling and Prediction. ACM Transactions on Information Systems (TOIS).

You see things; and you say, "Why?" But I dream things that never were; and I say, "Why not?"

– George Bernard Shaw

Contents

List of Figures

List of Tables

Abstract

Mark Twain famously said that "the past does not repeat itself, but it rhymes." In the spirit of this reflection, we present novel algorithms and methods for leveraging large-scale digital histories and human knowledge mined from the Web to make real-time predictions about the likelihoods of future human and natural events of interest.

The Web is a dynamic being, with constantly updating content, which is entangled with sophisticated user behaviors and interactions. Some of these behaviors have the ability to convey current trends in the present, e.g., economical growth (predicting automobile sales based on query volume [Varian and Choi, 2009]), popular movies [Kalev, 2011], and political unrest [Asur and Huberman, 2010; Joshi et al., 2010; Mishne, 2006]. We mine the ever-changing Web content and user Web behavior. We show that, not only the dynamics itself can be predicted, but also that it can be used for future real-world event prediction.

We mine decades of news reports (1851 – 2010) from the New York Times (NYT), and describe how we can learn to predict the future by generalizing sets of concrete transitions in sequences of reported news events. In addition to the news corpora, we leverage data from freely available Web resources, including Wikipedia, FreeBase, OpenCyc, and GeoNames, via the LinkedData platform [Bizer et al., 2009]. The goal is to build predictive models that generalize from specific sets of sequences of events to provide likelihoods of future outcomes, based on patterns of evidence observed in near-term Web activities. We propose the methods as a means of generating actionable forecasts in advance of the occurrence of target events in the world.

This thesis is one of the first works to demonstrate general, unrestricted artificial-intelligence prediction capacity. We present methods derived from heterogeneous Web sources to make knowledge-intensive reasoning about causality and future event prediction, using both automatic feature extraction and novel algorithms for generalizing over historical examples.

Chapter 1

Introduction

When an agent, situated in a complex environment, plans its actions, it reasons about future changes to the environment. Some of these changes are a result of its own actions, but many others are a result of various chains of events, possibly a result of actions of other agents situated in the same environment. In the past, computerized agents could not operate in complex environments due to their limited perceptive capabilities. The proliferation of the World Wide Web, however, changed all that. An intelligent agent can act in the virtual world of the Web, perceiving the current state of the world through extensive sources of textual information, including Web pages, tweets, news reports, and online encyclopedias, and performing various tasks such as searching, organizing, and generating information.

Indeed, the Web can be viewed as an extremely large environment where billions of agents act to achieve their goals. Some of these are human agents, consuming and generating information, and some are computerized agents, such as search engines, crawlers and Web servers. To act intelligently in such a complex virtual environment, the agents must be able to perceive the current state and reason about future states by predicting actions of other agents. In particular, computerized agents can significantly benefit by predicting the actions of the human users interacting with them. For example:

1. Predicting the frequency of a query on a certain day. Such a prediction can improve the performance of a query autocompletion mechanism of a search engine.

2. Given a query, predicting the probability that a result url will be clicked. This can help a search engine improve its ranking algorithm.

3. Predicting the popularity of a url on a certain day. This can help a crawler prioritize its page visit schedule.

4. Predicting the probability that a certain page will be changed on a certain day. Such prediction, again, can improve crawling.

The World Wide Web is highly dynamic and is constantly evolving to cover the latest information about the physical and social updates in the real world. The virtual environment of the Web reflects many aspects of the real world. The changes in the virtual world of the Web are entangled with new information needs and time-sensitive user interactions with information sources originating from the human agent. Therefore, to fully understand the human agents, the virtual agents must be able to perceive and reason about the real world. Web-based predictions can enable us to reason about future virtual states and predict, in fact, events that will take place in the real world, thus affecting the virtual one. For example, predicting the probability of a title to appear in the news can be helpful in conducting tasks such as identifying political unrest, detecting and tracking social trends, and generally supporting decision making by politicians, businesspeople, and individual users. Those in turn can help the virtual agents to predict future actions of the human agents in the virtual environment, such as search for information.

In this work, we present a new approach for conducting such prediction tasks. Our approach is unique in its use of two major sources of knowledge:

1. *Web Dynamics:* The Web is a dynamic being, which reflects the ever-changing information of our world, with the vast number of agents continuously acting upon the Web, consuming, producing and modifying the information it contains. This dynamics of actions is an extremely useful source of knowledge for predicting future states of the Web, yet, not many previous works exploit this dynamics. In this work, we present prediction algorithms that use Web dynamic extensively. For example, the history of modifications to a Web page can be used to predict the schedule of the page changes and can be used to design efficient crawling algorithms.

2. *World Knowledge:* Complex problem solving requires the usage of extensive common-sense knowledge [Lenat and Feigenbaum, 1991], especially in complex prediction tasks that involve large quantity of diverse textual information, such as the Web. Fortunately, the Web is also a source for such knowledge as it encapsulates much of the existing human knowledge. A treasure trove of information about historical events is contained in news archives and encyclopedias on the Web. Dynamic updates about current events are available through online newspapers, which update every minute. Additionally, the Web has many common-sense ontologies (e.g., Linked-Data, Wikipedia etc.) of entities and their relationships. All this knowledge can serve as the basis for performing true human-like prediction. For example, using a large collection of news articles and large Web-based ontologies,

our algorithms are able to predict future news items (and thus potentially future real-world events).

In this thesis, we present a set of algorithms for complex prediction tasks that is based on these two extensive resources.

In the first part of the thesis, we tackle prediction tasks that are associated with the human agents in the virtual world. We start with predicting user behavior based on historical dynamics and interactions on the Web, e.g., predicting search query popularity, predicting URL clicks, and predicting when an agent is likely to modify the content of a Web page. We represent those behaviors using time-series models and present significant gains over existing prediction methods. Additionally, we present novel methods for applying this framework to several search applications: time-aware query auto suggestions, time-aware ranking and improved crawling – all with significant gains over previous state-of-the-art works.

In the second part, we present algorithms for predicting future events in the real world based on the dynamics of the virtual world of the Web. We start with an algorithm that uses Web dynamics to predict the way that people perceive the real world – specifically, we try to predict the semantic similarity of real-world concepts. We continue and present an algorithm that mines this dynamics of large-scale Web resources to predict events that are likely to appear in the real-world news with high precision.

In the third part, we present an algorithm for predicting future events in the real world based on the knowledge accumulated on the Web. As specific events are very rare, the prediction algorithm must have the ability to learn and generalize from past events. For this reason, one must endow the machine the ability to obtain and encode the expertise which humans acquire throughout their lifetime with their interaction with the world. We mine millions of causality pairs extracted from 150 years of news archives on the Web, generalize them using Web ontologies, and present a system that generates human-like predictions in natural language given a predefined event.

Finally, we present an algorithm for predicting future events based on both Web dynamics and Web knowledge. The algorithm mines chains of events from the Web and then performs automated abstraction to enhance the size of training sets for learning and to generalize the inferences. The generalization is performed by moving from specific entities to broader classes of entities.

For example, in Figure 7.1, we present a real prediction example. The figure illustrates an alert of a cholera outbreak in Angola issued by the system. The alert was generated due a chain of events, beginning with the occurrence of a storm and then followed by a drought a year later. The system inferred, based on historical events, that this particular chain of events, occurring in the least developed countries or occurring in a landlocked countries with high population density, are most

likely to be followed by a cholera outbreak in that region. Although there were no sufficient number of examples to infer about such outbreak in Angola, there was enough data about such examples from other underdeveloped countries in Africa, such as Rwanda. The knowledge that Angola and Rwanda are located in Africa and their particular demographics and geological properties are extracted from Web ontologies, such as Wikipedia.

We propose the predictive modeling from the Web as a means of building insights about the likelihoods of transitions as well as for powering up automated alerting in the real world. We demonstrate this predictive power of mining thousands of news stories to create classifiers for several representative prediction challenges, including proactive alerting on forthcoming disease outbreaks, deaths, protests, and riots.

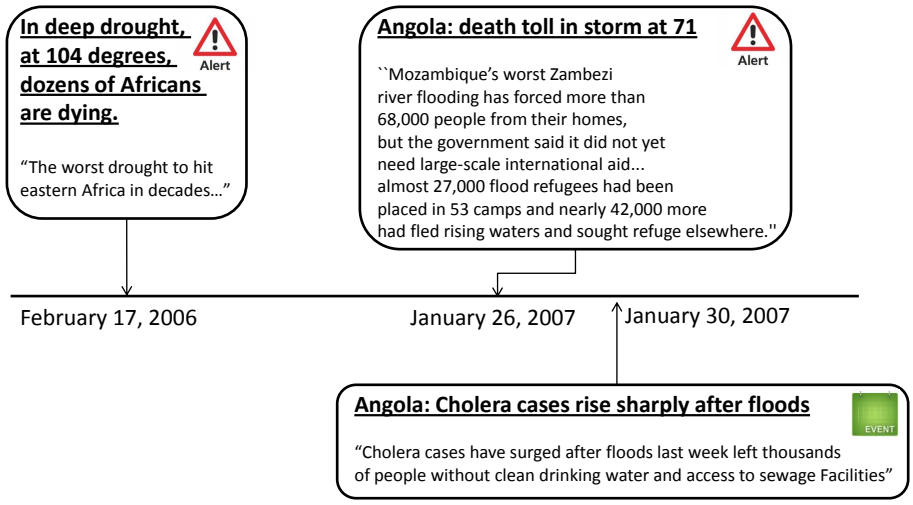

Figure 1.1: The system alerts about high likelihood of Cholera outbreak in Angola after observing a sequence of storms followed by a drought. The rule was learnt based on past Cholera outbreaks in Africa in landlocked countries with high population density.

The thesis is structured as follows: We start our discussion with how Web dynamics can be utilized for future prediction of user actions in the virtual world. We present an algorithmic framework for Web dynamics prediction – discussing both Web user behavior prediction (Section 2) and Web content change prediction (Section 3). We then discuss how this dynamics can be used to reflect the real world by novel utilization of this knowledge into a concept's semantic similarity task (Section 4). We then present an algorithm for utilizing this dynamics for future news evens prediction (Section 5). Furthermore, we discuss how the vast amount of information and different knowledge sources on the Web can be

combined into a system that generates future outcomes given an event (Section 6). We then present a system that combines both world knowledge from the Web and Web dynamics to alert on upcoming events (Section 7). Finally, we present the works related to our research (Section 8), and present the conclusions of this thesis (Section 9).

Chapter 2

Predicting Behavioral Dynamics on the Web

The way that people interact with Web search agents changes over time. However, in information retrieval today, most models that incorporate user behavior signals aggregate evidence over time and use it identically for all types of queries [Agichtein et al., 2006]. We explore the temporal dynamics of Web behavior, investigating how we can model and predict changes in the queries human agents issue, the informational goals corresponding to the queries, and the search results they click on during Web search sessions.

We learn to predict how the search behavior of users changes over time and use these prediction models to enhance retrieval. As an example, for a population of users, the frequency with which a query is issued and the number of times that search results are clicked for that query can change over time. In Figure 2.1, we show the *total clicks*[1] for the query *japan* over time. We can see a dramatic change in behavior associated with this query following the Japanese earthquake on March 11th, 2011. The number of clicks surges for the query for a period of time, and then slowly decays. Likewise, the URLs that people choose to click on following the same query may vary over time, indicating a change in what people consider relevant for that query. Figure 2.2 shows the change in click frequency for several popular URLs following the query *japan* around the time of the earthquake. The frequencies of access of some URLs (e.g., a US government site about Japan) mirror changes in query frequency, while others (e.g., a site for children to learn about Japan) do not change with query and click frequency.

We model and predict these different kinds of search dynamics, focusing in particular on predicting query and click frequency. Although the timing of the initial peak for the query *japan* may be hard to predict, once it is reached, the

[1]We define *total clicks* for a query q as the total number of times any URL returned by the search engine in response to it is clicked.

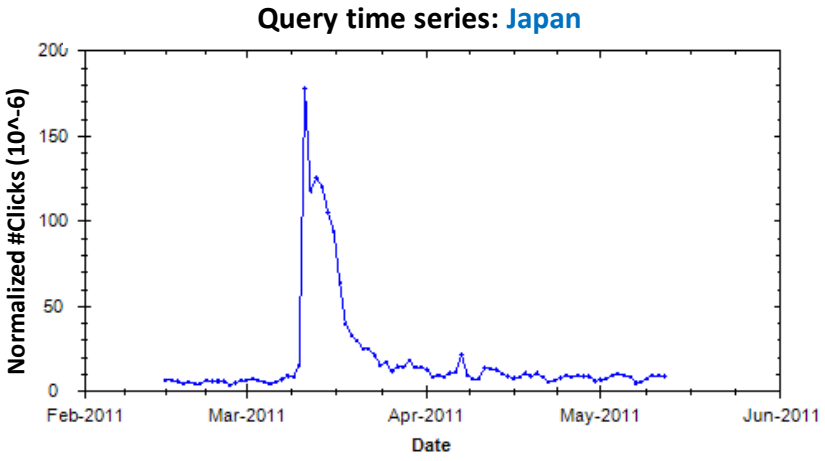

Figure 2.1: A time series (2/14–5/25, 2011) for the query *japan* (normalized by overall #clicks logged on each day based on Bing query logs).

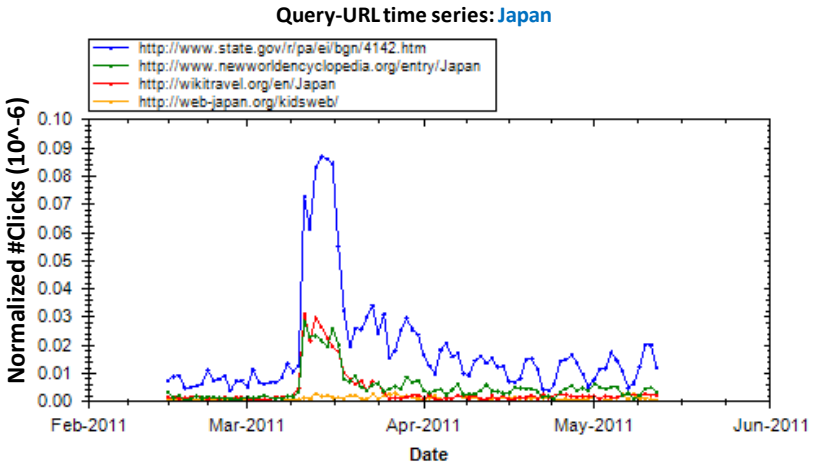

Figure 2.2: Time series (2/14–5/25, 2011) for sample clicked URLs for query *Japan* (normalized by total #clicks on each day based on Bing query logs).

subsequent behavior can be predicted. There are many other cases in which search behaviors are easy to predict. In Figure 2.3, the query *halloween* exhibits periodic trends, and *android* undergoes an increasing trend in popularity, but the query *justin bieber* fluctuates with no obvious trend. Using time-series modeling, we can estimate these different trends and periodicities and predict future values of the frequency of queries and clicks. A major challenge is to select the appropriate modeling. In this chapter, we present a novel algorithm for selecting the right time-series model for each behavior, called the *Dynamics Model Learner* (DML).

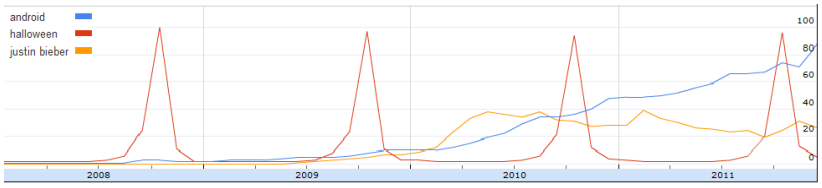

Figure 2.3: Different classes of temporal trends in query frequencies.

Our model considers many factors for this selection, ranging from the long-term shape of the time-series to query-dependent features.

Learned models of what people search for and click on can be used to improve the search experience. In this chapter we consider two search-related applications – result ranking and query suggestion. When user information needs change over time, the ranking of results should also change to accommodate these needs. Consider the example presented in Figure 2.4, which shows the frequency of URL clicks for the query *easter* at different times during the year. A user's information need several weeks before Easter (in mid-March) is likely to identify the exact date of the holiday; indeed, the URL `when-is-easter.html` is clicked more often during mid-March than other Easter-related pages. A few days before Easter, people issuing this query become more interested in planning activities for their kids, and click logs show that sites such as `holidays.kaboose.com` are clicked more often than other pages during this period of time. During Easter itself, people are more interested in the religious meaning and customs of the holiday, questions which are answered by a page such as `answers.com/topic/easter`. To account for changes in what people click on following a query like *easter*, we explore time-aware ranking mechanisms using two ranking scenarios: one which predicts user click behavior for a given day using only this prediction to rank URLs, and the other which uses temporal user behaviors as features (along with content-based features) in a learning-to-rank algorithm. We find that weighting user behavior for each query and URL pair based on temporal dynamics significantly improves the accuracy of the ranked results in both types of ranking scenarios across several types of queries.

We also explore the application of our methods for time-aware query auto-suggestion (QAS), also called auto-completion. Consider the example presented in Figure 2.5. On February 13th 2012 – a day before *Valentine's day* – Google suggested *verizon wireless* as a top candidate for *v* (underlined) and did not suggest any Valentine-related candidates in the QAS ranking (the top plot in Figure 2.5).[2] The query frequency trends in the bottom plot of Figure 2.5 however clearly show that *valentines day* is a more relevant suggestion during this time period.

[2]The query was issued from US with personalization disabled.

We use the same time-series models to predict the query frequency and demonstrate that modeling the temporal profile of queries can improve the ranking of auto-suggestion candidates.

In summary, the key contributions described in this chapter are as follows:

- We highlight the rich opportunities to study the dynamics of Web search behavior and explore several time-series models to represent and predict different aspects of search behavior over time. We discuss how these models can be learned from historical user-behavior data, and develop algorithms tailored to address several aspects of the dynamics of population behavior on the Web, including *trend, periodicity, noise,* and *surprise*.
- We present a novel learning algorithm which we refer to as the *dynamics model learner* (DML), that determines the appropriate model to use for predicting behavior, based on features extracted from large-scale logs of Web search behavior over time. We show that DML performs better than more traditional model selection techniques.
- We perform empirical evaluation of our approaches over large-scale logs of real-world user behavior (obtained from Bing), providing evidence for the value of temporal modeling techniques for capturing the dynamics of user behavior on the Web.
- We present applications of our temporal modeling methods to improve ranking and query suggestions, and provide evidence for the superiority of temporal modeling in both applications.

2.1 Temporal Modeling of Web Search Behavior

We now present a modeling technique based on state-space models that we use to capture the dynamic nature of Web behavior. Of special importance in modeling Web search behavior are global and local trends, periodicities, and surprises. We summarize in this section a general theory behind this model and discuss several modeling techniques we use to capture these important aspects. We conclude the section with a discussion of how the models can be used.

2.1.1 Model Framework: State-Space Models

The state-space model (SSM) is a mathematical formulation frequently used in work on systems control [Durbin and Koopman, 2008] to represent a physical system as a set of input, output, and state variables related by first-order differential

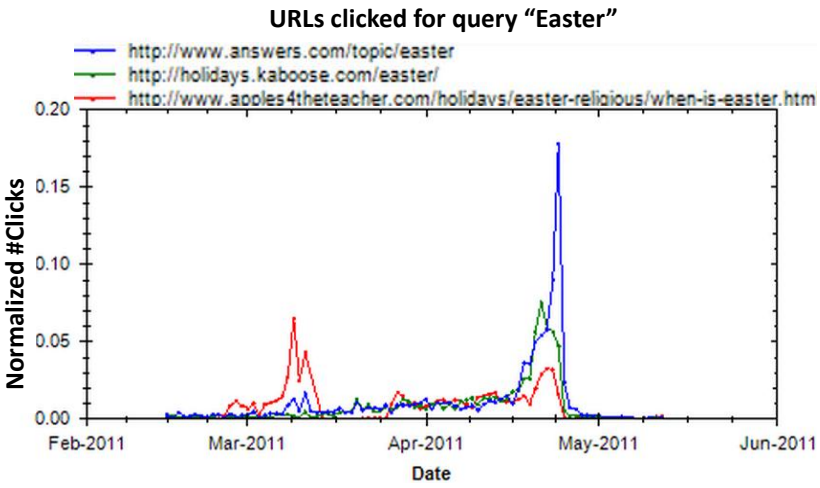

Figure 2.4: Clicked-URLs behavior (2/14–5/25, 2011) for the query *easter* (normalized by the total #clicks on each day and the position of the URL).

equations. It provides an easy and compact approach for analyzing systems with multiple inputs and outputs. The model mimics the optimal control flow of a dynamic system, using some knowledge about its state variables. These models allow for great flexibility in the specification of the parameters and the structure of the problem, based on some knowledge of the problem domain that provides information about the relations (e.g., linear) among the parameters. We use upper case letters for matrices and lower case for scalars. The linear space state model (with additive single-source error) defines a system behavior by the following two equations:

$$Y_t = W(\theta)X_t + \epsilon_t, \tag{2.1}$$

$$X_{(t+1)} = F(\theta)X_t + G(\theta)\epsilon_t, \tag{2.2}$$

where Y_t is the observation at time t, X_t is the state vector, ϵ_t is a noise series, and $W(\theta), F(\theta), G(\theta)$ are matrices of parameters of the model. For a longer prediction range h (also referred to as the *prediction horizon*), it is usually assumed that $Y_{t+h} = Y_t$. For simplicity, in the following sections we present equations for $h = 1$. In our work we assume (as commonly assumed in natural systems representations) that ϵ_t are independent and identically distributed following a Gaussian distribution with variance σ^2 and mean 0. Equations (2) and (3) are called the measurement and transition equations, respectively. To build a specific SSM, a structure for the matrices $W(\theta), F(\theta), G(\theta)$ is selected, and the optimal parameters θ and σ and an initial state X_0 are estimated.

The SSM representation encompasses all linear time-series models used in

Figure 2.5: (Top) The auto-suggestion candidates ranked by Google on the 13th of February 2012, a day before Valentine's Day in the USA. (Bottom) Query frequencies for *valentines day* vs. *verizon wireless* since 2004.

practice. We show in this section how the SSM can be applied to model query and click frequency in Web search. We model the state X_t using a trend component and a seasonal (or periodicity) component, as is often done for modeling time series [Hyndman et al., 2008]. The trend represents the long-term direction of the time series. The seasonal component is a pattern that repeats with a known periodicity, such as every week or every year. We first present models for user search behavior using just historical smoothing, and then present models with trend, periodicity, and their combination. Finally, we explore models that incorporate notions of surprise. Each model is represented by setting the scalars of the matrices $W(\theta), F(\theta), G(\theta)$.

2.1.2 Modeling with Smoothing (SMT)

For the query *justin bieber* in Figure 2.6, simple averaging of past frequencies may give too much weight to the relatively high popularity of the query before April, and hence if used for forecasting, it is likely to overestimate the future

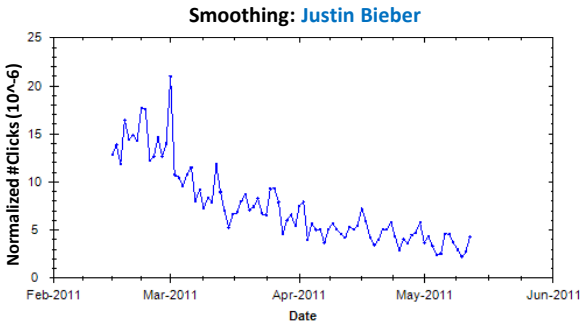

Figure 2.6: Query exhibiting behavior where historical data has little relevance for prediction of future clicks (normalized by overall #clicks on each day based on Bing query logs).

popularity.

Therefore models that simply average historical data, and then extrapolate a constant value as a prediction, may perform poorly for predicting the future. The simple Holt-Winters model [Holt, 2004] is a technique for producing an exponentially decaying average of all past examples, thus giving higher weights to more recent events. The model is represented by the following equation,

$$y_{t+1} = \alpha \cdot x_t + (1 - \alpha) \cdot y_t.$$

Here, for $y = x_0$, solving the recursive equation as follows,

$$y_{t+1} = \alpha x_t + \ldots + \alpha(1-\alpha)^{k-1} x_{t-k} + \ldots + (1-\alpha)^t x_0$$

produces a prediction y_{t+1} that weights historical data based on exponentially decay according to the time distance in the past. The parameter α is estimated from the data (see Section 2.1.7). Converting to SSM notation, let $l_t = y_{t+1}$ where l_t is the level of the time series at time t and $\epsilon_t = x_t - y_t$. The measurement and transition equations can be defined as:

$$\begin{aligned} Y_t &= y_t = l_{t-1}, \\ X_t &= l_t = l_{t-1} + \alpha\epsilon_t. \end{aligned} \tag{2.3}$$

In this case, $W = (1), F = (1), G = (\alpha)$.

2.1.3 Modeling Trends (TRN)

In many cases, such as the query *harold camping* in Figure 2.7, using only one coefficient to discount previous data is not expressive enough to capture the dynamics of the system. The figure shows the query-click behavior for the query *harold camping*, who predicted that the end of the world would commence on May 21st, 2011. The growing interest in this prediction around this date shows a clear local growth trend in the time series. In such cases, where the time series exhibits a local trend, simple smoothing of historical data cannot accurately model the dynamics of interest in this topic. A possible solution to this problem is the addition of a trend component b_t to the simple Holt-Winters model:

$$\begin{aligned} y_t &= l_{t-1} + d \cdot b_{t-1} + \epsilon_t, \\ l_t &= l_{t-1} + b_{t-1} + \alpha\epsilon_t, \\ b_t &= b_{t-1} + \beta^*(l_t - l_{t-1} - b_{t-1}), \end{aligned} \tag{2.4}$$

where l_t is called the level of the time series at time t, d is the damping factor, and b_t is called the estimation of the growth of the series at time t, which also can be written as

$$b_t = b_{t-1} + \alpha\beta^*\epsilon_t = b_{t-1} + \beta\epsilon_t.$$

Intuitively, the level l_t is usually the magnitude of the time series within a certain time interval, and b_t is an estimate of the growth based on the difference between successive levels.

Let $X_t = (l_t, b_t)'$, then:

$$\begin{aligned} Y_t &= \begin{pmatrix} 1 & d \end{pmatrix} X_{t-1} + \epsilon_t, \\ X_t &= \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} X_{t-1} + \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \epsilon_t. \end{aligned}$$

In this case, $W = \begin{pmatrix} 1 & d \end{pmatrix}, F = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, G = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$.

The parameters α, β are estimated from the data (see Section 2.1.7).

2.1.4 Modeling Periodicity (PRD)

Figure 2.8 shows query-click behavior for the query *consumer report* that exhibits weekly periodicity. Similarly, the query *halloween* in Figure 2.3 exhibits annual periodicity. For such queries, predictions based only on local trends or smoothing

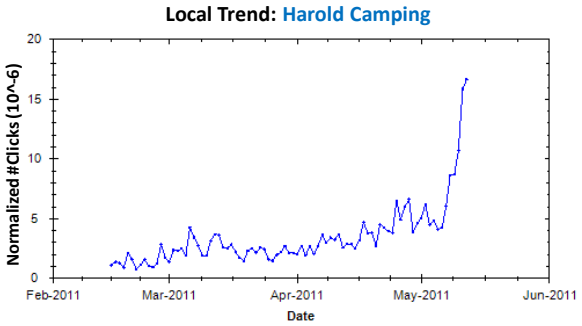

Figure 2.7: Query exhibiting behavior with local trend (normalized by overall #clicks on each day based on Bing query logs).

of the data will perform badly during the peaks. A possible solution is the addition of a periodic or seasonal component s_t to the simple Holt-Winters model,

$$\begin{aligned} y_t &= l_{t-1} + s_{t-m} + \epsilon_t, \\ l_t &= l_{t-1} + \alpha\epsilon_t, \\ s_t &= \gamma^* \cdot (y_t - l_{t-1}) + (1-\gamma^*)s_{t-m}, \end{aligned}$$

where m is the periodicity parameter that is estimated based on the data along with other parameters (see Section 2.7.1 for details). In SSM notation, the equation system can be written as:

$$\begin{aligned} y_t &= l_{t-1} + s_{t-m} + \epsilon_t, \\ l_t &= l_{t-1} + \alpha\epsilon_t, \\ s_{t-i} &= s_{t-i+1}, \\ &\dots, \\ s_t &= s_{t-m} + \gamma\epsilon_t, \end{aligned} \tag{2.5}$$

and for $X_t = (l_t, s_1, \dots, s_m)$, we can represent the parameters F, G, W in a form of the matrices similar to the Trend Holt-Winters model formulation. The parameters α, γ are estimated from the data (see Section 2.1.7).

2.1.5 Modeling Trends and Periodicity (TRN+PRD)

In the previous models, trend and periodicity were considered separately. However, for many queries, such as the query *vampire diaries* shown in Figure 2.9, the trend and periodicity components are mixed. In this case, the periodicity is

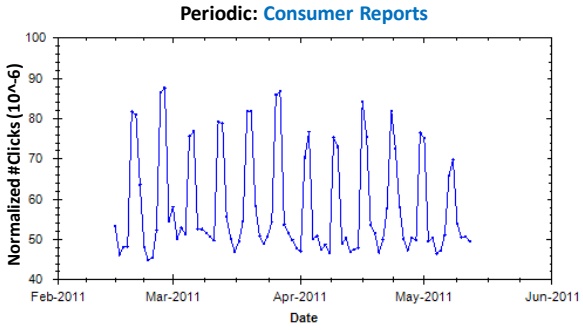

Figure 2.8: Query exhibiting a periodic behavior (normalized by overall #clicks on each day based on Bing query logs).

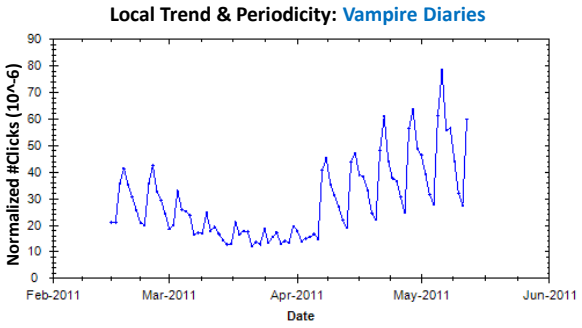

Figure 2.9: Query exhibiting periodic behavior with local trend (normalized by overall #clicks on each day).

unchanged but the frequency increases. The addition of trend and periodicity parameters produces the following model:

$$\begin{aligned} y_t &= l_{t-1} + d \cdot b_{t-1} + s_{t-m} + \epsilon_t, \\ l_t &= l_{t-1} + b_{t-1} + \alpha\epsilon_t, \\ b_t &= b_{t-1} + \beta\epsilon_t, \\ s_t &= s_{t-m} + \gamma\epsilon_t, \\ &\dots, \\ s_{t-i} &= s_{t-i+1}. \end{aligned} \tag{2.6}$$

The parameters α, β, γ are estimated from the data (see Section 2.1.7).

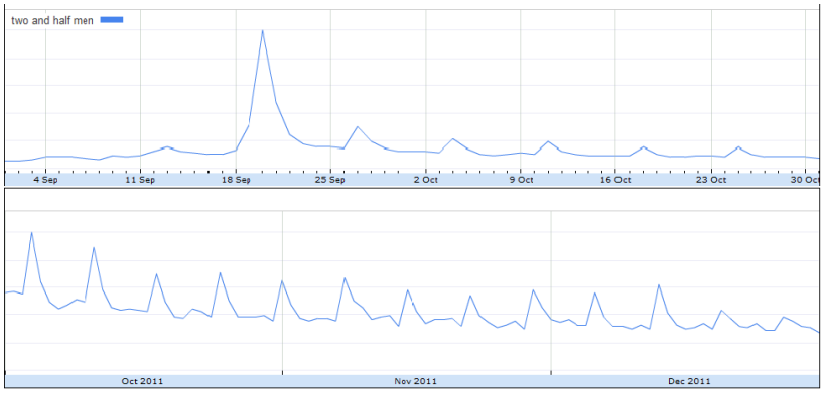

Figure 2.10: (Top) The frequency trends of query *two and half men* between September and October 2011. In addition to weekly cycles, there is a *surprise* spike on the September 20, 2011 with the beginning of the new season of the show. **(Bottom)** The frequency trends of query *two and half men* for the last three months of 2011. In addition to the weekly cycles, the decline *trend* in query frequency can be observed. Snapshots taken from Google insight for search.

2.1.6 Modeling Surprises (SRP)

The Holt-Winters models assume the conversion of a set of parameters to a single forecast variable Y_t. However, in many real-world time series, the model itself changes over time and is affected by external disturbances caused by non-modeled processes in the open world. For example, in Figure 2.11, we see a periodic query *fda* with a disturbance on March 4th, due to an external event concerning the announcement that some prescription cold products are unsafe. Another example can be seen in Figure 2.10, where we see a periodic query *two and half men* with a *surprising* spike on September 20th, 2011, due to the beginning of the new season of the show. Inclusion of such outliers might have a strong effect on the forecast and parameter estimation of a model. We want to identify characteristics in the temporal patterns which are not adequately explained by the fitted model, and try to model them for a better estimation of Y_t. If these characteristics take the form of sudden or unexpected movements in the series, we can model them by the addition of *disturbances* that capture the occurrences of *surprises* from the perspective of the model.

A disturbance or a surprise is an event which takes place at a particular point in the series, defined by its location and magnitude. In a time series, the effect of a disturbance is not limited to the point at which it occurs, but also propagates and creates subsequent effects that manifest themselves in subsequent observations.

We augment the standard Holt-Winters model with the addition of two *surprise parameters*: m_t, which is a surprise measurement at time t, and k_t, which

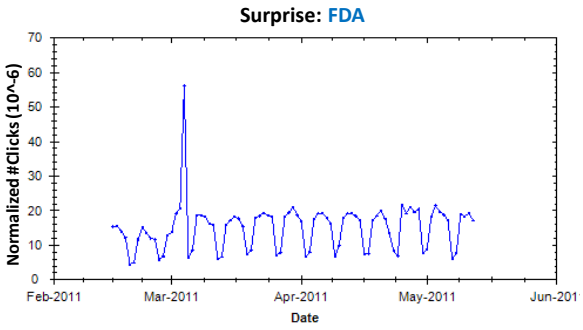

Figure 2.11: Query exhibiting behavior with surprises.

is the surprise trend at time t:

$$\begin{aligned} y_t &= l_{t-1} + d \cdot b_{t-1} + s_{t-m} + m_t + \epsilon_t, \\ l_t &= l_{t-1} + d \cdot b_{t-1} + \alpha\epsilon_t, \\ b_t &= b_{t-1} + k_t + \beta\epsilon_t, \\ s_t &= s_{t-m} + \gamma\epsilon_t, \\ &\dots, \\ s_{t-i} &= s_{t-i+1}. \end{aligned} \tag{2.7}$$

We discuss in Section 2.7.2 methods for identifying the surprises k_t in a time series, and in Section 2.1.7 discuss how the parameters α, β, γ are estimated from the data.

2.1.7 Using the Models to Forecast

Once the model structure is specified, the distribution of the future values of the time series can be evaluated, given past history. That is, we learn an SSM for time series $Y_1, \dots, Y_n$ jointly with the internal states $X_0, \dots, X_n$, and residuals $\epsilon_0, \dots, \epsilon_n$. During prediction, future states X_{n+1} are generated using the state transition equation (Eq. 2.2) and, based on these, a distribution of the future values Y_{n+1} is generated using the measurement equation (Eq. 2.1). The final prediction can be generated by using the expected value of this distribution.

The SSM family provides a predefined structure for forecasting, where the specific parameters of the model need to be evaluated from the data. We apply gradient descent [Snyman, 2005] to optimize the model parameters based on training data. We assume here the following loss function, which is a common

criterion for measuring forecast error:

$$F = \sum_{t=1}^{T} \epsilon_t^2. \tag{2.8}$$

The initial values of the models, X_0, are set heuristically, and refined along with the other parameters. The seasonal component m is estimated from the data by *autocorrelation* analysis (see Section 2.7.1 for details).

2.2 Learning the Right Temporal Model

As described in Section 2.1, different temporal models can be employed to represent user behaviors over time. We first present a known method for selecting which model is the best based on the information criterion of the time series (Section 2.2.1), and then provide a novel method for inferring the model based on extended and domain-specific characteristics of Web behaviors (Section 2.2.2).

2.2.1 Bayesian Information Criterion

The models described thus far are based on curve fitting and we have shown via examples that there is no single model that adequately models trends, periodicities, and surprise. An alternative approach would be to *fit* several models, and train a classifier that selects the best predictive model.

Each model adds more parameters. When fitting the models, there is a high likelihood of more complex models fitting the data better. This might result in overfitting, especially when applying gradient descent. The Bayesian information criterion (BIC) [Schwarz, 1978] resolves this problem by adding a penalty for the number of parameters in the model. That is, it presents a tradeoff between the accuracy and the complexity of the model. It is closely related to the Akaike information criterion (AIC), but the penalty term is larger in BIC than in AIC. The BIC criteria being optimized is defined as:

$$\text{BIC} = -2 \cdot \log(L) + q \cdot \log(n), \tag{2.9}$$

where q is the number of parameters, n is the length of the time series, and L is the maximized likelihood function. For a Gaussian likelihood [Priestley, 1981], this can be expressed as

$$\text{BIC} = n \cdot \log(\sigma_e^2) + q \cdot \log(n), \tag{2.10}$$

where σ_e is the variance of the residual in the testing period (estimated from the test data). The model with the lowest BIC is selected to represent the time series and to issue point forecasts.

2.2.2 Dynamics Model Learner (DML)

The BIC criterion introduced in Section 2.2.1 takes only the model behavior on the time-series values into account. However, in our domain we have access to richer knowledge about search behavior. For example, we know what query was issued, the number of clicks on a given URL for that query, and so on. We shall now discuss how to use domain knowledge to further improve behavior prediction. We focus on learning which of the trained temporal SSM models is most appropriate for each object of interest, and then estimating parameters for the chosen model.

Going Beyond Time-series for Temporal Modeling

We start by formally motivating the algorithm and defining the learning problem. Let T be the discrete representation of time and O be the set of objects, and let $f_i : O \times T \to F(f_i)$ be a set of features. For example, O can be a set of URLs, $f_1(o, t)$ can be the number of times URL o was clicked at time t, and $f_2(o, t)$ can be the dwell time on the URL at time t.

In time-series analysis, a *single object* o is modeled over some period $t_1, \ldots, t_n$, and a model $C : T \to \mathbb{R}$ can be trained based on those historical examples. In order to forecast future trends (classes) at time t_{i+1}, the model is provided with an example of the object seen at training time t_i (for example, predicting how many times the URL o is clicked on tomorrow based on its past history).

In regression learning, *multiple objects* $O' \subset O$ are modeled simultaneously by a single model $C : O \to \mathbb{R}$. During prediction, the regression model may be given an example of an object o_i, that has not been seen before, to produce the prediction of its numeric value. Notice that no notation of time is considered.

The time-series approach is capable of making specific predictions about a specific object at a certain time, but does not consider information about other objects in the system and therefore cannot generalize based on their joint behaviors. Regression learning, on the other hand, generalizes over multiple objects, but does not use the specific information about the object it receives during prediction, and therefore does not usually use the information about how this specific object behaves over time.

We combine the two approaches into a unified methodology that first considers generalized information about other objects to choose a model of prediction and then uses the specific knowledge of the predicted object to learn the specific parameters of the model for the object. Formally, given a set of objects $O' \subset O$ over some period of time $t_0, \ldots, t_n$, we produce a model C that receives an object $o \in O$ (not necessarily $o \in O'$) over some period $t_0, \ldots, t_n$, and produces the prediction of its numeric value at time t_{n+1}.

DML Learning

In this section, we present a new learning algorithm, *dynamics model learner (DML)*, for learning from multiple objects with historical data. Let $O' \subset O$ be a set of objects given as examples for training the learning model. Let $t_1, \ldots, t_{n+\sigma}$ be the times dedicated for training the model. For example, O' can be a set of queries for which we have user behavior information for the period of time $t_1, \ldots, t_{n+\sigma}$. We divide the objects into two sets — the learning set, $t_1, \ldots, t_n$, and the validation set $t_{n+1}, \ldots, t_{n+\sigma}$. For every temporal model described in Section 2.1, we train a model on the learning period, and check the mean squared error (MSE) over the validation period. Formally, let $o(t)$ be the behavior of object o at time t (e.g., how many times the query o was searched), then

$$MSE(o, t_1, \ldots, t_{n+\sigma}, m) = \frac{\sum_{t=t_{n+1}}^{t_{n+\sigma}} (o(t) - \widehat{o}(t))^2}{\sigma},$$

where $\widehat{o}(t)$ is the model m estimation at time t.

Let i be the index of the model with the lowest MSE on the test period for the object o. We construct a set of examples $E = \{\langle f_1(o,t), \ldots, f_n(o,t)\rangle, i | o \in O'\}$ — a vector representing an object and labeled with the index of the best-performing model for that object. We then use a learner (in our experiments, a decision-tree learner) along with the examples E to produce a classifier C. During prediction, C is applied on the object we wish to predict, o_{target}. The output $m = C(o_{target})$ represents the index of the most appropriate model for the object o_{target}. We train the model m using the behavior of o_{target} during $t_1, \ldots, t_{n+\sigma}$. A detailed algorithm is illustrated in Figure 2.12. An example of a learned decision tree is shown in Figure 2.13. In this figure, we see that the periodic model should be applied only on objects (queries) considered periodic (as defined in Section 2.7.1), along with other characteristics. If the query is not periodic, the trend or the smoothing model should be applied, depending on the query shape (see Section 2.2.2). Thus by using the DML model we learn to apply the best equations for modeling the time series.

DML Features

DML uses a set of features f_i about each object o. In this section, we discuss the specific features we use to train the DML model used in our experiments. We devise a total of 973 features (description of the features is available online [3]), and group them into three groups: aggregate features of the time series o, shape features of the time series o, and other domain-specific features such as the query class.

[3] `http://www.technion.ac.il/~kirar/Datasets.html`

Procedure DYNAMICS MODEL LEARNER($O', Learner$)
Train:
Train temporal models $m_1, \ldots, m_M$ for every $o \in O'$
$E = \{ \langle f_1(o,t), \ldots, f_n(o,t) \rangle,$
$\arg\min_i MSE(o, t_1, \ldots, t_{n+m}, m_i) | o \in O' \}$
Call *Learner* with E, and receive the
hypothesis classifier C
Prediction: Given unlabeled instance o_{target} at time t
chosenModel $\leftarrow$ Evaluate C on
$\langle f_1(o_{target}, t), \ldots, f_n(o_{target}, t_i) \rangle$
learntModel $\leftarrow$ Learn chosenModel parameters
using $o_{target}(t_1), \ldots, o_{target}(t_{n+m})$
Return Evaluate learntModel on $o_{target}(t)$

Figure 2.12: The procedure estimates the model type to learn based on past examples and their features. The model parameters are estimated and a prediction for the new object is performed.

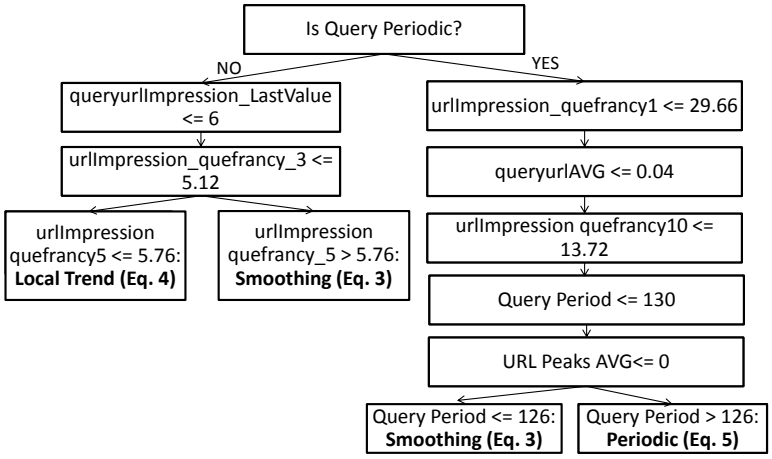

Figure 2.13: A part of the learned dynamic model.

Aggregate Features Features include the *average*, *minimum*, *maximum*, and *period* of the time series, e.g., the average of the query volume. Other features consider the dynamics of the series, e.g., the time series *periodicity* and *number of surprises*. We also consider the *size of the spikes* during the surprises (the magnitude of the disturbance).

Shape Features Shape features represent the shape of the time series. Our goal is to produce a representation that is not sensitive to shifts in time or the magnitude of the differences. Formally, for a time series $y[n] = x[n]$, we are looking for a representation that will be equivalent to series of the form $y[n] = x[n-h]$ and $y[n] = A \cdot x[n]$, for any shift h and any scalar A. Homomorphic signal processing is a solution that satisfies these conditions. Intuitively, application of the Finite Fourier transform (FFT) operator on a time series transforms it to what is called the *spectral domain*, i.e., produces a vector x of size k, where every x_k represents the k-th *frequency* of the time series.

$$x_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi k \frac{n}{N}}$$

This procedure eliminate shifts, as both $y[n] = x[n-h]$ and $y[n] = x[n]$ have similar frequencies. Application of a log function on the resulting frequency vector and an additional FFT operator transforms it to what is called the *cepstral domain* [Childers et al., 1977], and the values of x in this product are called *quefrencies*. The byproduct of these procedures is that series of the form $y[n] = A \cdot x[n-h]$ and $y[n] = x[n]$ are transformed to the same vector X in the ceptral domain. In speech recognition [Bogert et al., 1967], the values of $x_1, \ldots, x_{13}$ in the cepstral domain are considered a good representation of the series. We consider these features to represent the shape of the series.

Domain-Specific Features We also consider a set of temporal and static features that are domain specific. For the query-click time series we consider the total *number of clicked URLs*, and the *query-click entropy* at time t, which is defined as:

$$QueryClickEntropy(q,t) = -\sum_{i=1}^{n} p(click_t(u_i,q)) \log p(click_t(u_i,q)), \quad (2.11)$$

where $u_1, \ldots, u_n$ are the clicked URLs at time t, and $p(click_t(u_i,q))$ is the percentage of clicks on URL u_i for query q at time t, among all clicks on query q. If all users click on the same URL for query q, then $QueryClickEntropy(q,t) = 0$. We consider an aggregated version of query-click entropy as the average of the last $k = 7$ days before the prediction. For both the query and URL click time series, we consider the *topical distribution* of the URL or query. We used a standard topical classifier which classifies queries into topics based on categories from the Open Directory Project (ODP) [Bennett et al., 2010]. We chose to classify into approximately 40 overlapping topics, such as travel, health, and so on.

2.3 Overview of Applications of Time-Series Modeling for Search

Models that take advantage of historical user behavior data often aggregate the data uniformly, regardless of when the behavior is observed. These techniques fail to weight older data differently than newer data and may lead to a stale user experience. We now explore how the time-series modeling techniques presented in previous sections can be applied to resolve this problem. We first summarize how we predict future clicks and then describe two important search-related applications: ranking and query auto-suggestion. In Sections 2.4, 2.5, and 2.6 we provide the experimental results of those applications.

2.3.1 Experiments for Predicting Future Clicks

We start with an application of the methods we described for click prediction, and investigate three types of forecast: query click prediction, query-dependent URL click prediction, and query-independent URL click predictions. We provide a deep analyze the performance of different temporal models on the different types of predictions and their behavior for different prediction horizons (Section 2.4).

2.3.2 Experiments for Ranking Search Results

Web search engines often rely on usage data such as clicks and anchor text for ranking documents. Such techniques tend to favor old documents that have accumulated more behavioral data over time over fresher and potentially more relevant documents. As an example (shown in Figure 2.14), one of the highest ranked results for the query *WSDM conference* at the time of writing this chapter is the WSDM 2010 (on Bing) or WSDM 2011 page (on Google). However, the current intention behind this query is more likely about finding information on the forthcoming WSDM 2013 conference. The older pages have more historical data than the more recent one, which only has sparse click data because it did not even exist prior to 2012. Accurate predictions of future query and click frequency can be used directly to re-ranking the search results. We refer to those predictions as *temporal features*. Alternatively, those features can be used along with other features to train a ranking function, e.g., BM25 features [Robertson et al., 2004]. We refer to those features as *base features*.

We explore both of these scenarios in this chapter. In our first set of ranking experiments, we leverage temporal features as independent evidence for ranking. To understand how well our predictions can be used as independent evidence

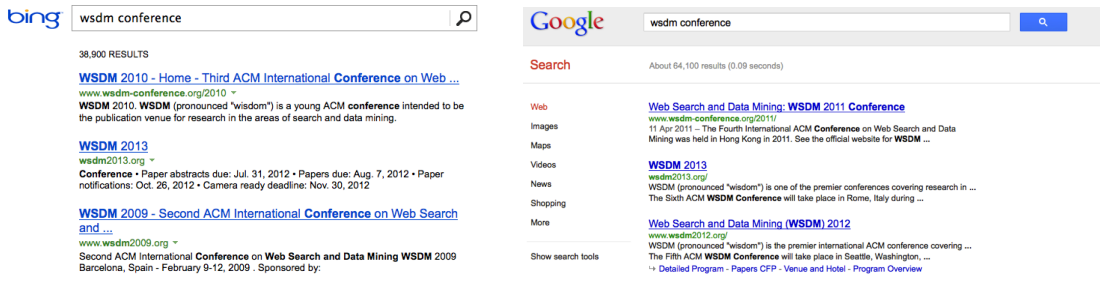

Figure 2.14: Search results for the query *WSDM conference* on August 5th 2012. The 2010 (left image) and 2011 (right image) conference websites are ranked higher than the 2013 conference website.

for ranking, we only use our prediction of future user behavior. For each query-URL pair we compute the predicted normalized number of clicks and use that prediction to rank the URLs. In the second set of ranking experiments, we use temporal features as input to a learning-to-rank algorithm. We employ a supervised learning technique to learn the ranking function that best predicts the ground-truth ranking of URLs for a given query, using a variety of query, query-URL and URL features, combining both base features and temporal features. We show that rankers that use temporal modeling consistently outperform rankers only considering static user behavior.

We provide an analysis of our temporal models applied for ranking search results in Section 2.5.

2.3.3 Experiments for Ranking Query-Suggestion Candidates

Query auto-suggestion (QAS) is a feature incorporated in most search engines, where the goal is to save user time by predicting user's intent and suggesting other possible queries matching the first few keystrokes typed. In a typical QAS scenario, the user is presented with a list of query suggestions that match the prefix text entered in the search box so far (e.g., "di" in Figure 2.15). For each prefix $\mathcal{P}$, the list of candidates $\mathcal{C}(\mathcal{P})$ consists of all previous queries that start with $\mathcal{P}$[4]. The list of candidates is dynamically updated at run-time with each new character typed by the user.

The common practice for ranking QAS candidates is to use past query frequencies [Bar-Yossef and Kraus, 2011; Chaudhuri and Kaushik, 2009] aggregation as a proxy for the *expected* popularity in the future. Bar-Yossef et al. referred

[4]Without loss of generality, we ignore more advanced fuzzy matching techniques [Chaudhuri and Kaushik, 2009; Ji et al., 2009] in our work.

to this general form of QAS ranking as *MostPopularCompletion* (MPC). Those approaches assume that user intent is static and does not change over time. However, the query popularity is dynamics and affected by different temporal trends. Consider the example in Figure 2.15 where a user has typed *di* in Google query box on Sunday, November 6th, 2011. At first glance, knowing that *dictionary* is generally a more frequent query than *disney*, it might be difficult to notice how the ranking might be improved. However, looking at the daily trends for these queries in Figure 2.16 reveals that *disney* is more popular on weekends. Hence, given that the first snapshot was taken on a Sunday, swapping *disney* and *dictionary* could lead to a better ranking at the time of this query. In summary, the past is not always a good proxy for future particularly for trendy and seasonal queries. Today's frequency for query *dictionary* is not necessarily the best estimate for its frequency tomorrow.

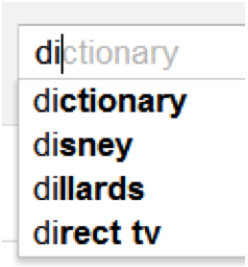

Figure 2.15: Google auto-suggestion candidates after typing *di* on Sunday, February 13th, 2012. The user was typing from a US IP address, with personalization turned off.

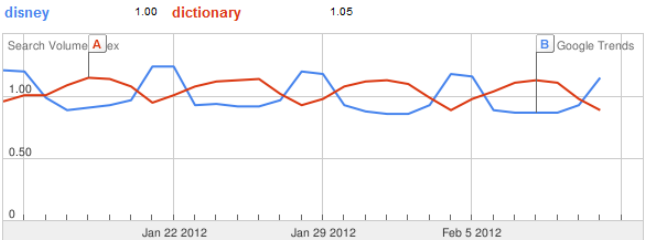

Figure 2.16: Daily frequencies for queries *dictionary* (red) and *disney* (blue) during January 2012 according to Google Trends (the snapshot was taken on Monday, 13-Feb-2012). Among the two queries, *disney* is more popular on weekends, while *dictionary* is issued more commonly by users on weekdays.

We propose to use our temporal modeling techniques to enhance this ranking.

In our time-sensitive QAS ranking model, the score of each candidate at time t is determined according to its predicted value calculated using time-series models that capture temporal trends and periodicity. Our time-sensitive QAS ranking model can be formalized as a variation of MPC model (Eq. 8.1),

$$TS(\mathcal{P}, t) = \arg\max_{q \in \mathcal{C}(\mathcal{P})} w(q|t), \quad w(q|t) = \frac{\hat{y}_t(q)}{\sum_{i \in \mathcal{Q}} \hat{y}_t(i)}, \tag{2.12}$$

where $\mathcal{P}$ is the entered prefix, and $\mathcal{C}(\mathcal{P})$ represents its list of QAS candidates, and $\hat{y}_t(q)$ denotes the estimated frequency of query q at time t.

We provide analysis of numerous temporal models applied for ranking query auto-complete candidates in Section 2.6.

2.4 Experiments for Predicting Future Clicks

We first describe the setup for the prediction experiments and perform several prediction experiments, namely predicting query, query-dependent URL click, and query-independent URL click frequencies. In every experiment, five SSM models (Section 2.1), two selection models (BIC (Section 2.2.1), DML (Section 2.2.2)), and four baseline models (Section 2.4.3) are evaluated. The prediction results are shown for a variant of the MSE to avoid numerical errors: $MSE(predicted) = E[|predicted - real|^{0.5}]$. The above error is averaged over 12 consecutive prediction days (April 14th to April 25th, 2011) to avoid over-fitting of a specific day. To compare the results of the different algorithms, we perform a t-test on the results.

2.4.1 Data

The dataset consists of query and URL activity obtained from Bing for the US market during the period December 15th, 2010 to April 25th, 2011. The data contains information about a query's daily click counts, a URL's daily click counts, and, for each query, all of the URLs presented along with their corresponding daily rank positions and daily click counts. We filter the data and consider only queries and URLs that have more than 5 clicks per day. For each query, we consider the top 4 URLs by click frequency. We normalize every activity time series by the total number of activities on that day, to mitigate known differences in daily query volume. For query and URL pairs, we also normalize by the number of clicks on the URL at the position at which it was displayed in the displayed ranked results for the query, producing a value which is not dependent on the position.

We describe the dataset in full detail in Section 2.4.2. Table 2.1 gives a summary of the data.

Data	#Queries	#URLs	Description
General	10000	35862	General queries randomly sampled (unique on a given day)
Dynamic	504	1512	Queries labeled as requiring fresh results
Temporal Reformulations	330	1320	Queries reformulated with temporal word added
Alternating	1836	7344	Randomly sampled queries whose URLs change rankings

Table 2.1: Summary of query types.

2.4.2 Queries

We investigate the effectiveness of different models on various types of queries. For this purpose, we use multiple sampling strategies to collect queries with different degrees of time-sensitivity.

General Queries We first present prediction over a general sample of queries issued to Bing. For this purpose, we use 10,000 queries randomly sampled without repetition on a given day, and 35,862 clicked URLs.

Time series modeling is especially interesting in cases where the behavior of the population of users changes over time. To study such changes, we identified three types of queries that we believe would benefit from temporal modeling.

Dynamic Queries Queries, such as *japan* described earlier, arise because of external events and require fresh content to satisfy users' needs. Trained judges labelled queries that required fresh results at specific points in time. We say that a query q is *Dynamic* if a human labelled it at time t as a query requiring fresh results. In the experiments, a total of 504 dynamic queries and 1512 labeled URLs were used.

Temporal-Reformulation Queries Another way of identifying queries associated with time-sensitive informational goals is to examine queries that explicitly refer to a period of time. We focused on queries that were reformulated to include an explicit temporal referent (e.g., an initial query *world cup* might be later reformulated as *world cup 2011* or *world cup latest results*). We say that a query q was reformulated to a query $q' = q + w$ if a user issuing a query q at time t issued the query q with an additional word w at time $t + 1$ in the same session. We say that a query q was *temporally reformulated* if w is of type year, day of week, or if w is one of the following words: current, latest, today, this week. Reformulation data was obtained for queries issued in the years 2007-2010. A total of 330 queries and 1320 URLs were sampled.

Alternating Queries Queries that exhibit interesting temporal behavior often show changes in the URLs that are presented and clicked on over time. We focus on a subset of queries, whose most frequently clicked URLs alternate over time. We say that a query q is alternating if $\exists t_1, t_2 \in T, i, j \in N : Click(u_i, t_1|q) > Click(u_j, t_1|q), Click(u_i, t_2|q) < Click(u_j, t_2|q)$, where $u_1, \ldots, u_n$ are the matching URLs to the query q, and $Click(u, t|q)$ is the number of clicks on u at time t for the query q. A total of 1836 queries and 7344 URLs were sampled.

2.4.3 Models

Baseline Methods

The most commonly used baseline for representing user search behavior is the averaging of activity over some period of time. Generally speaking, we consider baselines that perform some kind of uniform or non-uniform mapping of the data, and output the average of the mapping as the prediction. We call these different mappings *Temporal Weighting Functions*. The modeling in this case is of the form

$$y_t = \sum_{i=0}^{t-1} \frac{w(i, y_i) y_i}{\sum_{j=0}^{t-1} w(j, y_j)},$$

where $w(i, y_i)$ is a temporal weighting function. In this work, we consider the following baseline functions:

1. **AVG**: A simple average of the time series: $w(i, y_i) = 1$.

2. **LIN**: The linear weighting function: $w(i, y_i) = i$.

3. **POW**: The power weighting function: $w(i, y_i) = i^p$ (in the experiments we set $p = 2$).

4. **YES**: The yesterday weighting function that only considers the last value of the time series ("what happened yesterday is what will happen tomorrow"). The YES weighting function is as follows;

$$w(i, y_i) = \begin{cases} 1 & i = t - 1 \\ 0 & \text{otherwise.} \end{cases} \tag{2.13}$$

Temporal Methods

The temporal models that we use in these experiments are the five SSM models: the Smoothing model (SMT), the Trend model (TRN), the Periodic model (PRD), the Trend and Periodic model (TRN+PRD), and the Surprise model (SRP). In addition to the temporal models, we also evaluate two temporal model selection methods: BIC and the DML method.

2.4.4 Prediction Task

For each of the baseline and temporal methods we learn models from the data from December 15, 2010 to April 13, 2011, and predict click behavior for April 14th to April 25th, 2011. In this section, we present three prediction tasks: Query click prediction, URL click prediction task and Query-URL prediction task. The MSE error is averaged over 12 consecutive prediction days (April 14th to April 25th, 2011) to avoid over-fitting of a specific day. To compare the results of the different algorithms, we perform a t-test on the results comparing against the best performing model. Statistically significant results ($p < 0.05$) of the best models in each row are show in bold.

2.4.5 Predicting Query Clicks

We now summarize the prediction results for query and URL click frequencies. The query click prediction results are shown in Table 2.2. The table reports prediction errors, where smaller values indicate better prediction accuracy. The best performing model for each query type is shown in bold. For all of the query types, we observe that the DML method performs the best. DML always outperforms the well-known BIC method for model selection as well as all of the SSM models and baselines. This shows that *learning* which model to apply based on the different query features is useful for query-click prediction.

Comparing the temporal SSM models versus the baselines, we observe that for the General class of queries the model that smooths surprises performs the best. This result indicates that many queries are noisy and strongly influenced by external events that tend to interfere with model fitting. For the Dynamic class, temporal models that only take into account the trend or learn to decay historical data correctly perform the best. This result aligns with the intuition that queries that represent new events happening during the time of the prediction, thus requiring new results, need the most relevant new information. Thus, too old data interferes with prediction. Most of those queries exhibited a trend in the end of the period. Few disturbances (surprises) were detected, therefore the Surprise model was not useful for this set. For Temporal-reformulations, the best

	Baselines (Section 2.4.3)				Temporal SSM Models (Section 2.1.1)					Model Selection (Section 2.2)	
Query Type	AVG	LIN	POW	YES	SMT	TRN	PRD	TRN+ PRD	SPR	DML	BIC
General	0.17	0.40	0.40	0.18	0.19	0.18	0.17	0.16	0.15	**0.14**	0.19
Dynamic	0.54	0.68	0.68	0.56	0.49	0.49	0.58	0.59	0.60	**0.44**	0.48
Temp Reform	0.56	0.67	0.65	0.78	0.76	0.77	0.59	0.62	0.63	**0.52**	0.73
Alternating	0.30	0.34	0.33	0.33	0.30	0.30	0.44	0.45	0.45	**0.29**	0.33

Table 2.2: The prediction error of different models for predicting the *total clicks* for a query (lower numbers indicate higher performance). The best performing statistical significant results are shown in bold.

performing temporal models are those that take into account the periodicity of the query. For all query classes, the baseline that performs simple averaging of historical data yields the best results.

Overall, predictions are the most accurate for the General class of queries, indicating that many queries are predictable. The Temporal-reformulation class of queries provides the most difficult challenge for predictive models, showing prediction errors of 0.52 (best prediction model error). Most errors result from queries that are seasonal with a periodicity of more than a year, e.g., holiday related queries (*sears black Friday*) or other annual informational goals as exemplified by the query *2007 irs tax*. As we only had data for a period of 5 months, such longer-term periodicities were not detected.

We investigated how the models behaved over longer prediction periods h (the prediction horizon). For each such horizon we trained a new model. We present the error of the different models on the General set of queries as a function of the prediction horizon in Figure 2.17. We observe an interesting phenomenon — almost all methods have a sharp decrease in performance after a prediction horizon of approximately $h = 7$. This shows the complexity of the problem for predicting queries further than a week, perhaps due to the importance of weekly periodicities in queries. However, we did not observe the same phenomenon in the DML method, providing evidence that learning the correct model to apply remains stable throughout all prediction horizons. We observe that the relative performance of the methods remains the same, with one exception – the DML method, that did not experience any change in bigger prediction horizons.

2.4.6 Predicting Query-Independent URL Clicks

The predictions for URL clicks aggregated over all queries are given in Table 2.3. We see again that the DML procedure has the best prediction accuracy

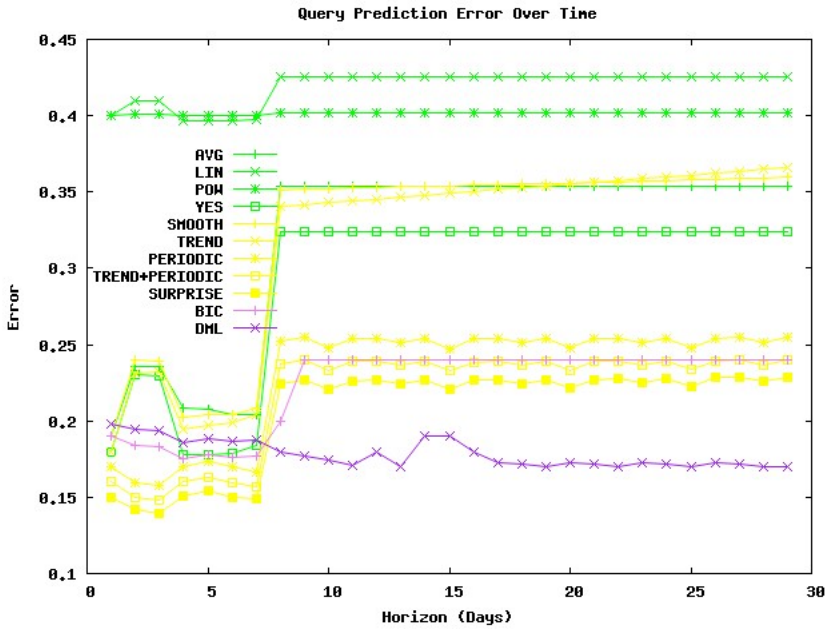

Figure 2.17: Query total click prediction error over time on the General queries set (lower values indicate higher performance). Static models are shown in green, Temporal models are shown in yellow, and Temporal model selection methods are shown in purple.

for Dynamic and Temporal-Reformulation queries. For these classes of queries, models that learn to weight historical behavior and trend models achieve the best performance. For Alternating queries, we observe that predicting clicked URLs benefits from seasonality modeling. For Temporal Reformulation queries, the baseline and DML models show the best prediction performance. Interestingly, the Periodic, Trend+Periodic and Surprise models perform poorly. Similar to what we observed for query prediction, the majority of errors stem from URLs that have periodic content with an annual periodicity lag, which is bigger than the 5 months of data obtained for training the models. The models incorrectly estimate the lag, which is outside of the scope of the data, and therefore the prediction accuracy is poor.

For URL prediction, the best accuracy is achieved for the General query set. The most difficult prediction challenge is found on the Dynamic set. The main reason for the lower prediction performance is that those queries often require new URLs, and are thus harder to predict when no long-term patterns appear.

We present the error of the different models as a function of the prediction horizon in Figure 2.18. Similar to the results for query prediction, we see that the performance of most models decreases after a prediction horizon of approximately 7 days. Notable exceptions are the DML methods and the LIN and POW methods

	Baselines (Section 2.4.3)				Temporal SSM Models (Section 2.1.1)					Model Selection (Section 2.2)	
Query Type	AVG	LIN	POW	YES	SMT	TRN	PRD	TRN+ PRD	SPR	DML	BIC
General	0.02	0.02	0.02	0.02	0.01	0.02	0.01	0.01	0.01	0.02	0.02
Dynamic	0.77	0.75	**0.71**	**0.72**	**0.73**	**0.72**	0.74	0.76	0.76	**0.72**	**0.73**
Temp Reform	0.31	0.30	**0.27**	**0.27**	0.32	0.29	0.78	0.72	0.72	**0.27**	0.28
Alternating	0.49	0.49	0.48	0.47	0.51	0.51	**0.41**	**0.42**	**0.42**	0.48	0.51

Table 2.3: The prediction error of different models for predicting the query-independent number of clicks for a URL (lower numbers indicate higher performance). Best performing statistical significant results are shown in bold.

that remain stable over all prediction horizons.

2.4.7 Predicting Query-Dependent URL clicks

In the previous section, we compared the forecast models for predicting the overall clicks on a URL aggregated across all queries. We now repeat the analysis but focus on query-dependent clicks instead. Query-URL pair click prediction results are shown in Table 2.4. We first observe that DML has the best performance for the General, Temporal Reformulation and Alternating queries. The temporal model which smooths across surprises (SPR) is the most accurate for the Dynamic query type.

Across Tables 2.2–2.4, there appear to be two groups of SSM models: (1) Smooth and Trend models, and (2) Periodic, Trend+Periodic and Surprise models. Smooth and Trend models show very similar performance to each other. Similarly the Periodic, Trend+Periodic and Surprise models behave in the same way. Sometimes one group performs better than the other, but the groupings are consistent across the three tables and four query types. The main reason for this is that the second group considers periodicities and therefore has a larger set of variables to evaluate. These models have higher expression complexity, but are harder to learn. However, when sufficient data exists and the data is less noisy we see those models perform better.

We present the error of the different models as a function of the prediction horizon in Figure 2.19. We observe very poor performance of temporal models that incorporate periodicity of any sort (Periodic, Trend+Periodic, Surprise), even for larger horizons. The problem becomes even more evident at a prediction horizon of $h = 7$ and longer. This provides evidence that many of the query-URL predictions are not easily fit by periodic models. However, the different selection models had very high and stable performance.

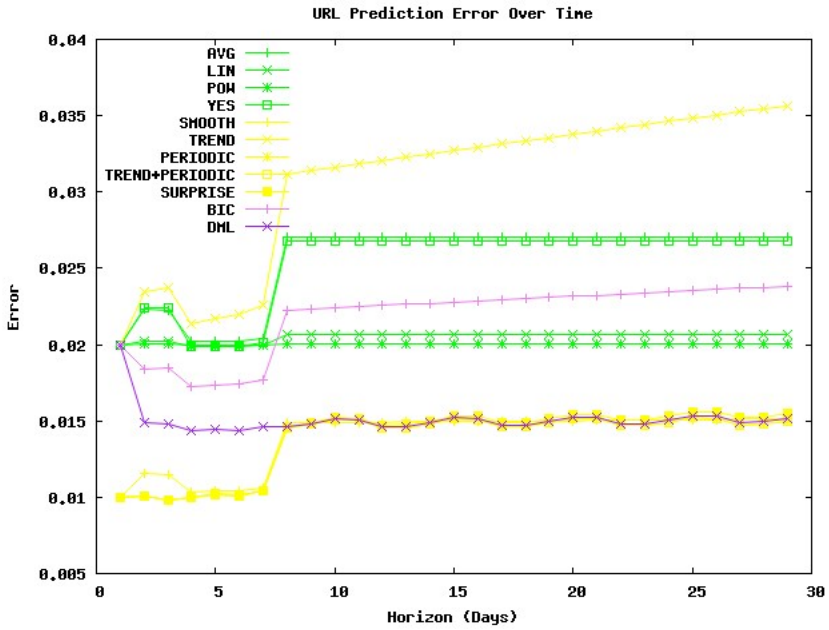

Figure 2.18: URL query-independent click prediction error over time on the General queries set (lower numbers indicate higher performance). Static models are shown in green, Temporal models are shown in yellow, and Temporal model selection methods are shown in purple.

2.5 Experiments for Ranking Search Results

We now apply the methods described in Section 2.1 to ranking of search results. The dataset used in this experiment is described in Section 2.4.1.We evaluate our techniques in two types of ranking experiments. In the first set of experiments, we use our predictions of future click behavior (derived from the models in 2.4.4) as the only source of evidence for ranking. In the second set of experiments, we combine our temporal predictions with other query-dependent features (e.g., number of words in query) and query-independent features (e.g., the URL depth) in a learning-to-rank framework.

2.5.1 Ground-truth Ranking

To evaluate the accuracy of a ranking system, it is common practice to use explicit human relevance judgments (labels) on query-URL pairs. Since user intent can change over time, relevance judgments may also change over time. Obtaining daily judgments on a large set of queries over a long period of time is a difficult challenge. Furthermore, it may be difficult for judges who are unfamiliar with a query to understand the time-varying intentions for the query. To overcome these

	Baselines (Section 2.4.3)				Temporal SSM Models (Section 2.1.1)					Model Selection (Section 2.2)	
Query Type	AVG	LIN	POW	YES	SMT	TRN	PRD	TRN+ PRD	SPR	DML	BIC
General	0.16	0.20	0.20	0.16	0.12	0.14	0.42	0.23	0.23	**0.10**	0.12
Dynamic	0.28	0.40	0.39	0.41	0.29	0.28	0.20	0.20	**0.19**	0.25	0.28
Temp Reform	0.53	0.68	0.67	0.69	0.66	0.69	0.58	0.58	0.59	**0.48**	0.63
Alternating	**0.08**	0.12	0.11	0.13	0.13	0.13	0.17	0.16	0.16	**0.09**	0.12

Table 2.4: The prediction error of different models for predicting the query-dependent number of clicks for a URL (lower numbers indicate higher performance). Best performing statistical significant results are shown in bold.

problems, we use *implicit judgments* drawn from query logs as our gold standard. The gold standard ranking for each day is determined by ordering by the click frequencies of theURL for a query on that day. Similar methods have been used to study personalized and contextual search [Teevan et al., 2005; White et al., 2010]. One challenge with using log data is that users are more likely to click on higher-ranked results than lower-ranked results [Yue et al., 2010]. In order to adjust for this position bias, we normalize by the probability of a click on the given URL at the displayed position for the given query. This method provides an unbiased estimate of the number of times a user would click on a given URL for a given query at a certain time.

2.5.2 Evaluation Metrics

We seek to compare the quality of the rankings generated by our models with those from the gold standard. A common way to compare two ranked lists is to consider the *correlation* between the two rankings. We computed both the Kendall's τ rank correlation and the Pearson product-moment correlation. Rank correlations assess the extent to which one variable increases as the other increases. Pearson correlation assesses the linear relationship between two orderings, and is especially effective for comparing two *regression* variables. The score not only measures the correctness of the ranking, but also takes into account the magnitude of the difference. Results obtained from the Kendall's τ rank correlation and the Pearson product-moment correlation are similar, so we present only the Pearson correlations in the work.

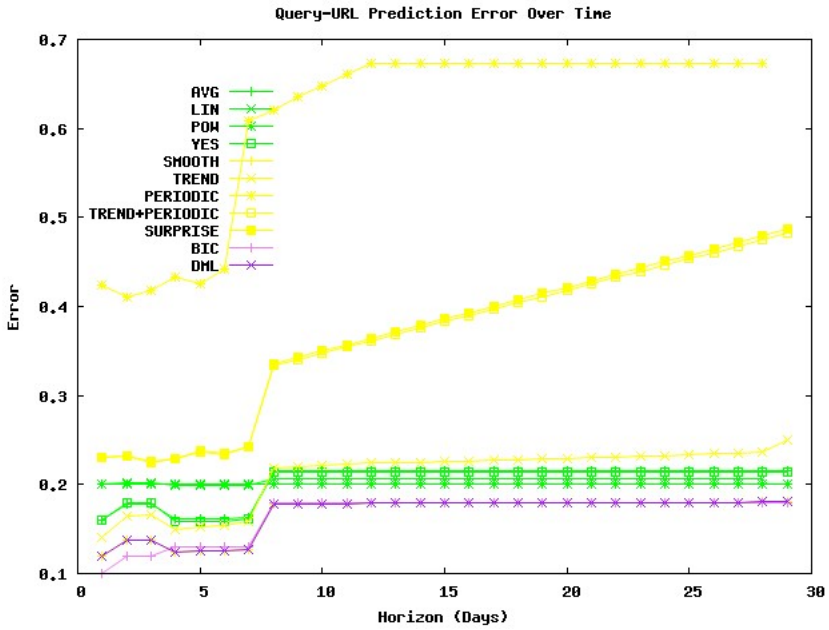

Figure 2.19: The query dependent URL click prediction error over time on the General queries set (lower numbers indicate higher performance). Static models are shown in green, Temporal models are shown in yellow, and Temporal model selection methods are shown in purple.

2.5.3 Baselines

In experiments where we only use temporal features as evidence for ranking, we compare the ranking performance of the temporal-modeling methods (Section 2.1) versus the performance of the static-modeling baselines (Section 2.4.3).

In the experiments where temporal features are added to the feature set of a *base* ranker, we compare the performance of the ranking algorithm using three different sets of features: only base features; the base features plus static modeling features; and the base features plus temporal modeling features. The feature set of the *base* ranker consists of approximately 200 typical information retrieval features such as several variants of BM25 [Robertson et al., 2004] to measure the similarity of a query to document body, title, and anchor texts. Additionally, we used other query dependent features, such as the matched document frequency of a term, bigram frequency, number of words in query, etc., and a set of query-independent features, such as the URL depth and the number of words in the document's body, title.

	Baselines (Section 2.4.3)				Temporal SSM Models (Section 2.1.1)					Model Selection (Section 2.2)	
Query Type	AVG	LIN	POW	YES	SMT	TRN	PRD	TRN+ PRD	SPR	DML	BIC
General	0.91	0.92	0.93	0.92	0.98	**0.98**	0.08	0.98	**0.99**	**0.99**	0.98
Dynamic	0.28	0.35	0.38	0.36	0.41	0.41	0.06	0.06	**0.46**	0.42	0.41
Alternating	0.80	0.82	0.84	0.82	**0.89**	**0.89**	0.06	0.06	0.06	**0.89**	0.87
Temp Reform	0.95	0.95	0.95	0.95	0.97	0.96	0.24	0.25	0.65	**0.99**	0.95

Table 2.5: Quality of URL ranking models produced by different forecast models using only predicted clicks, as measured by the Pearson correlation with the ground truth URL ranking. The best performing statistical significant results are shown in bold.

2.5.4 Ranking Task

For each query-URL pair we compute the predicted behavior and use that prediction to rank the URLs. We first divide the data into training and test sets. Behaviors from time $t_i, \ldots, t_{i+n}$ are used to estimate the model parameters and the resulting models are used to predict the user behavior on the following day t_{i+n+1}. For each query, we compare the predicted ranking to the gold standard ranking on day t_{i+n+1}. We investigate different prediction windows (of 1–10 days), and perform prediction on different days (training was performed on the dates 12-15-2010 until 4-15-2011 and the testing was done on on the subsequent 10 days). For each query type (described in Section 2.4.2), we calculate the mean of the Pearson correlation (ρ)over the queries in our test set.

2.5.5 Temporal Features as Only Evidence for Ranking

We begin by looking at how well the predictions for URL click behavior perform compared with several baseline predictions. The results can be found in Table 2.5, that presents the Pearson scores for each of our four query types.

For every type of query, the proposed temporal modeling approach outperforms all of the baselines, showing that learning the appropriate weighting for historical data is preferable to selecting a single weighting for all query and/or URLs. For most classes of queries, the Pearson scores are quite high, suggesting that the predicted ranking is very close to the ideal ranking. We observe that ranking based only on predicted user clicks is consistently better than the baselines for all types of queries. The correlations are noticeably lower for Dynamic queries, although temporal models still significantly outperform baseline models. Dynamic queries are ones that have been labeled as requiring fresh information.

For these queries, there is limited behavioral data available related to this fresh information need. Although we observe that models that weight more recent data more heavily (like the trend and power weight models) tend to perform well on this query class, the overall performance is still significantly lower than the accuracy in other query classes. A closer look at the data reveals that most of those queries are celebrity and sports-related queries that have sudden spikes. These sudden spikes are hard to predict, as they are event driven. However, after such a spike starts, the prediction of its peak and its decline are more easily predicted, as those spikes usually persist for 1-2 days. Most errors of those models occur in queries where the test date is drawn from the beginning of a spike.

For every query class, two of the temporal approaches that we explored, Smoothing (SMT) and Trend (TRN), show significant improvements over all of the baselines. Recall that ranking with Smoothing creates a short-range prediction assuming a stable mean, and Trend adds to this model the notion of a trend. The more complex temporal models, namely Periodic (PRD), and Trend + Periodic (TRN+PRD), does not generally perform as well. The Periodic model exhibits poor performance for most query sets. The Periodic model requires the estimation of many parameters – the size of the parameters is linear by the size of the periodicity. Estimating such a large number of parameters likely requires more data than we had available; we believe that this is the main reason for its lower performance. Furthermore, some of the periodicities were yearly, and therefore not identified well. The poor performance of the model on non-periodic queries stems from the miscalculation of the periodicity of those queries, which usually results in a noisy prediction.

We reported results for the top four URLs. We also examined performance of the models over different numbers of top ranked URLs. We have experimented on rankings of up to 12 URLs. We did not observe any statistically significant change in the results, and hence do not present them here.

2.5.6 Temporal Features as Input to a Ranking Algorithm

In this section, we investigate the effectiveness of predicted temporal features when added to a baseline learning-to-rank retrieval model. We employ a supervised learning technique to learn the ranking function that best predicts the ground-truth ranking of URLs for a given query, using a variety of query, and URL click features. In these experiments, we use boosted regression trees as our ranking function [Burges, 2010].

We divide the data into training queries Q^{train} (80% of the total number of queries Q) and test queries Q^{test} (20% of Q). The features used in Q^{train} are obtained using only data from the times $t_1, \ldots, t_n$. For every query in Q^{test}, we use the data from the times $t_1, \ldots, t_n$ to predict the user behavior for the

	Baselines (Section 2.4.3)				Temporal SSM Models (Section 2.1.1)					Model Selection (Section 2.2)	
Query Type	AVG	LIN	POW	YES	SMT	TRN	PRD	TRN+ PRD	SPR	DML	BIC
General	0.47	0.97	0.98	0.98	0.91	**0.99**	**0.99**	0.60	0.97	**0.99**	0.72
Dynamic	-0.08	0.30	0.30	0.39	0.59	0.46	0.01	0.30	**0.62**	0.61	0.35
Alternating	0.23	0.64	0.90	0.74	0.78	0.87	0.90	0.23	0.65	**0.98**	0.84
Temp Reform	0.19	0.73	0.97	0.96	**0.99**	**0.99**	0.98	0.98	**0.99**	**0.99**	0.96

Table 2.6: Quality of URL ranking models produced by different feature sets for learning-to-rank as measured by the Pearson correlation with the ground truth URL ranking. The best performing statistical significant results are shown in bold.

following day, and use this prediction as a feature fed into the learner for testing. Using separate train and test queries allows us to generalize to queries that have not previously been seen. Using the results of the ranker, we again calculate the average Pearson correlation for every query category.

We performed experiments comparing performance of a ranker using a variety of query-dependent and query-independent features in addition to temporal features based on historical patterns of user behavior.

The Pearson correlation results of these experiments can be found in Table 2.6. In nearly all cases, the inclusion of features based on user behavior in the ranker improved performance, regardless of how the behavior was modeled. The level of improvement observed was often quite significant, confirming that user behavior is a very important aspect of Web search result ranking [Agichtein et al., 2006]. The rankers that use temporal modeling consistently performed the best across all query classes.

2.5.7 Query Effects

We have seen that the different models perform differently on different classes of queries. In this section, we analyze characteristics of the queries that benefit from temporal behavioral signals when used for ranking. We perform an analysis of the results using the same 973 characteristics discussed in Section 2.2.2, and identify the characteristics that differentiate between the ranking results with and without the use of temporal models. For these experiments, we assess statistical significance using paired t-tests comparing the best performing temporal models to all other models.

Click Entropy We begin by looking at whether ambiguous queries are more or less likely to benefit from temporal modeling of user behavior. We use *query*

click entropy as a measure of the variation in what users click following a query, as defined in Eq. 2.11. We calculate the click entropy over the learning period for each of our queries.

We observe that the ranking of queries with higher click entropy (over 0.37) improves when using a model with temporal signals compared to using models with only baseline user behavior features. Queries with many different distinct search results clicked at different times benefit from temporal modeling. Because the click entropy was calculated over the entire study period, and not just a single slice of time, it is likely that high click entropy often indicates the search engine users' need for different information over time. For example, the query *lottery* has high click entropy, as the different lotteries results are reported on different days, depending on the location. Therefore, every day, users tend to click on the lottery site of the appropriate location. As the result are reported periodically in each location, the temporal models manage to predict the correct ranking of the results.

Click Predictability We measure the predictability of a query by calculating the average error in predicting its search volume (query clicks) using either a baseline model (e.g., Averaging) or a temporal model (e.g., Smoothing) over the seven days before the prediction. An analysis of the data reveals that temporal ranking performs poorly compared with the baseline models when the query clicks predictability is low. Regardless of the model used for predicting query clicks, the error is high (0.5-0.58) when the performance of the temporal model is worse than the baseline model. The error is low (0.27-0.36) when the performance of the temporal model is better than the baseline model. This indicates that if the query search behavior on its own is predictable, predicting the right ranking for it over time might be a simpler task.

Similarly, we found a high correlation between the predictability of the query-URL search click volume and the performance of the ranker with time-series modeled features. We measure this predictability by measuring the average error of prediction of every URL for the query. The error is measured by averaging over the temporal model prediction error for 7 days before the prediction over the URLs. Queries for which the time-sensitive ranker performed well indicate that query-URL features have high predictability. The error rate is lower (0.20) for instances where temporal modeling performs better than the baseline, and is higher (0.27) when the temporal modeling had worse performance. Wound those differences to be statistically significant. This query-URL number of clicks feature is important for ranking, as it is eventually the goal function being optimized. Queries and query-URLs whose search volume changes in an unpredictable manner pose harder challenges for temporal ranking. We believe that difficulties arise during learning because extracting meaningful statistics is more challenging

in those scenarios. In these cases, simple average modeling of the query-URL volume yields better results.

Spiking Trends It may seem intuitive that the behavior of queries associated with numerous spikes over time is hard to predict. We analyze the correlation between ranking performance and the number of spikes in the query, query-URL or URL time series. We found no significant correlation. We believe that this is because some spikes can be predictable with high precision, such as in the case of the periodic behavior.

Query Shapes

In this section we summarize how the different query shapes indicate which temporal model to apply. As discussed in Section 2.1, query shape often indicates whether a temporal model or a baseline should be applied in ranking. We clustered the different click behaviors in our dataset based on their shapes, using Expectation–maximization clustering algorithm. Figure 2.20 shows the four query and query-URL shapes where the temporal model yielded better rankings than baselines rankers. . These four shapes have upward or downward trends with different slopes. Those can be modeled well by the Smoothing and Trend techniques with a correctly learned slope. This clustering result is consistent with the results in Sections 2.5.5 and 2.5.6, where these two modeling techniques tended to yield the highest performance. We did not find distinctive query clusters for the cases when the baseline ranker outperformed the ranker using temporal modeling.

2.6 Experiments for Ranking Query Auto-Complete Candidates

In this section, we apply the techniques presented in Section 2.1 to predict query clicks for QAS ranking. The ground-truth QAS ranking for a prefix $\mathcal{P}$ is the list of all queries that start with $\mathcal{P}$ (or are somehow filtered for the prefix) ordered according to their *true* popularity. At any given time t, the true popularity values are set according to the observed query frequency values at that time. Obviously, this information is not available to QAS ranking models at runtime, and we have to rely on historical data at time $t-1$ and before to *predict* this unobserved ground-truth ($\mathcal{G}$) and rank candidates accordingly.

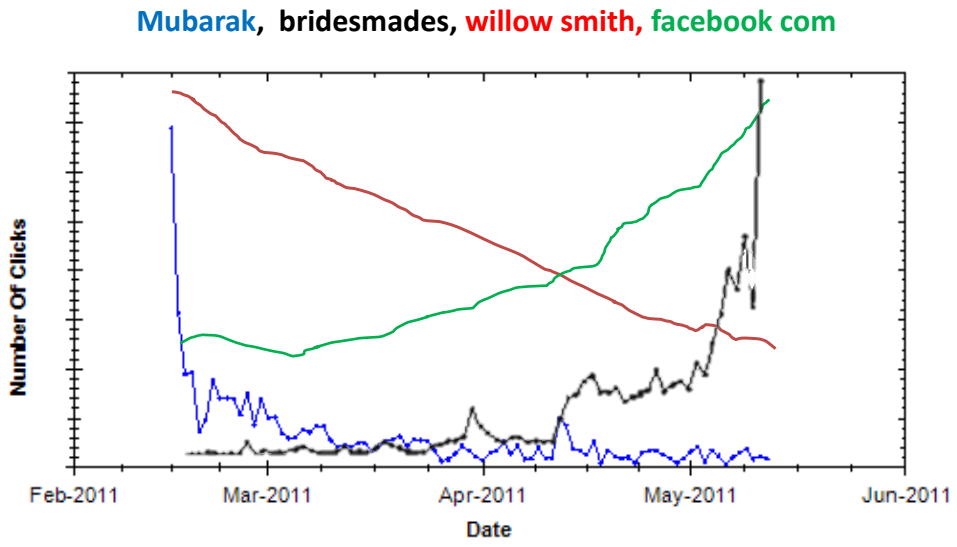

Figure 2.20: Dominant query shapes for queries where the proposed temporal model yields better rankings than the baseline rankers.

2.6.1 Data & Ground-truth Ranking

We use the dataset described in Section 2.4.1 and generate an auto-completion trie based on query frequency predictions for each of the days in the testing period. For each forecast method at time t, we rank the top-5 ground-truth candidates based on the data available at time $t-1$ and before. In an oracle list of QAS suggestions, candidates are sorted in descending order of their *ground-truth* popularity. We consider two types of ground-truth: (1) QAS Clicks Ground-truth; and (2) Total Clicks Ground-truth.

QAS Clicks Ground-truth We obtained click data for each QAS suggestion for the period of May 6th until May 12th, 2011 (a total of 4,687,814 prefixes and queries). The raw data consists of a date, a prefix typed, and the suggestion clicked for this prefix on that date. Therefore, our gold standard for a certain time and prefix is the ranking induced by the number of times the query suggestion was clicked for this prefix at this time.

Total Clicks Ground-truth The coverage of QAS clicks is limited to those queries that were available in the auto-completion trie and were selected from the auto-completion list by the users. However, a significant fraction of queries submitted by users are not available in the auto-completion trie. In addition, users sometimes issue the query directly from the search box even when it exists

in the auto-completion list. Therefore, we generate another *ground-truth* dataset in which we consider the total clicks on queries regardless of whether they were chosen from auto-completion lists or not. We obtained click data for each query for the period of May 6th until May 12th, 2011 (a total of 3,384,954 queries). We computed the probability of a click on a query by the volume of clicks on this query on that date. The raw data consists of a date, a query issued and how many times it was clicked at that date. Therefore, our gold standard for a certain time and prefix is the ranking induced by the number of times queries with this prefix were issued and had a click on that date. That is, for each prefix $\mathcal{P}$ at time t, we match all the queries that start with $\mathcal{P}$ and rank them according to their true total frequency at time t to generate the ground-truth ranking.

2.6.2 Evaluation Metrics

We compare the quality of the QAS generated by our models with those from the gold standard. As in our previous experiments, we use Pearson product-moment correlation to compare the orderings of the QAS.

2.6.3 QAS Ranking Results

In Table 2.7 we present the results for ranking query suggestions based on both ground-truths. We did not observe any significant difference in the results using temporal or static user behavior models. However, we see that the DML model, which is trained to choose among the more specific models, achieves the highest performance over all queries, particularly when total clicks are considered as ground-truth. The results were found to be statistically significant. Periodic models have shown low performance in both sets, since many of the queries are not periodic and therefore the periodic modeling hurts the performance. The results indicate once again that correct modeling of historical user behavior signals is beneficial.

2.6.4 Query Effects

We examined the data in more detail to investigate which queries benefit from temporal modeling. We categorized the queries into two groups: news and periodic queries.

News or event-related queries Many queries that have sudden spikes benefit from Trend or Smooth modeling. For example, the query *Navy Seals* started trending on May 2nd, 2011 during the Osama Bin-Laden mission. In the period following May 2nd, we observed that static modeling of this query for the prefix

	Baselines (Section 2.4.3)				Temporal SSM Models (Section 2.1.1)					Model Selection (Section 2.2)	
Ground-Truth	AVG	LIN	POW	YES	SMT	TRN	PRD	TRN+ PRD	SPR	DML	BIC
QAS	0.88	0.88	0.88	0.88	0.87	0.88	0.83	0.83	0.89	**0.91**	0.88
Total Click	0.93	0.94	0.94	0.93	0.93	0.93	0.85	0.85	0.95	**0.97**	0.94

Table 2.7: Quality of auto-completion ranking models produced by different learning-to-rank feature sets, as measured by the Pearson correlation with QAS (top) and total clicks (bottom) Ground-truth ranking. The best performing statistical significant results are shown in bold.

navy had lower performance than the Trend and Smooth modeling. This is because this event had not been observed in the past, so past data was not indicative for the current ranking. An interesting subclass of queries are celebrity queries. Queries about celebrities (such as *Chris Hemsworth* and *Marie Osmond*) are usually event related, and benefit from the temporal modeling. For example, at the time of the prediction, a new film starring Chris Hemsworth was released and Marie Osmond remarried her first husband. Both events caused spikes in the query volume which static models have difficulty modeling. Although the YES model performed well on this Hemsworth query predictions, it still had lower performance than that of the Trend model that correctly modeled the interest over time.

Periodic queries Periodic queries also benefited from the temporal periodic modeling. For example, the query *first friday* (for the prefix *first*) was very poorly modeled by the static behavior models, but the Periodic models modeled it with very high performance. The periodic models had a Pearson correlation of 0.99 vs. -0.34 for the static models over the periodic queries.

2.7 DML Component Analysis

This section summarizes the details of training surprise and periodicity detection models.

2.7.1 Learning To Detect Periodicity

Detecting the periodicity size of a time series is crucial for periodic models. For this process we use a signal processing method called *autocorrelation* that provides a signal's correlation value with itself [Dunn, 2005]. This method can be used to

detect repeating patterns with noise, and is defined as

$$Autocorrelation(f, h) = \int_{-\infty}^{\infty} f(t+h)f(t)\,\mathrm{dt}, \tag{2.14}$$

where h is the shift of the series, i.e., the periodicity. Several h values are experimented, and the highest value of the autocorrelation for those values is used. A query is classified as *periodic* based on a certain threshold ω:

$$Autocorrelation(f, h) > \omega, \tag{2.15}$$

Specifically in the Web domain, we found it beneficial to limit the possible h values to common Web-periodic intervals, such as weekly, monthly, and yearly.

We performed experiments to evaluate the periodicity detection algorithms. This component is crucial for initializing the SSM seasonal models. To capture long as well as short periodicity, search logs for query-click data for the period 2006–2011 were obtained, and a daily time series was generated for every query. We used the seasonality dataset [Shokouhi, 2011] that contained 259 queries, each annotated as seasonal or not by several judges. We compare our method for periodicity detection against the method described in [Shokouhi, 2011] (we refer to it as SC), which was applied to perform seasonal-trend decomposition on the time series, comparing the seasonal component to that of the time series before the decomposition.

To evaluate our periodicity model, we evaluate performance for different autocorrelation thresholds, ω. Figure 2.21 shows the precision-recall results for our autocorrelation model (Autocorrelation) compared to the baseline model (SC). The Autocorrelation method proposed in this work reaches the same maximum recall as the state-of-the-art SC autocorrelation method (around 0.85), and outperforms it in precision for every recall level by up to 15 percent. We also performed experiments for applying autocorrelation without any lag limiting. The result yields a recall of about 0.25–0.27 with precision of 0.5–0.6. We conclude that the lag limitation for regular Web periodicity ($h = 7, 28, \ldots, 31, 360 \ldots 365$) is important.

2.7.2 Learning To Detect Surprises

Queries can be modeled using one of the SSM models introduced previously but, as we discussed earlier, sometimes areas of the time series exhibit *surprising* behavior that is not well-fit by the model. We conjecture that, when a temporal model encounters a surprise in the data, the model's residual error stops behaving linearly. Intuitively, in the beginning of a spike the model *"under-predicts"* the data since the previous data did not include any indication of a spike. After the

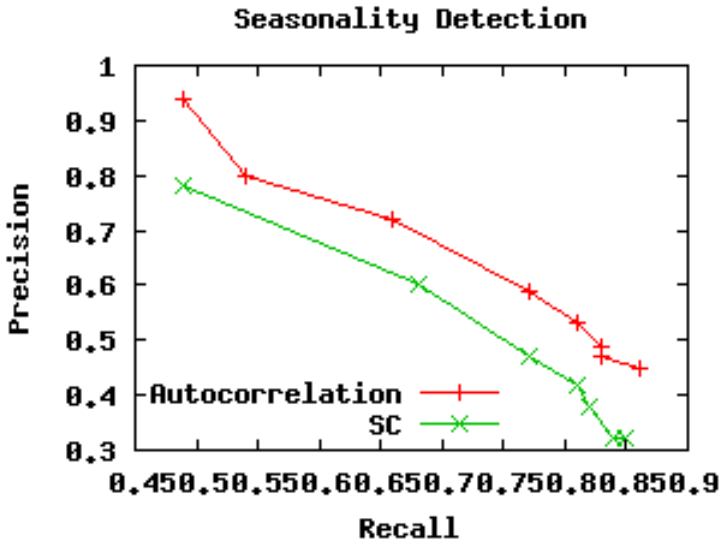

Figure 2.21: Comparison of SC method with Autocorrelation method for seasonality detection by precision (y axis) and recall (x axis).

spike, the model tends to *over-predict* the data, as it still considers the spike's data. We introduce *surprises* as significant changes in the residual during the period of an event. Let $r = r_1, \ldots, r_n$ be the residuals of the temporal model for the times $t_1, \ldots, t_n$, where $r_t = o(t) - \widehat{o}(t)$, and $\widehat{o}(t)$ is the prediction of the model for time t. Let $t'_1, \ldots, t'_m$ be surprise candidates time points, such that $r_{t'_1-1} \cdot r_{t'_1} < 0$ for each $t' \in \{t'_1, \ldots, t'_m\}$, i.e., locations in the time series where the residual changes signs. Let r_{t_1} and r_{t_2} be two neighbouring sign-change points, such that $r_{t_1-1} \cdot r_{t_1} < 0, r_{t_2+1} \cdot r_{t_2} < 0, r_t \cdot r_{t_1} > 0, t_1 \leq t \leq t_2$. We define an impact of an event as

$$Impact(t_1, t_2) = MSE(o, t_1, t_2, m) = \frac{\sum_{t=t_1}^{t_2} r_t^2}{t_2 - t_1}.$$

Intuitively, only unexpected dynamics that have long impact on the model should be considered as surprises. We propose a greedy procedure that adds the surprise locations starting from highest to lowest impact to the model and measures the improvement of the model (using BIC criterion). When the model stops improving, we output the surprises. The surprise detection algorithm is given in Figure 2.22.

We performed experiments to evaluate our surprise detection algorithm. We obtained a set of 24 queries judged by humans as news-related queries. Judges were also asked to provide a label (event or not) for every week in the series (a

Procedure SURPRISE DETECTOR($o(t_1), \ldots, o(t_n)$,m)
  $BIC_{i-1} \leftarrow \infty$
  $BIC_i \leftarrow \infty$
  $EventCandidates = \{t'_1, \ldots, t'_m | r_{t'_{i-1}} \cdot r_{t'_i} < 0\}$
  $Events = \{\}$
  Do
    $curEvent \leftarrow \arg\max_{t'_i} Impact(t'_i)$
    $EventCandidates \leftarrow EventCandidates / \{curEvent\}$
    $m_i \leftarrow$ Build model m with events $Events$
      using training data $o(t_1), \ldots, o(t_n)$
    $BIC_i \leftarrow BIC(m_i)$
    If $BIC_i > BIC_{i-1}$
      $Events \leftarrow Events \cup \{curEvent\}$
      $BIC_{i-1} \leftarrow BIC_i$
    Else Return $Events$
  While $EventCandidates \neq \{\}$
  Return $Events$

Figure 2.22: Detecting surprises in time series.

total of 313 data points for queries behavior for the years 2006-2011). A total of 61 events were detected. For every query we trained the surprise detection model and a baseline method, and compare their results. The baseline model is a method that detects peaks as events. This is a reasonable baseline, as many of the events can be seen as peaks in the query stream.

Table 2.8 shows results of our surprise detection algorithm compared to the baseline peak detection algorithm. We see the surprise detector has high precision and low recall. On the other hand, the peak detector identifies all surprises but with very low precision. As most peaks in queries are usually noise, and not real events, we prefer the Surprise Detector over the peak detection method. For example, the query *credit rating* has some seasonal peaks after S&P declarations, which should not be considered a surprise. However, the peak detector identifies them as such, while the surprise detector does not recognize it as an event. On Aug 8th, when U.S. credit rating was downgraded, the query volume exhibited an unusual peak, which was identified by the surprise detector.

Table 2.8: Surprises Detector Results. Recall and precision of the different surprises detectors. Statistically significant results (as measured by a t-test) are shown in bold.

	Surprises Detector	Peak Detector
Precision	**96.42%**	46.66%
Recall	59.92%	**100%**

2.8 Conclusions

We developed methods for modeling the dynamics of the query and click behaviors seen in a large population of Web searchers. We modeled temporal characteristics that are often observed in query and URL click behavior, including trend, periodicity, and surprise disruptions. We presented several different temporal representations and learning procedures and showed how we can construct models that predict future behaviors from historical data.

In almost all cases, we found that temporal models performed better than the baseline models that use the same weighting of historical data for all queries for predicting clicks, ranking search results and ranking query suggestions. We also found that different temporal models are appropriate for different types of queries. Query click behavior tends to be rife with unmodeled disturbances (surprises). Thus, extending models with the ability to identify and smooth out such disturbances can enhance predictive power. For specific types of queries, such as Dynamic and Alternating queries, trend models are more appropriate. URL click behavior tends to exhibit somewhat different temporal behavior showing fewer disturbances (as we did not see much gain with using the surprise temporal model). Thus, models that understand trend and smooth historical data using learned models tend to perform better for these predictions. For Alternating queries, periodic modeling tends to work better. The dynamics seen in query-URL click behavior over time provides interesting insight into changes in users' intentions over time. For General and Alternating queries smoothing and trend models are the best temporal models, whereas for Dynamic and Temporal Reformulations modeling periodicities improves performance. We observed that, in general, the application of smoothing or incorporating trend is the best methodology.

We introduced the Dynamics Model Learner (DML) algorithm that learns the correct model to apply for queries, URL, or query-URL pairs without a priori categorization of queries into groups. We compared the predictive performance of this method with static baselines, temporal models, and a standard model selection algorithm (BIC), and found that it has superior performance for all groups of queries. We believe this result paves the way for incorporating multiple

types of models for better time-aware search procedures.

The kinds of time-aware modeling of user behavior that we introduced in this work can be incorporated in many search-related applications. In this work, we examined basic click predictions, as well as two end-to-end applications – ranking of results and query suggestions. Query click prediction can be used to improve query auto-completion to present the most appropriate suggestions at the time the query is issued. We studied this application and provided evidence that, in most cases, the temporal models are more successful in query suggestions. Query-URL prediction can induce better ranking that is more aware of the user query-URL temporal intent. We evaluated this claim in a variety of experiments providing support that temporal modeling improves ranking for many classes of queries.

There are several directions for future work. We believe that further experiments examining how long it takes for temporal aspects of a new query or URL to sufficiently permeate the search stream so that it can be aided by temporal prediction are needed to enhance many applications. A more in-depth study with human subjects can also reveal whether users perceive a qualitative difference in the user experience when temporal techniques to improve ranking, QAS, etc. Additionally, studies of other model selection techniques, such as AIC or multimodel approaches, e.g., using Bayesian model averaging, might also shed light on the pragmatic use of these techniques.

Additional applications of these techniques are also possible and include the use of URL click prediction to improve re-crawling strategies, by focusing crawling efforts on URLs that are likely to be clicked. In general, we believe similar techniques might be used to optimize indexing, storage, and retrieval strategies to improve response time. Furthermore, the models presented can also be used at different time granularities and applied on different types of objects (Query, URL and Query-URL), either aggregated for all users or applied on specific user behavior, thus creating time-aware personalized retrieval, where the temporal intent is modified to accommodate individual searchers.

Chapter 3

Predicting Content Dynamics on the Web

Numerous aspects of the Web change over time, from subtle dynamics such as the user Web behavior we have discussed in the previous chapter, to more salient changes as the addition and evolution of content by human agents. However, in many retrieval frameworks, Web pages are still often treated as static documents, whereas, in reality, pages are dynamic channels whose contents change with time. Some pages, like news entry points, change on a daily basis with regularity. While others, like faculty home pages, may be updated with far less frequency and regularity. These changes may involve significant changes in content, the addition or deletion of hyperlinks, or the addition of hyperlinks to previously unknown web pages. Furthermore, these changes in content drive changes in the query distribution that lead to a page, and impact the relevance of a page to the query.

The ability to predict key types of changes can be used in a variety of settings. For example, accurate prediction of a significant change in page content enables an improved incremental crawling strategy that only recrawls pages when necessary [Olston and Najork, 2010]. Similarly, accurate prediction of a new hyperlink to a previously unknown (i.e., non-indexed) web page, enables one to efficiently discover pages that should be indexed. In a personalized retrieval setting, accurate prediction of re-visitation to a page with new content enables one to push new content to a user.

When modeling any variable of interest as a function of time, it is natural to consider the past observed values as a basic means of prediction [Cho and García-Molina, 2003b]. For example, in this setting, when predicting whether a page will change, we can simply predict based on the change frequency for the page observed over some past historical window. However, there are other signals that provide information beyond change frequency.

In particular, the content of a page enables better prediction of its change

[Barbosa et al., 2005; Tan and Mitra, 2010; Tan et al., 2007]. It provides a finer distinction of the degree of change that a page is undergoing, and the relationship among these changes. For example, in a series of ten days, a page may experience small changes on every day or large changes. These changes in content might be similar to each other or vastly dissimilar. For a page that experiences vastly dissimilar changes on a regular basis, the changes may be considered more significant (since the content change implies a need for reindexing). On the other hand, similar changes with the same regularity may be considered less significant. To distinguish these cases, the appropriate features need to be encoded.

Pages that are related to the prediction page may also change in a similar, although not necessarily, synchronous fashion [Cho and Ntoulas, 2002; Fetterly et al., 2003a,b; Tan and Mitra, 2010; Tan et al., 2007]. These pages would then provide a useful signal for determining whether the prediction page will change. The *strength* and the *type* of the relationship between those pages is likely to be indicative of how well one can predict the change of the other. In the web setting, natural choices of relationship types include: distance in the web graph, similarity in content, and similarity in temporal change pattern. We explore all of these relationship types and evaluate their relative contributions over the page's content alone. In particular, our exploration of similarity in content *over time* through the use of cross-correlation and the dynamic time warping (DTW) algorithm is especially novel. Both of these provide more robust computation of similar temporal change patterns by dealing with the case where content between two pages may be correlated in time but slightly out of phase.

Additionally, even after identifying other related information sources, there is a question as to how to best incorporate this information into a machine learning algorithm. We develop an expert combination approach where the experts are selected according to their degree of "relatedness" and demonstrate the superiority of this modeling choice to other combination and relational learning techniques.

This chapter is organized as follows: In Section 3.1, we present the formal problem description and outline the challenges and research questions. We then present a general characterization of the types of information sources available when predicting page change using past history (Section 3.2.1). Using this characterization, we develop specific feature sets that capture aspects of each of these information sources in a useful way for modeling change (Section 3.2.2). In particular, in addition to the past history of a page's change, we also incorporate features based on the page content of the page as well as the content of *related* pages. For each information source we provide several algorithms to combine the information in a learning setting. In particular, we present a novel algorithm to combine the information in an expert prediction framework (Section 3.3). We examine techniques for identifying related pages, including pages that are close in the web graph, pages with similar content, and pages with correlated content

across time – where the last two prove to be especially effective (Section 3.4). Finally, in an empirical study, we not only demonstrate the effectiveness of our methods, but also explore alternative modeling choices, analyze the qualitative differences in the expert selection methods, and present the superiority of our prediction methods in a crawling application, which strives to maximize the *freshness* of its index (how many pages in the index remain up-to-date as compared with the online web copies).

3.1 Problem Formulation

We consider the general setting where we want to estimate a function h for each page. This h can indicate several types of page change (Section 3.1.1). This function can be estimated using different information types (Section 3.1.2), which have different observability when estimating h (Section 3.1.3).

3.1.1 Types of Web Page Change

Let O be a finite set of pages, and let $h : O \times \mathfrak{T} \to \mathbb{R}$ be a temporal goal function, that provides information about the state of a page $o \in O$ at time $t \in \mathfrak{T}$, where $\mathfrak{T}$ is a discrete representation of time. Depending on the targeted application, a function h can be defined in numerous ways. For example, the change may be:

1. Whether the page $o \in O$ changed significantly or the amount of its change. This can be used to identify when a page is stale enough to be recrawled.

2. Whether the change in page $o \in O$ corresponds to a change from non-relevant previous content to relevant current content for some specified query load. The function not only indicates whether the page is stale, but also that the change will result in the page being surfaced for a query.

3. Whether there is a new out link from page $o \in O$ and how many of those. This helps indicate when recrawling a page will lead to discover a new portion of the web graph, discover new anchor text, or impact link analysis metrics.

We focus on page changes of type 1, although our approach is readily applicable to the other change types. We consider a *significant page change* to be a change in content between the content vector at time t and the content vector at time $t + 1$ whose word distribution is different in a statistically significantly sense (as measured by a chi-squared test) from that of the previous day. That is, the p-value produced by performing the test satisfies:

$$p^{\chi^2}(\text{ContentDist}(t), \text{ContentDist}(t + 1)) < 0.05 \tag{3.1}$$

3.1.2 Information Sources

As we discussed before, when estimating h for each page, we can consider several types of information. In many previous studies [Ali and Williams, 2003; Cho and García-Molina, 2000, 2003a; Edwards et al., 2001], only information about the changing object was used, e.g., the change rate of the specific page. We propose a general-unifying framework (formally discussed in Section 3.2.1) of the possible information types, ranging from only information about the changing object to information about all the other related objects. We divide the information types into the following information settings:

1. 1D setting: In this setting, only information about h of the page itself in previous time-stamps is used to estimate h. For example, only information involving the rate of the page change itself is used to estimate future page change. This setting was explored in the past [Cho and García-Molina, 2003a; Edwards et al., 2001], and we consider it as one of the baselines.

2. 2D setting: In this setting, a wider range of information about the page h itself is used to estimate h. We consider the page as an entity with features which can be used to understand its future h, e.g., future change rate. This setting was explored by several works [Ali and Williams, 2003; Cho and García-Molina, 2000], and we consider it as one of the baselines as well.

3. 3D setting: In this setting, not only is the information about the specific page used, but also the relations to other pages are considered in determining the probability of a future h value of a page, e.g., the future change of a page.

3.1.3 Information Observability

In a *fully-observed scenario*, at each time stamp t, all the above information from the previous n time stamps is available. This is a scenario, where, given a set of k periodically observed pages over a period of n, the goal is to induce a function h that can be used to estimate page changes within this group in the following time-stamp. Figure 3.1 illustrates the three information settings in the fully-observed scenario.

The more general scenario, is a *partially-observed scenario*, where only partial information about the objects in previous times is given. That is, at each time stamp t, only the information about some subgroup is available to the learner. This is an online scenario, where, for example a crawling system doesn't observe the information about non-crawled pages. It only crawls pages it believes to have changed, and therefore only information about those crawled pages is given to

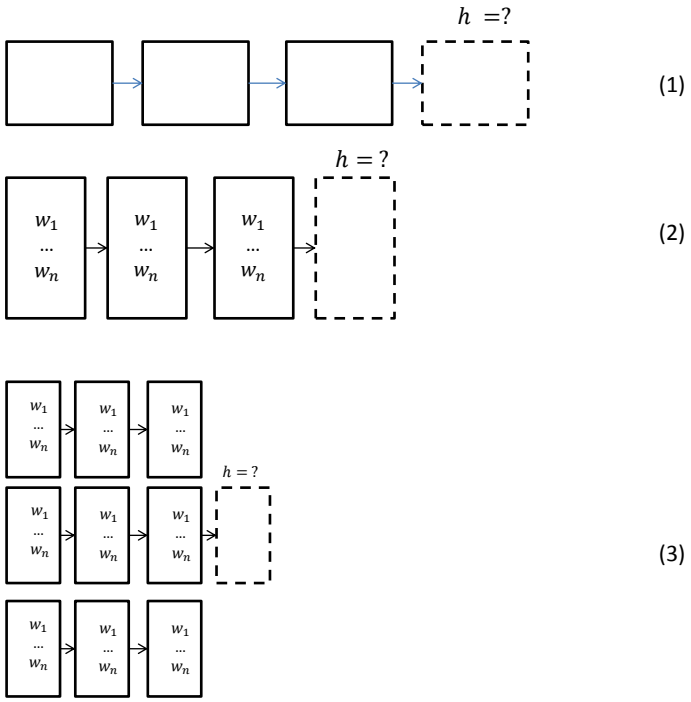

Figure 3.1: Fully observed setting.

the learner to induce an approximation of h. The three information settings in the partially-observed scenario are illustrated in Figure 3.2.

In this work we focus on the fully-observed setting and provide algorithms based on the 1D, 2D and 3D settings. We wish to derive insights and understanding of this setting before generalizing to the partially-observable scenario. Additionally, in Section 3.6.3 we empirically show a simple application of our algorithms for the partially-observed setting, where the objects are iteratively sampled to gain more information. Section 3.2.1 formalizes the problem from a machine learning perspective, and Section 3.3 outlines our algorithmic solution to the problem.

3.2 Solution Framework

In this section, we formalize the learning problem and provide an algorithmic framework for predicting change. Unlike standard learning algorithms that assume a static feature set and a single object type, we present a formalism that handles temporal features, derived from both single and multiple objects over

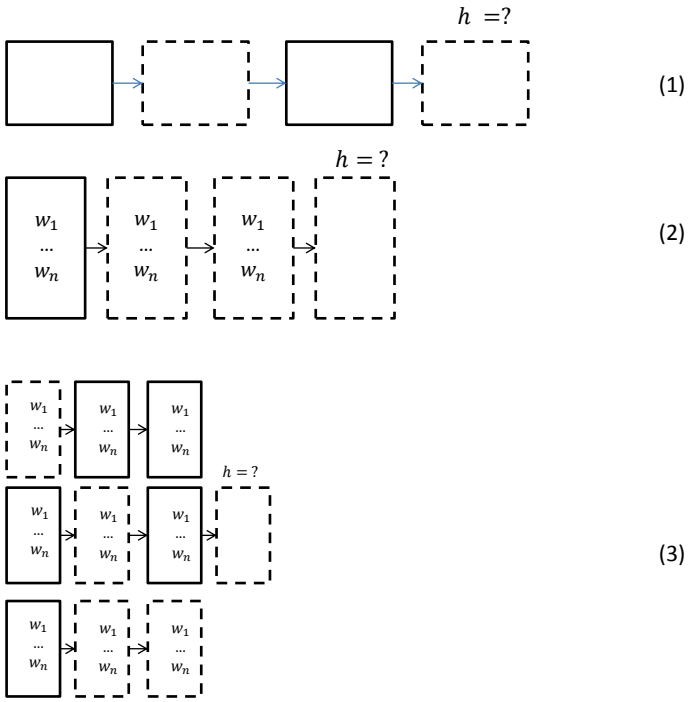

Figure 3.2: Partially observed setting.

time. We then discuss the specific features for page change prediction that we build using this framework.

3.2.1 Formal Framework

Let O be a set of objects. Let $C \subseteq O$ be a goal concept. In our problem, O is the set of pages and C is the set of changing pages. In the traditional (binary) supervised learning setup, we define the goal function $h : O \rightarrow \{0, 1\}$, where $h(o) = 1 \leftrightarrow o \in C$. Let $E = \{\langle o_1, h(o_1)\rangle, ..., \langle o_K, h(o_K)\rangle\}$, where $o_i \in O$ and $h(o_i) \in \{0, 1\}$, be a set of training examples. To extend the basic setup to include classes and features that dynamically change over time, we assume $\mathfrak{T} = \{0, 1, ..., \infty\}$ to be a discrete representation of time. We define $h : O \times \mathfrak{T} \rightarrow \{0, 1\}$ to be a *temporal goal function*, and
$E = \{\langle\langle o_1, t_1\rangle, h(o_1, t_1)\rangle, ..., \langle\langle o_K, t_n\rangle, h(o_K, t_n)\rangle\}$ to be a set of *temporal training examples*, where $o_i \in O$, $t_j \in \mathfrak{T}$ and $h(o_i, t_j) \in \{0, 1\}$.

The basic algorithm for predicting page change, as currently used in many applications [Cho and García-Molina, 2000, 2003a; Edwards et al., 2001], is prediction based on the $h(o_i, t_j)$ of past training examples of o_i. Using the previous

notation, we call this information setting the *1D setting*, as it only utilizes the class (in our case, page change) behavior.

Let $F = \{f_1, ..., f_{|F|}\}$ be a set of feature functions, where $f_i : O \rightarrow Im(f_i)$. A traditional learning algorithm takes E and F as input and produces a hypothesis $\hat{h}$ which is a good approximation of h. We define $F = \{f_1, ..., f_{|F|}\}$ to be a set of *temporal feature functions*, where $f_i : O \times \mathfrak{T} \rightarrow Im(f_i)$. That is, $f_i(o, t)$ is the value of the feature i of object o at time t. A *temporal learning algorithm* receives as input E and F and produces a temporal hypothesis $\hat{h}$.

Traditionally in machine learning, the building blocks of a classifier $\hat{h}(o)$ are the features of the object $f_1(o), \ldots, f_{|F|}(o)$, e.g., changes in a page might depend only on its features, e.g., its terms. In our extended framework, $\hat{h}(o, t)$ can use, in addition to $f_i(o, t)$, also $f_i(o, t')$ for $t' < t$. For example, the page change might depend on the change direction values over the last several days. Using the previous notation, we call this information setting a *2D setting*, as it only utilizes the features of the page itself.

Alternatively, in our framework, $\hat{h}(o, t)$ can use features of other objects, $f_i(o', t')$ for $o' \neq o, t' < t$. For example, changes in other pages from previous observable times. We also assume a relation function $R_i : O \times O \leftarrow R$ between the objects. These functions can also be considered as a weighted graph structure over the objects. Thus, $\hat{h}(o, t)$ can use features of only related objects using metrics over the structure of relations R_i. In the most general setup, $\hat{h}(o, t)$ can combine the above two approaches and use $f_i(o', t')$ for $t' < t$ and $o' \neq o$. For example, to determine whether a certain page will change tomorrow, the classifier can use various parameters of other pages over the last week. A more extended view of the features f_i, are those that take into account several objects in an aggregated feature, i.e., $f_i : O^K \rightarrow Im(f_i)$. For example, a feature that determines the content similarity of the predicted page o with the other objects that are fed as input to produce $\hat{h}(o, t)$. In the previous notation, we call this setting a *3D setting*, as it utilizes the features of the page itself and other pages as well.

3.2.2 Framework Implementation

The previous section discussed the formal framework of the information sources. We now discuss the specific features that we implement for content change prediction.

2D Features

Given the time of classification t, the goal is to predict whether page A will change at time $t + \sigma$, where σ is the prediction interval. We denote the representation of a page to be classified at time t as $A(t)$. Let l be the regression parameter,

i.e., the window size from which features are generated. For example, for $l = 2$ we create page features using its behavior over the last 2 time stamps (not to be confused with the number of examples n, which is the number of objects usually fed into a classifier). We define the following features for learning page change:

1. Frequency of change in page A during the window l. This feature is equivalent to the information captured by the baseline in Section 3.3.1.

2. Term unigram probability distribution of the words in page $A(t)$: $\{\frac{c_{i,A(t)}}{\sum_j c_{j,A(t)}}\}$ where $c_{i,A(t)}$ is the number of occurrences of the considered term (w_i) in page $A(t)$, and the denominator is the sum of number of occurrences of all terms in page $A(t)$, that is, the size of the document $A(t)$. These features are meant to capture the intuition that particular words on a page might be predictive of page change. For example, on a local organization's page the words "upcoming meeting" might be predictive of a change that notes of the meeting will soon be posted.

3. Features representing the magnitude of the change vector between the content at every time point in the window and the content of the page at time t. Formally, a set of l features representing how big of a change A experienced compared to each previous day in the window: $\{||A(t) - A(t-i)|||i \in [1 \dots l]\}$. Intuitively, the magnitude of change in content between the last observed content (t) and the content on any given day in the window may be predictive of change. For example, a stable set without change may increase the chance of no change. While a series of small changes may presage an upcoming update. This will be dependent on the particular page and these features provide the expressiveness to learn such behaviors.

4. For each pair of prediction-interval separated timesteps in the window, we define features measuring the magnitude of the change vector in page content in a single timestep. Formally, a set of l features representing how big of a change A was experiencing compared to previous days in the window: $\{||A(t-i) - A(t-i-\sigma)|||i \in [1 \dots l]\}$. Intuitively, this captures the magnitude of change typically seen given the prediction interval size.

5. Features representing whether the page changed on every time in the window. We define change of page at time t as:

$$\text{change}(A(t)) = \begin{cases} 1 & \text{if } A(t) \text{ changed} \\ 0 & \text{if } A(t) \text{ did not change} \end{cases}$$

   We consider the following features:
   $\text{change}(A(t-1)), \dots, \text{change}(A(t-l))$ representing

whether the page changed significantly on the days before the prediction. Intuitively, these features are useful when an upcoming change might be predicted by any change (or lack thereof) a fixed number of days in the past or a regularity of change/stability might indicate a continuation of the pattern.

3D Features

Additionally, we can incorporate features of other pages B on the web. Their change and features might also reflect on the change in page A. The first five types of features we define are exactly analogous to those defined in Section 3.2.2, and they are meant to capture similar motivations but with the shift that if the feature types from the same page are useful, then the same ones from a related page may prove useful. We also add two new feature types targeted at representing the relationship between the two pages. We define the following set of features for each page B:

1. Frequency of change in page B during the window l.

2. Term unigram probability distribution of the words in page $B(t)$: $\{\frac{c_{i,B(t)}}{\sum_j c_{j,B(t)}}\}$.

3. A set of l features representing how big of a change B experienced compared to previous days in the window: $\{||B(t) - B(t-i)|||i \in [1 \dots l]\}$.

4. A set of l features representing how big of a change B was experiencing compared to previous days in the window: $\{||B(t-i) - B(t-i-\sigma)|||i \in [1 \dots l]\}$.

5. Feature $\text{change}_{t-\sigma-1}(B), \dots, \text{change}_{t-l}(B)$ representing whether the page changed significantly on the days before the prediction.

6. A set of l features representing the similarity of page B to the prediction page A: $\cos(A(t), B(t))$. Intuitively, the degree of content similarity of the pages at time $t \in [t \dots t-l]$ may indicate the strength of predictivity between the pages.

7. A set of l features representing the similarity in change of page B to the content of page A. Formally, let $\delta(B)_i = B(t) - B(t-i)$, and $i \in [1 \dots l]$, thus this feature is defined as: $\cos(\delta(B), A)$. Intuitively, not only may the degree of content similarity be indicative of the relationship strength, but also the similarity between the content of B that changed and A's full content. We also consider a set of l features representing the similarity of the changes of

page B to the changes of the prediction page A: $\cos(\delta(B(t-i)), \delta(A(t-i))$ for $i \in [1 \dots l]$.

3.3 Learning Algorithms

In this section we outline the specific algorithms based on the solution framework we described in the previous section.

3.3.1 Baseline Algorithm

The basic algorithm for predicting page change, as currently used in many applications [Cho and García-Molina, 2000, 2003a; Edwards et al., 2001], is prediction based on the probability of the page to change significantly based on the past training examples. That is, $h(o_i, t_j) = 1$ with probability:

$$p(h(o_i, t_j) = 1 \mid h(o_i, t_k) \in E \text{ where } t_k < t_j \text{ and } (t_j - t_k) \leq l).$$

The parameter l is typically referred to as a window size and is a user-specified parameter.

3.3.2 Single Expert Algorithm

The single expert is an algorithm that represents the pages with a set of features. Given M time stamps, we create training examples with the features described in Section 3.2.2. For example, if we want to predict for 5 days in the future ($\sigma = 5$) and we have a total of 300 days for training, we can prepare training objects for a learner. Each object is made of features of l days, where l is the regression parameter. For example, for $l = 3$ a total of $300 - \sigma - l + 1 = 293$ examples are created to train a learner with. Given prediction interval σ, each example is labeled with the h value of the page in σ time stamps, i.e., $h(o, t + \sigma)$. Those E examples are used to train a classifier C. In our experiments we use SVM classifiers for this purpose.

During prediction, the set of the above features is extracted for the test instance and given as input to C. In our example, we are given another set of consecutive 21 days (separate from training period), from which $21 - \sigma - l + 1 = 14$ testing objects are created, on which the learner will be tested. In this work we consider the case where $\sigma = 1$ and $l = 1$, i.e., we observe the page as it is today and predict how it will be tomorrow. We then output the prediction of the classifier.

3.3.3 Algorithms Using Related Objects

In this section we discuss algorithms that consider both the page's features and features of other pages. We first present a novel algorithm that models the related pages as experts, and follows a voting scheme (Section 3.3.3) In order to investigate whether this framework is in fact advantageous we also investigate several alternative mechanisms for incorporating information from related pages: an algorithm that collects the information of the other sources into a single expert (Section 3.3.3), and algorithms that use multiple models to boost performance (Section 3.3.3).

Multiple-Experts Algorithm

In this section we propose a novel algorithm, that performs a temporal relational choice of experts. By temporal experts we mean related pages that predict change. The novel algorithm we present here takes into consideration the structure of the object-relations network, and builds relational experts. Each page will be considered an expert, and will include the features described in Section 3.2.2 and Section 3.2.2 (features are calculated in respect to page $A(t)$ – the object of classification). Each expert is trained with the target function, h, of the *relevant page*. Intuitively, each expert can predict based on the changes it experienced, its content and similarity to page A at time t and previous times, whether page A will experience a change in the future. For example, if we wish to predict a change in an Information Retrieval professor's page, we might train a classifier based on the features of the WSDM page, where the label would be whether the professor's page changed or not.

While the combination of the pages decision can be done using any ensemble combination method we use weighted majority voting (see weighting techniques in Section 3.4). Formally, given prediction interval σ, for each object $o \in O$ train a classifier using examples (with the features described in Section 3.2.2) and labeled with the h value of the page A in σ time stamps, i.e., $h(o, t+\sigma)$. Those examples are used to train a classifier C (in our experiments we use SVM classifiers for this purpose). Eventually O classifiers are trained. During prediction, for each classifier C_i (corresponding to an object o_i), the set of the above features is extracted for the test instance and given as input to C_i. We then output the combined prediction of the classifier by selecting the majority weighted vote. See full algorithm in Figure 3.3.

Relational-Union Algorithm

In many relational learning scenarios relational features are created by collecting features over related objects, which are then fed into a classifier. This type of

Procedure MULTIPLE EXPERTS($WeakLearner, O, o_{target}$)
Train:
Foreach $o_j \in O$
$E(o_j) = \{ \langle f_1^t(o_j, t_i), \ldots, f_n^t(o_j, t_i) \rangle, h(o_{target}, t_i + \sigma) | \}$
Call *WeakLearner* with $E(o_j)$,
and receive the hypothesis classifier h_j
Add h_j to the ensemble, ξ
Test: Given an unlabeled instance o_{target} at time t
Evaluate the ensemble $\xi = \{h_1, \ldots, h_{|O|}\}$ on $o_{target}(t)$
Let $V_{t,j} = \begin{cases} R_i(o_{target}, o_j) & \text{if } h_t \text{ picks class } c_j \\ 0 & \text{otherwise} \end{cases}$
be the vote given to class c_j by classifier h_t
Obtain total vote received by each class
$V_j = \sum_{t=1}^{T} v_{t,j}$ for $j \in \{0, 1\}$
Return the class that receives the highest total vote.

Figure 3.3: Temporal experts algorithm. The inputs are a weak learner, the pages O and the page to be classified o_{target}. R_i is a weighting of the relation between two objects. The prediction interval is σ.

prediction has been used widely on the web Getoor and Mihalkova [2011]. One such example is the ad-click prediction problem Richardson et al. [2007]. Here the nodes are the ads, bids and queries, and the edges are the relations between the three. The goal is to predict a certain click on an ad. Similarly here, the nodes are the pages and the edges are the hyperlinks. The goal is to predict a certain page change.

Given M time stamps, we create training examples with the features described in Section 3.2.2. Given prediction interval σ, each example is labeled with the h value of the page of prediction in σ time stamps, i.e., $h(o, t+\sigma)$. Those E examples are used to train a classifier C. In our experiments we use SVM classifiers for this purpose. During prediction, the set of the above features is extracted for the test instance and given as input to C, which provides the prediction.

Relational-Stacking Algorithms

Stacking methods use multiple models to improve prediction over just a single model. Stacking algorithms, similar to the previous models, combine many features from related objects, and then train several "base" models either by bagging or reweighting the example distribution (boosting) before combining these models in the final model output.

Bagging [Breiman, 1996] is one such method, where randomly sampled examples from the training set are used to train different base models. The base models predictions are then combined, where every model has an equal weight. Let k be the number of base models and let E be the set of examples as in Section 3.3.3. Given a training set of size $|E|$, this method generates k new training sets, each of size $|E|$, by sampling examples from the original training set uniformly and with replacement. A model is trained for each of the k training sets, and combined by majority voting. This method was shown to be useful in improving many linear models [Breiman, 1996].

An extension of this idea, is **Boosting**. Intuitively, instead of randomly choosing instances, the ensemble model is built incrementally, where every new base model is trained with the training instances that previous models did not perform well on. Let E be the examples (as mentioned in Section 3.3.3), k number of base models, and let C be a classifier. AdaBoost [Freund et al., 2003] defines a vector of weights W for each example in E. It is first initialized as:

$$W_0(j) = \frac{1}{|E|}.$$

The algorithm performs k iterations, at each iteration reweighing the examples and retraining a classifier C_i with the weighted examples. The examples are reweighed in the following way:

$$W_{i+1}(j) = \frac{W_i(j)\exp(-\alpha_t y_j C_i(x_j))}{Z_i}$$

where Z_i is a normalization factor, $C_i(x)$ is the prediction of the classifier at iteration i for example x, and

$$\alpha_i = \frac{1}{2}\ln\frac{1-\epsilon_i}{\epsilon_i}$$

is the weighted error rate at iteration i, where

$$\epsilon_i = \sum_{j=1}^{|E|} W_i(j)[y_j \neq h_t(x_j)]$$

is the error rate at iteration i. The output of the ensemble model is a combination of the k base models created at the end of each iteration weighted by the error α_i they experienced.

3.4 Expert Selection

The number of pages on the web, and therefore the number of temporal experts (or number of 3D features as used by the relational algorithm), is extremely large.

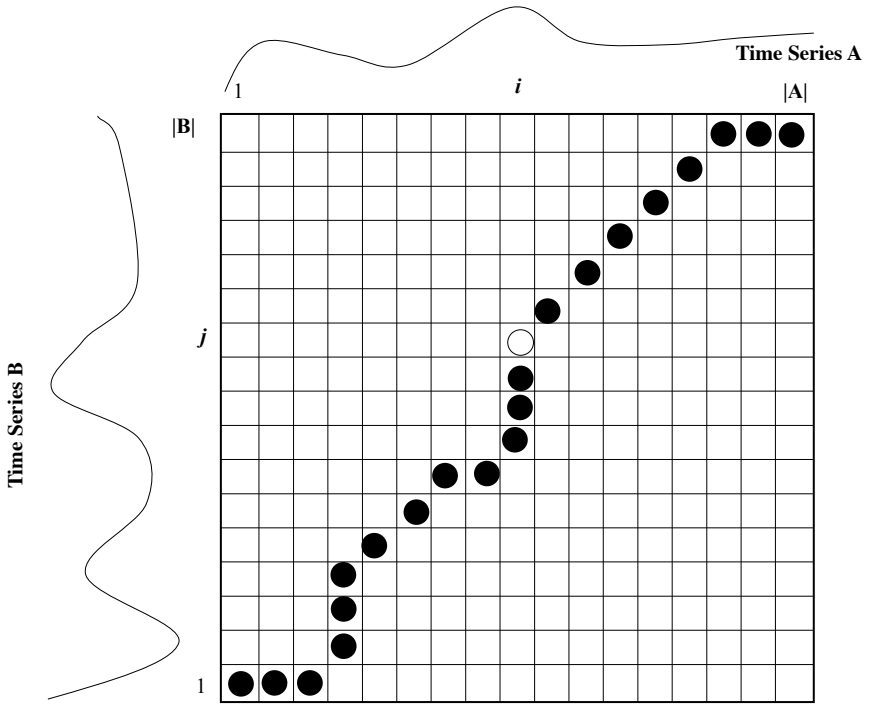

Figure 3.4: DTW finds the best alignment between time series A and B by maintaining the illustrated cost matrix $C(i, j)$.

Due to this computational considerations, only a subset of the pages in O can be considered and monitored for every page. Based on our formalism from Section 3.2.1, we assumed a relation function $R_i : O \times O$ between the objects. We explore several such functions that choose the experts to monitor for every page – what we refer to as the *Expert Selection* problem.

Let p be the page whose h value is being predicted, and let e be a candidate expert. Given a predefined constant k of pages the system is capable of monitoring, the goal of an expert selection algorithm is to find the k experts such that the prediction accuracy of page h values is the highest. Notice that simple cross validation is a hard task due to its computation complexity (selecting all possible set of experts). We next present several heuristic algorithms for estimation of these experts.

3.4.1 Graph Distance KNN

We hypothesize that pages that are linked to one another in the web graph have higher probability of experiencing similar changes. In this case, we identify the experts for a page by using in-links and out-links and finding the closest neighbor

using a single step using an undirected representation of the links. In other words, the experts are selected using a K-nearest-neighbors procedure in the web graph, and we weight each page equally ($R_i(o, neighbor(o) = 1)$. We used a web graph based on a sample of 22 million web pages (reached from good reputation sites seed) and 345 million edges, extracted from a web crawl performed in 2006 (for more information see [Leskovec et al., 2007]), at the same time the content crawling was performed.

3.4.2 Content Similarity KNN

We hypothesize that content similarity is a potential catalyst of mutual page changes. As the expert and the page discuss similar topics, a change in one might lead to a change in the other. Each page can be represented in some vector space with a predefined distance metric, and then a k nearest algorithm can be applied to find the most similar experts in content. In this work, we represent each page as a bag of words and the distance metric chosen is the cosine similarity (i.e., $R_i(o, neighbor(o) = cos(text(o), text(neighbor(o)))$.

3.4.3 Temporal Content Similarity KNN

The above mentioned approaches can be considered as finding k nearest neighbors in some static snapshot (either the web graph or web content). In this section, we also leverage the content change dynamics of the pages. The change patterns of the expert and the page over time are essential information to understand their mutual temporal behavior. Intuitively, a good predictive expert of change will have some correlation in content not just at a static time, but also will tend to trend in change together with the predicted page across a range of time – although one may lag/lead the other. Furthermore, the change correlation will be in similar topics. That is, the temporal correlation of the content-similarity of the change behavior of the two pages is high. We explore the monitoring of content similarity using cross-correlation and dynamic time warping (DTW). Similar temporal semantic modeling techniques have been used successfully in the past to improve semantic relatedness tasks Radinsky et al. [2011]. Cross-correlation is a measure of similarity of two time series as a function of some time-lag applied to them, and defined as

$$\frac{\sum_i [(ts_1[i] - E(ts_1)) \cdot (ts_2[i-d] - E(ts_2))]}{N \cdot \sigma_{ts_1} \sigma_{ts_2}},$$

where $d = 1 \ldots N$ is a possible delay, N is the minimal time series length, σ is the time series variance, and E is the means of the corresponding series. The delay with the best correlation is chosen as the similarity. DTW (see Figure 4.6), on

the other hand is a measure of similarity of two time series as a function of both time-lag and speed. For example, DTW will give high score to the similarity of page A and page B, even if A changes more than B, but they still have a similar change pattern, i.e., B is influenced by some of the changes in A. DTW is a dynamic programming algorithm, that tries to find an optimal match of the time series by stretching and compressing section of those time series. It holds a cost matrix C, where $C(i, j)$ is the minimum distance between two points of the time series i and j (see illustration in Figure 3.4). Cross-correlation can be thought of a simple case of DTW, where only diagonal steps are allowed in the matrix.

We define *temporal content similarity* over time as the weighted combination of the correlation over the words in the two documents. Let a document d be represented by a sequence of words $w_1, \ldots, w_m$. Let $t_1, \ldots, t_T$ be a sequence of consecutive discrete time points (e.g., days). We define the *dynamics* of a word w in a document d to be the time series of its frequency of appearance in d:

$$\mathrm{Dyn}(w, d) = \langle \mathrm{tfidf}(w, d(t_0)), \ldots, \mathrm{tfidf}(w, d(t_T)) \rangle$$

where $d(t_i)$ us the snapshot of document d at time t_i.

Let Q be a correlation function between time series (such as DTW or cross-correlation). We define a distance metric between p and e as follows:

$$\mathrm{Distance}(p, e) \quad = \quad \sum_{i=1}^{k} \omega(w_i)[Q(\mathrm{Dyn}(w_i, p), \mathrm{Dyn}(w_i, e))]$$

where $\omega(w_i)$ is a weighting function for every word, which can represent how dominant the word was in past changes, how many times it changed, etc. In our work we consider a simple version where the function represents the significance of the word for the document in the last snapshot of page: $\omega(w_i) = \mathrm{tfidf}(w_i, p(T))$. As before, the closest k experts can now be selected using the distance metric Distance, and the weights are the values of their correlation.

3.5 Experimental Setup

We implemented the 1D,2D and 3D algorithms and conducted a variety of experiments to test their performance, compared to the state of the art in the prediction literature.

3.5.1 Dataset

We evaluate our method on a real-world data collection. The data consists of 54,816 pages crawled on an hourly basis for a 6 months period (2007-06-30 till

Procedure DTW(ts_1, ts_2, C)
$n \leftarrow Min(|ts_1|, |ts_2|)$
$dtw(ts_1, ts_2) \leftarrow$ new $[|ts_1| \times |ts_2|]$
For $i = \{1 \dots n\}$
$dtw(i, 1) \leftarrow dtw(i-1, 1) + C(i, 1)$
$dtw(1, i) \leftarrow dtw(1, i-1) + C(1, i)$
For $i = \{1 \dots n\}$
For $j = \{1 \dots n\}$
$dtw(i, j) = |ts_1(i) - ts_2(j)| +$
$Min(dtw(i-1, j), dtw(i, j-1), dtw(i-1, j-1))$
Return $dtw(n, n)$

Figure 3.5: Dynamic time warping algorithm (DTW)

2007-12-31). We collected URLs from the logs of logged-in users of the Microsoft Live Search toolbar. The pages to crawl were not random, but rather driven by user visitation patterns. We are interested in pages whose change is going to affect the satisfaction of many users, and therefore we sampled pages from those visited by 612,000 people for a five week period. For more information see the work by Adar et al. [Adar et al., 2009]. In order to prevent trivial predictions, we filtered out pages that exhibit a constant behavior, e.g., pages that either change all the time or don't at all. Additionally, for pages below a change frequency threshold (whose change patterns appear to be random), one can recrawl them on an infrequent schedule paying an amortized cost for having a stale copy. Let change$(o, t) = 1$ if the page o changes at time t, and change$(o, t) = 0$ otherwise. We require $|\{t|\,\text{change}(o, t) \neq \text{change}(o, t+1)\}| \geq \alpha$, where in our experimentation $\alpha = 0.3$, i.e., that page changes behavior at least in 30% of the data.

3.5.2 Empirical Methodology

In order to perform correct statistical evaluations of the algorithms over sufficient data, the empirical evaluation over each dataset is performed as follows:

- In each prediction, for each algorithm, we use the previous n days as training examples (we experimented on several n values). We construct each example where the task is to predict what happens on the following day ($\sigma = 1$). In order to construct each example the two days previous are used for the dynamic change features ($l = 1$).
- We repeated the predictions $|O|$ (number of pages) times. Each time, a different page was used as the target of prediction.

1D features (Baseline)	2D features (Single Expert)	3D features (Multiple Experts)
54.84%	62.93%	**72.72%**

Table 3.1: Prediction accuracy average on a sliding window. Results statistically significant using a t-test ($p < 0.05$) when compared to Multiple Experts.

For each training set of size n and the consecutive test day, all algorithms train on all but last time point in the training set, and classify o_i with the resulting classifier. The classifier always takes as input the last ($l = 1$) time points. We use the same SVM polynomial classifier (degree=3) to implement all of the learning approaches to control for variance. All parameters of the learning algorithm were calibrated on the training data. The accuracy of a prediction algorithm for a page is averaged over all sliding windows (i.e., over $t_n - (t_1 + \sigma)$ predictions).

3.6 Results

In this section we discuss the different experiments we have conducted. We first start with the main result showing that the use of information found in other pages provides higher precision for page change prediction. We then proceed with different analysis experiments of our algorithm.

3.6.1 Main Result

The accuracy results of all algorithms are presented in Table 3.1, with $k = 20$ experts selected using the cross-correlation method, trained with training size of $n = 50$. These particular values, as well as the number of experts and selection of cross-correlation as the primary temporal selection method, were chosen over a small sample of data. We investigate sensitivity of performance of these methods to varying choices of the amount of history used (n) as well as the number of experts selected (k) in Sections 3.6.2. Each value in the table represents an average performance over all objects. The results provide evidence that the temporal experts outperform the single experts and baseline, and that the single experts outperform the baseline (both results are statistically significant, as measured by a paired t-test).

We further experiment with different training sizes (Figure 3.6). There was a statistical significant difference in all periods of times in favor of the multiple expert approach, and the single expert approach outperformed the baseline over all training sizes. The behavior of the algorithm exhibits almost constant behavior over the different training sizes, indicating that a crawler only monitoring the last

Relational	Relational Stacking		Multiple
Union	Boosting	Bagging	Experts
62.91%	63.44%	63.58%	**72.72%**

Table 3.2: Prediction accuracy using different relational algorithms. Results statistically significant using a t-test ($p < 0.05$) when compared to Multiple Experts.

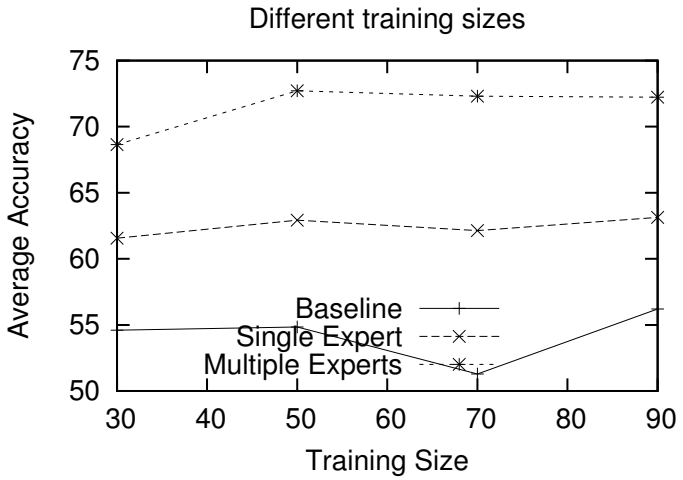

Figure 3.6: Comparison of Baseline, Single Expert and Multiple Experts over different training sizes. Each point in the graph represents the prediction accuracy (y label) of every of the method using the specified size of training examples (x label).

30 days of every page can perform with high accuracy. All learning algorithms performed poorly with less than 30 training examples, and therefore are not shown in the graph.

3.6.2 Parameter Selection Analysis

In this section we analyze the different multiple experts algorithms, the different expert selection method and how the number of experts affects performance.

Different Expert Algorithms

The accuracy results of all algorithms are presented in Table 3.2, with 20 experts selected using the cross-correlation method, trained with a training size of 50. We observe that the multiple experts algorithm we have suggested in this work reaches the highest performance (significant as measured by a paired t-test), providing evidence that this type of mechanism is appropriate for combining information

Graph Distance	Content Similarity	Temporal Content Similarity	
		Cross-Correlation	DTW
64.40%	67.27%	**72.72%**	70.33%

Table 3.3: Prediction accuracy average of multiple experts using the different selection algorithms. Results statistically significant using a t-test ($p < 0.05$) when compared to DTW.

from multiple sources. We conclude that, although all algorithms were trained using the same set of features, just combining the features without the taking into account the structure, whether one trains a simple classifier, bags, or boosts, is not enough.

Expert Selection Methods

In this section we present the results for the different expert selection methods (see Table 3.3). We performed a paired t-test on the results, and found statistical significance difference between the cross-correlation approach and the non-temporal approaches with p-value $p < 0.05$. However, we found no statistical significance between the different temporal content approaches. All expert selection algorithms show statistically significantly higher performance than the performance of methods using 2D and 1D features.

The results provide an interesting view point on the change behavior and its indicators. Although graph distance may seem an intuitive measure for the web, it provides little impact for change prediction. Correlation over time in the same topics is the more beneficial approach in those cases.

Number of Selected Experts

In this section we investigate the performance of the multiple expert algorithm as a function of the number of selected experts. We show the results in Figure 3.7. While there is some sensitivity in performance to the number of experts selected, the gains over the single expert using the 2D features is robust across a range of values for number of experts. Up to 20 experts it seems the system gains more precision by the addition of more experts. However, after reaching this performance peak the addition of other features seems to be adding unnecessary noise to the system. Additionally, we see that even a small number of experts has enough predictive power for predicting page content change.

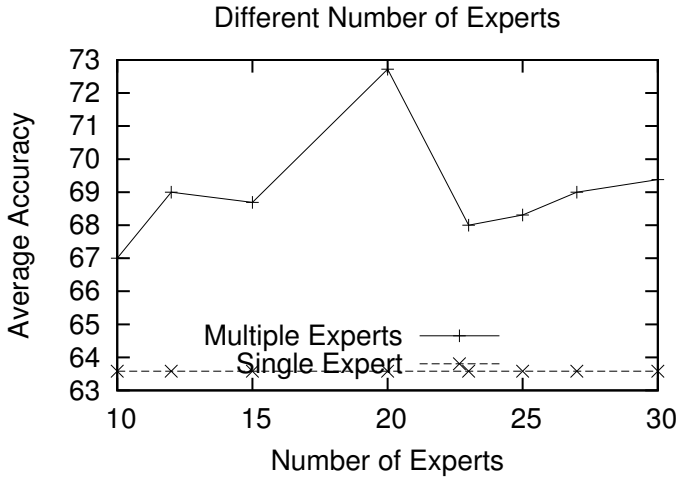

Figure 3.7: Number of experts as a function of accuracy. Each point in the graph represents the prediction accuracy (y label) of the multiple experts algorithm using the specified size of experts (x label).

3.6.3 Application to Crawling: Maximizing Freshness

We show an application of our work to a crawling task in which the goal is to maximize the freshness of the indexed pages. It is standard practice to assume that the set of crawled pages is fixed and to assume a maximum crawl rate r [Olston and Najork, 2010]. We set a crawling limit of 10% of the total number of pages to simulate a setting where the number of pages to be crawled far exceeds the crawling resources, and adopt a scheduling policy that targets to re-visit pages as closely as possible to their change frequency by flipping a biased coin based on the estimated change probability. This probability is estimated by fitting logistic regression models (Platt recalibration [A. Smola and Schuurmans, 1999]) to the outputs of the SVM classifier for the multiple and single experts, and for the baseline it is estimated based on the page frequency. We follow a batch crawling scenario, therefore the classifiers are not re-trained during the crawling period. Once a page is selected to be crawled it is used to update the training of all the methods. We compare the overall freshness of the index using different methods of estimating this probability as a function of the number of days the crawl was performed. Figure 3.8 shows the performance according to average freshness as a function of the number of observed crawling days (*i.e.*, the amount of data available to train the classifier). Both the 2D and 3D methods improve over the baseline with significant gains in freshness occurring after observing 50 days of

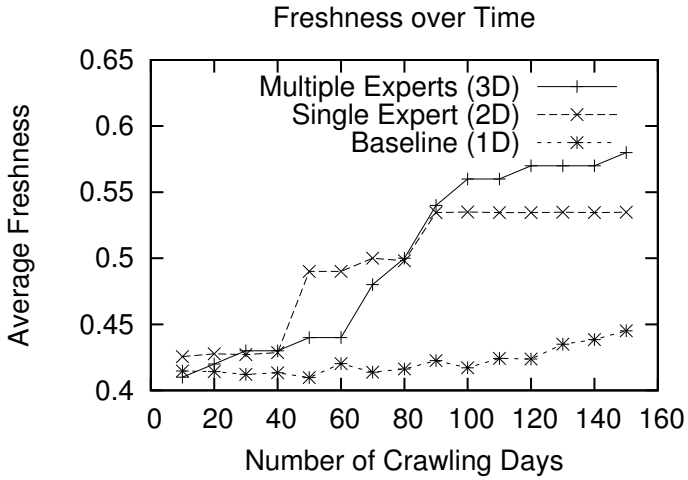

Figure 3.8: Freshness as a function of crawling days

crawl data. The 3D method shows performance on par with the 2D method after 80 days with superior and significant performance after 100 days of crawl data. Overall, the results show the superiority of the 3D algorithm over the baseline after a small number of days were sampled and over the 2D approach after a moderate number of days, showing that the approach is also applicable for when some of the features are missing (as presented in the partially-observed setting).

3.7 Discussion

In this section we give some qualitative examples to provide insight into the multiple experts and expert selection algorithms. We then also discuss how our methods can be efficiently calculated and scaled.

3.7.1 Qualitative Examples

In this section we provide some qualitative examples of the prediction results.

An example where related page expert selection outperforms the 1D and 2D approaches is for the URL `http://www.thekirkreport.com/archives.html`. This is a URL irregularly giving updates on stock trading. The content similarity KNN method (Section 3.4.2) chose the page of the American first credit union `http://www.amerfirst.org/`, that provides updates on rates, credits and loans. And indeed, in many of the times that the kirkreport had changes, it followed a significant update of all the rates on the amerfirst site. However,

using simple content similarity pages as experts might also yield unsatisfactory result for change prediction. For example, the experts selected for the Southern Methodist university (`http://smu.edu/`, were other university sites such as the university of Windsor `http://www.uwindsor.ca`, and the university of Vermont `http://www.uvm.edu/`. Although all the above university pages change frequently, their change is usually due to updating university news, such as awards and local news. Therefore using those experts as indicators for the change in smu was unhelpful. As we use the page itself as an expert of its own change, in this case the ensemble algorithm weighted the smu page very high, yielding similar results to the 2D approach.

Those type of errors were easily fixed by the temporal content similarity method using cross-correlation (Section 3.4.3), that not only takes content into account, but also the actual correlation across time. For example, the best correlating pages for `http://www.boxofficeindia.com` were `http://www.musicnmovies.com` and `http://www.bollywoodworld.com`, as the pages experience change in the same topics when a new movie comes out to the theater.

The graph distance similarity (Section 3.4.1) usually selects pages with no clear connection to the temporal dynamics; therefore the experts extracted using this method are not satisfactory. For example, for the URL `http://www.mtv.com/music/video` the experts selected were relevant pages such as a radio site `http://www.ksidradio.com`, but also irrelevant pages, such as a wrestling site `http://prwrestling.com`. For comparison, the cross-correlation experts for this same URL were `http://www.mtv.com/ontv/dyn`, `http://www.imdb.com/Sections/Quotes` and `http://www.newsoftheworld.co.uk/story_pages`. The first URL is an aggregate of new content in MTV, the second indicates a new popular movie that indicates new YouTube films about it, the last one is a correlation indicating that after specific news events there will be a new YouTube movie.

3.7.2 Efficient Calculation

In this work we showed that information from related pages strongly contributes to better page change detection. We investigated numerous ways of identifying such related pages – either by graph distance, content similarity or temporal content similarity. We have showed empirically that the temporal content similarity methods outperform the first two. However, identifying those related pages can be computationally expensive. Fortunately, several methods have been proposed in the literature, that can be used to speed up identifying such pages in an efficient way. For example, Keogh and Pazzani [Keogh and Pazzani, 2000] discuss several ways of efficiently identifying similar time series in a large database. The method reduces the size of every time series by providing a method to divide it into frames, on which a time-series similarity algorithm can later be performed.

The algorithm was applied using DTW algorithm, but the extension to the application of cross-correlation is similar.

Additional concern in such large scale prediction tasks is that storing snapshots from multiple iterations is prohibitively expensive. However, the cross-correlation method for finding temporal experts, that had the best performance in our empirical study, can be computed online without the need to actually store the previous snapshots. As more information arrives the correlation can be updated with only the correlation of the new snapshot. We saw in the empirical study that gains in performance asymptote after 30 days of historical data. Some classifiers can be trained online as well, preventing the need to keep the last snapshots. Additionally, in our experiments the object used for prediction is composed of only the past two snapshots ($l = 1$), and we did not see a significant higher gain when using bigger l values. This result is akin to the result by Ali & Williams [Ali and Williams, 2003], that showed that simply using only the previous two snapshots of a page to determine when to recrawl significantly reduces crawling compared to recrawling all recently changed pages.

3.8 Conclusions

In this chapter we tackled the problem of predicting significant content change on the web. Our results shed light on how and why content changes on the web, and how it can be predicted. We explored several information sources that can be used to predict such change – the commonly used frequency of change, the page content and related pages content. We have seen that the addition of the page content improves prediction when compared to simple frequency-based prediction. Additionally, the addition of information of related pages content improves over the usage of page's content alone. We experimented with several methods for selecting related pages, and have shown that related pages that correlate over time in their content change contain valuable information for page content change prediction. We have investigated several ways of combining information from related pages, and proposed a novel algorithm that outperformed the common learning mechanisms.

Chapter 4

Predicting the Real World using Web Dynamics

Varian and Choi [2009] show in their work that Web queries reflect the real world. They present several economical phenomenons and how they can be inferred directly from user Web behavior. We believe that Web dynamics can show much more. In this chapter, we show that not only our economical perception of the world can be derived from Web behavior, but our entire perception of the concepts of the "real world". We discuss a common task of semantic relatedness of texts, which reflects human perception of concept's similarity, and how it can be enriched using Web dynamics.

The ability to quantify semantic relatedness of texts underlies many fundamental tasks in natural language processing, including information retrieval, word sense disambiguation, text clustering, and error correction. Previous approaches to computing semantic relatedness used various linguistic resources, such as WordNet, Wikipedia, or large-scale text corpora for methods like Latent Semantic Analysis (LSA). Yet all of these approaches essentially considered the underlying linguistic resource as a static collection of texts or concepts. In this chapter we argue that there is an additional source of rich information about semantic relatedness of words, which can be revealed by studying the patterns of word occurrence over time.

Consider, for example, words such as "war" and "peace". While these words are clearly related, they might rarely be mentioned in the same documents. However, they are likely to be mentioned roughly around the same time (say, in different articles posted during the same day, or in adjacent days). In this work, we use the New York Times archive spanning over 130 years. For each word, we construct the time series of its occurrence in New York Times articles. We posit that if there is a correlation between the time series of two words, then the meanings of the two words are related.

In principle, there are a number of cases when temporal information could offer a complementary source of signal, which is not captured by other models. Synonyms (that is, words with similar meanings) are rarely used in the same article since an author usually sticks to one set of terms, yet they can be used by different authors in different articles describing the same events. Looking at their coordination in time allows us to leverage the opinions of multiple authors collectively. As another example, consider pairs of words that form stock phrases, such as "luxury car." Taken individually, the two words in each pair have very different meanings, and are likely to be judged as such by existing methods. On the other hand, these words are indeed related, and the frequency of their use over time exhibits non-trivial correlation. Especially interesting are pairs of words that have implicit relationships such as "war" and "peace" or "stock" and "oil," which tend to correlate in frequency of use over time. Figures 4.1 and 4.2 depict these correlations in time. The proposed method, Temporal Semantic Analysis, captures such correlations, and is able to better estimate semantic relatedness than methods that only use static snapshots of linguistic resources.

Our contributions here are threefold. First, we propose to use temporal information as a complementary source of signal to detect semantic relatedness of words. Specifically, we introduce *Temporal Semantic Analysis* (TSA), which leverages this information and computes a refined metric of semantic relatedness. Second, we construct a new dataset for semantic relatedness of words, which we have judged with the help of Amazon's Mechanical Turk service. In contrast with the previous standard benchmark, WS-353, our new dataset has been constructed by a computer algorithm (also presented below), which eliminates subjective selection of words. We make the new dataset publicly available for further research in the field. Finally, empirical evaluation shows that TSA exhibits superior performance compared to the previous state of the art method (ESA), and achieves higher correlation with human judgments on both datasets.

4.1 Temporal Semantic Analysis

We propose Temporal Semantic Analysis (TSA), which is composed of two novel components: a new approach, described in this section, for *representing* the semantics of natural language words, and a new method, described in Section 4.2, for *computing* the semantic relatedness between words.

Our method is based on associating each word with a weighted *vector of concepts*. Such concepts can be derived from crowd intelligence folksonomies such as Wikipedia, co-tagging in Flickr, or from online bookmarking services such as del.icio.us. This is similar to recent semantics approaches such as ESA

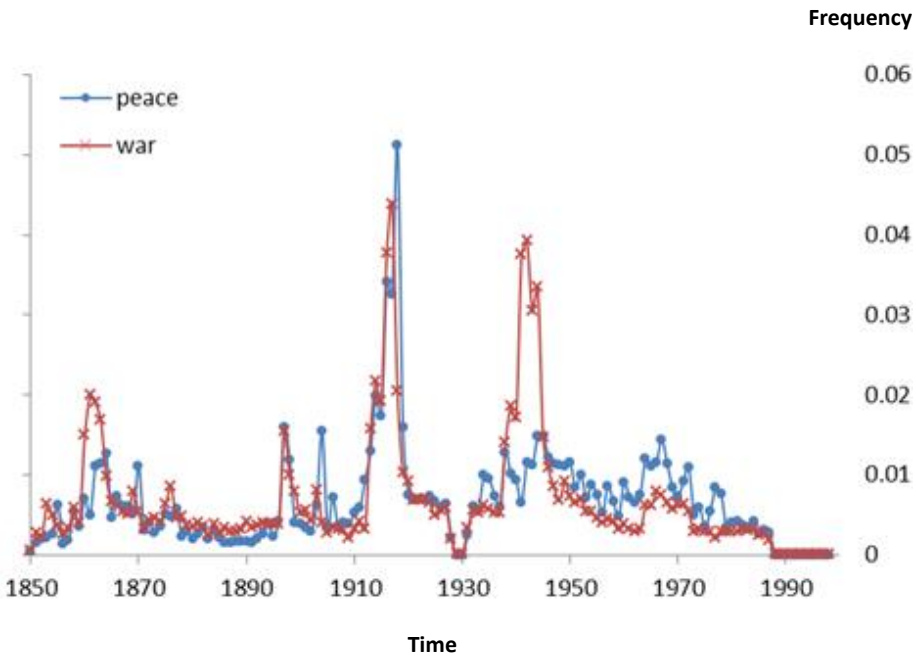

Figure 4.1: Time series (1870-1988) of the words "war" (red) and "peace"(blue). The words correlate over time.

[Gabrilovich and Markovitch, 2007]. However, while ESA uses a *static* representation of each concept, we use the concept *dynamics*—its behavior over time, represented by the time series of the concept occurrence. Thus, instead of representing a word with a vector of unit concepts, vectors of time series are manipulated, where each time series describes concept dynamics over time. Our hypothesis is that *concepts that behave similarly over time, are semantically related.* Such a rich representation of words (adding the extra temporal dimension) could facilitate the discovery of implicit semantic relationships between the original words. As we will show experimentally, the naive approach of directly computing temporal correlation between words (without the concept vector representation) is not effective.

Thus, our TSA method consists of three main steps:

1. **Represent words as concept vectors**: using a concept repository of choice (e.g., Wikipedia or Flickr image tags), represent a word as a set of associated concepts with weights (Section 4.1.1).

2. **Extract temporal dynamics for each concept**: using a corpus of choice (e.g., New York Times archive), quantify concept occurrence for each time

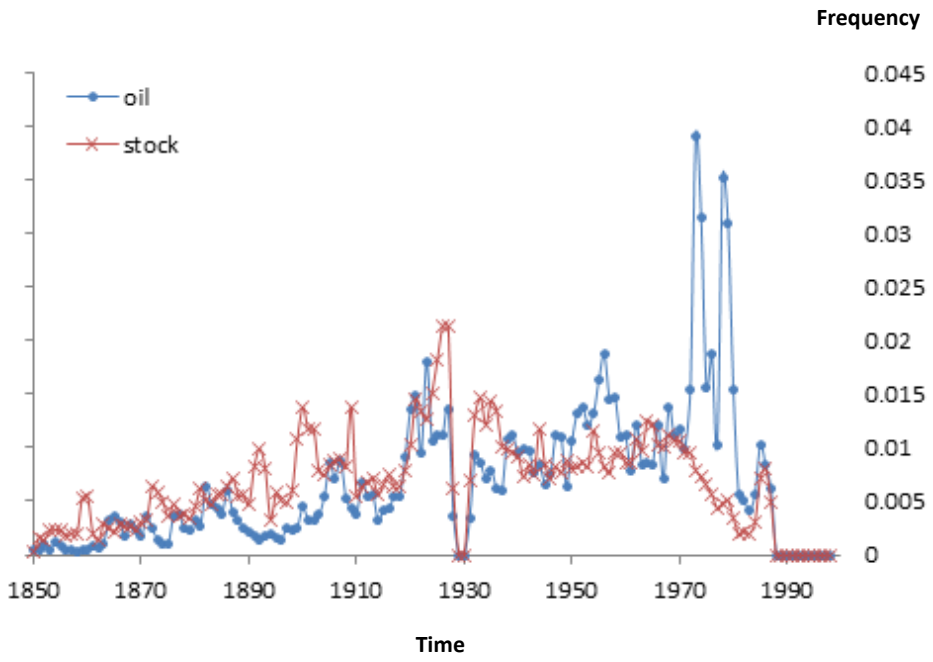

Figure 4.2: Time series (1870-1988) of the words "stock" (red) and "oil"(blue). The words correlate over time.

period (e.g., a day) and build its time series (Section 4.1.2).

3. **Extending static representation with temporal dynamics**: finally, scale each concept's time series according to the concept's original weight from item 1 above (Section 4.1.3).

4.1.1 Representing Words as Concept Vectors

In our representation, each word is mapped into a vector of concepts — a *concept vector*. For each concept a static weight is computed. We consider several such representations over multiple folksonomies:

1. Wikipedia Concepts — Wikipedia is among the largest knowledge repositories on the Web, which is written collaboratively by millions of volunteers around the world, and almost all of its articles can be edited by any user. Wikipedia is available in dozens of languages, while its English version is the largest of all with more than 500 million words in over three million articles. Vector space models based on this ontology have been used by many works for semantic relatedness [Gabrilovich and Markovitch, 2007; Strube

and Ponzetto, 2006]. In those representations, each entry in the concept vector is a TFIDF-based function of the strength of association between the word and the concept in Wikipedia.

2. Flickr Image Tags — Flickr is an online image hosting community. Photo submitters have the option to add metadata to each image in the forms of natural language tags. This feature enables searchers to find images related to specific topics. The natural concepts in this representation are the Flickr tags.

3. Del.icio.us Bookmarks — del.icio.us is a social URL bookmarking service, with the possibility to search and explore new bookmarks. The service had, by the end of 2008, more than 5.3 million users and 180 million unique bookmarked URLs. Del.icio.us users can tag each of their bookmarks with free text, and we use these tags as concepts.

4.1.2 Temporal Concept Dynamics

Let c be a concept represented by a sequence of words $wc_1, \ldots, wc_k$. Let d be a document. We say that c *appears* in d if its words appear in the document with a distance of at most ε words between each pair wc_i, wc_j, where ε is a proximity relaxation parameter (in the experiments we set $\varepsilon = 20$). That is, a concept *appears* in a document if there is a window of size ε where all the concept words appear. For example, for the concept c — "Great Fire of London" — we say that the c appears in a document d, if the words "Great", "Fire", "of", "London" appear in the document with a distance of at most ε between each word.

Let $t_1, \ldots, t_n$ be a sequence of consecutive discrete time points (e.g., days). Let $H = D_1, \ldots, D_n$ be a *history* represented by a set of document collections, where D_i is a collection of documents associated with time t_i. We define the *dynamics* of a concept c to be the time series of its frequency of appearance in H:

$$Dynamics(c) = \langle \frac{|\{d \in D_1 | appears(c,d)\}|}{|D_1|}, \ldots, \frac{|\{d \in D_n | appears(c,d)\}|}{|D_n|} \rangle \quad (4.1)$$

In the experiments described in this chapter we used New York Times articles since 1870 for history. Each time point is a day, and the collection of documents associated with a day is the set of articles appearing on that day.

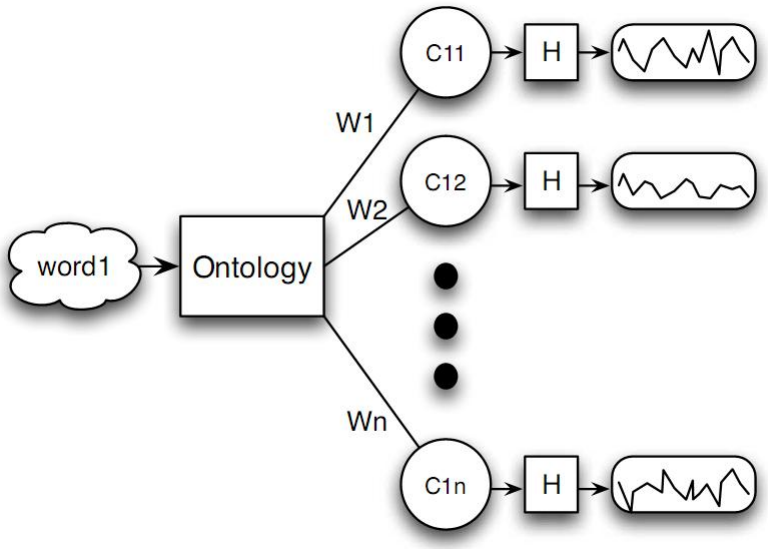

Figure 4.3: Each word is represented by a weighted vector of concept time series (produced from a historical archive H). The weight w_i of each concept corresponds to the concept "importance" w.r.t. the original word.

4.1.3 Extending Static Representation with Temporal Signals

Our approach is inspired by the desire to augment text representation with massive amounts of temporal world knowledge. Hence, we represent a word as a weighted mixture of concept time series, where the weights correspond to the concept "importance" w.r.t. the original word (Figure 4.3).

In common semantic representations (such as ESA [Gabrilovich and Markovitch, 2007]) a word is represented as a weighted vector of concepts (derived from Wikipedia articles). In ESA, each vector entry contains a single (static) TFIDF weight, which expresses the strength of association of the word and the concept. Our TSA method extends ESA so that each entry in the vector corresponds to a time series, computed as described above.

4.2 Using TSA for Computing Semantic Relatedness

To compute semantic relatedness of a pair of words we compare their vectors (as defined in Section 4.1.3) using measurements of weighted distance between multiple time series, combined with the static semantic similarity measure of the concepts. This approach, therefore, integrates both temporal and static semantic behavior of the words.

4.2.1 TSA-based Semantic-Relatedness Algorithm

The ESA method for computing semantic relatedness is based on the assumption that related words share highly-weighted concepts in their representations. The TSA approach does not assume so. We only assume that highly-weighted concepts of the related words are *related*.

Suppose we are trying to find the relatedness between words t_1 and t_2. Assume that t_1 is mapped to a set of concepts $C(t_1) = \{c_1^1, \ldots, c_n^1\}$ and t_2 is mapped to $C(t_2) = \{c_1^2, \ldots, c_m^2\}$. Suppose we have a function Q that determines relatedness between two individual concepts using their dynamics (as defined in Section 4.1.2). Assuming w.l.o.g $n \leq m$, we can define the relatedness R between t_1 and t_2 as the maximal sum of pairwise concept relatedness over all ordered subsets of size n of $C(t_2)$:

$$R(t_1, t_2) = \max_{j_l \in (1 \ldots \binom{m}{n})} \sum_{l=1,\ldots,n} Q(c_l^1, c_{j_l}^2) \tag{4.2}$$

This exhaustive search over all possible pairs is, however, infeasible. Therefore we take an alternative greedy approach, which is formally described in Figure 4.4. The procedure at each step finds a pair of time series with the highest relatedness Q (line 5 in the algorithm), removes them and proceeds (lines 7 and 8). Iteratively, the relatedness $R(t_1, t_2)$ is computed as the sum of relatedness of the matching concepts (line 6). This procedure complexity is $O(n \cdot m \cdot max(|ts|))$, where $|ts|$ is the length of the time series representing the concepts.

4.2.2 Similarity Between Individual Time Series

The relatedness Q between two concepts is determined by comparing their dynamics. Our basic assumption is that related concepts correlate in their temporal behavior. For comparing the concepts associated time series, we use two existing methods for measuring time series similarity — cross correlation and dynamic time wrapping (DTW).

Procedure SEMANTIC RELATEDNESS(t_1, t_2)
(1) $C(t_1) = \{ts_1^1, \ldots, ts_n^1\}$
(2) $C(t_2) = \{ts_1^2, \ldots, ts_m^2\}$
(3) $R(t_1, t_2) \leftarrow 0$
(4) **Repeat** $Min(m, n)$ **times**
(5) $\langle \hat{ts_1}, \hat{ts_2} \rangle = arg\max_{\langle ts_1, ts_2 \rangle \in C(t_1) \times C(t_2)} Q(ts_1, ts_2)$
(6) $R(t_1, t_2) \leftarrow R(t_1, t_2) + Q(\hat{ts_1}, \hat{ts_2})$
(7) $C(t_1) \leftarrow C(t_1) \backslash \{\hat{ts_1}\}$
(8) $C(t_2) \leftarrow C(t_2) \backslash \{\hat{ts_2}\}$
(9) **Return** $R(t_1, t_2)$

Figure 4.4: A greedy algorithm for computing the semantic relatedness between two words. The procedure assumes the availability of a function Q that determines relatedness between a pair of time series ts_i associated with two concepts.

Cross Correlation

In statistics, cross correlation is a method for measuring statistical relations, e.g., measuring similarity of two random variables. A common measurement for this purpose is the Pearson's product-moment coefficient which is defined as:

$$\begin{aligned} \mathrm{corr}(X,Y) &= \frac{\mathrm{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X - E(X))(Y - E(Y))]}{\sigma_X \sigma_Y} = \\ & \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \overline{x})^2}\sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \overline{y})^2}} \end{aligned} \tag{4.3}$$

In signal processing, cross-correlation is used as a measure of similarity of two signals as a function of a time-lag applied on one of the signals — a variation of the Pearson coefficient to different time delays between two time series in Figure 4.5. An innate characteristic of this measure is identification of similar time series in volume, with consideration of time shifts. In our representation, where words are represented as time series, words whose frequencies correlate in volume, but with a time lag, will be identified as similar. When we wish to evaluate the correlation of the two words' time-series, we compare the time series starting from the first time point they both started appearing, until the time point when one of the words stopped appearing. For example, the word "computer" did not appear during the 1800s, and started to appear only around 1930. Therefore, when we compare it to the word "radio", we calculate the cross correlation only during the period starting at 1930.

Procedure CROSS CORRELATION(ts_1, ts_2)
(1) $similarity(ts_1, ts_2) = 0$
(2) $cov(ts_1, ts_2) = 0$
(3) **For** $delay = \{-delay_{min} \dots delay_{max}\}$
(4) **For** $index = \{0 \dots Min(|ts_1|, |ts_2|)\}$
(5) $delayedIndex = index + delay$
(6) $cov(ts_1, ts_2) \leftarrow cov(ts_1, ts_2) + (ts_1[index] - E(ts_1)) * (ts_2[delayedIndex] - E(ts_2))$
(7) $corr@delay(ts_1, ts_2) \leftarrow \frac{cov(ts_1, ts_2)}{N \cdot \sigma_{ts_1} \sigma_{ts_2}}$
(8) $similarity(ts_1, ts_2) \leftarrow Max(similarity(ts_1, ts_2), corr@delay(ts_1, ts_2))$
(9) **Return** $similarity(ts_1, ts_2)$

Figure 4.5: Time series cross correlation

Dynamic Time Warping

The DTW algorithm [Bellman and Kalaba, 1959] measures the similarity between two time series that may differ in time scale, but similar in shape. In speech recognition, this method is used to identify similar sounds between different speakers whose speech speed and pitch might be different. The algorithm defines a local cost matrix $C \in R^{|ts_1| \times |ts_2|}$ of two time series ts_1 and ts_2 as

$$C_{i,j} = \|ts_1[i] - ts_2[j]\|, i \in \langle 1 \dots |ts_1| \rangle, j \in \langle 1 \dots |ts_2| \rangle \tag{4.4}$$

where $\|ts_1[i] - ts_2[j]\|$ is a distance metric between two points of the time series.

Given this cost matrix, DTW constructs an alignment path that minimizes the cost over this cost matrix. This alignment p is called the "warping path", and defined as a sequence of points pairs $p = (pair_1, \dots pair_k)$, where $pair_l = (i, j) \in \langle 1 \dots |ts_1| \rangle \times \langle 1 \dots |ts_2| \rangle$ is a pair of indexes in ts_1 and ts_2 respectively. Each consequent pair preserves the ordering of the points in ts_1 and ts_2, and enforces the first and last points of the warping path to be the first and last points of ts_1 and ts_2. For each warping path p we compute its cost as $c(p) = \sum_{l=1}^{k} C(pair_l)$. The DTW is defined to be the minimum optimal warping path

$$DTW(ts_1, ts_2) = min\{c(p) | p \in P^{|ts_1| \times |ts_2|}\} \tag{4.5}$$

where P are all possible warping paths. A dynamic programming algorithm (similar to the one in Figure 4.6) is usually applied to compute the optimal warping path of the two sequences.

This similarity measurement, as opposed to time series cross-correlation distance (cf. Section 4.2.2) is much more flexible, hence we decided to experiment with it as well.

Procedure DTW(ts_1, ts_2, C)
(1) $n \leftarrow Min(|ts_1|, |ts_2|)$
(2) $dtw(ts_1, ts_2) \leftarrow$ new $[|ts_1| \times |ts_2|]$
(3) **For** $i = \{1 \dots n\}$
(4) $\quad dtw(i, 1) \leftarrow dtw(i-1, 1) + c(i, 1)$
(5) $\quad dtw(1, i) \leftarrow dtw(1, i-1) + c(1, i)$
(6) **For** $i = \{1 \dots n\}$
(7) $\quad$ **For** $j = \{1 \dots n\}$
(8) $\quad\quad dtw(i, j) = \|ts_1(i) - ts_2(j)\| + Min(dtw(i-1, j), dtw(i, j-1), dtw(i-1, j-1))$
(9) **Return** $dtw(n, n)$

Figure 4.6: Dynamic time warping algorithm

Temporal Weighting Function

As the meaning of the words changes over time, more recent concept correlation are more significant than past correlation. Therefore, when measuring the distance between two individual time series, higher weights to recent similarities should be given. We apply several linear and non-linear weighting functions to the above time series distance functions (see Section 4.4.2). Let $f(i, j)$ be such a function, whose parameters are two time points i, j. Thus, we modify DTW definition of $\|ts_1(i) - ts_2(j)\|$ to

$$\|ts_1(i) - ts_2(j)\| \cdot f(i, j) \tag{4.6}$$

and the covariance definition in the cross-correlation distance now changes to

$$\begin{aligned} cov(ts_1, ts_2) \leftarrow & cov(ts_1, ts_2) + \\ & f(i, j) \cdot [(ts_1[index] - E(ts_1)) \cdot (ts_2[delayedIndex] - \\ & -E(ts_2))] \end{aligned} \tag{4.7}$$

We described how our TSA method represents words as concepts (Section 4.2.1), and how the temporal dynamics of the concept usage over time can be used to compute semantic relatedness (Section 4.2.2). We now turn to the experimental evaluation of our approach.

4.3 Experimental Setup

We implemented our TSA approach using the New York Times archive (1863-2004). For each day we had an average of 50 abstracts of articles, which after parsing yielded 1.42 GB of texts with a total of 565,540 distinct words. In this

section we describe the methodology we used in our experiments and then describe a novel algorithm for automatically creating benchmarks for word relatedness tasks.

Both ESA and TSA were implemented on the concepts extracted from the folksonomies presented in Section 4.1.1, and therefore use the same vector representations. This will allow us to isolate the performance of the temporal dimension in the TSA semantics.

4.3.1 Experimental Methodology

Methods compared: We compare our algorithm and representations to the state of the art semantic representation — Explicit Semantic Analysis (ESA), which has been shown to be significantly superior to other approaches [Gabrilovich and Markovitch, 2007]. This approach projects words into a high-dimensional space of concepts derived from Wikipedia. Using machine learning techniques, it represent the meaning of a word as a weighted vector of Wikipedia-based concepts. Each concept in the vector is weighted by relevance to the word. Assessing the relatedness of words in this space is done by utilizing cosine distance – a conventional metric of comparison of high-dimensional vectors.

Evaluation metrics: As in prior published studies, in our evaluation we use Spearman correlation coefficient to compare the predicted relatedness scores with human judgements. The comparison is applied on both our algorithm and representations and the current state of the art.

Statistical Significance: We compare the rank correlation coefficient of our method, $rank_1$, to the competitive methods rank coefficient, $rank_2$, and calculate statistical significance, using the following standard formula:

$$p = 0.5 \cdot ErrorFunction(\frac{|z_1 - z_2|}{\sqrt{2} \cdot \sqrt{\frac{2}{N-3}}}) \tag{4.8}$$

where N is the number of word pairs the dataset, $z_i = 0.5 \cdot \ln(\frac{1+rank_i}{1-rank_i})$, and $ErrorFunction(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$ is the standard Gauss error function.

4.3.2 Dataset Construction Algorithm

Evaluating word relatedness is a natural ability humans have and is, therefore, considered a common baseline. To assess word relatedness, we use the WS-353 benchmark dataset, available online [Finkelstein et al., 2002], which contains 353 word pairs. Each pair was judged, on average, by 13-16 human annotators. This dataset, to the best of our knowledge, is the largest publicly available collection of

this kind, which most prior works [Gabrilovich and Markovitch, 2007; Yeh et al., 2009; Zesch and Gurevych, 2010; Zesch et al., 2008] use in their evaluation.

As an effort to provide additional evaluation data in this problem domain, we created a new dataset[1] to further evaluate our results upon. We present a principled method to create additional datasets, as opposed to the WS-353 benchmark where the word pairs were extracted manually. We propose to draw the word pairs from words which frequently occur together in large text domains. The relatedness of these pairs of words is then evaluated using human annotators, as done in the WS-353 dataset.

Selecting word pairs to evaluate: To create a balanced dataset of both related words and unrelated words, we applied the following procedure: Let W be a set of all words in the New York Times news articles. As we wish to compare between entities, we intersect this collection with entities extracted from DBpedia. We further proceed with removing stop words and rare words (words appearing less than 1000 over the entire time period), and stemmed the remaining words. We annotate this collection as W'. For each word pair $(a_i, a_j) \in W' \times W'$, their point-wise mutual information (PMI) is computed over the entire set of the articles, i.e., a group G of all possible word pairs, ordered by their PMI values

$$\begin{aligned} G &= \{(a_1, b_1), \ldots, (a_n, b_n) \mid PMI(a_i, b_j) \\ &\quad \leq PMI(a_{i+1}, b_{j+1}), (a_i, b_j) \in W' \times W'\} \end{aligned} \tag{4.9}$$

where PMI is defined as:

$$PMI(a_i, a_j) = \log \frac{p(a_i, a_j)}{p(a_i)p(a_j)} \tag{4.10}$$

Eventually, given a pre-defined number n of desired test pairs, every $\frac{|W' \times W'|}{n}$-th pair from the G' ordering is chosen. Formally, we construct the final set

$$D = \{g_{1+i \cdot \frac{|W' \times W'|}{n}} | g_j \in G, j \leq |W' \times W'|\} \tag{4.11}$$

Intuitively, this process performs a stratified sampling, containing both frequently and infrequently co-occurring words, with decent coverage of the entire spectrum of co-occurrence values (as measured by mutual information).

Obtaining human ratings from Amazon MTurk workers: The human "ground truth" judgements were obtained by using the Amazon's Mechanical Turk workers, in batches of 50 word pairs per assignment, resulting in 280 word pairs labeled overall. Up to 30 workers per batch were assigned, with the average of 23 MTurk workers rating each word pair, on average. Ten (distinct) pairs from WS-353 dataset were injected into each batch, in order to provide a calibration

[1] http://www.technion.ac.il/ kirar/Datasets.html

baseline to discard poor-quality work. Additionally, a simple "captcha" requiring to solve a simple math problem was given to each worker. As a result, the work of the annotators with ratings that correlated less than 50% on the WS-353 subset of the batch, or those that failed the "captcha" was discarded (approximately 7% of the submitted ratings were discarded through this procedure).

4.4 Experimental Results

We first report the main experimental results comparing TSA to ESA on the WS-353 and MTurk datasets described above. Then, we analyze the performance of TSA in more detail on the WS-353 dataset to gain more insights into the effects of the different system parameters.

4.4.1 Main Results

In this section we compare the results of TSA to known similarity measurements. The section first provides empirical evidence that temporal signals contribute to measuring semantic relatedness of words, and then we show that our representation as a vector of concepts combined with temporal data outperforms previous temporal similarity techniques.

TSA vs. ESA

The comparison results of TSA on the WS-353 dataset are reported in Table 4.1. TSA results shown in the table are computed using cross correlation with a quadratic weighted function as the distance metric between single time series (the parameter tuning was performed on a separate validation set).

Table 4.1: TSA algorithm vs. ESA (WS-353 dataset)

Algorithm	Correlation with humans
ESA-Wikipedia [Gabrilovich and Markovitch, 2007]	0.75
ESA-ODP [Gabrilovich and Markovitch, 2007]	0.65
TSA (Section 4.2)	**0.80**

As reported in Table 4.1, TSA performs significantly better compared to the ESA-Wikipedia approach, with $p < 0.05$.

We also evaluate the performance of ESA-Wikipedia and TSA, on the additional dataset we created (we refer to it as the MTurk dataset). The results are presented in Table 4.2. Again, TSA performs substantially better than ESA, confirming that temporal information is useful on other datasets.

Table 4.2: TSA algorithm vs. state-of-the-art (MTurk Dataset)

Algorithm	Correlation with humans
ESA-Wikipedia [Gabrilovich and Markovitch, 2007]	0.59
TSA (Section 4.2)	**0.63**

TSA vs Temporal Word Similarity

Some works [Chien and Immorlica, 2005] proposed measuring semantic similarity of queries through temporal correlation analysis alone – without expending to a vector of semantic concepts. We therefore compare to additional two baselines: Word-Similarity using cross correlation and Word-Similarity using DTW as the distance measurement of the time-series of the two words. The results using the WS-353 and Mturk dataset can be seen in Table 4.3. In both datasets TSA significantly outperformed the baselines. This suggests that temporal vector similarity combined with static similarity is essential.

Table 4.3: TSA algorithm vs. temporal word similarity (WS-353 dataset)

Algorithm Dataset	WS-353	MTurk
Word-Similarity (cross correlation)	0.51	0.56
Word-Similarity (DTW)	0.59	0.58
TSA (Section 4.2)	**0.80**	**0.63**

4.4.2 TSA Performance Analysis

This section analyzes the performance of TSA for varying settings to gain more insights into the advantages and limitations of the TSA method.

Word Frequency Effects

To further analyze the performance of our algorithm we conducted experiments to test on which type of word pairs our algorithm outperforms the state of the art to this end. We chose to focus on word frequency. We investigated whether our algorithm performs better on frequent or rare words. We measured frequency in both domains — Wikipedia and New York Times. In order to evaluate the joint frequency of a pair of words, we combine their frequency by three types of measurements: minimum frequency of the two words, average, and maximum frequency of the two words. We divide the word pairs into three buckets, each

containing an equal number of data points. We compute Spearman correlation separately in each bucket.

The results for the minimum criteria for the New York Times and Wikipedia corpora are reported in Tables 4.4 and 4.5, respectively. Similar results were obtained for the average and maximum frequency measurements. The results show that TSA performs significantly better than ESA on low-frequency words. This can be attributed to the fact that ESA is based on statistical information about words and concepts, which requires sufficient number of occurrences. Low-frequency words do not have enough statistical data, hence any additional signal, such as the temporal behavior of the words, can improve the performance.

Table 4.4: Grouping word pairs by NYT word frequency (WS-353 dataset)

Type of Bucket	ESA Correlation with humans	TSA Correlation with humans
Low	0.73	**0.82**
Medium	0.74	0.76
High	0.76	0.79

Table 4.5: Grouping word pairs by Wikipedia word frequency (WS-353 dataset)

Type of Bucket	ESA Correlation with humans	TSA Correlation with humans
Low	0.72	**0.79**
Medium	0.68	0.68
High	0.78	0.81

The results on the Mturk Dataset comparing ESA-Wikipedia, and TSA are reported in Table 4.6. While the absolute values of the TSA and ESA correlations with humans are lower, the trend persists: TSA significantly outperforms ESA, particularly on words with low frequency. The lower absolute values are likely due to increased level of noise in the MTurk ratings, despite performing best-of-practice filtering of poor-quality MTurk work [Snow et al., 2008], and as explained in Section 4.3.2.

Size of Temporal Concept Vector

In this subsection we experiment with several sizes of the temporal concept vector in several different natural representations. In many of the folksonomy domains presented in Section 4.1.1, we are able to obtain only vectors of about 10 concepts (based on API limitation in Flickr and Del.icio.us), i.e., for each word we are

Table 4.6: Grouping word pairs by Wikipedia word frequency (Mturk dataset)

Type of Bucket	ESA Correlation with humans	TSA Correlation with humans
Low	0.52	**0.61**
Medium	0.50	0.48
High	0.77	0.79

not able to produce all the words and their co-occurrence weight, but only the word's related tags. Due to this limitation, a traditional cosine measurement cannot be computed between those partial vectors — as each vector contains different concepts. We define a *size of a concept vector* to be the number of concepts. The main advantage of the distance measurement we defined in Section 4.2.1 is the ability to measure distance between vectors with different concepts representation, and even vectors of different sizes. We ran the experiments on various vector sizes. We deduce from the results (as appear in Table 4.7) that the optimal vector size is 10. Additional improvements for larger vector sizes might be achieved with additional feature selection.

Table 4.7: Effect of concept vector size on performance (WS-353)

Vector Size	5	10	50	100
Correlation with humans	0.78	**0.80**	**0.80**	0.79

Time Series Distance Functions

In this subsection we experiment with several distance functions, that are applied during the measurement of the semantics distance of the temporal concept vectors. Cross correlation outperforms DTW in each setting, where TSA with cross-correlation performance is 0.80, and with DTW it drops to 0.74. This indicates that, for the purpose of measuring similarity of concept's vectors, correlations in time series volume are more significant than measuring general similarity in time series structure (as in DTW).

Temporal Weighting Functions

Several weighting functions can be applied on the words' time series to produce higher weighting to more recent correlations (as we discussed in Section 4.2.2). In this work, we define several variations for a weighing function $f(t_1, t_2)$. This function receives two time points of two time series, and is used to weigh the

distance between the time-series at these points. The functions we experiment upon are:

1. Constant Weighting Function: $f(t_1, t_2) = Constant$, which weighs all time points equally.

2. Power Weighting Function: $f(t_1, t_2) = (Max(t_1, t_2))^n$ which is a power model of weight, in which volume differences in more recent time points are weighted higher based on the power of the function. We have experimented on $n = 1, 2$.

3. Exponential Weighting Function: $f(t_1, t_2) = e^{Max(t_1,t_2)}$ which is an exponential model of weight, in which volume differences in recent time points are weighted exponentially higher.

The results of the performance for the TSA algorithm (with cross correlation distance function over WS-353) are presented in Table 4.8. The results provide evidence for the need to weigh the recent changes in time series distance measurement higher than the ancient changes. While linear, quadratic, and exponential temporal weighting functions perform similarly, the quadratic performs best, and we use it for all the experiments described in this chapter. A few examples to

Table 4.8: Effect of temporal weighting function

Temporal Weighting Function	Correlation with humans
Constant	0.70
Linear	0.79
Quadratic	0.80
Exponential	0.80

illustrate those changes in performance can be seen in Table 4.9. It is clear from the rankings presented in the table, that quadratic weighting yields more significant correlation with human ranking than the constant weighting function. The correlation of such words, such as "Mars" and "water" in 1900 should be weighted differently from the correlation they exhibit in 2008, when NASA images suggested the presence of water on Mars.

4.5 Discussion

In order to gain more intuition on which cases TSA approach should be applied, we provide real examples of the strengths and weaknesses of our methods

Table 4.9: Temporal weighting influence

Word 1	Word 2	Humans Rank	TSA-Const Rank	TSA-Quadratic Rank
Mars	water	46	210	94
peace	plan	102	220	108

compared to the state of the art ESA method. The results are derived from the application of the TSA algorithm with cross correlation and a quadratic weighting function as the distance metric between single time series.

4.5.1 Strengths of TSA

Synonyms are the first type of words for which the TSA method seems to outperform the ESA method. The reason for that is that synonyms have similar patterns of occurrence over time, as writers in the news corpus tend to use them interchangeably. Therefore, the two synonyms time-series in the same corpus strongly correlate over time. On the other hand, ESA represents each word as a vector of concepts - where the weight is the TFIDF value. For each concept's Wikipedia article the number of distinct authors is limited, and therefore, the language model, and, as a consequence, the use of different synonyms is quite limited. For this reason, the TFIDF values of the synonyms in the ESA representation tend to be quite different. A sample of those cases can be seen in Table 4.10, where we present for each pair of synonyms the ranking given by human judgements, the ESA rank and the TSA rank. The rankings are based on the rank of the similarity of the pair of words out of the 353 pairs in the WS-353 dataset.

Table 4.10: Synonyms

Word 1	Word 2	Human Rank	ESA Rank	TSA Rank
asylum	madhouse	338	61	336
coast	shore	347	232	341
boy	lad	337	198	291
problem	challenge	209	74	252

As our method also captures co-occurrences of words in a single article (as we construct time-series aggregated over all articles on a certain date), phrases can also be identified well. ESA represents each word as a vector of Wikipedia concepts, weighted by the TFIDF of the word in the concept's article. Therefore,

when measuring similarity of "hundred" and "percent' the similarity score is quite low - as the words appear in different articles and acquire completely different meanings in different contexts. Therefore, word phrases like "hundred-percent" are not identified well by ESA. More of those examples are presented in Table 4.11.

Table 4.11: Word Phrases

Word 1	Word 2	Humans Rank	ESA Rank	TSA Rank
luxury	car	189	341	235
hundred	percent	247	78	166
game	series	166	276	151

Implicit relations are one of the differentiating strengths of the TSA representation and the new distance metric we presented. For example, causality relations, such as "summer causes draught", are easily detected using correlation of the words' time-series. Relations of "type-of" (such as canyon is a type of landscape) are also relations we have found to be common when TSA outperforms ESA. We attribute that to the fact that many words in Wikipedia are associated to the general concepts (in our example "landscape") and therefore, when measuring the distance between the concepts' TFIDF vectors, the relation of each sub-object (such as "canyon") declines. Table 4.12 presents additional examples of pairs belonging to these relations and the ranking of human judgments, ESA and TSA algorithms for the WS-353 dataset.

Table 4.12: Implicit Relations

Word 1	Word 2	Humans Rank	ESA Rank	TSA Rank
closet	clothes	296	180	297
summer	drought	237	86	282
disaster	area	172	44	206
cup	tableware	217	7	283
cup	liquid	146	23	173
canyon	landscape	263	131	253
tiger	jaguar	296	201	302

4.5.2 Limitations of TSA

Although we have seen many results in which TSA performs better than ESA, we also present in this work some examples in which TSA performs worse.

One of the strengths of the algorithm sometimes also serves as its weakness. Although this phenomenon is not too common, TSA identifies very complex implicit relations, which are not always straightforward to humans. For example, the correlation between "drink" and "car". In the news, many times alcohol drinking correlates with car accidents, however humans tend not to find them related at all. Another representative example is "psychology" and "health". These words are considered very related by humans, however no true correlation in the news was found between the two words. More information about these examples can be seen in Table 4.13.

Table 4.13: Complex Implicit Relations

Word 1	Word 2	Humans Rank	ESA Rank	TSA Rank
drink	car	50	40	203
psychology	health	239	268	107

Some problems of our representation arise from the corpus we have selected to represent the concept's temporal behavior. The corpus is, unfortunately, sparse in certain topics — mostly specific topics such as technology and science (see Table 4.14). Therefore, correlations between words such as "physics" and "proton" are not identified well. A possible solution for this problem would be to add other sections of the New-York-Times news (such as sports, technology and science), and weigh the words frequency by the appearance in each one of those sections (so small sections will not be "discriminated"). Considering adding additional temporal corpus like blogs, tweets and so on, might also be useful. Unfortunately, we did not have access to this kind of data at the time.

Table 4.14: News Corpus Bias

Word 1	Word 2	Humans Rank	ESA Rank	TSA Rank
physics	proton	306	298	31
network	hardware	316	244	94
boxing	round	271	326	106

Chapter 5

Predicting the Future using Web Dynamics

In the previous chapters, we presented methods for predicting human and Web agents interactions using Web dynamics. Additionally, we presented evidence that this dynamics reflects the human perception of the "real world". In this chapter, we go even further, and present an algorithm for utilizing Web dynamics for future event prediction in the real world.

Many organizations invest significant efforts in trying to predict events that are likely to take place in the near future. Such predictions can be beneficial for various purposes, such as planning, resource allocation and identification of risks and opportunities. Predicting global events in politics, economics, society, etc. is a difficult task that is usually performed by human experts possessing extensive domain-specific and common-sense knowledge. Is it possible to design an algorithm that will automate this process?

Many events are hard to predict due to the fact that their occurrence is spread over a long time period, in a very tangled net of relations and mutual influence. However, some of them share a common pattern. Events that indicated other events in the past, might do so again as history repeats itself. Some of the events have preliminary signs. Identifying these signs may enable us to predict the events themselves.

Global events usually draw the attention of Web users, triggering many of them to submit queries related to that events. We assume that the popularity of terms appearing in such queries peaks when the event occurs. These spikes can be used as supporting evidence for the occurrence of the event.

In this chapter we present a novel method, PROFET, that mines large-scale web resources to predict terms that are likely to appear in the news of the near future. Specifically, we predict 100 terms that will prominently appear in the news up to one week from now. The main resource used by our method is a

search-query history archive (specifically, in this work we use *Google Trends*). In order to predict whether an event will appear in tomorrow's news, we examine if the terms representing today's events (extracted from today's queries) indicated this event in the past. This is done by analysis of patterns in user queries for these terms.

We test our algorithm by examining if the terms it predicts indeed appear in the news. We compared its performance to a baseline method which assumes that the news of today will be the news of tomorrow and found our algorithm to be significantly better, especially for longer prediction periods.

Our main contributions are: First, we introduce a new method for prediction of global future events using their patterns in the past. Second, we present a novel usage of aggregated collection of search queries. Finally, we introduce a testing methodology for evaluating such news prediction algorithms.

5.1 Predicting using Patterns in Web Search Queries

In this work we obtain history of user queries from two main sources:

1. **Google Trends** is a service that provides charts representing the popularity of given search terms over time.

2. **Google Hot Trends** is a service that presents the 100 top searched queries on a certain day, that deviate the most from their historic search pattern. The service also provides related searches for each of the top terms (also called Google Related Terms service).

5.1.1 Formal framework

Let $W = \{w_1, w_2, ..., w_k\}$ be a set of terms characterizing events. Let $D = \langle d_1, ..., d_n \rangle$ be an ordered set of days. Assume that for each term w_i we are given a binary vector $g(w_i) = \langle d_1^i, ..., d_n^i \rangle$. Intuitively, $d_j^i = 1$ indicates that on day d_j the term w_i appeared prominently.

This definition is suitable for a one day prediction, i.e. prediction of a 1 interval (tomorrow's events). The same definition can be generalized for a prediction of k interval:

We define prediction for a k days interval (events that will occur in k days): A term w_j **indicates term** w_i **in an interval k**, denoted by $w_j \xrightarrow{k} w_i$, if : $P(d_t^i = 1) + \Delta < P(d_t^i = 1 | d_{t-k}^j = 1)$, where t is uniformly drawn from the interval $[1, n]$ and $0 < \Delta < 1$.

To make the formal setup more practical we make the following simplifications:

1. We conclude if a term w_i *prominently appeared* on day d_t ($d_t^i = 1$) by testing for peaks in its frequency of search, specifically the peaks from its corresponding *Google Trends*' chart.

2. Due to complexity considerations, we do not address the entire set $W \times W$ to find indications. We test for indications only from the subset $Salient \times Related(Salient) \subset W \times W$, specifically in this work we used *Google Hot Trends* to determine saliency, and *Google Related Trends* to determine relatedness.

5.1.2 Peak Detection

The raw data for our peak detection is $TermChart(w_i) = \langle f_1^i, ..., f_n^i \rangle$, where $f_j^i \in [0, 1]$ represents the normalized value of the search volume of the term w_i on day $d_j \in \langle d_1, ..., d_n \rangle$.

We assume that events occur when a "deep maximum" is present, that is, people search for the term abnormally more than usual. Therefore, we must define criteria that will separate "events' peaks" from local noise.

The algorithms first extracts local maximum and local minimum points from $TermChart(w_i)$. Each maximum point m has at most two neighboring minimum points. We consider a maximum point a peak if $m > \Delta_1$ and the difference between the max point and the lowest of its neighboring minimum points is above Δ_2.

After extracting the peaks of w_i, we can construct $g(w_i) = \langle d_1^i ... d_n^i \rangle$, where $d_j^i = 1$ if a peak was observed on day d_j.

5.1.3 The PROFET Algorithm

The goal of the PROFET algorithm is to predict which terms will peak in k days.

The algorithm returns a list of terms with their associated weights. A higher weight means that the algorithm has a stronger belief that it will appear in k days. The weight of each candidate term (which belongs to the union of the related terms) is computed as the sum of indication weights for all terms salient today. When $w_1 \xrightarrow{k} w_2$, the indication weight of w_2 given w_1, where w_1 is one of today's salient terms and w_2 is the prediction, is calculated based on:

- Probability of a peak in w_2 to occur k days after a peak in w_1, i.e. $P(d_t^2 | d_{t-k}^1)$.

- How salient the term w_1 is today. Where *Saliency* of a term is based on how strongly the frequency of search for this term today deviates from its historic search pattern.

Procedure PREDICTION(k, day)
$W^S \leftarrow$ MOSTSALIENT(day)
$W^R \leftarrow \{\bigcup Related(w) | w \in W^S\}$
Return $\{\langle w, \sum_{t \in W^S} \text{W-PREDICTION}(t, w, k)\rangle | w \in W^R\}$

Procedure W-PREDICTION(w_1, w_2, k)
$N = |TermChart(w_1)|$
$g(w_1) \leftarrow \langle d_1^1, ..., d_n^1\rangle)$, $d_j^1 = \begin{cases} 1 & j \in Peaks \\ 0 & otherwise \end{cases}$
$g(w_2) \leftarrow \langle d_1^2, ..., d_n^2\rangle)$, $d_j^2 = \begin{cases} 1 & j \in Peaks \\ 0 & otherwise \end{cases}$
$p(w_2) \leftarrow |Peaks(g(w_2))|/N$
$hits \leftarrow |\{\ d\ |g(w_1)[d-k] = 1 \text{ and } g(w_2)[d] = 1, k \leq d \leq N\}|$
$p(w_2|w_1) \leftarrow hits/|Peaks(g(w_1))|$
$Saliency(w_1) \leftarrow \frac{TermChart(w_1)[today]}{Average(TermChart(w_1))}$
$IndicationWeight \leftarrow p(w_2|w_1) \cdot Saliency(w_1)$
Return $\begin{cases} IndicationWeight & p(w_2|w_1) - p(w_2) > \Delta \\ 0 & otherwise \end{cases}$

Figure 5.1: The Prediction algorithm

We call this method of computing the strength of prediction of one term based on another *W-Prediction*. For a more formal description of the algorithm see Figure 6.6.

5.2 Experimental Evaluation

We have implemented the PROPHET algorithm and evaluated its performance. In the learning phase we trained seven predictors, $p_1, \ldots, p_7$, for prediction intervals of one to seven days. The training data included history of 5 years of user queries obtained from *Google Trends*.

For testing we used a very large dataset of news obtained from **Google News**. It contains around 5,000,000 items daily and covers most of the available news sources (4,500 news sources). Our learner did not have access to this data.

The task we set was to predict 100 terms that will *significantly appear* on a day: Let t be a given term. A term t **significantly appears** on day d if $\frac{1}{365}\sum_{i=d-365}^{d-1} frequency(t,i) < \frac{1}{\mu} frequency(t,d)$.

In our tests, $\mu = 20$. Thus, a term *significantly appears* on a certain day if its frequency of appearance on that day is substantially bigger than its average daily appearance in the past year.

To test a predictor p_i, we run it on 25 consecutive days (June 6th, 2007 – July 1st, 2007). For each input day d, the task of the p_i was to output 100 terms it predicts to *significantly appear* in day $d+i$. We then used the real data to count what fraction of the 100 terms indeed *significantly appeared*. Thus, the principle performance measurement of our test was precision at 100.

Precision at k documents retrieved is a common measure of information retrieval performance, when recall on the set of relevant documents is hard to estimate. In our case, retrieving all news on a certain day was indeed hard. This measure also reflects the utility of retrieval results more directly.

We had difficulties finding appropriate baseline algorithms to compare with, as this task was not tried before. The baseline method we eventually chose is the "today = tomorrow" algorithm. This method predicts that the same terms that are prominent today will be also prominent tomorrow (or in k days, when the prediction interval is k). The baseline method works quite well, since events mentioned in the news tend to last more than one day. An additional method we suggested is one that randomly chooses terms from the related terms, rather than use the indication mechanism.

Before we started the experiments, we had preformed parameter tuning of Δ_1 and Δ_2 using a time range different than the one used for testing (1-18 of January, 2008). The performance peaks with $\Delta_1 = 0.2$ and $\Delta_2 = 0.1$, which we therefore select as the default values.

5.2.1 The performance of PROFET

Graph 5.2(a) shows the performance of PROFET and the baseline on the test set with prediction interval (k) of 1 to 7. Each point in the graph represents the average of the precision (at 100) over the 25 test days for the given prediction interval. In Figures 5.2(b) and 5.2(c), we give a more focused view on a 1 day prediction and a 7 days prediction respectively. The X-axis and Y-axis represent the precision-at-100 of the baseline method and PROFET respectively. Each point represents the precision of each algorithm on one test day.

The pattern of difference in performance is quite clear. As the prediction interval increases, the advantage of PROFET becomes more apparent. Indeed, a paired t-test shows that the advantage of PROFET is significant ($p < 0.05$) for all values of k, except $k = 1$. The baseline algorithm has shown to be a strong competitor to PROFET for a prediction of one day. This is no surprise, as news does not fade away quickly. However, for prediction for longer periods, we observe a significant advantage of our algorithm.

5.2.2 Qualitative results

To have a better understanding of the prediction abilities of PROFET, we list here several predictions it made.

1. **Hurricane and Gas Prices** – On September 9th 2008, several terms related to Galveston Hurricane which hit Texas appeared as salient. One of the related terms this day was "gas prices" (was related to the search term "gas buddy"). In the past, queries about "Hurricane" were followed 3 days later by queries about "gas prices", probably due to the fact that hurricanes in the past, such as Katrina and Rita in 2005, affected the gas prices dramatically, and people search queries conveyed this situation. Due to the fact that 9 salient terms from that day indicated the term "gas prices" this result got a high weight. PROFET predicted that this term will appear on September 12th on the news. And indeed, it appeared in the news, due to a record in gasoline prices.

2. **Apple Share** – On January 15th, terms related to Steve Jobs keynote on Apple's new products were salient in the search queries that day. PROFET revised the related terms, one of which was "AAPL" (Apple share). In the past queries about "Steve Jobs" indicated the term "AAPL" in an interval of 7 days, and thus PROFET predicted it would appear in the news a week later. Indeed, a week later, "AAPL" has been unusually dominant in the news in several economy reviews about Apple.

5.2.3 Random indicator vs PROFET

One potential critique of PROFET could be that the main power of the algorithm is embedded in Google's related terms, and the indication mechanism is redundant. To evaluate the importance of the indication mechanism to the performance of PROFET, we tested a version of PROFET without it. This *No-Indication* algorithm will not conduct any checking of indications, but rather randomly output 100 of the related terms. Figures 5.3(a) and 5.3(b) compare the performance of the *No-Indication* algorithm with that of PROFET and show that indeed the indication process is crucial. All the results were found to be statistically significant ($p < 0.05$).

5.2.4 W-Prediction vs Cross-Correlation

The motivation for the development of our *W-prediction* method was the lack of sufficient statistics for using more traditional correlation methods such as cross correlation. For example, Chien and Immorlica [2005] have investigated the idea

of finding semantically-related search-engine queries based on the correlation coefficient of their frequency functions. In this subsection we experiment with a version of PROFET where the w-prediction component is replaced by cross correlation (for a fixed delta). Figure 5.3(c) compares the results of this method with that of PROFET for $k = 1$. PROFET shows a much better performance and its advantage was found to be statistically significant ($p = 0.0015$). While w-prediction was found to perform better on this data, it is quite possible that with much larger quantities of data (perhaps spread over longer periods) cross correlation would have been a better method.

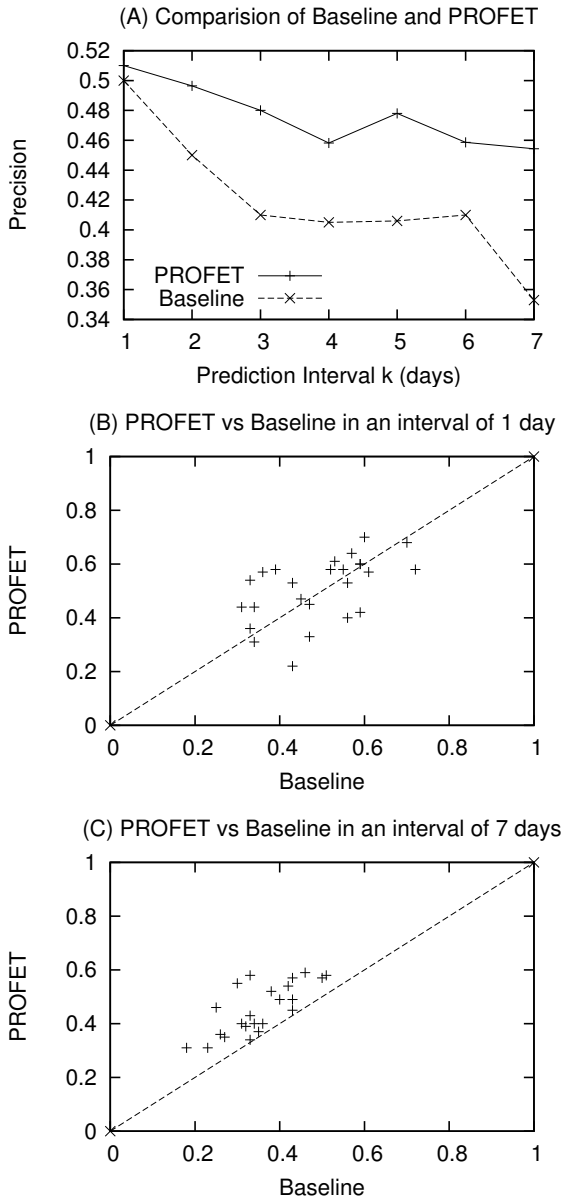

Figure 5.2: (a) Comparison of the performance of Baseline and PROFET on several prediction intervals. (b), (c) Performance on a prediction interval of 1 and 7 days

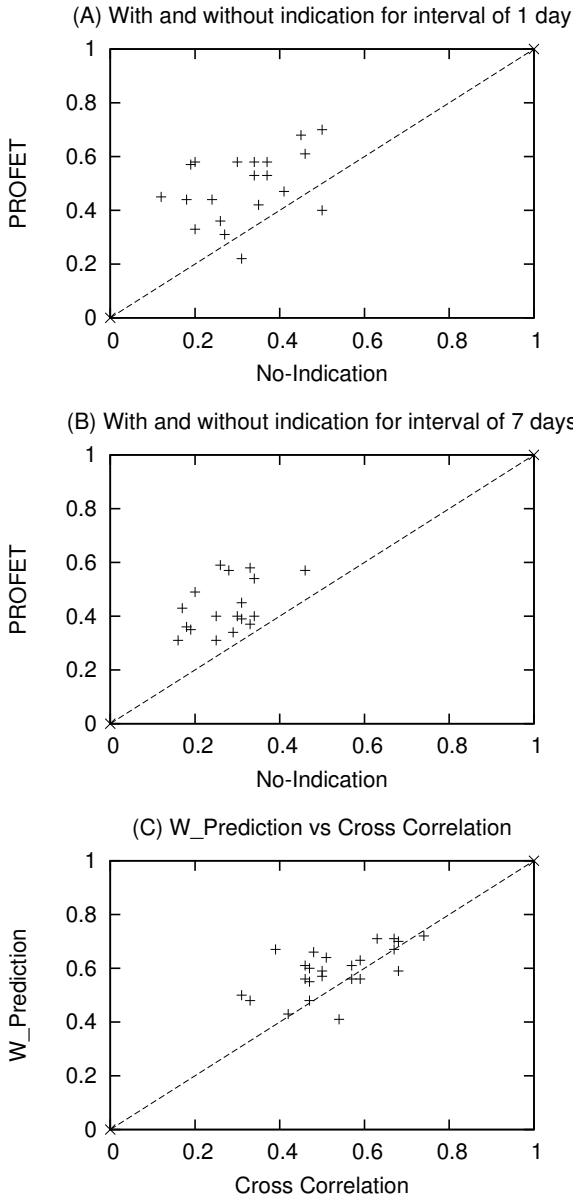

Figure 5.3: (a), (b) Performance with and without the indication mechanism on a prediction interval of 1 day and 7 days. (c) W-Prediction compared with Cross-Correlation on a prediction interval of 1 day. Each point represents the precision of the algorithm on each of the days of the test period.

Chapter 6

Predicting the Future using Web Knowledge

In the previous chapters, we have discussed how Web dynamics can predict future human actions. However, the power of the Web is not limited only to its dynamics. It treasures much of our knowledge, which also can enrich predictions about the real human world. The algorithms we discussed before perform well when enough data exists about the specific event. However, specific events are very rare. Therefore, the algorithm has to have the ability to learn and generalize the events it has seen in the past. For this reason, one must endow the machine the ability to obtain and encode the expertise which humans acquire throughout their lifetime with their interaction with the world. The Web encapsulates much of such human knowledge. A treasure trove of information about historical events is contained in news archives and encyclopedias. Dynamic updates about current events are available through online news papers, which update every minute. Additionally, the Web has many common sense ontologies of entities and their relationships. All this knowledge can serve as the basis for performing true human-like prediction, with the ability to learn, understand language, develop intuitions and generate general world knowledge. This knowledge can serve as the basis for understanding causality as human agents perceive it.

In this chapter, we discuss causality in its most basic form, as has been discussed from the early days of Aristotle. The same causality that was discussed in the modern work of David Hume (1711–1776), who referred to it as the strongest and most important associative relation, that which lies at the heart of our perception and reasoning about the world, as "it is constantly supposed that there is a connection between the present fact and that which is inferred from it."

While many works have been devoted to extracting information from text (e.g., Banko et al. 2007 and Carlson et al. 2010), little has been done in the area of causality extraction, with Khoo et al. 2000 and Girju et al. 2002 being notable

exceptions. Furthermore, the algorithms developed for causality extraction try to *detect* causality and cannot be used to *predict* it, that is, to generate new events the given event might cause.

Our goal in this paper is to provide algorithms that perform causal reasoning, in particular causality prediction, in textually represented environments. We have developed a causality learning and prediction algorithm, Pundit, that, given an event represented in natural language, predicts future events it can cause. Our algorithm is trained on examples of causality relations. It then uses large ontologies to generalize over the causality pairs and generate a prediction model. The model is represented by an abstraction tree, that, given an input cause event, finds its most appropriate generalization, and uses learned rules to output predicted effect events.

We have implemented our algorithm and applied it to a large collection of news reports from the last 150 years. To extract training examples from the news corpus, we do not use correlation, by means of which causality is often misidentified. Instead, we use *textual causality patterns* (such as "X because Y" or "X causes Y"), applied to news headlines, to identify pairs of structured events that are supposedly related by causality. The result is a semantically-structured causality graph of 300 million fact nodes connected by more than one billion edges. To evaluate our method, we tested it on a news archive from 2010, which was not used during training. The results are judged by human evaluators.

To give some intuition about the type of predictions the algorithm generates, we present here two examples of actual predictions made by our system. First, given the event "Magnitude 6.5 earthquake rocks the Solomon Islands," the algorithm predicted that "a tsunami-warning will be issued for the Pacific Ocean." It learned this from past examples on which it was trained, one of which was
$\langle$*7.6 earthquake strikes island near India*, *tsunami warning issued for Indian Ocean*$\rangle$.
Pundit was able to infer that an earthquake occurring near an island would result in a tsunami warning being issued for its ocean. Second, given the event "Cocaine found at Kennedy Space Center," the algorithm predicted that "a few people will be arrested." This was partially based on the past example $\langle$*police found cocaine in lab* $\rightarrow$ *2 people arrested*$\rangle$.

The contributions of this work are threefold: First, we present novel and scalable algorithms for generalizing causality pairs to causality rules. Second, we provide a new method for using casualty rules to predict new events. Finally, we implement the algorithms in a large scale system and perform an empirical study on realistic problems judged by human raters. We make the extracted causality information publicly available for further research in the field [1].

[1] `http://www.technion.ac.il/~kirar/Datasets.html`

6.1 Learning and Predicting Causality

"Predicting is very difficult, especially if it's about the future." (Niels Bohr)

In this section, we describe the Pundit algorithm for learning and predicting causality. We start with an overview of the learning and prediction process. During training, the learning algorithm receives causality event pairs, extracted from historical news archives (Section 6.2). The algorithm then generalizes over the given examples using world knowledge and produces an abstraction tree (AT)(Section 6.1.4). For each node in the AT, a prediction rule is generated from the examples in the node (Section 6.1.5). Then, during the prediction phase, the algorithm matches the given new event to nodes in the AT, and the associated rule is applied on it to produce possible effect events (Section 6.1.6). Those events are then filtered (Section 6.1.7) and an effect event output. The output event itself is also given in natural language, in sentence-like form. The process is illustrated in Figure 7.2.

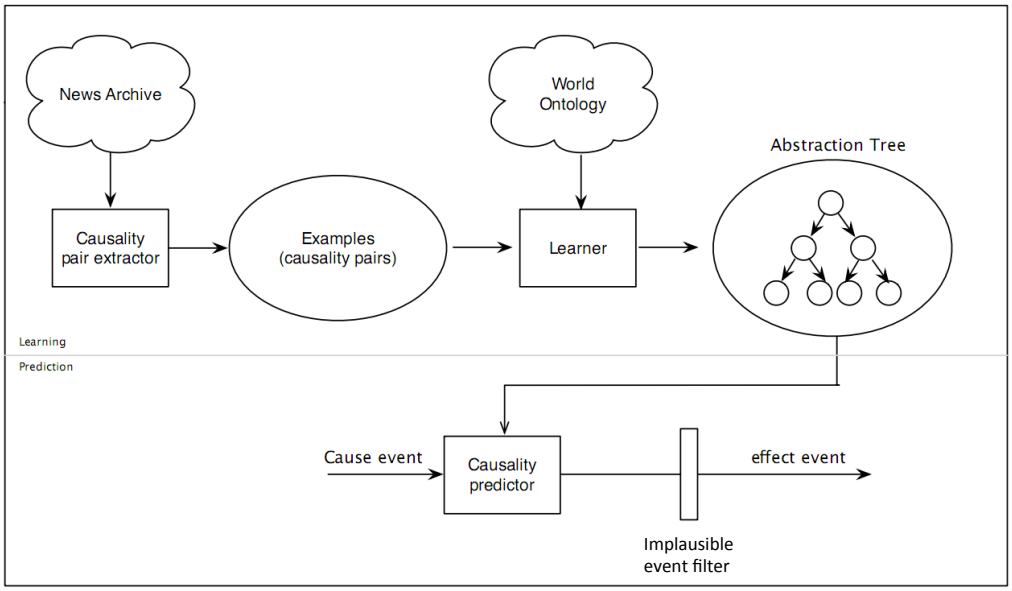

Figure 6.1: Structure of the Pundit prediction algorithm

6.1.1 Event Representation

"The representation of causality is an essential component in causal cognition as it...indicates what people learn when they induce causal relationships and what people mean when they use causal language." Wolff [2007].

The basic element of causal reasoning is an event. The Topic Tracking and Detection (TDT) community Allan [2002] has defined an event as "a particular thing that happens at a specific time and place, along with all necessary preconditions and unavoidable consequences." Other philosophical theories consider events as exemplifications of properties by objects at times Kim [1993]. For example, Caesar's death at 44 BC is Caesar's exemplification of the property of dying at time 44 BC. Those theories impose structure on events, where a change in one of the elements yields a different event. For example, Shakespear's death is a different event from Caesar's death, as the objects exemplifying the property are different. In this section, we will discuss a way to represent events following the exemplification theory Kim [1993] that will allow us to easily compare them, generalize them, and reason about them.

There are three common approaches for textual event representation: The first approach describes an event at sentence level by running text or individual terms Blanco et al. [2008]; Sil et al. [2010]. Event similarity is treated as a distance metric between the two events' bag of words. While such approaches can be useful, they often fail to perform fine-grained reasoning. Consider, for example, three events: "US Army bombs a warehouse in Iraq," "Iraq attacks US base," and "Terrorist base was attacked by the US Marines in Kabul." Representing these events by individual terms alone might yield that the first two are more similar than the first and the last as they have more words in common. However, such approaches disregard the fact the actors of the first and last event are military groups and that Kabul and Iraq are the event locations. When these facts are taken into account, the first and last events are clearly more similar than the first and second.

The second approach describes events in a syntax-driven manner, where the event text is transformed into syntax-based components, such as noun phrases Chan and Lam [2005]; Garcia [1997]; Girju and Moldovan [2002]; Khoo et al. [2000]. In our example, this representation again erroneously finds the second and third events to be most similar due to the syntactic similarity between them. Using the first two approaches, it is hard to make practical generalizations about events or to compare them in a way that takes into account all the semantic elements that compose them.

The third approach is semantic (similar to the representation in Cyc Lenat and Guha [1990]), and maps the atomic elements of each event to semantic concepts. This approach provides grounds for canonic representation of events that are both

comparable and generalizable. In this work, we follow the third approach and represent events semantically.

Given a set of entities O that represent physical objects and abstract concepts in the real world (e.g., people, instances, and types), and a set of actions P, we define an event as an ordered set $e = \langle P, O_1, \ldots, O_4, t \rangle$, where:

1. P is a temporal action or state that the event's objects exhibit.
2. O_1 is the *actor* that performed the action.
3. O_2 is the *object* on which the action was performed.
4. O_3 is the *instrument* with which the action was performed.
5. O_4 is the *location* of the event.
6. t is a time-stamp.

For example, the event "The U.S Army destroyed a warehouse in Iraq with explosives," which occurred on October 2004, is modeled as: Destroy (Action); U.S Army (Actor); warehouse (Object); explosives (Instrument); Iraq (Location); October 2004 (Time). This approach is inspired by Kim's property-exemplification of events theory 1993.

6.1.2 Learning Problem Definition

"History repeats itself."(Anonymous proverb)

We treat causality inference as a supervised learning problem. Let Ev be the universe of all possible events. Let $f : Ev \times Ev \to \{0, 1\}$ be the function

$$f(e_1, e_2) = \begin{cases} 1 & \text{if } e_1 \text{ causes } e_2, \\ 0 & \text{if otherwise.} \end{cases}$$

We denote $f^+ = \{(e_1, e_2) | f(e_1, e_2) = 1\}$. We assume we are given a set of possible positive examples $E \subseteq f^+$.

Our goal is not merely to *test* whether a pair of events is a plausible cause-effect pair by f, but to *generate* for a given event e the events it can cause. For this purpose we define $g : Ev \to 2^{Ev}$ to be $g(e) = \{e' | f(e, e') = 1\}$; that is, given an event, output the set of events it can cause. We wish to build this predictor g using the examples E.

Learning f from E could have been solved by standard techniques for concept learning from positive examples. The requirement to learn g, however, presents the challenging task of structured prediction from positive examples.

6.1.3 Generalizing Over Objects and Actions

"An idea is always a generalization, and generalization is a property of thinking. To generalize means to think." (Georg Wilhelm Friedrich Hegel)

Our goal is to develop a learning algorithm that automatically produces a causality function based on examples of causality pairs. The inferred causality function should be able to predict the outcome of a given event, even if it was never observed before. For example, given the training examples ⟨earthquake in Turkey, destruction⟩ and ⟨earthquake in Australia, destruction⟩, and a current new event of "earthquake in Japan," a reasonable prediction would be "destruction." To be able to handle such predictions, we must endow our learning algorithm with generalization capacity. For example, in the above scenario, the algorithm must be able to generalize Australia and Turkey to countries, and to infer that earthquakes in countries might cause destruction. This type of inference and the knowledge that Japan is also a country enables the algorithm to predict the effects of new events using patterns in the past.

To generalize over a set of examples, each consisting of a pair of events, we perform generalization over the components of these events. There are two types of components – objects and actions.

To generalize over objects, we assume the availability of a semantic network $G_o = (V, E)$, where the nodes $V \subseteq O$ are the objects in our universe, and the labels on the edges are relations such as isA, partOf and CapitalOf. In this work, we consider one of the largest semantic networks available, the LinkedData ontology Bizer et al. [2009], which we describe in detail in Section 6.2.

We define two objects to be similar if they relate to a third object in the same way. This relation can be a label or a sequence of labels in the semantic network. For example, Paris and London will be considered similar because their nodes are connected by the path $\xrightarrow{Capital-of}\xrightarrow{In-continent}$ to the node Europe. We now formally define this idea.

Definition 1: Let $a, b \in V$. A sequence of labels $L = l_1, \ldots, l_k$ is a **generalization path** of a, b, denoted by GenPath(a,b), if there exist two paths in G, $(a, v_1, l_1), \ldots (v_k, v_{k+1}, l_k)$ and $(b, w_1, l_1), \ldots (w_k, w_{k+1}, l_k)$, s.t. $v_{k+1} = w_{k+1}$.

Overgeneralization of events should be avoided – e.g., given two similar events, one occurring in Paris and one in London, we wish to produce the generalization "city in Europe" ($\xrightarrow{Capital-of}\xrightarrow{In-continent} Europe$) rather than the more abstract generalization "city on a continent" ($\xrightarrow{Capital-of}\xrightarrow{In-continent}\xrightarrow{IsA} Continent$). We wish our generalization to be as specific as possible. We call this *minimal generalization* of objects.

Definition 2: The **minimal generalization path**, denoted by $MGenPath(a, b)$, is defined as the set containing the shortest generalization paths. We denote $dist_{Gen}(a, b)$ as the length of the $MGenPath(a, b)$.

Path-based semantic distances such as the one above were shown to be successful in many NLP applications. For example, the semantic relatedness of two words was measured by means of a function that measured the distance between words in a taxonomy Rada et al. [1989]; Strube and Ponzetto [2006]. We build on this metric and expand it to handle events that are structured and can contain several objects from different ontologies.

To efficiently produce $MGenPath$, we designed an algorithm (described in Figure 6.2), based on dynamic programming, that computes the $MGenPath$ for all object pairs in G. For simplicity, we describe an algorithm that computes a single path for each two nodes a and b, rather than the set of all shortest paths. At step 1 a queue that holds all nodes with the same generalization is initialized. At step 2, the algorithm identifies all nodes (a, b) that have a common node (c) connecting to them via the same type of edge (l). c can be thought of as a generalization of a and b. The $Mgen$ structure maps a pair of nodes to their generalization ($Mgen.Gen$) and their generalization path ($MGen.Pred$). At step 3, in a dynamic programming manner, the algorithm iterates over all nodes (a, b) in $Mgen$ for which we found a minimal generalization in previous iterations, and finds two nodes – one (x) connecting to a and one (y) connecting to b via the same type of edge l (step 3.4). Thus, the minimal generalization of x and y is the minimal generalization of a and b, and the path is the $MGenPath$ of a, b with the addition of the edge type l. This update is performed in steps 3.4.1–3.4.4. Eventually, when no more nodes with minimal generalization can be expanded (i.e., the algorithm cannot find two nodes that connect to them via the same edge type), it stops and returns $Mgen$ (step 4). During prediction, if several $Mgen$ exists, we consider both during the prediction with their corresponding $MGenPath$.

We define a distance between actions using an ontology G_p, similarly to the way we defined distance between objects. Specifically, we use the VerbNet Kipper [2006] ontology, which is one of the largest English verb lexicons. It has mapping to many other online resources, such as Wordnet Miller [1995]. The ontology is hierarchical and is based on a classification of the verbs to the Levin classes Dang et al. [1998]. This resource has been widely used in many natural language processing applications Giuglea and Moschitti [2006]; Shi and Mihalcea [2005]. Using this ontology we describe the connections between verbs. Figure 6.10 shows a node in this ontology that generalizes the actions "hit" and "kick."

6.1.4 Generalizing Events

"Avoid context and specifics; generalize and keep repeating the generalization." (Jack Schwartz) In order to provide strong support for generalization, we wish to find similar events that can be generalized to a single abstract event. In our example,

Procedure MINIMAL GENERALIZATION PATH(G)
(1) $Q \leftarrow$ new Queue
(2) **Foreach** $\{(a,c,l),(b,c,l) \in E(G)|$
$a,b,c \in V(AT), l \in Lables\}$
(2.1) $Mgen(a,b).Gen = c$
(2.2) $Mgen(a,b).Pred = l$
(2.3) $Mgen(a,b).Expanded = false$
(2.4) $Q.enqueue((a,b))$
(3) **While** $Q \neq \emptyset$:
(3.1) $(a,b) \leftarrow Q.dequeue()$
(3.2) **If** $Mgen(a,b).Expanded \neq true$:
$Mgen(a,b).Expanded = true$
(3.4) **Foreach** $\{(x,a,l),(y,b,l) \in E(AT)|$
$x,y \in V(AT), l \in Lables\}$
(3.4.1) $Mgen(x,y).Gen = Mgen(a,b).Gen$
(3.4.2) $Mgen(x,y).Pred = Mgen(a,b).Pred||l$
(3.4.3) $Mgen(x,y).Expanded = false$
(3.4.4) $Q.enqueue((x,y))$
(4)**Return** $Mgen$

Figure 6.2: Procedure for calculating the minimal generalization path for all object pairs

we wish to infer that both ⟨earthquake in Turkey, destruction⟩ and ⟨earthquake in Australia, destruction⟩ are examples of the same group of events. Therefore, we wish to cluster the events in such a way that events with similar causes and effects will be clustered together. As in all clustering methods, a distance measure between the objects should be defined. Let $e_i = \langle P^i, O_1^i, \ldots, O_4^i, t^i \rangle$ and $e_j = \langle P^j, O_1^j, \ldots, O_4^j, t^j \rangle$ be two events. In the previous subsection we defined a distance function between objects (and between actions). Here, we define the similarity of two events e_i and e_j to be a function of distances between their objects and actions:

$$SIM(e_i, e_j) = f\big(dist_{Gen}^{G_p}(P^i, P^j), dist_{Gen}^{G_o}(O_1^i, O_1^j), \ldots, dist_{Gen}^{G_o}(O_4^i, O_4^j)\big), \quad (6.1)$$

where, $dist_{Gen}^{G}$ is the distance function $dist_{Gen}$ in the graph G, and f is an aggregation function. In this work, we mainly use the average as the aggregation function, but also analyze several alternatives.

Likewise, a similarity between two pairs of cause-effect events $\langle c_i, e_i \rangle$ and $\langle c_j, e_j \rangle$ is defined as:

$$SIM(\langle c_i, e_i \rangle, \langle c_j, e_j \rangle) = f\big(SIM(c_i, c_j), SIM(e_i, e_j)\big). \quad (6.2)$$

Using the similarity measure suggested above, the clustering process can be thought of as a grouping of the training examples in such a way that there is a low variance in their effects and a high similarity in their cause. This is similar to information gain methods where examples are clustered by their class. We use the HAC hierarchical clustering algorithm Eisen et al. [1998] as our clustering method. The algorithm starts by joining the closest event pairs together into a cluster. It then keeps repeating the process by joining the closest two clusters together until all elements are linked into a hierarchical graph of events we call an abstraction tree (AT). Distance between clusters is measured by the distance of their representative events. To allow this, we assign to each node in the AT a representative cause event, which is the event closest to the centroid of the node's cause events. During the prediction phase, the input cause event will be matched to one of the created clusters, i.e., closest to the representative cause event of the cluster.

6.1.5 Causality Prediction Rule Generation

"The expression of causal necessity is a projection of the functional change onto the objects involved in the causal connection." (Wikipedia entry about David Hume)

The last phase of the learning is the creation of rules that will allow us, given a cause event, to generate a prediction about it. As the input cause event is matched against the node centroid, a naive approach would be to return the effect event of the matched centroid. This, however, would not provide us with the desired result. Assume an event e_i="Earthquake hits Haiti" occurred today, and that is matched to a node represented by the centroid: "Earthquake hits Turkey," whose effect is "Red Cross help sent to Ankara." Obviously, predicting that Red Cross help will be sent to Ankara because of an earthquake in Haiti is not reasonable. We would like to be able to abstract the relation between the past cause and past effect and learn a predicate clause that connects them, for example "Earthquake hits [Country Name]" yielding "Red Cross help sent to [Capital of Country]." During prediction, such a clause will be applied to the input event e_i, producing its predicted effect. In our example, the logical predicate clause would be *CapitalOf*, as *CapitalOf(Turkey)*= Ankara. When applied on the current event e_i, *CapitalOf(Haiti)* = Port-au-Prince, the output will now be "Red Cross help sent to Port-au-Prince." Notice that the the clauses can only be applied on certain types of objects – in our case, countries. The clauses can be of any length, e.g., the pair ⟨"suspect arrested in Brooklyn," "Bloomberg declares emergency"⟩ produces the clause *Mayor(BoroughOf(x))*, as Brooklyn is a borough of New York, whose mayor is Bloomberg.

We will now show how to learn such clauses for each node in the AT graph. Recall that the semantic network graph G_O is an edge-labeled graph, where each

edge is a triplet $\langle v_1, v_2, l \rangle$, where l is a predicate (e.g., "CapitalOf"). The rule-learning procedure is divided into two main steps. First, we find an undirected path p_i of length at most k in G_O between any object of the cause event to any object of the effect event. Note that we do not necessarily look for paths between two objects with the same role. In the above example, we found a path between the location of the cause event (Brooklyn) to the actor of the effect event (Bloomberg). Second, we construct a clause using the labels of the path p_i as the predicates. We call this the *predicate projection* of size k, $pred = l_1, \ldots, l_k$ from an event e_i to an event e_j. During prediction, the projections will be applied to the new event $e = \langle P^i, O_1, \ldots, O_4, t \rangle$ by finding an undirected path in G_O from O_i with the sequence of labels of *pred*. As k is unknown, the algorithm, for each training example $\langle c_t, e_t \rangle$ in a node in the AT, finds all possible predicate paths with increasing sizes of k from the objects of c_t to the objects of e_t in the G_O graph. Each such path is weighted by the number of times it occurred in the node, the *support* of the rule. The full predicate generation procedure is described in Figure 6.3. The function $LearnPredicateClause$ calls the inner function $FindPredicatePath$ for different k sizes and different objects from the given cause and effect events. $FindPredicatePath$ is a recursive function that tries to find a path between the two objects in a graph of length k. It returns the labels of such a path if found. The rule generated is a template for generating the prediction of a future event given the cause event. An example of such a rule can be seen in Figure 6.4. Rules that return NULL are not displayed in the figure. In this example, when we generate object O_1 of the future event, we try to apply the path $\xrightarrow{l_1}\xrightarrow{l_2}$ on the object O_4 of the cause, thus generating possible objects that can be object O_1 of the prediction (see Section 6.1.6). Similarly, the path $\xrightarrow{l_1}\xrightarrow{l_2}$ is applied on O_2, generating more possible objects. For object O_2 of the prediction, a simple path of one label was generated. Therefore, during prediction, the possible objects for O_2 are the ones that connect to O_4^{cause} with the label l_8 (if any). For object O_3 of the prediction, we use the O_3^{cause}. For O_4 no special rule was generated ($FindPredicatePath$ returned NULL for all objects), and the final prediction will have O_4^{effect}.

6.1.6 Prediction

"Causal necessity is an expression of a functional change in the human mind, whereby certain events are predicted or anticipated on the basis of prior experience." (Wikipedia entry about David Hume)

Given a trained model $\hat{g}$, it can be applied to a new event $e = \langle P^i, O_1, \ldots, O_4, t \rangle$ in order to produce its effects. The process is divided into two main steps – propagating the event in the AT to retrieve a set of matched nodes, and applying the rules of each matched node to produce the possible effects.

Procedure FindPredicatePath($curEntity, goalEntity, depth$)
  If $curEntity = goalEntity$ **Return** $\emptyset$
  Else
    If $depth = 0$ **Return** $NULL$
    Else
      Foreach $relation \in outEdges(curEntity)$
        $solution \leftarrow$ FindPredicatePath($relation(curEntity), goalEntity, depth - 1$)
        If $solution \neq NULL$
          Foreach $existingSolution \in Solutions$:
            Return $Solutions \bigcup (existingSolution||relation||solution)$
      Return $Solutions$

Procedure LearnPredicateClause($\langle P^c, O_1^c, \ldots, O_4^c, t^c\rangle, \langle P^e, O_1^e, \ldots, O_4^e, t^e\rangle, depth$)
  Foreach $O_i^c \in O^c$, $O_j^e \in O^e, k \in \{1 \ldots depth\}$
    rule(j) $\leftarrow \emptyset$
  Foreach $O_i^c \in O^c$, $O_j^e \in O^e, k \in \{1 \ldots depth\}$
    rule(j) $\leftarrow$ rule(j) $\bigcup$ $\{\langle O_i^c,$ FindPredicatePath$(O_i^c, O_j^e, k)\rangle\}$
  Return rule

Figure 6.3: Procedure for generating a rule between two events by inferring paths between the two events in the causality graph.

Given a new event, Pundit traverses the AT starting from the root. For each node in the search frontier, the algorithm computes the similarity ($SIM(e_i, e_j)$) of the input event to the centroid of each of the children on this node, and expands those children with better similarity than their parent. This idea can be stated intuitively as an attempt to find the nodes which are the least general but still similar to the new event. The full algorithm is illustrated in Figure 6.5. An illustration of the process can be seen in Figure 6.6. Here, an event of a bombing in Baghdad is received as input. The system searches for the least general cause event it has observed in the past (for simplicity we only show a short notation of the cause events in the AT). In our case, it is a generalized cluster: "bombing in city." Other candidates selected are the "military communication" cluster and the "bombing" cluster (as the node "bombing in worship area" has a lower score than "bombing").

For each node retrieved in the previous stage, its predicate projection, *pred*, is applied to the new event $e = \langle P^i, O_1, \ldots, O_4, t\rangle$. Informally, we can say that *pred* is applied by finding an undirected path in G_O from O_i with the labels of *pred*. This rule generates a possible effect event from the retrieved node. The projection results are all the reached objects in the vertex. The formal explanation is that

$Rule(cause, effect) =$

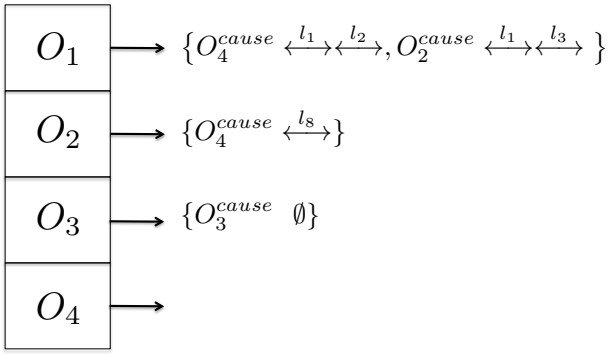

Figure 6.4: An example of a generated rule

pred can be applied if $\exists V_0 : O \subseteq V_0, \exists V_1 \ldots V_k : (V_0, V_1, l_1), \ldots (V_{k-1}, V_k, l_k) \in Edges(G_O)$. The projection results are all the objects $o \in V_k$. The projection results of all the nodes are weighted by the similarity of the target cause to the node *MGen* and then ranked by the support of the rule (for tie breaking). If several *MGen* exists, the highest similarity is considered. See Figure 6.7 for a complete formal description of the algorithm. In our example (Figure 6.6), the candidate "bombing in [city]" has the following rules:

1. $P^{effect} = happen$, $O_1^{effect} = riot$, $O_4^{effect} = O_4^{cause}$
2. $P^{effect} = happen$, $O_1^{effect} = riot$, $O_4^{effect} = O_4^{cause} \overset{main-street-of}{\longleftrightarrow}$
3. $P^{effect} = kill$, $O_2^{effect} = people$
4. $P^{effect} = condemn$, $O_1^{effect} = O_4 \overset{mayor-of}{\longleftrightarrow} \overset{borough-of}{\longleftrightarrow}$, $O_2^{effect} = attack$

For clarity, for objects where no rule can be applied (the rule for the object is NULL), we use the effect concept of the matched training example.

For the event Baghdad Bombing ($O_1 = Bombing$, $O_4 = Baghdad$), applying the rules yields the following:

1. Baghdad Riots ($P^{effect} = happen$, $O_1^{effect} = riot$, $O_4^{effect} = Baghdad$).

Procedure PROPAGATION($e = \langle P^i, O_1, \ldots, O_4, t \rangle$)
(1) $Candidates \leftarrow \{\}$
(2) $Q \leftarrow$ new Queue
(3) $Q.enqueue(G.root)$
(4) **While** $Q \neq \emptyset$:
(4.1) $cur \leftarrow Q.dequeue()$
(4.2) **Foreach** $edge \in cur.OutEdges$:
If $SIM(e, edge.Source) > SIM(e, edge.Destination)$:
$Candidates \leftarrow Candidates \bigcup \{(edge.Source, SIM(e, edge.Source))\}$
Else :
$Q.enqueue(edge.Destination)$
(5) **Return** $Candidates$

Figure 6.5: Procedure for locating candidates for prediction. The algorithm saves a set of possible matched results (Candidates), and a queue holding the search frontier (Q). In step 4, the algorithm traverses the graph. In step 4.2, for each edge, the algorithm tests whether the similarity of the new event e to the parent node (edge.Source) is higher than to the child node (edge.Destination). If the test succeeds, the parent node, with its similarity score, is added to the possible results. After all edges are exposed, the algorithm returns the possible results in step 5.

2. Caliphs Street Riots ($P^{effect} = happen$, $O_1^{effect} = riot$, $O_4^{effect} = Caliphs\ Street \overset{main-street-of}{\longleftrightarrow} O_4^{cause}$).
3. People killed ($P^{effect} = kill$, $O_2^{effect} = people$).
4. This rule cannot be applied on the given event, as there is no outgoing edge of type borough-of for the node Baghdad.

6.1.7 Pruning Implausible Effects

"Proof [is] evidence having a shade more of plausibility than of unlikelihood. The testimony of two credible witnesses as opposed to that of only one."(Ambrose Bierce)

In some cases, the system generated implausible predictions. For example, for the event ⟨lightning kills 5 people⟩, the system predicted that ⟨lightning will be arrested⟩. This prediction was based on generalized training examples in which people who killed other people got arrested, such as: ⟨*Man kills homeless man*, *man is arrested*⟩. But if we could determine how logical an event is, we could avoid such false predictions. In this section we discuss how we filter them out.

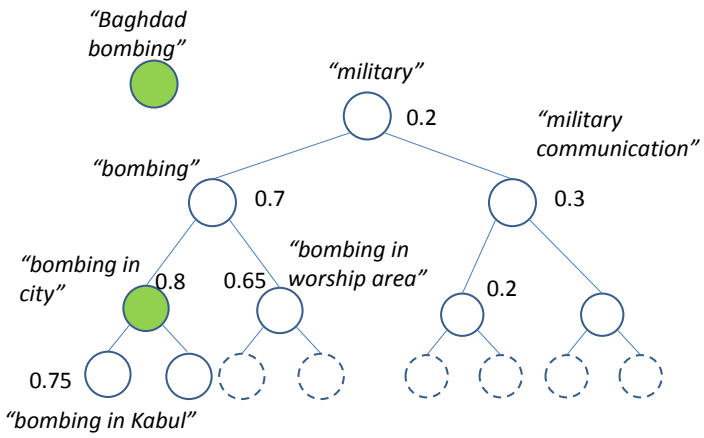

Figure 6.6: An event of a bombing in Baghdad is received as input. The system searches for the least general cause event it has observed in the past. In our case it is a generalized cluster: "bombing in city". The rule at this node now will be applied on the Baghdad bombing to generate the prediction.

The goal of our filtering component is different from that of the predictor. While the predictor's goal is to generate predictions about future events, this component's goal is to monitor those predictions. While the predictor learns a causality relation between events, this component learns their plausibility.

The right way to perform such filtering is to utilize common sense knowledge for each action. Such knowledge would state the type of the actor and the object that can perform the action, the possible instruments with which the action can be preformed and the possible locations. If such knowledge would have existed, it would have identified that for the action arrest the object can be only human. However, such common sense knowledge is currently not available. Therefore, we had to resort to the common practice of using statistical methods.

In the information extraction literature, identifying the relationship between facts and their plausibility has been widely studied. These methods usually estimate the prior probability of a relation by examining the frequency of its pattern in a large corpus, such as the Web Banko et al. [2007]. For example, for the relation ⟨People,arrest,People⟩ these methods return that this phrase was mentioned 188 times on the Web, and that the relation ⟨People,arrest,[Natural Disaster]⟩ was mentioned 0 times. Similarly, we estimate the prior probability of an event to occur from its prior appearance in the New York Times, our primary source of

news headlines. We then filter out events that, a priori, have very low probability to occur.

We present the algorithm in Figure 6.8. We calculated how many times the semantic concepts representing the event, or their immediate generalizations, actually occurred together in the past in the same semantic roles. In our example, we check how many times lightning or other natural phenomena were arrested. Formally, we define point-wise mutual concept information (PMCI) between two entities or verbs o_i, o_j (e.g., lightning and arrest) in roles r_i, r_j (e.g., actor and action) be defined as

$$PMCI(o_i, o_j, r_i, r_j) = \log \frac{p(o_i@r_i, o_j@r_j)}{p(o_i@r_i)p(o_j@r_j)}. \tag{6.3}$$

Given an event, we calculate the average PMCI of its components. The algorithm filters out predicted events that have low average PMCI. We assume that the cause and effect examples in the training are the ground truth, and should yield a high PMCI. Therefore, we evaluate the threshold for filtering from this training data. That is, we collected all the effects we observed in the training data and estimated their average PMCI on the entire NYT dataset.

The reader should note that applying such rules might create a problem. If in the past no earthquake occurred in Tokyo, the pruning procedure might return low plausibility. To handle these type of errors, we calculate the PMCI of the upper level categories of entities (e.g., natural disasters) rather than specific entities (e.g., earthquakes). We therefore restrict ourselves to only the two upper level categories.

6.2 Implementation Details

"You can build learning machines, and the learning machine could painstakingly try to learn, but how would it learn? It would have to read the New York Times every day."(Roger Schank)

In the previous section, we presented a high-level algorithm that requires training examples T, knowledge about entities G_O, and event action classes P. One of the main challenges of this work was to build a scalable system to meet those requirements.

We present a system that mines news sources to extract events, constructs their canonical semantic model, and builds a causality graph on top of those events. The system crawled, for more than 4 months, several dynamic information sources (see Section 6.2.1 for details). The largest information source was the NYT archive, on which optical character recognition (OCR) was performed. The overall gathered data spans more than 150 consecutive years ($1851 - 2009$).

For generalization of the objects, the system automatically reads Web content and extracts world knowledge. The knowledge was mined from structured and semi-structured publicly available information repositories. The generation of the causality graph was distributed over 20 machines, using a MapReduce framework. This process efficiently unites different sources, extracts events, and disambiguates entities. The resulting causality graph is composed of over 300 million entity nodes, one billion static edges (connecting the different objects encountered in the events), and over 7 million causality edges (connecting events that were found by Pundit to cause each other). Each rule in the AT was generated using an average of 3 instances with standard deviation of 2.

On top of the causality graph, a search and indexing infrastructure was built to enable search over millions of documents. This highly scalable index allows a fast walk on the graph of events, enabling efficient inference capabilities during the prediction phase of the algorithm.

6.2.1 World Knowledge Mining

"You have to live to really know things." (Dan Simmons)

The entity graph G_o is composed of concepts from Wikipedia, ConceptNetLiu and Singh [2004], WordNetMiller [1995], YagoSuchanek et al. [2007], and OpenCyc; the billion labeled edges of the graph G_o are the predicates of those ontologies. In this section we describe the process by which this knowledge graph is created and the search system built upon it.

Our system creates the entity graph by collecting the above content, processing feeds, and processing formatted data sets (e.g., Wikipedia). Our crawler then archives those documents in raw format, and transforms them into RDF (Resource Description Framework) format Lassila et al. [1998]. The concepts are interlinked by humans as part of the Linked Data project Bizer et al. [2009]. The goal of Bizer et al.'s Linked Data project is to extend the Web by interlinking multiple datasets as RDF and by setting RDF links between data items from different data sources. Datasets include DBPedia (a structured representation of Wikipedia), WordNet, Yago, Freebase, and more. By September 2010 this had grown to 25 billion RDF triples, interlinked by around 395 million RDF links.

We use SPARQL queries as a way of searching over the knowledge graph. Experiments of the performance of those queries on the Berlin benchmark Bizer and Schultz [2009] provided evidence for the superiority of Virtuoso open source triple structures for our task.

6.2.2 Causality Event Mining and Extraction

"For the extraction of causal relationships, rule based methods are preferred which require Human Intervention to develop rules." Feldman et al. [2002].

Our supervised learning algorithm requires many learning examples to be able to generalize well. As the amount of temporal data is extremely large, spanning over millions of articles, the goal of obtaining human annotated examples becomes impossible. We therefore provide an automatic procedure to extract labeled examples for learning causality from dynamic content. In this work, we used the NYT archives for the years $1851 - 2009$, WikiNews, and the BBC – over 14 million articles in total (see data statistics in Table 6.1). Extracting causal relations between events in text is a hard task. The state-of-the-art precision of this task is around 37% Do et al. [2011]. Our hypothesis is that most of the information regarding an event can be found in the headlines. These are more structured and therefore easier to analyze. Many times the headline itself can contain both the cause and effect event. We assume that only some of the headlines are describing events and developed an extraction algorithm to identify those headlines and to extract the events from them. News headlines are quite structured, and therefore the accuracy of this stage (performed on a representative subset of the data) is 78% (see Section 6.3.2). The system mines unstructured natural language text found in those headlines, and searches for causal grammatical patterns. We construct those patterns using *causality connectors* Levin and Hovav [1994]; Wolff et al. [2002]. In this work we used the following connectors:

1. Causal Connectives: the words *because*, *as*, and *after* as the connectors.

2. Causal prepositions: the words *due to* and *because of*.

3. Periphrastic causative verbs: the words *cause* and *lead to*.

We constructed a set of rules for extracting a causality pair. Each rule is structured as: ⟨Pattern, Constraint, Priority⟩, where Pattern is a regular expression containing a causality connector, Constraint is a syntactic constraint on the sentence on which the pattern can be applied, and Priority is the priority of the rule if several rules can be matched. The following constraints were composed:

1. Causal Connectives: The pattern [sentence1] after [sentence2] was used with the following constraints: [sentence1] cannot start with "when," "how," "where," [sentence2] cannot start with "all," "hours," "minutes," "years," "long," "decades." In the pattern "After [sentence1], [sentence2]" we add the constraint that [sentence1] cannot start with a number. This pattern can match the sentence "after Afghan vote, complaints of fraud surface" but will not match the sentence "after 10 years in Lansing, state lawmaker

Tom George returns". The pattern "[sentence1] as [sentence2]" was used with the constraint of [sentence2] having a verb. Using the constraint, the pattern can match the sentence "Nokia to cut jobs as it tries to catch up to rivals", but not the sentence "civil rights photographer unmasked as informer."

2. Causal prepositions: The pattern [sentence1]["because of," "due to"] [sentence2] only required constraints that [sentence1] does not start with "when," "how," "where."

3. Periphrastic causative verbs: The pattern [sentence1] ["leads to," "Leads to," "lead to," "Lead to," "led to," "Led to"] [sentence2] is used, where [sentence1] cannot contain "when," "how," "where," and the prefix cannot be "study" or "studies." Additionally, as we consider periphrastic causative *verbs*, we do not allow additional verbs in [sentence1] or [sentence2].

The result of a rule application is a pair of sentences – one tagged as a cause, and one tagged as an effect.

Given a natural-language sentence (extracted from an article headline), representing an event (either during learning or prediction), the following procedure transforms it into a structured event:

1. Root forms of inflected words are extracted using a morphological analyzer derived from WordNet Miller [1995] stemmer. For example, in the article headline from 10/02/2010: "U.S. attacks kill 17 militants in Pakistan", the words "attacks," "killed" and "militants" are transformed to "attack," "kill," and "militant" respectively.

2. Part-Of-Speech tagging Marneffe et al. [2006] is performed, and the verb is identified. The class of the verb is identified using the VerbNet vocabulary Kipper [2006], e.g., kill belongs to P =murder class.

3. A syntactic template matching the verb is applied to extract the semantic relations and thus the roles of the words (see example in Figure 6.10). Those templates are based on VerbNet, which supplies for each verb class a set of syntactic templates. These templates match the syntax to the thematic roles of the entities in the sentence. We match the templates even if they are not continuous in the sentence tree. This allows the match of a sentence even where there is an auxiliary verb between the subject and the main transitive verb. In our example, the template is "NP1 V NP2," which transforms NP1 to "Agent", and NP2 to "Patient." Therefore, we match U.S. attacks to be the *Actor*, and the militant to be the *Patient* .

Data Source	Number of Titles Extracted
NYT	14,279,387
BBC	120,445
WikiNews	25,808

Table 6.1: Data Summary.

If no template can be matched, the sentence is transformed into a typed-dependency graph of grammatical relations Marneffe et al. [2006]. In the example, U.S. attacks is identified as the subject of the sentence (candidate for Actor), militants as the object (candidate for Patient), and Pakistan as the preposition (using Locations lexicons). Using this analysis, we identify that the *Location* is Pakistan.

4. Each word in O_i is mapped to a Wikipedia-based concept. If a word matches more than one concept, we perform disambiguation by computing the cosine similarity between the body of the news article and the body of the Wikipedia article associated with the concept. For example, U.S was matched to several concepts, such as United States, University of Salford, and Us (Brother Ali album). The most similar by content was the Wikipedia concept United States. If a word in O_i is not found in Wikipedia, it is treated as a constant, i.e., generalization will not be applied on it, but it will be used during similarity calculation. That is, $dist_{Gen}(const_1, const_2) = 0$ if $const_1 = const_2$, or $dist_{Gen}(const_1, const_2) = k$ otherwise. In our experiments, we set $k = 4$, as it was the length of the longest distance found between two concepts in G_O.

5. The time of the event t is the time of the publication of the article in the news, e.g., t =10/02/2010.

In our example, the final result is the event $e = \langle$Murder-Class, United States of America, Militant, NULL, Pakistan, 10/02/2010$\rangle$. The final result of this stage is a causality graph composed of causality event pairs. Those events are structured as described in Section 6.1.1. We illustrate such a pair in Figure 6.9.

In certain cases, additional heuristics were needed in order to deal with the brevity of news language. We used the following heuristics:

1. Missing Context – In "McDonald's recalls glasses due to cadmium traces," the extracted event "cadmium traces" needs additional context – "Cadmium traces [in McDonald's glasses]." If an object is missing, the first sentence ([sentence1]) subject is used.

2. Missing entities and verbs – the text "22 dead" should be structured to the event "22 [people] [are] dead." If a number appears as the subject, the word people is added and used as the subject, and "be" is added as the verb.

3. Anaphora resolution – the text "boy hangs himself after he sees reports of Hussein's execution" is modeled as "[boy_1] sees reports of Hussein's execution" causes "[boy_1] hangs [boy_1]" Lappin and Leass [1994].

4. Negation – the text "Matsui is still playing despite his struggles" should be modeled as: "[Matsui] struggles" causes the event "Matsui is [not] playing". Modeling preventive connectors (e.g., despite) requires negation of the modeled event.

6.3 Experimental Evaluation

In this section, we describe a set of experiments performed to evaluate the ability of our algorithms to predict causality. We first evaluate the predictive precision of our algorithm, continue with analyzing each part of the algorithm separately, and conclude with a qualitative evaluation.

6.3.1 Prediction Evaluation

The prediction algorithm was trained using news articles from the period $1851 - 2009$. The world knowledge used by the algorithm was based on Web resource snapshots (Section 6.2) dated until 2009. The evaluation was performed on separate data – Wikinews articles from the year 2010. We refer to this data as the *test data*.

As the task tackled by our algorithm has not been addressed before, we could not find any baseline algorithm to compare against. We therefore decided to compare our algorithm's performance to that of human predictors. Our algorithm and its human competitors were assigned the basic task of predicting what event a given event might cause. We evaluate each such prediction using two metrics. The first metric is *accuracy*: whether the predicted event actually occurred in the real world. There are two possible problems with this metric. First, a predicted event, though plausible, still might not actually have occurred in the real world. Second, the predicted event might have happened in the real world but was not caused by the given event, for example, in trivial predictions that are always true ("the sun will rise"). We therefore use an additional metric, event *quality*, the likelihood that the predicted event was caused by the given event.

The experiments were conducted as follows:

1. Event identification – our algorithm assumes that the input to the predictor h is an event. To find news headlines that represent an event, we randomly sample $n = 1500$ headlines from the test data. For each headline, a human evaluator is requested to decide whether the headline is an event that can cause other events. We denote the set of headlines labeled as events as E. We again randomly sample $k = 50$ headlines from E. We denote this group as C.

2. Algorithm event prediction – on each headline $c_i \in C$, Pundit performs event extraction, and produces an event $Pundit(c_i)$ with the highest score of being caused by the event represented by c_i. Although the system provides a ranked list of results, to simplify the human evaluation of theses results, we consider only the highest score prediction. If there is a tie for the top score, we pick one at random. The results of this stage are the pairs: $\{(c_i, Pundit(c_i))|c_i \in C\}$.

3. Human event prediction – For each event $c_i \in C$, a human evaluator is asked to predict what that event might cause. Each evaluator is instructed to read a given headline and predict its most likely outcome, using any online resource and with no time limit. The evaluators are presented with empty structured forms with the 5 fields for the output event they need to provide. The human result is denoted as $human(c_i)$. The results of this stage are the pairs: $\{(c_i, human(c_i))|c_i \in C\}$.

4. Human evaluation of the results –

   (a) Quality: We present $m = 10$ people with a triplet $(c_i, human(c_i), Pundit(c_i))$. The human evaluators are asked to grade $(c_i, human(c_i))$ and $(c_i, Pundit(c_i))$ on a scale of 0-4 (0 is a highly implausible prediction and 4 is a highly plausible prediction). They were allowed to use any resource and were not limited by time. The human evaluators were different from those who performed the predictions.

   (b) Accuracy: For each predicted event, we checked the news (and other Web resources), up to a year after the time of the cause event, to see whether the predicted events were reported.

Human evaluation was conducted using Amazon Mechanical Turk, an emerging utility for performing user study evaluations, which was shown to be very precise for certain tasks Kittur et al. [2008]. During the evaluation, tasks are created by routing a question to random users and obtaining their answers. We filtered the raters using a CAPTCHA. We restricted to only US-based users, as

Human Event Identification	Human Event Prediction	Human evaluation (Quality)	Human evaluation (Accuracy)
1 min 26 sec	4 min 10 sec	1 min 44 sec	6 min 24 sec

Table 6.2: Response times of human evaluators for the different evaluation tasks.

the events used by our system are extracted from the NYT. We did not perform any other manual filtering of the results. The average times for all human tasks are reported in table 6.2. We observed that the most time-consuming task for humans was to verify that the event indeed happened in the past. The other time-consuming task was Human Event Prediction. This is not surprising, as both cases required more use of external resources, whereas the quality evaluation only measured whether those events make sense. Additionally, we manually investigated the human evaluations in each category, and did not find correlation between the response time and quality of the human prediction. As we used Mechanical Turk, we do not know which external resources the evaluators used. We measured inter-rater reliability using Fleiss' kappa statistical test, where κ measures the consistency of the ratings. For the raters in our test, we obtained $\kappa = 0.3$, which indicates fair agreement Landis and Koch [1977]; Viera and Garrett [2005]. This result is quite significant, for the following reasons:

1. Conservativeness of this measure.

2. Subjectivity of the predictions – asking people whether a prediction makes sense often leads to high variance in responses.

3. Small dataset – the tests were performed with 10 people asking to categorize into 5 different scales of plausibility over 50 examples.

4. Lack of formal guidelines for evaluating the plausibility of a prediction – no instructions were given to the human evaluators regarding what should be considered plausible and what is not.

Additionally, for comparison, similar tasks in natural language processing, such as sentence formality identification Lahiri et al. [2011], usually reach kappa values of $0.1 - 0.2$.

The quality evaluation yielded that Pundit's average predictive precision is 3.08/4 (3 is a "plausible prediction"), as compared to 2.86/4 for the humans. For each event, we average the results of the m rankers, producing an average score for the algorithm's performance on the event, and an averaged score for the human predictors (see Table 6.3). We performed a paired t-test on the k paired scores. The advantage of the algorithm over the human evaluators was found to be statistically significant, with $p \leq 0.05$.

	[0-1)	[1-2)	[2-3)	[3-4]	Average Quality
Pundit	0	2	19	29	**3.08**
Humans	0	3	24	23	2.86

Table 6.3: Quality results. The histogram of the rankings of the users for humans and the algorithm.

Algorithm	Average Accuracy
Pundit	**63%**
Humans	42%

Table 6.4: Prediction accuracy for both human and algorithm.

The accuracy results are reported in Table 7.1. We performed a Fisher's exact test (as the results are binary) on the k paired scores. The results were found to be statistically significant, $p \leq 0.05$.

6.3.2 Component Analysis

In this section, we report the results of our empirical analysis of the different parts of the algorithm.

Evaluation of the Extraction Process

In Section 6.2.1, we described a process for extracting causality pairs from the news. These pairs are used as a training set for the learning algorithm. This process consists of two main parts: causality identification and event extraction. We perform a set of experiments to provide insights on this extracted training data quality.

Causality Extraction Experiment The first step in building a training set consists of using causality patterns to extract pairs of sentences for which the causality relation holds. To assess the quality of this process, we randomly sampled 500 such pairs from the training set and presented them to human evaluators. Each pair was evaluated by 5 humans. We filtered the raters using a CAPTCHA and filtered out outliers. The evaluators were shown two sentences the system believed to be causally related and they were asked to evaluate the plausibility of this relation on a scale of 0-4.

The results show that the averaged precision of the extracted causality events is 3.11 out of 4 (**78%**), where 3 means a plausible causality relation, and 4 means a highly plausible causality relation. For example, the causality pair: "pulling over

	Action	Actor	Object	Instrument	Location	Time
Quality Precision	93%	74%	76%	79%	79%	100%

Table 6.5: Extraction precision for each of the 5 event components using the causality patterns.

a car" → "2 New Jersey police officers shot," got a very high causality precision score, as this is a plausible cause-effect relation, which the system extracted from the headline "2 New Jersey Police Officers Shot After Pulling Over a Car."

For comparison, other temporal rule extraction systems Chambers et al. [2007] reach precision of about 60%. The better performance of our system can be explained by our use of specially crafted templates (we did not attempt to solve the general problem of temporal information extraction).

Most causality pairs extracted were judged to be of high quality. The main reason for errors was that some events, although reported in the news and matching the templates we have described, are not common-sense causality knowledge. For example, "Aborted landing in Beirut" → "Hijackers fly airliner to Cyprus", was rated unlikely to be causally related, although the event took place on April 09, 1988.

Event Extraction Experiment After a pair of sentences is determined to have a casualty relation, our algorithm extracts a structured event from each of the sentences. This event includes the following roles: *action*, *actor*, *object*, *instrument*, and *time*.

To assess the the quality of this process, we used the 500 pairs from the previous experiment and presented each of the 1000 associated sentences to 5 human evaluators. The evaluators were shown a sentence together with its extracted roles: *action*, *actor*, *object*, *instrument*, and *time*, and they were asked to mark each role assignment as being right or wrong.

Table 6.5 shows that the precision for the extracted event components ranges from $74 - 100\%$. In comparison, other works Chambers and Jurafsky [2011] for extracting entities for different types of relations reach $42 - 53\%$ precision. The higher precision of our results is mainly due to the use of domain-specific templates.

We performed additional experiments to evaluate the matching of every entity from the above experiment to the world-knowledge ontology. The matching was based on semantic similarity. Each ranker was asked to indicate whether the extracted entity was mapped correctly to a Wikipedia URI. The results are summarized in Table 6.6.

Actor Matching	Object Matching	Instrument Matching	Location Matching	Action Matching
84%	83%	79%	89%	97%

Table 6.6: Entity-to-ontology matching precision.

Minimum	Maximum	Average
1.9	3.5	3.9

Table 6.7: Comparison of the different aggregations for the event-similarity f.

Evaluation of the Event Similarity Algorithm

Both the learning and prediction algorithms strongly rely on the event similarity function *dist* described in Section 6.1.4. To evaluate the quality of this function, we randomly sampled 30 events from the training data and found for each the most similar event from the entire past data (according to the similarity function). A human evaluator was then asked to evaluate the similarity of these events on a scale of 1–5. We repeated the experiment, replacing the average aggregator function f with minimum and maximum functions.

The results are presented in Table 6.7. The general precision of the average function was high (3.9). Additionally, the average function performed substantially (confirmed by a t-test) better than over the minimum and maximum. This result indicates that distance functions that aggregate over several objects of the structured event (rather than just selecting the minimum or maximum of one of the events) yield the highest performance.

The Importance of Abstraction

Given a cause event whose effect we wish to predict, we use the algorithm described in Section 6.1.4 to identify similar generalized events. To evaluate the importance of this stage, we compose an alternative matching algorithm, similar to the nearest-neighbor approach (as applied by Gerber et al. 2010), that matches the cause event to the cause events of the training data. Instead of building an abstraction tree, the algorithm simply finds the closest cause in the past based on text similarity. We then rank the matched results using TF-IDF measure.

We applied both our original algorithm and this baseline algorithm on the 50 events used for prediction. For each event, we asked a human evaluator to compare the prediction of the original and the baseline algorithm. The results showed that in **83%** of the cases the predictions with generalization were rated as more plausible than those of the nearest-neighbor approach without generalization.

Analysis of Rule Generation Application

In order to generate an appropriate prediction with respect to the given cause event, a learned rule is applied, as described in Section 6.1.5. We observe that in **31%** of the predictions, a non-trivial rule was generated and applied (that is, a non-NULL rule that does not simply output the effect it observed in the matched past cause-effect pair example). Out of those, the application predicted correctly in more than **90%** of the cases and generated a plausible object in the effect. These results indicate that generalization and rule-generation techniques are essential to the performance of the algorithm.

Analysis of Pruning Implausible Causation

To eliminate situations in which a generated prediction is implausible, we devised an algorithm (Section 6.1.7) that prevents implausible predictions. We randomly selected 200 predictions from the algorithm predictions based on the human-labeled events extracted from the Wikinews articles (see Section 6.3.1). A human rater was requested to label predictions that are considered implausible. We then applied our filtering rules on the 200 predictions as well. The algorithm found 15% of the predictions to be implausible with 70% precision and 90% recall with respect to the human label. A qualitative example of a filtered prediction is "Explosion will surrender" for the cause event "Explosion in Afghanistan kills two."

6.3.3 Qualitative Analysis

For a better understanding of the algorithm's strengths and weaknesses we now present some examples of results. Given the event "Louisiana flood," the algorithm predicted that [number] people will flee. The prediction process is illustrated in Figure 6.11.

1. *Raw data*:

   The above prediction was based on the following raw news articles:

   (a) 150000 flee **as** hurricane nears North Carolina coast.

   (b) A million flee **as** huge storm hits Texas coast.

   (c) Thousands flee **as** storm whips coast of Florida.

   (d) Thousands in Dallas Flee Flood **as** Severe Storms Move Southwest.

2. *Causality pair extraction*:

   The "as" template was used to process the above headlines into the following structured events:

(a) *Cause Event*: near (Action); hurricane (Actor); Coast(Object); North Carolina (Object Attribute) ; (Instrument); Carolina (Location); 31 Aug 1993 (Time).
Effect Event: flee (Action); People (Actor); 150000(Actor Attributes); Carolina (Location); 31 Aug 1993 (Time).

(b) *Cause Event*: hit (Action); Storm (Actor); Huge (Actor Attributes); Coast(Object); Texas (Object Attribute); Texas (Location); 13 Sep 2008 (Time).
Effect Event: flee (Action); People (Actor); million(Actor Attributes); Texas (Location); 13 Sep 2008 (Time).

(c) *Cause Event*: whip (Action); Storm (Actor); Coast(Object); Florida (Object Attribute); Florida (Location); March 19, 1936 (Time).
Effect Event: flee (Action); People (Actor); thousands(Actor Attributes); Florida (Location); March 19, 1936 (Time).

(d) *Cause Event*: move (Action); Storm (Actor); Severe (Actor Attributes); Dallas (Location); May 27, 1957 (Time).
Effect Event: flee (Action); People (Actor); thousands(Actor Attributes); Flood(Object); Dallas (Location); May 27, 1957 (Time).

3. *Learning the abstraction tree*:

   The above four events were clustered together in the AT. They were clustered in the same node because the causes were found to be similar: the actors were all weather hazards and the location was a state of the United States. The effects were found to be similar as the actions and actors were similar across all events, and the actor attributes were all numbers. For this generalization, the following world knowledge was used:

   (a) Storm, hurricane and flood are "weather hazards" (extracted from the in-category relation in Wikipedia).

   (b) Carolina, Texas, and California are located in the "United States" (extracted from the located-in relation in Yago).

4. *Prediction*:

   During the prediction, the event "Louisiana flood" (which did not occur in the training examples) was found most similar to the above node, and the node rule output was that [number] people will flee.

As another example, given the event "6.1 magnitude aftershock earthquake hits Haiti," the highest matching predictions were: "[number] people will be dead," "[number] people will be missing," "[number] magnitude aftershock earthquake will strike island near Haiti" and "earthquake will turn to United States

Virgin Islands." The first three predictions seem very reasonable. For example, the third prediction came from a rule that natural disasters hitting countries next to a shore tend to affect nearby countries. In our case it predicted that the earthquake will affect the United States Virgin Islands, which are geographically close to Haiti. The fourth prediction, however, is not very realistic as an earthquake cannot change its course. It was created from a match with a past example of a tornado hitting a country on a coast. The implausible causation filters this prediction, as it has very low PMCI, and the output of the system is "[number] people will be dead". This example is also interesting, as it issues a prediction using spatial locality (the United States Virgin Islands are [near] Haiti).

Additional examples out of the 50 in the test and their predictions can be seen in Table 6.8.

6.3.4 Discussion

In our experiments we only report the precision of our algorithms. Further experiments measuring the recall of the system are necessary. However, in our experiments each validation step required human intervention. For example, validating that a prediction occurred in the future news. In order to perform a full recall experiment one should apply the algorithm on all the news headlines reported on a certain day and measure the appearance of all the corresponding predictions in the future news. Unfortunately, performing human validation on such a large prediction space is hard. We leave the task of performing experiments to provide a rough estimate of recall to future work.

It is common practice to compare system performance to previous systems tackling the same problem. However, the ambitious task we tackled in this work had no immediate baselines to compare with. That is, there was no comparable system neither in scale nor in the ability to take an arbitrary cause event in natural language and output an effect event in natural language. Instead, we compared to the only agents we know capable of performing such a task – humans.

Although the results indicate the superiority of the system over such human agents, we do no claim that the system predictions perform better than humans. We rather provide evidence that the system provides similar predictions to that of humans, and sometimes even outperforms human ability to predict, as can be supported by the superiority of the system in the accuracy evaluation.

To fully support the claim of superiority of the system over humans, wider experiments should be performed. Experiments larger by an order of magnitude can provide results with higher agreement between raters and shed light on the different types of events where the system's performance is better. Additionally, more experiments comparing the system performance to that of experts in the fields of each individual prediction can be valuable as well. At this point, we

assume the performance of experts would be higher than that of our algorithm. The main reason for this is the causality knowledge used to train the algorithms. This knowledge is extracted from headlines that tend to have simple causality contents, which is easily understandable by the general population. This type of knowledge limits the complexity of the predictions that can be made by Pundit. Pundit predictions therefore that tend to be closer to common knowledge of the average human. In order to predict more complex events we would need to rely on better training examples than news headlines alone.

The evaluation presented in this section provides evidence of the quality of the predictions that the system can provide. Our results are impressive in the sense that they are comparable to that of humans, thus providing evidence to the ability of a machine to perform one of the most desirable goals of general AI.

Procedure FINDPREDICATEPATHOBJECTS($entity, path = \langle l_1 \dots l_k \rangle$)

(1) $Candidates \leftarrow \{\}$
(2) $Q \leftarrow$ new Queue
(3) $Q.enqueue(entity)$
(4) $labelIndexInPath = 1$
(5) **If** path.Count == 0: **Return** {entity}
(6) **While** $Q \neq \emptyset$:
  $cur \leftarrow Q.dequeue()$
    Foreach $edge \in \{edge \in cur.OutEdges | edge.label = path[labelIndexInPath]\}$:
      If $labelIndexInPath = k$:
        $Candidates \leftarrow Candidates \bigcup \{edge.Destination\}$
      Else :
        If $labelIndexInPath > k$: **Return** $Candidates$
        $Q.enqueue(edge.Destination)$
        $labelIndexInPath \leftarrow labelIndexInPath + 1$
(7) **Return** $Candidates$

Procedure APPLYPREDICATECLAUSE($\langle P, O_1, \dots, O_4, t \rangle, rule$)

Foreach $i = 1...4$
  $O_i^{prediction} \leftarrow \emptyset$
  Foreach $path = \{O_j, \{l_1 \dots l_k\}\} \in rule(i)$
    $O_i^{prediction} \bigcup$ FindPredicatePathObjects($O_j, \langle l_1 \dots l_k \rangle$)
Return $\langle O_1^{prediction} \dots O_4^{prediction} \rangle$

Figure 6.7: Procedure for applying a rule to a new given event. The main procedure is ApplyPredicateClause. This procedure generates the objects of the predicted event $O_1 \dots O_4$ given a rule. The rule is a list of lists of tuples. Each tuple is a concept and a path. For each such tuple the function FindPredicatePathObjects is applied. This procedure finds objects that have a path whose labels connect to the given concept. Those objects are stored in Candidates (step 1). The algorithm holds a queue Q with the frontier (step 2). The queue first holds the given entity (step 3). The procedure holds a counter indicating whether we followed the entire given path (step 4). The algorithm then checks whether there is an edge with the label path[labelIndexInPath] going out of the object at the head of the frontier. When the algorithm reaches the end of the given path ($labelIndexInPath = k$), it returns the candidates.

Procedure PRUNING IMPLAUSIBLE EFFECTS($ev = \langle P_i, O_1, \ldots, O_4, t \rangle, generalizationBound$)
(1) **Foreach** $j \in 1 \ldots 4$:
generalizationPath $= \{\}$
For $i \in 0 \ldots generalizationBound$
$Gen(O_i) \leftarrow FindPredicatePathObjects(O_j, generalizationPath)$
generalizationPath $\leftarrow generalizationPath \bigcup \{IsA\}$
(2) **Return** $Average_{i,j,i \neq j}(Max_{o_1 \in Gen(O_i), o_2 \in Gen(O_j)} PMCI(o_1, o_2, i, j))$

Figure 6.8: A procedure for calculating the PMCI of an event. The procedure, at step 1, first generates all generalizations of type IsA of an object (with a path whose length is at most generalizationBound). For this purpose it uses the function FindPredicatePathObjects (defined in Figure 6.7). The generalization procedure is repeated on all objects comprising the event ev, and the result is stored in Gen. The final result of the algorithm is calculated in step 2. For two objects (o_1, o_2) in the generalization (Gen), which also contains the original objects, we find the maximum PMCI. We then compute the final result by averaging over this maximum PMCI.

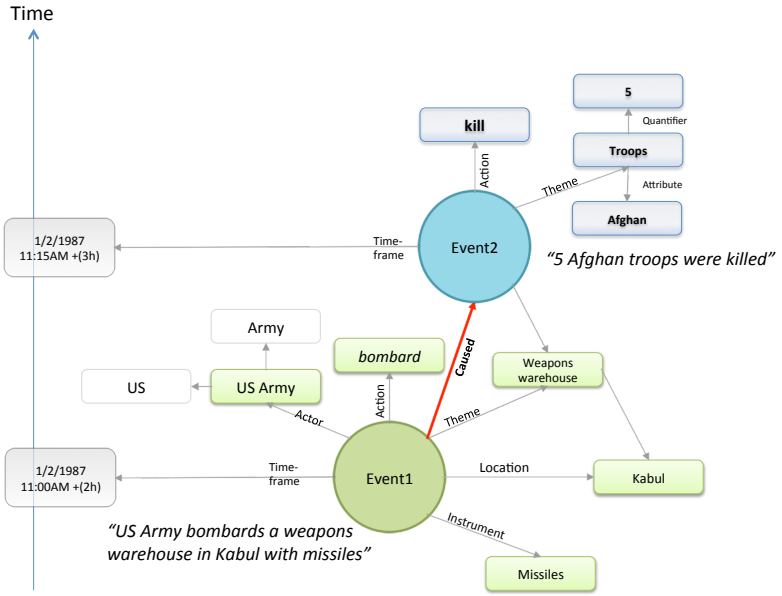

Figure 6.9: A pair of events in the causality graph. The first represents a cause event and the second represents the effect event. Both were extracted from the headline published on 1/2/1987: 5 Afghan troops killed after US army bombards warehouse in Kabul.

Class Hit-18.1		
Roles and Restrictions: Agent[int control] Patient[concrete] Instrument[concrete]		
Members: bang, bash, hit, kick, ...		
Frames:		
Example	Syntax	Semantics
Paula hit the ball	Agent V Patient	cause(Agent, E) manner(during(E), directedmotion, Agent) !contact(during(E), Agent, Patient) manner(end(E),forceful, Agent) contact(end(E), Agent, Patient)

Figure 6.10: VerbNet Template.

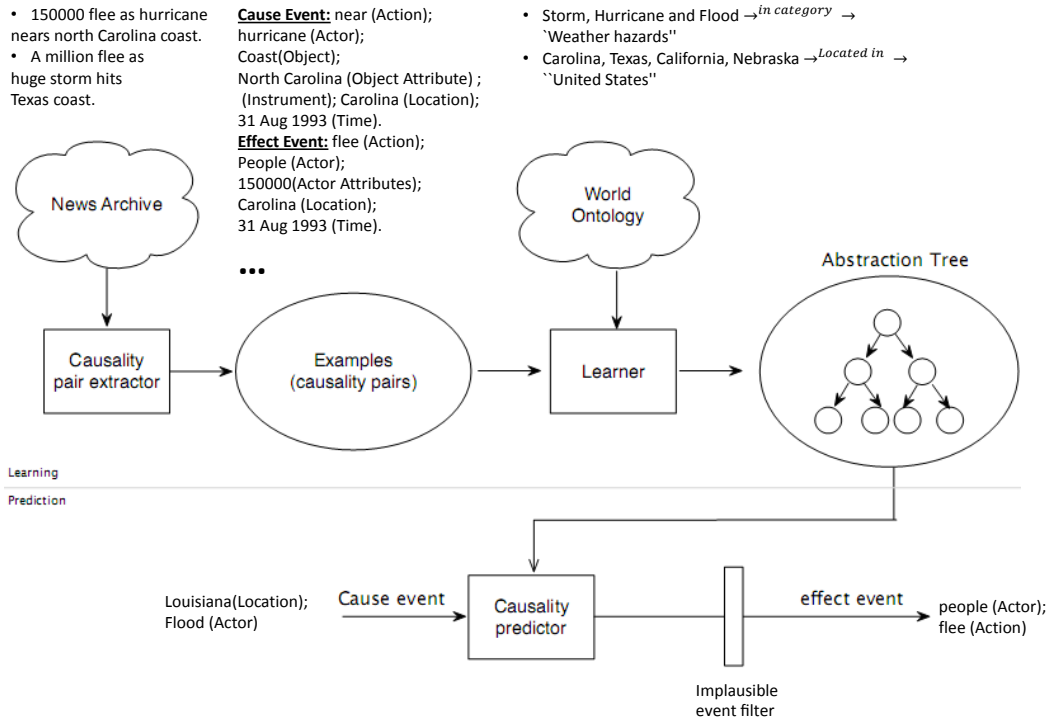

Figure 6.11: Examples of a prediction

Cause event	Human-predicted effect event	Algorithm-predicted effect event
Al-Qaida demands hostage exchange	**Al-Qaida exchanges hostage**	A country will refuse the demand
Afghanistan's parliament rejects Karzai's cabinet nominations	**Parliament accepts Karzai's cabinet nominations**	**Many critics of rejection**
Remains of 1912 expedition plane found in Antarctica	Europe museums vie for remains	**Enduring mystery will be solved in Antarctica**
North Korea seeks diplomatic relations with the US	UN officials offer mediation services	**North Korea rift will grow**
Volcano erupts in Democratic Republic of Congo	Scientists in Republic of Congo investigate lava beds	**Thousands of people flee from Congo**
Iceland's President vetoes repayment of Ice save losses	**Banks in Reykjavik report record withdrawals**	**Official administration reaction issued**
Death toll from Brazil mudslides rises to sixty	**Rescuers in Brazil abandon rescue efforts**	**Testimonies will be heard**
7.0 magnitude earthquake strikes Haitian coast	**Tsunami in Haiti affects coast**	**Tsunami warning is issued**
2 Palestinians reportedly shot dead by Israeli troops	Israeli citizens protest against Palestinian leaders	**Israeli troops will face scrutiny**
Professor of Tehran University killed in bombing	Tehran students remember slain professor in memorial service	**Professor funeral will be held**
Alleged drug kingpin arrested in Mexico	Mafia kills people with guns in town	**Kingpin will be sent to prison**
UK bans Islamist group	Islamist group would adopt another name in the UK	**Group will grow**
China overtakes Germany as world's biggest exporter	German officials suspend tariffs	**Wheat price will fall**
Cocaine found at Kennedy Space Center	Someone will be fired	People will be arrested

Table 6.8: Human and algorithm predictions for events. Predictions in bold were labeled by the evaluators as correct predictions.

Chapter 7

Predicting the Future using Web Knowledge and Dynamics

In the previous chapter, we presented methods for leveraging Web knowledge and large-scale digital histories of news reports from the New York Times (NYT) archive to generate on-demand predictions in natural language. In this chapter, we utilize both Web dynamics and Web knowledge for predictions. We mine sequences of events and present methods to predict the future by generalizing sets of concrete *transitions* in sequences of those reported news events, extracted from a news archive spanning the years 1986–2007. The generalization is performed by means of leveraging of Web knowledge. The goal is to build predictive models that generalize from specific sets of sequences of events to provide likelihoods of future outcomes, based on patterns of evidence observed in near-term newsfeeds. We propose the methods as a means of generating actionable forecasts in advance of the occurrence of target events in the world.

As an example of the inferences provided by the system, the system can alert about the increased likelihood of a forthcoming cholera outbreak. Automated alerting based on evolving news stories could serve as adjuvants to monitoring and communication services, such as the World Health Organizations (WHO) Global Alert and Response (GAR) system for coordinating responses to public health emergencies [1]. Alerts about jumps in the likelihoods of a future cholera outbreak, based on the monitoring of news stories, would provide valuable warnings: Cholera is a fast-paced infection of the small intestine that causes over 100,000 deaths per a year. The illness is caused by the bacterium Vibrio cholera. Toxins secreted by Vibrio cholera cause a severe diarrhea which leads to rapid dehydration. Untreated, the mortality rate of cholera is believed to be over 50% [Sack et al., 2004]. With prompt rehydration therapy, the mortality rate drops to

[1] `www.who.int`

less than 1%. Early knowledge and active warnings of an impending cholera epidemic could be helpful in proactive planning for acquiring and transporting clean water, electrolytes, and antibiotics to sites where the infection is expected [Todar]. Results of our system evaluation (Table 7.4) demonstrate that such alerts about upcoming disease outbreaks can be provided on average 12 days preceding the actual event.

We propose methods for predictive modeling from news archives as a means for both building insights about the likelihoods of transitions among events of interests, as well as for providing real-time automated alerting. We show examples of such alerts in Figure 7.9, where the system automatically inferred likelihoods and timings of transitions from floods in Bangladesh to major Cholera outbreaks. A large study of real data regarding cholera epidemics in Bangladesh [Michel et al., 1992], analyzing government figures and independently collected data during 1985–1991, reached the same conclusions that our system inferred in an automated manner. The analysis showed that the number of Cholera cases and deaths in 1987 and 1988 was significantly higher than in other years (300,000-1,000,000 cases vs. approximately 50,000 cases in other years), as severe floods affected Bangladesh in both in 1987 and 1988. The study concludes that in many cases those deaths could have been prevented if access to medical care was given in key non-rural areas. When appropriate intervention was made in early stages, the death rates were significantly smaller. In tests, we found that our system would have provided an alert a week before the cholera epidemic broke.

Similar inferences can be made by experts such as epidemiologists who study the relationships between disease and natural disasters. However, such studies are typically sparse in number, employ heuristic assessments, and are typically done in a retrospective manner. In contrast, a computational system has the ability to learn patterns from large amounts of data, monitors large numbers of information sources, constantly learns new probabilistic associations, and can continue to do real-time monitoring, prediction, and alerting on rises in likelihoods of concerning events. Beyond knowledge that is easily discovered in studies or available from experts, new relationships and context-sensitive probabilities of outcome can be discovered by a computational system. As an example, the inferential system identified a relationship in Angola between droughts and storms that in turn might cause cholera outbreaks, issuing alerts about a downstream risk of cholera nearly a year in advance (Figure 7.1). Such long-term interactions may be overlooked by human experts who focus on shorter-time horizons. However, computational systems can consider multiple time granularities and horizons in pursuing the probabilistic influences among events. Beyond alerting to actionable situations based on raised likelihoods of forthcoming outcomes, the system can also assist by providing alerts when, in contrast to expert expectations of pending event, its inferences indicate a significantly lower likelihood based on the large

set of observations and feeds being considered in an automated way. Finally, the system could have faster and more comprehensive access to news stories that might seem less important (e.g., a story about a funeral published in a local newspaper that does not reach the main headlines), but that might provide valuable evidence in the evolution of larger, more important stories (e.g., massive riots).

The inferential methods we describe operate on newsfeeds and provide large numbers of predictions. We demonstrate the predictive power of mining thousands of news stories to create classifiers for a broad range of prediction problems – trying to predict all the news events in 2006–2007 (Table 7.1). In addition, we also dwell on three prediction challenges, including proactive alerting on forthcoming disease outbreaks, deaths, protests, and riots (Figures 7.9, 7.1, 7.10). Those are of specific interest, based on the possibility to provide actionable prevention methods if identified early. We compare the prediction power to several baselines and demonstrate that the precision of our system (positive predictive value) in these domains ranges from 70% to 90% with a recall (sensitivity) of 30% to 60%.

7.1 Event Prediction

We assume that events in the real-world are generated by a probabilistic model that also generates news reports corresponding to these events. We use the text of news stories to build an inferential model of the form $P\big(ev_j(\tau + \Delta)\big|ev_i(\tau)\big)$ for some future event ev_j at time $\tau + \Delta$ and past event ev_i happening at time τ (e.g., today). For example, the model learns that the probability of a news report about a drought (ev_j) happening after a news report about a flood (ev_i) to be 18%. This probability approximates the relationship between the two real-world events.

Given a target future event (such as cholera outbreak), calculating this probability for every possible future time $\tau + \Delta$ and every possible ev_i is an intractable problem. We simplify the analysis by focusing on a small subset of event sequence candidates that may be causally linked, and define sets of events ev_i that are linked to target events ev_j in this manner. In particular, we define and extract from the NYT archive news *storylines*—sets of topically cohesive ordered segments of news that include two or more declarative independent clauses about a single story. As an example, the following events form a storyline: {(drought in Africa, 02/17/2006), (storm in Rwanda, 01/26/2007), (flood in Rwanda, 01/27/2007), (cholera outbreak in Rwanda,
01/30/2007)}. We then use such storylines as a heuristic for identifying possible causal relationships among events. The process is performed by clustering news

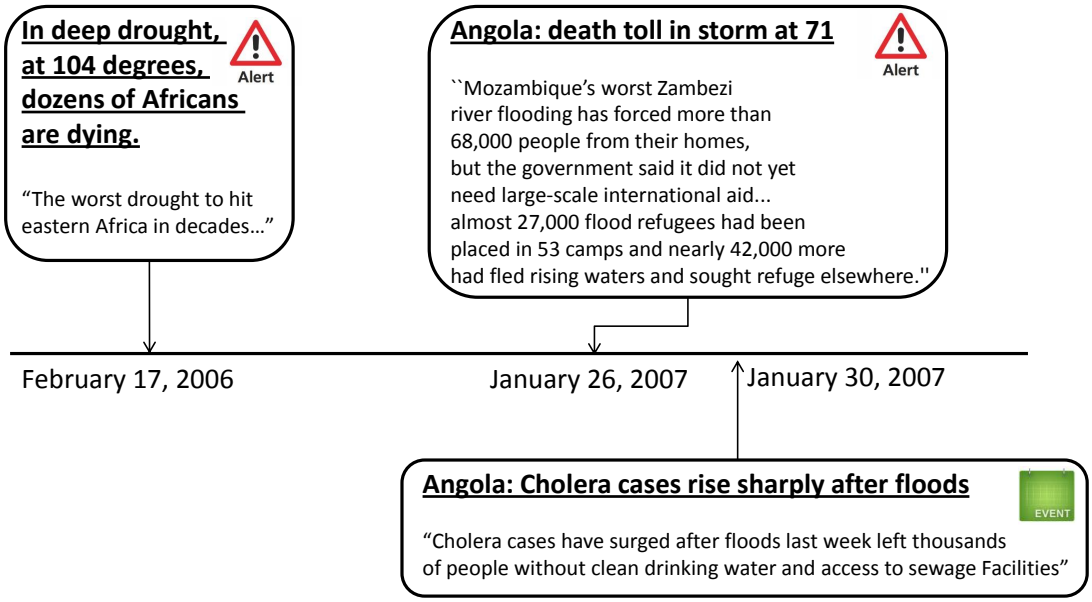

Figure 7.1: The system learned from a news archive of multiple incidents that the likelihood of a cholera outbreak is higher after droughts, as observation of drought reports are linked to increasing probability of forthcoming water-related disasters, which, in turn, are linked to increases in the likelihood of waterborne diseases. The system learns that not all droughts are associated with jumps in likelihood of such disease outbreaks, identifying specific sets of preconditions that influence the likelihood that a transition from a report of drought to a report of cholera outbreak will occur: (1) dense populations (such as the refugee camps in Angola and Bangladesh) that are (2) located in underdeveloped countries and are (3) proximal to water resources.

stories with similar text and semantic entities, as detailed in Section 7.1.1.

We show a componentized view of the method in Figure 7.2. At the start of the learning phase, the system mines the NYT news corpora and extracts storylines, using techniques adapted from well-known topic tracking and detection algorithms [Allan, 2002; Carbonell et al., 2000; Cieri et al., 2000], that cluster similar texts together (Section 7.1.1). We next enrich the storylines with information extracted from Web knowledge sources via the LinkedData project(Section 7.1.2). We extract a wide variety of facts, including such information as the population density in Rwanda, percentage of land in Rwanda covered by water, and the gross domestic product of the country. We generalize both features and events to increase the number of equivalent samples for constructing predictive models

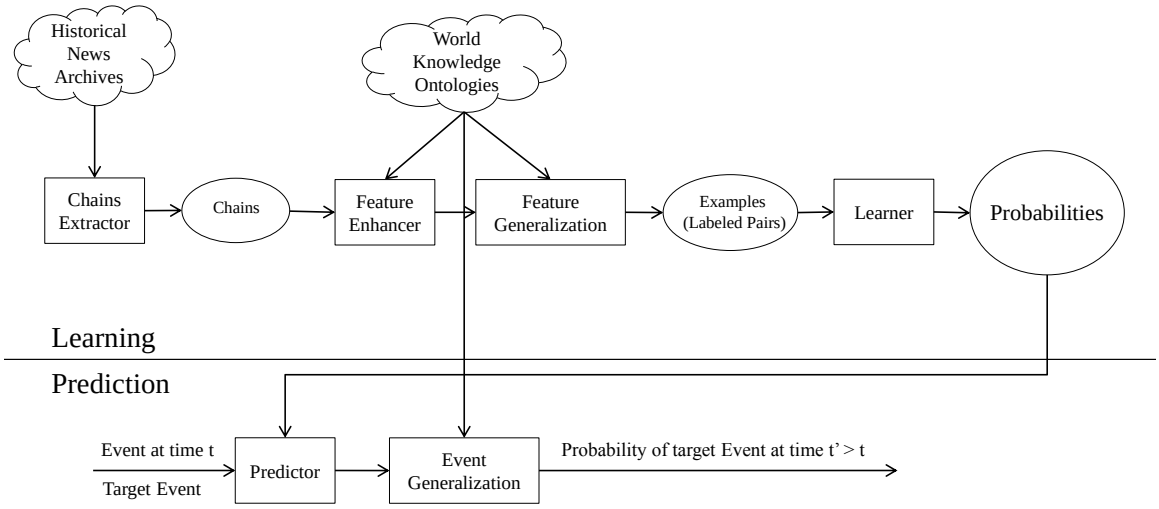

Figure 7.2: Main components and flow of analysis of event prediction pipeline.

(Section 7.1.3). For example, we can learn from data about events in specific countries (e.g., Angola and Rwanda) to build classifiers that consider the likelihood of events of interest on a larger scale (e.g., larger continent of Africa) or to regions characterized by particular demographic and geological properties. At the end of the learning phase, the system estimates the probabilities $P\big(ev_j(\tau + \Delta)\big|ev_i(\tau)\big)$ and builds a probabilistic classifier for use in the prediction phase. The classifier can be used to provide real-time probabilities of events of interest, such as an impending "cholera outbreak in Angola" based on the previous knowledge obtained in a storyline about Angola or its generalization, Africa. The classifier we construct provides binary predictions of whether an event will occur following an observed event sequence. In experiments, we also evaluate both the occurrence of the target event and the mean time between the prediction and occurrence. We show results on providing such alerts nearly three weeks prior to the actual predicted event. We leave the prediction of the exact date of events of interest based on specific dynamics to future work.

7.1.1 Extracting Event Chains

We define and extract news storylines from the NYT archive as a heuristic for identifying potential causal relationships among events. A storyline is a set of

topically cohesive ordered segments of news that includes two or more declarative independent clauses about a single story. As an example, a story line about the arrest of Carlos the Jackal includes the stories about verification of his identity, his transport to prison, and so on. Methods for extracting such storylines are referred to as *topic detection and tracking* (TDT) [Cieri et al., 2000]. Topic detection involves identifying a series of linked events in streams of stories. To identify storylines, we modified the Inc.LM method, an approach to topic tracking found to be most successful for this task in several competitions [Allan, 2002]. Consider $Chains \in 2^{2^{|T| \times Time}}$ as the set of all possible storylines, where T is all the news articles and $Time$ is a discrete representation of time. We denote with $t_1 <<_c t_2$ an event represented by the news article t_1 occurring before an event represented by the news article t_2 in a chain $c \in Chains$. We use the notation $t(\tau)$ to represent an event as defined by the appearance of the text t of a news story at time $\tau \in Time$. Under the assumption that causality occurs only within storylines, the prediction challenge is reduced to calculating the probability $P\big(t(\tau_i > \tau) \big| t_j(\tau)\big)$ for $\{t_j | \exists c \in Chains, t_j <<_c t\}$.

Similar to other vector space approaches for topic detection [Carbonell et al., 2000], we first cluster documents with similar text. We consider news articles as documents and represent each news article as a vector $(\sigma_1^t \dots \sigma_n^t)$, such that

$$\sigma_i^t = \mathrm{tf}_{w,t} \cdot \log \frac{|T|}{|\{t' \in T \,|\, w_i \in t'\}|},$$

where $|T|$ is all the news articles, and $\mathrm{tf}_{w,t}$ is the frequency of the word w in article t. We then perform a nearest-neighbor analysis, where we find for each article the k closest (in our experiments $k = 50$) articles to it using a cosine similarity measurement, defined as

$$\mathrm{sim}(t_a, t_b) = \frac{\sum_{i=1}^{N} \sigma_i^{t_a} \sigma_i^{t_b}}{\sqrt{\sum_{i=1}^{N} {\sigma_i^{t_a}}^2} \sqrt{\sum_{i=1}^{N} {\sigma_i^{t_b}}^2}},$$

with a constraint on temporal proximity. Articles are either generated within a threshold time horizon or the date of an article is mentioned in a text of a more recent article in the chain. We performed several experiments using the time threshold on the TDT4 corpus[2], and reached our best performance when limiting the chains to 14 days. This type of analysis has a high recall for identifying articles that cover the same topic, referring to the fraction of relevant instances that are retrieved. However, the procedure has a low precision; a large fraction of the identified instances are false positives. We wish to enhance the precision, while maintaining the high recall. As an approach to reducing the false positives,

[2] `http://www.nist.gov/TDT`

we overlay a preference that the entropy of the entities $\{e \in Entities\}$ of the story articles C, defined as

$$StoryEntropy(C) = -\sum_{i=1}^{n} P(e_i \in C) \log P(e_i \in C)),$$

grows "slowly" as the story evolves over time. A similar approach has been shown to provide major improvements on a related topic-detection task [Ahmed et al., 2011a]. We employ conditional random fields (CRF), trained on a heterogeneous corpus [Finkel et al., 2005], to identify entities of the types *location*, *people*, and *organizations*. We define a vector of counts of entities and operations of addition and removal of an article from a chain. We use a greedy algorithm that selects at each step the next best document to add or decides to halt if all remaining documents increase the entropy by more than a threshold amount α, which we evaluate from a validation set. We performed experiments showing that this extension improves precision while maintaining levels of recall (see Section 7.2.4).

We performed the latter process on an offline corpus. We note that studies have addressed the usage of similar techniques for extraction from online streams of news (e.g., [Ahmed et al., 2011a]). Such online approaches could be adapted in our approach to forecasting future events.

7.1.2 Lexical and Factual Features

We seek to infer the probability of a predefined future news event of interest given a vector representing the news events occurring up to a certain time. To perform this task, we create training cases for each target event, where each case is represented using a set of observations or features. We define both *lexical* and *factual* features. We set the label for each case as true only if the text representing the future target event occurs in a document dated at a later time in the chain.

Let $w_1 \ldots w_n$ be the words representing the concepts of the event at time τ and let $a_1 \ldots a_m$ be additional real-world characteristics of the event concepts. We refer to these attributes respectively as *lexical and* factual features. The words w_i are extracted from the text of each news article using the Stanford Tokenizer, and filtered using a list of stop words. The factual characteristics a_i are extracted from the different LinkedData sources, specifically the properties under the type rdf:Property for the event concepts w_i. For example, given the text of the news story title, "Angola: Cholera Cases Rise Sharply After Floods," the feature vector contains both the tokenized and filtered words of the text (Angola, cholera, rise, sharp, flood), and other features describing characteristics of Angola (GDP, water coverage, population, etc.). We map each concept in the event text to a LinkedData concept. If several concepts are matched, we

perform disambiguation based on the similarity between the concept text (e.g., its Wikipedia article) and the news article, using a bag-of-words representation.

We denote $f_1(ev) \ldots f_{n+m}(ev)$ to be the features of the event ev (either lexical or factual features). We make a naive simplifying assumption that all features are independent, and describe the probability $P(ev_j(\tau + \Delta) \big| ev_i(\tau))$ as follows:

$$P(ev_j(\tau + \Delta) \big| ev_i(\tau)) \propto \prod_{k=1}^{n+m} P\big(ev_j(\tau + \Delta) \big| f_k(ev_i(\tau))\big).$$

Using Bayes rule, we can derive that

$$P\big(ev_j(\tau + \Delta) \big| f_k(ev_i(\tau))\big) = \frac{P\big(ev_j(\tau + \Delta), f_k(ev_i(\tau))\big)}{P(f_k(ev_i(\tau)))},$$

where $P\big(ev_j(\tau+\Delta), f_k(ev_i(\tau))\big)$ is evaluated from the data by counting how many times the event ev_j happens in the same storyline after an event having the value of feature f_k of the event ev_i at time τ. Similarly, $P(f_k(ev_i(\tau))$ is evaluated from the data by counting the portion of times that an event with this feature value happens in the corpus. We build a predictor based on these learned probabilities. The predictor outputs the probability that each future event represented in the system will occur. Changes in this probability can be noted and thresholds set for alerting.

We may be interested in predicting various characteristics of a future event as *scalar* values. For example, beyond predicting the likelihood that deaths will be associated with a later accident or disruption occurring within a horizon, we may wish to predict the number of people who will perish given the occurrence of a target event that causes or is associated with deaths. To do this, we bin target predictions of numbers of deaths into a set of buckets capturing a mutually exclusive and exhaustive set of ranges of numbers of deaths, e.g., less than ten deaths, greater than 10 but less than 100 deaths, and greater than 100 deaths. We say that ev_j belongs to bin_k, if ev_j holds the bin_k relation. As an example, for the less than 10 deaths bin, we say that $ev_j \in bin_{0-10}$ if the text representing ev_j contained text indicating how many people died and this number was less than 10. We learn predictors that estimate the probability of the event to belong to a certain bin k, $P\big(ev_j(\tau + \Delta), ev_j(\tau + \Delta) \in bin_k \big| ev_i(\tau)\big)$, and output the bin with the highest probability. We present results on a study of the accuracy of inferring the number of deaths caused by an event in Section 7.2.2.

7.1.3 Learning to Predict with Abstractions

Event sequences are relatively sparse in the domains we explored. For example, the target event "Rwanda cholera outbreak" appeared only 33 different times in

the news archive. Such sparsity may degrade classifier performance due to the poor estimation of both $P\big(ev_j(\tau+\Delta), f_k(ev_i(\tau))\big)$ and $P(f_k(ev_i(\tau)))$. In other cases, the feature values might not appear with high enough frequency in the data for deriving high-confidence inferences about the future. For example, there is not even a single mention of the African country of Comoros in the news corpus. Such sparsity in lexical features can lead to poor estimation of the probability $P\big(ev_j(\tau+\Delta), f_k(ev_i(\tau))\big)$, and therefore to poor predictors. As an example, for the target prediction of the likelihood of a forthcoming large-scale evacuation, a hurricane originating in Comoros might be important information for predicting a storm in nearby countries which might be useful in predicting evacuations in those countries. Similarly, if the system is focused on making predictions about upcoming storms in Comoros, i.e., "storm in Comoros" is the target event, there may not be enough data to evaluate the aforementioned probabilities.

We address the joint challenges of event and feature sparsity via employing procedures for automated abstraction. Instead of considering only "Rwanda cholera outbreak," an event with a small number of historical cases, we consider more general events of the form: "[Country in Africa] cholera outbreak." We turn to world knowledge available on the Web. Some LinkedData resources provide hierarchical ontologies. For example, Fabian et al.[Suchanek et al., 2007] created an *isA* ontology from Wikipedia content. This ontology maps Rwanda to the following concepts: Republics, African countries, Landlocked countries, Bantu countries, etc. Similarly, WordNet provides hypernym relations, that map Rwanda to the concept *country*.

We developed a an automated method for guiding abstraction. The method determines when to generalize events and features. As features are evaluated separately, estimations are made about the value of each feature abstraction to enhance the accuracy of predicting the target event. We evaluate for each feature and its abstractions the precision over the training data using cross validation. We note that it is insufficient to measure the precision associated with using an abstracted feature without altering the target event. Consider the abstracted feature [Country in Africa], and the target event "Death in Kigali." The probability of a death in Kigali, the capital of Rwanda, caused by an event in some country in Africa, is small. Therefore, given an event in [Country in Africa], the probability of death being caused by the event in CapitalOf([Country in Africa]) may typically be more appropriate. We now formalize this intuition.

Let the semantic network graph G be an edge-labeled graph, where each edge is a triplet $\langle v_1, v_2, l\rangle$ and l is a predicate (e.g., "CapitalOf"). We look for a path of maximum length k (in our experiments $k = 3$)that connects the concept representing the abstraction and the concepts described in the target event. For example, given a chain where the first event discusses the large attendance at the opera "The Nose," and the second event discusses an award

that the opera writer Dmitri Shostakovich receives, we find the following path in the Wikipedia graph, connecting the articles: "The Nose" $\xrightarrow{OperasBy}$ Dmitri Shostakovich. Later, we can use a similar observation, observing a large attendance at the opera "The Murder of Comrade Sharik," to predict an award for the opera writer William Bergsma using the path "The Murder of Comrade Sharik" $\xrightarrow{OperasBy}$ William Bergsma. Given two events' concepts c_1, c_2, represented by the nodes v_1 and v_2 in G, we call the labels of the k-sized path, connecting v_1 and v_2, an *abstraction path* $abs(c_1, c_2) = l_1, \ldots, l_k$. *Applying an abstraction* on a node v determines the node v' that satisfies the abstraction path, i.e.,
$ApplyAbs(v, abs(c_1, c_2)) = v'$,
s.t $\exists v_i \in V(G)(v, v_1, l_1) \ldots, (v_{k-1}, v', l_k) \in E(G)$.

When inferring probabilities for each entity en in the target event and a feature of the causing event, we iteratively abstract each feature f to a more general concept $gen(f)$, using a semantic hierarchical graph G^H, calculating the abstraction path $abs(f, en)$ (based on the semantic graph G), and instead of $P\big(ev_j(\tau + \Delta), f_k(ev_i(\tau))\big)$, we calculate the probability for the more abstract event,

$$P\left(ApplyAbs\left(ev_j(\tau + \Delta), abs(f_k, en)\right), gen(f_k)(ev_i(\tau))\right).$$

A similar process is conducted when generalizing the target event. In this case, the probabilities are calculated for the abstracted target event, and the precision is calculated on the concrete target event. For each entity en in the target event and a feature of the causing event, we wish to iteratively abstract each feature f to a more general concept $gen(f)$, using a semantic hierarchical graph G^H (in our experiments we used the IsA and InCategory relations).

Figure 7.3 shows pseudocode for the abstraction process. Given a target and a possible causative event, the goal of the procedure is to estimate the probability of the causative event or any of its abstractions to cause the target event. The algorithm is given as input several parameters: a target event (e.g., cholera Outbreak in Rwanda), denoted as $target$, and an event occurring at time τ ($cause$), the storylines the system extracted, denoted as $Chains$, the hierarchical graph G^H, the semantic graph (G), and some parameter specifying a maximum degree of abstraction (k). The system evaluates the probability

$$P\left(ApplyAbs\left(ev_j(\tau + \Delta), abs(f_k, en)\right), gen(f_k)(ev_i(\tau))\right).$$

At stages 1-2, the system builds a classifier estimating the probability that any of the entities of the lexical features of the causative event precede an appearance of the target event in a text of an event in a storyline. For example, the

Procedure ABSTRACT($target, cause, Chains, G^H, G, k$)
(1) **Foreach** $\{entity \in Entities(cause)\}$
(1.1) $PositiveExamples \leftarrow \{(ev_1, ev_2) | ev_1 <<_{c \in Chains} ev_2, entity \in ev_1,$
$\forall e \in Entities(target) : e \in ev_2\}$
(1.2) $NegativeExamples \leftarrow \{(ev_1, ev_2) | ev_1 <<_{c \in Chains} ev_2, entity \in ev_1,$
$\exists e \in Entities(target) : e \notin ev_2\}$
(2) $bestClassifier \leftarrow Build(PositiveExamples, NegativeExamples)$
(3) **Foreach** $\{entity \in Entities(cause), absEntity \in Abstractions(entity, G^H)\}$
(3.1) $absPaths \leftarrow FindPaths(absEntity, Entities(target), G, k)$
(3.2) $absTargets \leftarrow ApplyAbs(absEntity, absPaths, G)$
(3.2) **Foreach** $absTaret \in absTargets$
(3.2.1) $PositiveExamples \leftarrow \{(ev_1, ev_2) | ev_1 <<_{c \in Chains} ev_2, absEntity \in ev_1,$
$\forall e \in Entities(absTarget) : e \in ev_2\}$
(3.2.2) $NegativeExamples \leftarrow \{(ev_1, ev_2) | ev_1 <<_{c \in Chains} ev_2, absEntity \in ev_1,$
$\exists e \in Entities(absTarget) : e \notin ev_2\}$
(3.2.3) $absClassifier \leftarrow Build(PositiveExamples, NegativeExamples)$
(3.2.4) **If** $CV(bestClassifier, Chains) < CV(absClassifier, Chains)$
$bestClassifier \leftarrow Update(absClassifier)$
(4) **Return** bestClassifier

Figure 7.3: Procedure for generalizing features via abstraction. *Build* takes as input positive and negative examples and estimates the probability of our target event. *FindPaths* finds all predicate paths of size k between two nodes in the graph given as input. *ApplyAbs* applies the predicate path on a node, returning nodes that are connected to the given node via the predicates of the directed paths. *CV* calculates the precision via cross validation of a classifier on the training data.

$bestClassifier$ at this stage will have estimations of the probability of "cholera Outbreak in Rwanda" given the entity Kigali (Rwanda's capital). At stage 3, the algorithm iteratively estimates the probability of the target event happening given any of the abstracted features (extracted using the hierarchical graph G^H). For example, one iteration can be the evaluation of the number of times the entity "Capital of Africa" preceded "cholera Outbreak in Rwanda" in our storylines. Stages 3.1-3.2 evaluate the needed transformations to the target event given the abstracted cause entity. For example, instead of looking for cases where an event with an entity belonging to "Capitals in Africa" occurred and an event regarding "cholera Outbreak in Rwanda" followed, we look for examples where an event of the type "cholera Outbreak in Africa" followed. We then train and evaluate new classifier using the transformed training data. If its performance, as measured by cross validation on the training data, is superior to that of the classifier in advance of the abstraction, we update the best classifier found.

7.2 Experimental Evaluation

We now describe the experiments that we conducted to test the methodology, and present the results of the studies of inferences performed on a test portion of the news archive held out from the training phase.

7.2.1 Experimental Setup

In this Section we outline the data we obtained for the experiments, the experimental methodology, and the baselines we compared against.

Data

We crawled and parsed the NYT archive containing news articles for the years 1986–2007. We say that a chain of events belongs to a domain D, if it consists one of the domain relevant words, denoted as $w_i(D)$. For example, for the challenge of predicting future deaths, we consider the words "killed," "dead," "death," and their related terms.[3] For the challenge of predicting future disease outbreak, we consider all mentions of "cholera, " "malaria, " and "dysentery."

During prediction, we hold out from the learning phase a test set of a decade of events for the period of 1998–2007 (the *test period*). We say that a chain is a *test-domain chain* if (1) the dates of all of its events occurred in the test period dates, and (2) the first chronological event in the chain does not contain one of the domain terms, e.g., the first event did not contain a mention of death (otherwise the prediction might be trivial). Formally, let $C = \{e_1 \dots e_k\}$ be a test chain, thus $\forall i : w_i(D) \notin e_1$.

Experimental Methodology

For each prediction experiment we first select a target event e_{target} from a test-domain chain. The procedure differs depending on the type of the experiment:

1. Predicting general events in the period 2006–2007. In this type of experiment, a target event is any news headline published during 2006–2007, i.e., we build a classifier for each possible headline.

2. Predicting events in specific three domains: deaths, disease outbreaks, and riots. In this case, any news story containing one of the domain words is selected. Additionally, we validate manually that those events actually

[3]We consider all the similarity relations in Wordnet: Synonyms, pertainyms, meronyms/holonyms, hypernyms/hyponyms, *similar to*, *attribute of*, and *see also* relations.

contain an event from the domain. If several of the target events exist, we choose the first one appearing chronologically to be the identified target event, i.e., $e_{target} = argmin_j\{e_j|\exists i : w_i(D) \in e_j\}$. As e_{target} is selected from a test-domain chain $j > 1$, i.e., it is not the first event in the chain. That is, we consider only event chains that are not observed by the system during the years 1998–2007, and do not contain words implying the target event within a domain (e.g., the word death) during the first event chain. The first event of the chain is given as input to the system.

In summary, the general events predictions represents prediction of *all* the events in 2006-2007. The system is given an event from 2006–2007 as input, and we measure the success in predicting the event. For the domain-specific predictions (death, disease outbreak, and riots), we manually check to see if the event occurs using the domain representative words or their synonyms as filters. We consider only event chains that are not observed during the years 1998–2008, and do not contain words implying the target event within a domain (e.g., the word death) during the first event chain. The first event of the chain is given as an input.

We train from the data via evaluating the probabilities of e_{target} happening for events occurring up until the date of the first event in the chain. During the test, the algorithm is presented with the first event of the chain e_1 and outputs its prediction about e_{target}. In the experiments, we consider the predictor as indicating that the target event will occur if

$$P(e_{target}\big|e_1) > P(\neg\, e_{target}\big|e_1),$$

i.e., the probability of the event happening given e_1 is bigger than the probability of it not happening. We perform these experiments repeatedly over all the relevant chains, and evaluate for each:

$$\text{precision} = \frac{|\{\text{events reported}\} \cap \{\text{predicted events}\}|}{|\{\text{predicted events}\}|}$$

and

$$\text{recall (sensitivity)} = \frac{|\{\text{events reported}\} \cap \{\text{predicted events}\}|}{|\{\text{events reported}\}|}.$$

Comparative Analysis

We are not aware of any methods in the literature that are aimed at tackling the prediction of probabilities of future news events. Thus, we compare the generated predictions with two baselines:

	General Predictions		Death		Disease Outbreak		Riots	
	Prec.	Recall	Prec.	Recall	Prec.	Recall	Prec.	Recall
Full model	24%	100%	83%	81%	61%	33%	91%	51%
Frequency	<1%	100%	59%	<1%	13%	3%	50%	1%
Co-occurrence	7%	100%	46%	61%	40%	<1%	61%	14%

Table 7.1: Precision and recall of predictions for several domains.

1. using prior probabilities of the occurrence of an event e given the appearance of its corresponding text in the training set, $P(e)$;

2. using an estimate of how well people do on predicting these types of events.

For the latter, we implement a method Glickman et al. [2005] that provides approximations of whether people, given two events represented by text of news stories, would agree that the first event implies the truth of the later event. This baseline evaluates the probabilities of co-occurrence in text rather than in time.

7.2.2 Prediction Results

We performed experiments evaluating the precision and recall for the general predictions and for each of the different domains. We compare our model (Full model) with the frequency-based model (Frequency), and the co-occurrence-based method (Co-occurrence). The results are presented in Table 7.1. We observe that in all cases the Full model outperforms the baselines.

We performed additional experiments to evaluate numerical predictions, such as forecasts of numbers of deaths. For this purpose, we searched for specific patterns in the news stories of the form "[number] died" or "[number] killed". The number matching [number] is used as the bin classification. We focus only on chains containing those patterns and evaluate our algorithms on those. In Table 7.2, we show a confusion matrix for the numbers of deaths. The content of each cell i, j (i is row and j is column) represents the percentage of the data that in reality belongs in bin i and is classified as belonging in bin j. For example, 4% of the events resulting in tens of deaths are predicted erroneously as events associated with only a few (less than ten) deaths. We see high performance in those types of classifications and most mistakes are observed in adjacent bins.

7.2.3 Algorithm Analysis

We now describe additional experiments performed to measure the performance of the procedures and the contribution of specific components of the analysis.

		Predicted		
		Few	Tens	Hundreds
Real	Few	40%	6%	1%
	Tens	4%	32%	1%
	Hundreds	1%	6%	9%

Table 7.2: Confusion matrix showing predicted versus actual number of deaths.

	General Predictions		Death		Disease		Riots	
	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.
News alone	19%	100%	80%	59%	44%	34%	88%	38%
News + factual features	19%	100%	81%	62%	52%	31%	87%	42%
News + generalization	21%	100%	81%	67%	53%	28%	88%	42%
Full model	24%	100%	83%	81%	61%	33%	91%	51%

Table 7.3: Precision and recall for different algorithm configurations.

Gain from Factual Features and Generalization

We first consider the influence of adding different sources of world knowledge on the accuracy of predictions. The results are displayed in Table 7.3. We consider predictors based solely on lexical features (News alone), on both lexical and factual features (News + factual features), on lexical features and abstractions (News + generalization), and on using all categories of features along with the abstraction procedure (Full model). We find that adding knowledge, either when abstracting or when adding factual features, improves the performance of predictions. We see the biggest performance gain when employing both refinements.

Algorithm Performance at Different Threshold cut-offs

The algorithm we described in this work outputs that the target event will occur if

$$P(e_{target}|e_1) + \alpha > P(\neg\, e_{target}|e_1),$$

i.e., the probability of it happening given e_1 is bigger than the probability of its not happening. In the experiments described in this work we used $\alpha = 0$. We investigated several other thresholds: $\alpha = 0.1, 0.2, 0.3, 0.5, 0.7$, and we present the behavior curves in Figures 7.4, 7.5 and 7.6 for disease, riots and death predictions. As expected, the higher the precision becomes the lower the recall.

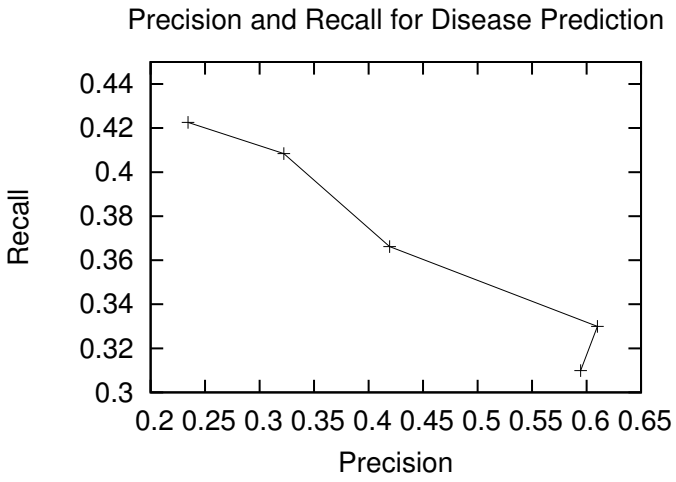

Figure 7.4: Precision and Recall of the algorithm as a function of different prediction thresholds for disease prediction.

General Predictions		Death		Disease Outbreak		Riots	
Med.	Avg.	Med.	Avg.	Med.	Avg.	Med.	Avg.
9	21	8	41	12	273	18	30

Table 7.4: Median and average time between alerts based on inferred probabilities of outcome and target events in the world (days).

Predicting Times of Forthcoming Events

Table 7.4 displays the average and median times between the inference-based alerts and the occurrence of reports that embody the target event for the three types of predictions we study. We consider only examples where deaths appear in the news story title and match a handcrafted template (patterns in the text of the form "[number] died" or "[number] killed") to identify certain deaths only on test chains. This procedure results in 951 death predictions. In many cases, we find that the alerts would come more than a week in advance of the target event. We illustrate this phenomenon in Figure 7.7, where we show predictions of the number of deaths that come at a future time within a storyline predicted at different times between the alert and the occurrence of the deaths. A more detailed view of the timing of alerts at two and fifteen days before the event are displayed in Figure 7.8.

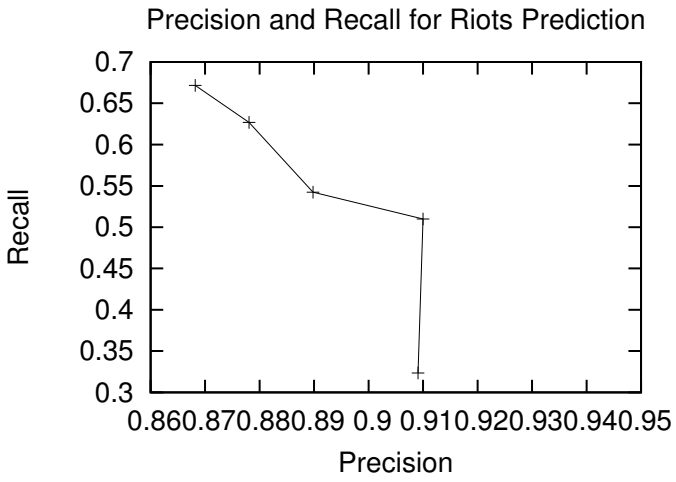

Figure 7.5: Precision and Recall of the algorithm as a function of different prediction thresholds for riots prediction.

7.2.4 Event Chain Extraction Evaluation

To evaluate the quality of the extracted event chains, we performed experiments on the TDT4 corpus[4], filtering only NYT articles. This corpus contains about 280,000 documents from the dates 04/01/2003–09/30/2003. Human annotation for labeling storylines was performed by the organizers of the TDT challenge. For each chain, we calculate the average precision—the percentage of articles we extracted as being in a chain that were indeed part of the storyline. We also compute the average recall, the number of articles actually in the chain that the system retrieved. We compared the event chain extractor using the entity entropy measure with the extractor working without the entropy measure. The results are summarized in Table 7.5. The results show that, while the recall of text clustering is very high (by 10%), the precision is significantly lower than the methods we have presented (by 30%). We therefore prefer the second method, as it provides more flexibility in training predictive models in a more precise way, with influence on the number of examples used for training the learner.

	Precision	Recall
Text Clustering	34%	**80%**
Text Clustering + Entity Entropy	**70%**	63%

Table 7.5: Precision and recall for chain extraction procedure.

[4] `http://www.nist.gov/TDT`

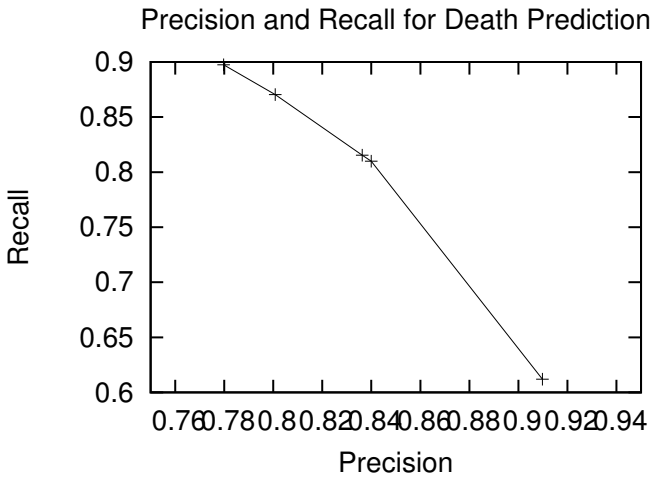

Figure 7.6: Precision and Recall of the algorithm as a function of different prediction thresholds for death prediction.

Cause	Effect	Probability
Drought	Flood	18%
Flood	cholera	1%
Flood in Rwanda	cholera in Rwanda	67%
Flood in Lima	cholera in Lima	33%
Flood in Country with water coverage $> 5\%$	cholera in Country	14%
Flood in Country with water coverage $> 5\%$, population density > 100	cholera in Country	16%

Table 7.6: Probability transitions for several examples.

7.2.5 Sample Likelihoods and Storylines

The learning and inference methodology we have described can be used to output the probabilities of key transitions of interest from sequences of observations. The system continues to refine its learning with updates of news and related data on the Web. As we mentioned, the system can provide real-time alerting from news stories on sets of specific outcomes that it is monitoring. Examples of statistics of representative learned transition probabilities are displayed in Figure 7.6. These transition probabilities and mean times to transition highlight the ability of the methods to provide inferences about a variety of levels of abstraction.

We now present details on several additional storylines, along with inferences and timing. Consider the example displayed graphically in Figure 7.1. On January 26th, 2007, the New York Times published an article about storms and

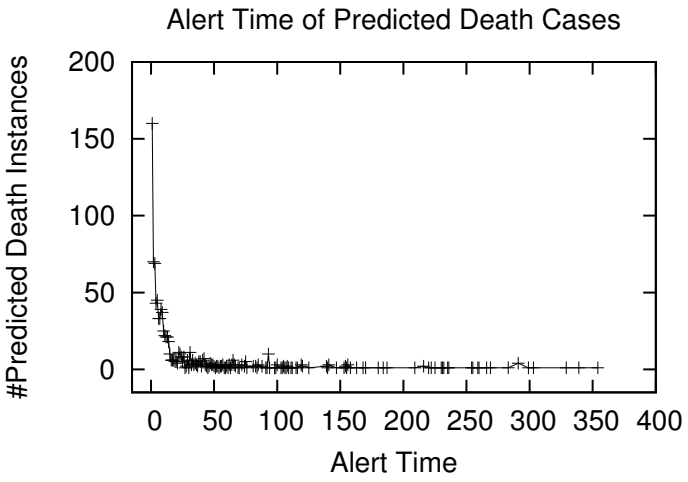

Figure 7.7: Number of times deaths of any number were predicted as a function of alert time (days before event).

floods in Africa. News of a cholera epidemic were reported four days later. In response to this stream of news, the methodology we describe yields two alerts, one when observing the drought reports in Angola at the beginning of 2006, and another one after news of the reported storms. The system learned from numerous similar incidents in its training set that the likelihood of a cholera outbreak is higher after droughts, specifically as reports on observations of drought are linked to increases in the probability of later reports of water-related disasters, which, in turn, are linked to increases in the likelihood of reports of waterborne diseases. Examples of such transitions and likelihoods include a set of Bangladesh droughts analyzed by the system. 19 significant cases of drought were reported in Bangladesh between 1960–1991 [Nagarajan, 2009]. We observed that in the story lines describing those droughts, a cholera outbreak was reported later in the storyline in 84% of cases. After the 1973 drought, which was responsible for the famine in 1974, the NYT reported on October 13, 1975: "cholera epidemic hits Bangladesh; may prove worse than one that set record in '74...". On March 13 1983, a year after the 1982 drought that "caused a loss of rice production of about 53000 tons while in the same year, flood damaged 36000 tons ...", the NYT published an article entitled, "Bangladesh cholera deaths." Several months later, an article appeared entitled "cholera reportedly kills 500 in 3 outbreaks in Bangladesh". Based on these past story lines the system infers the outbreak of cholera at the end of January in 2007.

The prediction method learns that not all droughts are associated with jumps

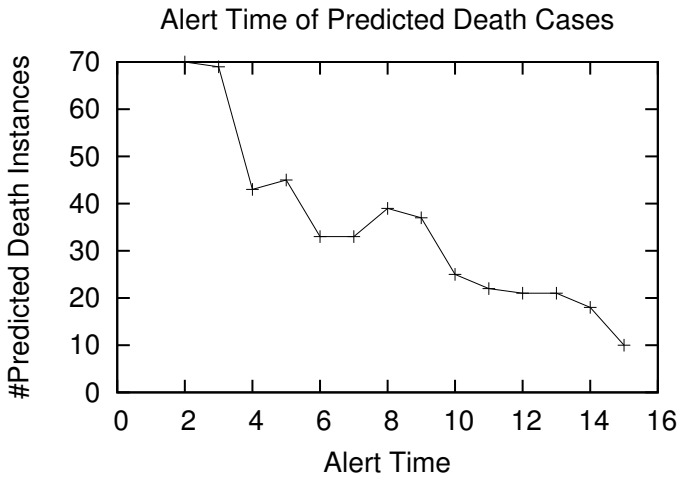

Figure 7.8: Number of times deaths of any number were predicted as a function of alert time (days before event).

in the likelihood of such outbreaks of disease. Specific sets of preconditions influence the likelihood of seeing a transition from a report of drought to a report of cholera outbreak. The method was able to recognize that the drought experienced in New York City on March 1989, published in the NYT under the title: "Emergency is declared over drought" would not be associated with a disease outbreak. The only consequence was that New York City declared water curbs, which ended on May 16th of that year. The system estimates that, for droughts to cause cholera with high probability, the drought needs to happen in dense populations (such as the refugee camps in Angola and Bangladesh) located in underdeveloped countries that are proximal to bodies of water.

As an additional example of predictions, we focus on the case of the 1991 cholera epidemic in Bangladesh. This cholera outbreak is estimated to have included 210,000 cases of cholera with more than 8,000 deaths [Michel et al., 1992]. In our experiments, we found that the running prediction system would have produced an alert four days before the beginning of the cholera outbreak, following observation of the major floods. In Figure 7.9, we display graphically the storyline detected. The system identifies that reports of major floods with high probability will be followed by reports of significant disease outbreak in Bangladesh. The inferences of the system are supported by a large study of cholera epidemics in Bangladesh [Michel et al., 1992], based on analyses of government figures, as well as data collected independently in 400 rural areas in Bangladesh between the years 1985–1991. The analysis shows that the number of cholera cases and

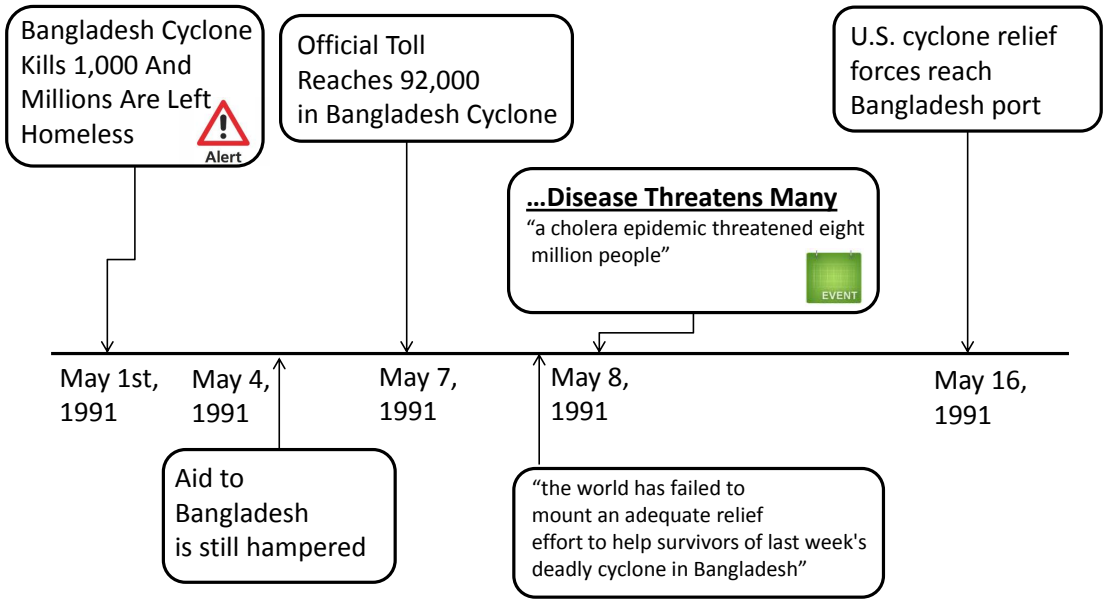

Figure 7.9: Example of cholera alert following storms in Bangladesh. Triangular alert icons represent inferences of significant upswings in likelihood of forthcoming cholera outbreak.

deaths in 1987 and 1988 is significantly higher than in other years (300,000-1,000,000 cases vs. 50,000 cases in other years). In 1987 and 1988, severe floods occurred in Bangladesh. The study concludes that access to medical care was one of the main reasons for high death rates in many non-rural areas. In areas where appropriate interventions were made at early stages, the death rates were significantly smaller.

We also study examples of prior deaths and riots. In Figure 7.10, we present a partial storyline and the alerts inferred for the Dialo case of 1999. Some of the other storylines are presented in detail in Tables 7.7, 7.8, 7.9. The system identified in an automated manner that for locations with large immigrant populations (e.g., Ohio and New York), the shooting of an unarmed person by the police can cause protests. Additional events in the news, such as reports of the funeral of the people who have been killed in similar way, of the beginning of the trial of the policemen who performed the shooting, of support for the policemen, and of the end of the trial are all associated with increases in the likelihood of later reports of protests. Sample storylines at the basis of the inferred probabilities are presented in Table 7.10.

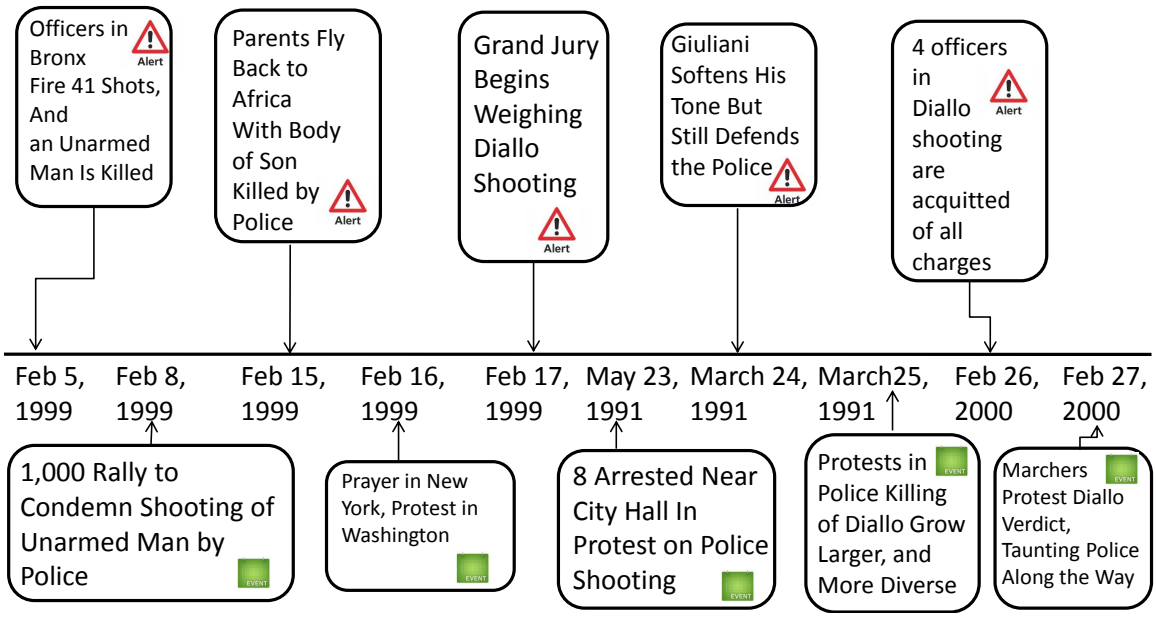

Figure 7.10: Example of alerts on the likelihood of forthcoming riots after shooting of unarmed minority. Triangular alert icons represent inferences of significant upswings in likelihood of a forthcoming riot.

Date	Title	
Feb 5, 1999	**Officers in Bronx Fire 41 Shots, And an Unarmed Man Is Killed**	**System Alert**
Feb 6, 1999	U.S. Examining Killing of Man In Police Volley	
Feb 6, 1999	In a Quest for Peace and Opportunity, West Africans Find Anger	
Feb 7, 1999	Four Officers Might Testify if Federal Interrogation Is Ruled Out	
Feb 8, 1999	1,000 Rally to Condemn Shooting of Unarmed Man by Police	Riot Event
Feb 8, 1999	Police Killing Draws National Notice	
Feb 9, 1999	Inquiry Into Police Shooting Draws the Gaze of Guinea	
Feb 13, 1999	Anger and Protest At Rite for African Killed by the Police	Riot Event
Feb 14, 1999	**Mayor Says Officers' New Ammunition Will Be Safer**	**System alert**
Feb 14, 1999	Killing Heightens the Unease Felt by Africans in New York	
Feb 15, 1999	**Parents Fly Back to Africa With Body of Son Killed by Police**	**System Alert**
Feb 16, 1999	Top Officials In Guinea Meet Plane Carrying Peddler's Body	
Feb 16, 1999	Prayer in New York, Protest in Washington	Riot Event

Table 7.7: Dialo Case

Date	Title	
Feb 17, 1999	**Grand Jury Begins Weighing Diallo Shooting**	**System Alert**
Feb 17, 1999	Slain Man's Mother Is Center of Attention in Guinea	
Feb 18, 1999	The Big City; Once, Racism Meant Leaving Harlem Alone	
Feb 18, 1999	F.B.I. Examines Site in Bronx Where Police Fatally Shot Unarmed Man	
Feb 19, 1999	After Shooting, an Eroding Trust in the Police	
Feb 19, 1999	Family of Man Slain by Officer in Diallo Case Sees Failure to Investigate	
Feb 23, 1999	8 Arrested Near City Hall In Protest on Police Shooting	Riot Event
Mar 8, 1999	Diallo lawyers level new charge	
Mar 9, 1999	Safir Denies Charge Police Tried to Taint Diallo's Reputation	
Mar 9, 1999	Lawsuit Seeks to Curb Street Crimes Unit, Alleging Racially Biased Searches	
Mar 10, 1999	12 Arrested During Sit-In To Protest Diallo Killing	Riot Event
Mar 15, 1999	Mayor Expresses Regret Over Police Shooting of Immigrant	
Mar 16, 1999	Poll in New York finds many think police are biased	
Mar 17, 1999	In America; Protesting Police Behavior	Riot Event
Mar 19, 1999	Daily Protesters in Handcuffs Keep Focus on Diallo Killing	Riot Event
Mar 20, 1999	Black Officer Gets No. 2 Post On Crime Unit	
Mar 23, 1999	Police Head Defends His Force As Protests in Diallo Case Go On	Riot Event
Mar 24, 1999	Officers in Diallo Shooting Won't Testify for Grand Jury	
Mar 24, 1999	Giuliani Softens His Tone But Still Defends the Police	Riot Event
Mar 25, 1999	Protests in Police Killing of Diallo Grow Larger, and More Diverse	Riot Event
Mar 26, 1999	Police Headquarters Rallies Won't End, Protesters Vow	Riot Event
Mar 27, 1999	Unlikely Protesters: Diallo Case Draws Diverse Group	Riot Event
Mar 30, 1999	Daily Protests End at Police Headquarters	Riot Event
Mar 31, 1999	Diallo Family to Get Preview of Charges From the District Attorney	
Mar 31, 1999	Officers Are Said to Tell of Suspicions About Diallo	
Apr 1, 1999	4 Officers Enter Not-Guilty Pleas To Murder Counts in Diallo Case	
Apr 3, 1999	Charges May Be Dropped in Diallo Protest	
Apr 8, 1999	Court Drops All Charges In Protests For Diallo	
Apr 9, 1999	Diallo's Tale Raises the Bar For Refugees	

Table 7.8: Dialo Case

Date	Title	
Feb 24, 2000	Diallo Jurors Begin Deliberating In Murder Trial of Four Officers	
Feb 26, 2000	**4 officers in Diallo shooting are acquitted of all charges**	**System Alert**
Feb 26, 2000	Rage Boils Over, and Some Shout 'Murderers' at Police	
Feb 26, 2000	Civil Rights Prosecution Is Considered	
Feb 27, 2000	Marchers Protest Diallo Verdict, Taunting Police Along the Way	Riot Event
Feb 27, 2000	2 jurors defend Diallo acquittal	
Mar 3, 2000	Diallo Family Meets With Justice Officials to Press for Federal Prosecution of Officers	

Table 7.9: Dialo Case

Date	Title
Jan 12, 1987	Trial of officer in Bumpurs case starting with request for no jury
Feb 12, 1987	Testimony Ends On Bumpurs's Death
Feb 18, 1987	Amid protest, Bumpurs case nears its end
Feb 27, 1987	Judge acquits Sullivan in shotgun slaying of Bumpurs
Feb 28, 1987	Officer Defends Bumpurs Shooting
Apr 11, 1990	Officer Kills Teen-Age Boy In Teaneck
Apr 12, 1990	Accounts Differ in Officer's Fatal Shooting of Youth
Apr 13, 1990	Police Officer Is Suspended; Tougher Response Is Vowed
Apr 17, 1990	Youth Slain by Teaneck Police Officer Is Buried
Apr 17, 1990	Anger Lingers at Slain Youth's Funeral
Apr 22, 1990	900 Protest Officer's Fatal Shooting of a 16-Year-Old in Teaneck
Apr 24, 1990	Town Reactions To Death Upset Teaneck Police
Aug 2, 1990	Evidence Shows Youth's Hands Up When Teaneck Officer Killed Him
Aug 23, 1990	Police Protest 2d Inquiry On Shooting in Teaneck
Nov 29, 1990	2d Jury Indicts Teaneck Officer In Youth's Death
Jan 29, 1990	Police Officer Fatally Shoots Youth in Back
Feb 2, 1990	Gritty Eulogies and Anger For Youth Killed by Police
Feb 4, 1990	Marchers Protest Police Killings of 2 Teen-Agers in Bushwick
Feb 7, 1990	Brown Lifts Officer's Suspension in Slaying
Feb 14, 1990	Officer Who Shot Unarmed Youth Claims Self-Defense and Is Cleared
Jul 2, 1991	Police Slaying of a Black Man Brings Protest
Jul 3, 1991	Melee Follows Protests Against Police Slaying
Jan 16, 1992	Jury in Shooting by Officer Hears Conflicting Accounts
Feb 11, 1992	Closing Arguments Conflict on Killing by Teaneck Officer
Feb 12, 1992	Officer Acquitted in Teaneck Killing
Feb 13, 1992	Acquitted Officer Expresses Only Relief, Not Joy
Feb 16, 1992	250 March in Rain to Protest Teaneck Verdict
May 24, 1992	Police and Mourners Clash at Funeral Procession
May 25, 1992	A Fatal Shooting by Police Ignites Protest in Bushwick
May 27, 1992	Brooklyn Officer Indicted In Unarmed Man's Death
Feb 23, 1997	Officer Kills Cook in a Raid On a Social Club in Queens
Feb 24, 1997	Police Actions In Club Killing Ignite Protests
Mar 1, 1997	Family's Anger Lingers In Police Killing of Cook
Apr 9, 1997	Actions by Police Under Scrutiny in Recent Fatal Shootings

Table 7.10: Some of the partial story lines from which probability was inferred for the Dialo case

Chapter 8

Related Work

In this chapter, we outline the related works. We the prior works into two classes – those studying Web dynamics and those studying Web knowledge for predictions on the Web.

8.1 Predicting using Web Dynamics

Several lines of research are related to modeling and predicting peoples' Web search behavior. We begin our discussion of related work with a review of studies that have characterized temporal search behavior dynamics on the Web. We next summarize previous research that has used temporal evidence for ranking, mostly focusing on content dynamics rather than behavioral dynamics. Finally, we describe related work in automatic query suggestion.

8.1.1 Web Search Behavioral Dynamics

The variations of query volume over time have been studied extensively in prior work. For example, some researchers have examined changes in query popularity over time [Wang et al., 2003] and the uniqueness of topics at different times of the day [Beitzel et al., 2004]. Some studies [Jones and Diaz, 2007] identified three general types of temporal query profiles: atemporal (no periodicities), temporally unambiguous (contain a single spike), and temporally ambiguous (contain more than one spike). They further showed that query profiles were related to search performance, with atemporal queries being associated with lower average precision. Kulkarni et al. [2011] explored how queries, their associated documents, and the intents corresponding to the queries change over time. The authors identify several features by which changes in query popularity can be classified, and show that presence of these features, when accompanied by changes in result content, can be a good indicator of change in the intent behind queries. Others [Chien and

Immorlica, 2005; Wang et al., 2007] used temporal patterns of queries to identify similar queries or words.

Vlachos et al. [2004] were among the first to examine and model periodicities and bursts in Web queries using methods from Fourier analysis. They also developed a method to discover important periods and to identify query bursts. Shokouhi [2011] identified seasonal queries using time-series analysis.

Shimshoni et al. [2009] studied the predictability of search trends using time-series analysis. Kleinberg et al. [Kleinberg, 2002, 2006] developed general techniques for summarizing the temporal dynamics of textual content and for identifying bursts of terms within content.

Researchers have also examined the relationship between query behavior and events. Ginsberg et al. [2009] used queries for predicting H1N1 influenza outbreaks. Similarly, Adar et al. [2007] identified when changes in query frequencies lead or lag behind mentions in both traditional media and blogs. From a Web search perspective, breaking news events are a particularly interesting type of evolving content. Diaz [2009] and Dong et al. [2010b] developed algorithms for identifying queries that are related to breaking news and for blending relevant news results into core search results. König et al. [2009] studied click prediction for news queries by analyzing the frequency and location of keywords in a corpus of news articles. Information about time-varying user behavior was also explored by Koren [2009], who used matrix factorization to model user biases, item biases, and user preferences over time.

Although much has been done to understand user Web search behavior over time, few efforts have sought to construct underlying models of this behavior and then used these models to predict future behavior. We present the construction of models for behaviors over time that can explain observed changes in the frequency of queries, clicked URLs, and clicked query-URL pairs.

8.1.2 Using Temporal Dynamics for Ranking

Agichtein et al. [2006] were the first to show that user behavior data could significantly improve ranking. In that work, the user behavior was represented as the simple average of behaviors over time, independent of query, URL clicks, or interactions between the two.

Researchers have examined how temporal attributes can be used to improve ranking using various kinds of content analysis. Dakka et al. [2008] defined a class of time-sensitive news queries, and suggested an approach that identifies important time intervals for those queries and augments the weight of those documents for ranking. Metzler et al. [2009] investigated a subset of temporal "year queries" – i.e., queries that often include the addition of terms representing a year such as *SIGIR 2012*, and modified the language model of the document so that that

years found in the document are weighted more heavily. Similarly, Efron and Golovchinksy [2011] and Li and Croft [2003] added temporal factors into models of language likelihood and relevance for re-ranking results based on the publication date of documents. Elsas and Dumais [2010] incorporated the dynamics of content changes into document language models to improve relevance ranking, showing that there is a strong relationship between the amount of change and term longevity and document relevance. Efron [2010] considered term popularity in a document collection over time, to adjust term weights for document ranking. In summary, the above works examine how general changes in content or specific content features (like dates) can be used to improve ranking.

Another line of research centers on methods for improving the ability of search engines to rank recent information effectively. Diaz [2009] studied how news can be integrated into search results by examining changes in query frequency, the popularity of query terms in the news collection, and query click feedback on presented news articles. Dong et al. [2010b] used Twitter data to detect and rank fresh documents. Similarly, Dong et al. [2010a] identified queries that are time-sensitive, developed features that represent the time of Web pages, and learned a recency-sensitive ranker. Dai et al. [2011] presented a ranking optimization with temporal features in documents, such as the trend and seasonality of the content changes in title, body, heading, anchor, and page or link activities. In summary, this line of research focuses on the desirability of providing fresh results for some kinds of queries.

To the best of our knowledge, temporal characteristics of queries and click behavior have not yet been used to improve general ranking of documents. In this work, we use time-series models to represent the dynamics of search behavior over time and show how this can be used to improve ranking and query suggestions.

8.1.3 Query Auto-Suggestion

In addition to using temporal models for ranking, we also use them to improve query auto-suggestion (QAS). Previous work on query auto-suggestion can be grouped into two main categories. The first group (also referred to as *predictive* auto-completion [Chaudhuri and Kaushik, 2009]) uses information retrieval and NLP techniques to generate and rank candidates on-the-fly as the user enters new words and characters [Darragh et al., 1990; Grabski and Scheffer, 2004; Nandi and Jagadish, 2007]. For instance, Grabski and Scheffer [2004], and Bickel et al. [2005] studied sentence completion based on lexicon statistics of text collections. Fan et al. [2010] ranked auto-suggestion candidates according to a generative model learned by Latent Dirichlet Allocation (LDA) [Blei et al., 2003]. White and Marchionini [2007] developed a real-time query expansion system that produces an updated list of candidates based on the top-ranked documents as the user

types new words in the search box.

In the second group of QAS techniques — including the one presented here — candidates are pre-generated and stored in tries and hash tables for efficient *lookup*. The list of candidate suggestions is updated by new lookups with each new input from the user. The filtering of candidates is typically based on exact prefix matching. Recently, Chaudhuri and Kaushik [2009] and Ji et al. [2009] proposed flexible fuzzy matching models that are tolerant to small edit-distance differences between the query (prefix) and candidates.

In the context of Web search, the most conventional approach is to rank candidate query suggestions according to their past popularity. Bar-Yossef and Kraus [2011] referred to this approach as *MostPopularCompletion* (MPC):

$$MPC(\mathcal{P}) = \arg\max_{q \in \mathcal{C}(\mathcal{P})} w(q), \quad w(q) = \frac{f(q)}{\sum_{i \in \mathcal{Q}} f(i)}. \tag{8.1}$$

where, $f(q)$ denotes the number of times the query q occurs in a previous search log $\mathcal{Q}$. They propose a context-aware technique in which the default static scores for the candidates are combined with contextual scores based on recent session history to compute the final ranking. Under the MPC model, the candidate scores do not change as long as the same query log $\mathcal{Q}$ is used. We also take MPC [Bar-Yossef and Kraus, 2011] as our QAS ranking baseline and show that it can be improved significantly by considering the temporal characteristics of queries.

Our approach is distinct from prior work in several important ways. We use time-series analysis to learn how to differentially weight historical click data for queries, URLs, and query-URL pairs. This enables recent behavior to be weighted more highly for some queries (or URLs or query-URL pairs) but not others; and similarly for periodic behaviors to be important for some queries but not others; etc. Second, we extend previous research on ranking fresh results by addressing the more general challenge of finding the most appropriate results, even if they are not recent. We extend earlier work by Agichtein et al. [2006] on search ranking by harnessing time-series analyses to learn the most appropriate weighting of previous behaviors rather than simply averaging previous behavior. Finally, we show that the temporal profiles of queries created by our time-series models can be used to improve the ranking of query auto-suggestion candidates that have historically been based on static measures of popularity.

8.1.4 Web Content Dynamics

Attempting to predict the probability of a web page changing is not a new problem and has been studied by a variety of researchers [Barbosa et al., 2005; Cho and García-Molina, 2003b; Cho and Ntoulas, 2002; Tan and Mitra, 2010; Tan

et al., 2007]. This has primarily been motivated by the incremental web crawling setting [Olston and Najork, 2010]. In this setting, recrawling a web page is linked to the probability of its change, and the goal is to maximize some tradeoff between freshness and coverage of a web index. A crawl policy that optimizes this tradeoff usually focuses only on the probability of change for pages that do not change too often or very rarely. The reason for this is that, under certain assumptions, recrawling pages that change very frequently dominates crawl resources and actually hurts the overall utility of a crawl by ensuring only these very frequently changing pages are kept fresh [Cho and García-Molina, 2003a; Coffman et al., 1998; Shahaf et al., 2011]. In line with these studies, we restrict our study to pages above a minimum threshold of change and below a maximum threshold.

However, while freshness may be one metric for measuring the utility of recrawling a web page, it may not reflect impact on the user of a search engine. As a result, Wolf *et al.* [Wolf et al., 2002] look at minimizing embarrassment – the probability a search engine user clicks on a result URL whose live page doesn't match the query. This introduces a notion of the negative utility a user may experience but not what gains may be experienced from crawling. Therefore, Pandey & Olston [Pandey and Olston, 2005] consider a user-centric utility. This utility weights each page by: (1) an observed query load on a search engine, and (2) the impact that downloading a new copy of the page has on the search ranking (as estimated via previous change). Similarly, Ali & Williams [Ali and Williams, 2003] measure the impact under a specific query load of several simple prediction schemes. They found that simply recrawling pages whose previous two observed snapshots changed by more than (approximately) 20 words significantly reduced crawling compared to recrawling all recently changed pages. This policy also maintained effective ranking as measured under the query load. In line with these, we restrict ourselves to predicting significant changes to a search page. In our case, we define this as cases where the page's change relative to the last observed snapshot is above a significance threshold (as defined in Equation 1).

In terms of methods of predicting change, several works [Cho and García-Molina, 2000, 2003a; Edwards et al., 2001] use past change-frequency and change-recency of a page, typically together with an assumption of a Poisson process, in order to predict whether the page will change. This is essentially equivalent to our baseline model described in Section 3.3.1. Both Barbosa *et al.* [Barbosa et al., 2005] and Tan & Mitra [Tan and Mitra, 2010] demonstrate that using content-based features can improve prediction over just change-frequency alone [Cho and García-Molina, 2000]. The first of these works primarily uses static features (content features, such as file size available from a single snapshot) in addition to change history, while the second also considers dynamic features (features that change between two snapshots, such as the cosine similarity between the

page contents). In our work, we focus on dynamic content features since we are concerned with predicting *significant* content change, and as argued by Ali & Williams [Ali and Williams, 2003], many of those static features are usually predictive of any type of change (e.g., new advertisements, images, etc.), that are usually uninteresting or do not impact indexing. Nevertheless, our methods are general and could easily incorporate such additional features.

Fetterly *et al.* [Fetterly et al., 2003a] note that not only is the previous degree of content change a good predictor of future change, but that change is also correlated at the top-level domains of web pages. Cho & Ntoulas [Cho and Ntoulas, 2002] exploited this type of correlation structure at the website level by using a sample of webpages from a website to estimate the change probability of any page from the website. Tan *et al.* [Tan and Mitra, 2010; Tan et al., 2007] extend this idea to the more general case by clustering pages based on static and dynamic content features into clusters that are more homogeneous in terms of change patterns before conducting the sampling based approach. Both of these can be viewed as methods of identifying related pages. We also identify related pages, but use them to directly predict the probability of change without having sampled from the current time slice. Additionally, we investigate directly estimating relatedness via temporal content similarity – which could easily be used to cluster in a sampling based approach.

In contrast to previous work: (1) we focus on the task of predicting *significant* changes rather than any change to a web page; (2) we develop a wide array of dynamic content-based features that may be useful for the more general temporal mining case beyond crawling; (3) we explore a wide variety of methods to identify related pages including content, web graph distance, and temporal content similarity – where the last is both especially novel and effective; (4) we derive a novel expert prediction framework that effectively leverages information from related pages without the need for sampling from the current time slice.

8.2 Predicting using Web Knowledge

We are not aware of any work that attempts to perform the task we face: receive arbitrary news events in natural language representation and predict events they can cause. Several works, however, deal with related tasks. In general, our work does not focus on better information extraction or causality extraction techniques, but rather on how this information can be leveraged for prediction. We present novel methods of combining world knowledge with event extraction methods to represent coherent events, and present novel methods for rule extraction and generalization using this knowledge.

8.2.1 Prediction from Web Behavior, Books and Social Media

Several works have focused on using search-engine queries for prediction in both traditional media and blogs [Adar et al., 2007]. Ginsberg et al. [2009] used queries for predicting H1N1 influenza outbreaks. In the context of causality recognition, Gordon et al. [2011] present a methodology for mining blogs to extract common-sense causality. The evaluation is done on a human dataset where each test consists of a fact and two possible effects. Applying point-mutual information to personal blog stories, the authors select the best prediction candidate. The work differs from ours in that the authors focus on personal common-sense mining and do not consider whether their predictions actually occurred. Other works focused on predicting Web content change itself. For example, Kleinberg [2002, 2006] developed general techniques for summarizing the temporal dynamics of textual content and for identifying bursts of terms within content. Similarly, Amodeo et al. [2011] built a time series model over publication dates of documents relevant to a query in order to predict future bursts. Social media were used to predict riots [Kalev, 2011] and movie box office sales [Asur and Huberman, 2010; Joshi et al., 2010; Mishne, 2006]. Other works [Jatowt and Yeung, 2011; Michel et al., 2011; Yeung and Jatowt, 2011] have explored the use of text mining techniques over news and books to explain how culture develops, and what people's expectations and memories are.

Our work differs from the above in several ways: First, we present a *general-purpose* prediction algorithm rather than a domain-specific one. Second, unlike the above works, ours combines a variety of *heterogenous online sources*, including world knowledge mined from the Web. Finally, we focus on generation of future event predictions represented entirely in natural language, and provide techniques to enrich and generalize historical events for the purpose of future event prediction.

8.2.2 Textual Entailment

A related topic to our work is that of textual entailment (TE) [Glickman et al., 2005]. A text t is said to entail a textual hypothesis h if people reading it agree that the meaning of t implies the truth of h. TE can be divided into three main categories: recognition, generation, and extraction. In this section, we provide a short summary of the first two categories (for a more detailed overview we refer the reader to Androutsopoulos and Malakasiotis [2010]), and discuss the specific task of causality extraction from text in Section 8.2.3.

Textual Entailment recognition

In this task, pairs of texts are given as input, and the output is whether TE relations hold for the pair. Some approaches map the text to logical expressions (with some semantic enrichment, using WordNet, for example) and perform logical entailment checks, usually using theorem provers [Bos and Markert, 2005; Raina et al., 2005; Tatu and Moldovan, 2005]. Other approaches map the two texts to a vector space model, where each word is mapped to strongly co-occurring words in the corpus [Mitchell and Lapata, 2008], and then similarity measurements over those vectors are applied. Some measure syntactic similarity by applying graph similarity measure on the syntactic dependency graphs of the two texts [Zanzotto et al., 2009]. Similarly, other methods measure the semantic distance similarity between the words in text [Haghighi, 2005], usually exploiting other resources such as WordNet as well. The last set of approaches represents the two texts in a single feature vector and trains a machine learning algorithm, which later, given two new texts represented via a vector, can determine whether they entail each other [Bos and Markert, 2005; Burchardt et al., 2009; Hickl, 2008]. For example, Glickman et al. [2005] show a naive Bayes classifier trained on lexical features, i.e., the number of times that words of t appeared with words of h. Other features usually include polarity [Haghighi, 2005], whether the theorem prover managed to prove entailment [Bos and Markert, 2005], or tagging of the named entities to the categories people, organizations, or locations.

Textual Entailment Generation

Here we discuss TE generation, where, given an expression, the system should output a set of expressions that are entailed by the input. This task is most closely related to the one presented in this work: in TE generation, a text is received and an entailed text is generated as output. Androutsopoulos and Malakasiotis [2010] mention that no benchmarks exist to evaluate this task, and the most common and costly approach is to evaluate using human judges. We also encountered this difficulty in our own task, and performed human evaluation.

TE generation methods can be divided into two types: those that use machine translation techniques and those that use template-based techniques. Those that use machine translation techniques try to calculate the set of transformations with the highest probability, using a training corpus. Quirk et al. [2004] cluster news articles referring to the same event, select pairs of similar sentences, and apply the aforementioned techniques. Other methods use template-based approaches on large corpora, such as the Web. Some methods [Idan et al., 2004] start with an initial seed of sentences (composed of entities), and use a search engine to find other entities for which these entailment relations hold. Those relations are used as templates. To find additional entities for which these relations hold,

the relations themselves are then searched again. The TE generation system, given a text, matches it to a template and outputs all the texts that matched this template. Others [Ravichandran et al., 2003] also add additional filtering techniques on those templates.

Our work is most closely related to the template-based approach. We have crafted a new set of templates to extract causality pairs from the news.

8.2.3 Information Extraction

Information Extraction is the study of automatic extraction of information from unstructured sources. We categorizes the types of information extracted into three types: entities, relationships between entities, and higher-order structures such as tables and lists. The most closely related tasks to ours are those of entity extraction and relation extraction; for the rest we refer the reader to [Sarawagi, 2008]. The former task, similar to our process of extracting concepts, deals with extracting noun phrases from text. In the latter task, given a document and a relation as input, the problem is to extract all entity pairs in the document for which this relation holds. Whereas the above works deal only with one element of our problem – extraction of information needed to understand a given causality, we deal with the actual causality prediction. We do not claim to create more precise information extraction methods, but rather try to leverage all this knowledge to perform an important AI task – future event prediction.

Entity Extraction

For entity extraction, two categories of methods exist – rule-based and statistical methods. Rule-based methods [Ciravegna, 2001; Hobbs et al., 1993; Jayram et al., 2006; Maynard et al., 2001; Riloff, 1993; Riloff and Jones, 1999; Shen et al., 2007] define contextual patterns consisting a regular expression over features of the entities in the text (e.g., the entity word, part-of-speech tagging). Those rules are either manually coded by a domain expert or learned using bottom-up [Califf and Mooney, 1999; Ciravegna, 2001] or top-down learners [Soderland, 1999]. Others follow statistical methods that define numerous features over the sentence and then treat the problem as a classification problem, applying well-known machine learning algorithms (e.g., Hidden Markov Models [Agichtein and Ganti, 2004; Borkar et al., 2001]). Our system does not deal with the many challenges in this field, as we propose a large scale domain-specific approach driven by specific extraction templates.

Relation Extraction

Relation extraction has been developed widely in the last years from large text corpora [Schubert, 2002] and, in particular, from different Web resources, such as general Web content [Banko et al., 2007; Carlson et al., 2010; Hoffmann et al., 2010], blogs [Jayram et al., 2006], Wikipedia [Suchanek et al., 2007], and news articles (e.g., the topic detection and tracking task (Section 8.2.3)). Given two entities, the first task in this domain is to classify their relationship. Many feature-based methods [Jiang and Zhai, 2007; Kambhatla, 2004; Suchanek, 2006] and rule-based methods [Aitken, 2002; Jayram et al., 2006; Mcdonald et al., 2004; Shen et al., 2007] have been developed for this task. Most methods use different features extracted from the text, such as the words, the grammar features, such as parse tree and dependency graphs, and features extra ion from external relation repositories (e.g., Wikipedia Infobox) to add additional features Hoffmann et al. [2011]; Nguyen and Moschitti [2011]. Labeled training examples, from which those feature are extracted, are then fed into a machine learning classifier, sometimes using transformations such as kernels [Bunescu and Mooney, 2005; Culotta and Sorensen, 2004; Nguyen et al., 2009; Wang, 2008; Zelenko et al., 2003; Zhang et al., 2006; Zhao and Grishman, 2005], which, given new unseen entities, will be able to classify them into those categories.

Given a relation, the second common task in this domain is to find entities that satisfy this relation. Out of all information extraction tasks, this task is most relevant to ours, as we try to find structured events for which the causality relation holds. Most works in this domain focus on large collections, such as the Web, where labeling all entities and relations is infeasible [Agichtein and Gravano, 2000; Banko et al., 2007; Bunescu and Mooney, 2007; Rosenfeld and Feldman, 2007; Shinyama and Sekine, 2006; Turney, 2006]. Usually seed entity databases are used, along with some manual extraction templates, and then expanded and filtered iteratively. Sarawagi [2008] states that "in spite of the extensive research on the topic, relationship extraction is by no means a solved problem. The accuracy values still range in the neighborhood of 50%–70% even in closed benchmark datasets ... In open domains like the Web, the state-of-the-art systems still involve a lot of special case handling that cannot easily be described as principled, portable approaches." Similarly, in our task the size of our corpus does not allow us to assume any labeled sets. Instead, like the common approaches presented here, we also start with a predefined set of patterns.

Temporal Information Extraction

The temporal information extraction task deals with extraction and ordering of events from many events over time. Temporal information extraction can be categorized into three main subtasks – predicting the temporal order of events or

time expressions described in text, predicting the relation between those events, and identifying when the document was written. This task has been found to be important in many natural language processing applications, such as question answering, information extraction, machine translation and text summarization, all of which require more than mere surface understanding. Most of these approaches [Chambers et al., 2007; Lapata and Lascarides, 2006; Ling and Weld, 2010; Mani et al., 2003; Tatu and Srikanth, 2008; Yoshikawa et al., 2009] learn classifiers that predict a temporal order of a pair of events from predefined features of the pair.

Other related works deal with topic detection and tracking [Cieri et al., 2000]. This area includes several tasks [Allan, 2002]. In all of them, multiple, heterogenous new sources are used, including audio. The story segmentation task aims to segment data into its constituent stories. The topic tracking task –e.g., the work by [Shahaf and Guestrin, 2010] – aims to find all stories discussing a certain topic. A subtask of this is the link detection task which, given a pair of stories, aims to classify whether they are on the same topic. The topic detection task –e.g. the work by [Ahmed et al., 2011b; Yang et al., 1998] – aims to detect clusters of topic-cohesive stories in a stream of topics. The first-story detection task aims to identify the first story on a topic [Jian Zhang and Yang, 2004]. In this chapter, we focused on short text headlines and the extraction of events from them. Our work differs from that of temporal information extraction, in that we generate predictions of future events, whereas temporal information extraction tasks focus on identifying and clustering the text corpus into topics.

Causality Pattern Extraction and Recognition

In the first stage of our learning process we extract causality pairs from text. Causality extraction has been discussed in the literature in the past, and can be divided into the following subgroups:

1. Use of handcrafted domain-specific patterns. Some studies deal with causality extraction using specific domain knowledge. Kaplan and Berry-Rogghe [1991] used scientific texts to create a manually designed set of propositions which were later applied on new texts to extract causality. These methods require handcrafted domain knowledge, which is problematic to obtain for real-world tasks, especially in large amounts.

2. Use of handcrafted linguistic patterns. These works use a more general approach by applying linguistic patterns. For example, Garcia [1997] manually identified 23 causative verb groups (e.g., to result in, to lead to, etc.). If a sentence contained one of those verbs, it was classified as containing a causation relation. A precision of 85% was reported. Khoo et al. [2000] used manually extracted graphical patterns based on syntactic parse trees,

reporting accuracy of about 68% on an English medical database. Similarly, Girju and Moldovan [2002] defined lexicon-syntactic patterns (pairs of noun phrases with a causative verb in between) with additional semantic constraints.

3. Semi-supervised pattern learning approaches. This set of approaches uses supervised machine learning techniques to identify causality in text. For example, Blanco et al. [2008] and Sil et al. [2010] use syntactic patterns as features that are later fed into classifiers, whose output is whether the text implies causality or the cause and effect themselves.

4. Supervised pattern learning approaches. There have been many works on design inference rules to discover extraction patterns for a given relation using training examples [Agichtein and Gravano, 2000; Lin and Pantel, 2001; Riloff, 1996; Riloff and Jones, 1999]. Specifically, [Chan and Lam, 2005] dealt with the problem of creating syntactic patterns for cause-effect extraction.

In the domain of causality pattern extraction, our work most resembles the handcrafted linguistic patterns pattern approaches. We evaluated their performance on our specific domain. Since our goal was to obtain a very precise set of examples to feed into our learning, we chose to follow such an approach as well.

8.2.4 Learning Causality

We have drawn some of our algorithmic motivation from work in the machine learning community. In this section, we give a partial review of the main areas of machine learning that are relevant to our work.

Bayesian Causal Inference

The functional causal model [Pearl, 2000] assumes a set of observables $X_1 \dots X_n$, which are the vertices of a directed acyclic graph G. The semantics of the graph is that parents of a node are its directed causes. It was shown to satisfy Reichenbach's common cause principle, which states that for a node Z with children X, Y, if X and Y are statistically dependent, then there is a Z causally influencing both. This model, similar to a Bayesian network, satisfies several conditions: (1) Local Causal Markov condition: a node is statistically independent of non-descendants, given its parents; (2) Global Causal Markov condition: d-separation criterion; (3) Factorization criterion: $P(X_1, \dots, X_n) = \prod_i P(X_i | Parents(X_i))$. The theoretical literature on the inference and learning of causality models is extensive. Those models resemble our work in the use of structural models. The literature

on inference and learning of causality models is extensive, but to our knowledge there are no solutions that scale to the scope of tasks discussed in this work. In contrast with Bayesian approach, the causality graph in our work contains less detailed information. Our work combines several linguistic resources that were learned from data with several heuristics to build the causality graph.

Structured Learning

An important problem in the machine learning field is structured learning, where the input or the output of the classifier is a complex structure, such as relational domain, where each object is related to another, either in time or in its features. Our task resembles structured learning in that we also use structured input (structured events given as input) and produce a structured event as output.

Many generative models have been developed, including hidden Markov models, Markov logic networks, and conditional random fields, among others. Other approaches use transformations, or kernels, that unite all the objects, ignoring the structure, and then feed it into a standard structured classifier, e.g., kernelized conditional random fields [Lafferty et al., 2004], maximum margin Markov networks [Taskar et al., 2003], and others [Bakir et al., 2007]. When dealing with complex output, such as annotated parse trees for natural language problems, most approaches define a distance metric in the label space between the objects, and they again apply standard machine learning algorithms (e.g., structured support vector machines [Joachims, 2006]).

Learning from Positive Examples (One Class Classification)

As our system is only fed examples of the sort "a causes b," and no examples of the sort "a does not cause b," we must deal with the problem of learning from positive examples only. This is a challenge for most multi-class learning mechanisms, which require both negative and positive examples. Some theoretical studies of the possibility of learning from only positive unlabeled data are provided in [Denis, 1998] (probably approximately correct (PAC) learning) and [Muggleton, 1996] (Bayesian learning).

Most work Manevitz and Yousef [2000]; Manevitz et al. [2001]; Tax [2001] in this domain develops algorithms that use one-class SVM Vapnik [1995] and learn the support using only positive distribution. They construct decision boundaries around the positive examples to differentiate them from all possible negative data. Tax and Duin [1991] use a hyper-sphere with some defined radius around some of the positive class points (support vector data description method). Some also use kernel tricks before finding this sphere [Tax, 2001]. Schölkopf et al. [1999, 2000] develop methods that try to separate the surface region of the positive labeled data from the region of the unlabeled data.

Chapter 9

Conclusions

The World Wide Web increasingly reflects the current state of the physical and social world, manifested both in traditional news media sources along with user-generated publishing sites such as Wikipedia and Twitter. At the same time, Web interactions, such as information seeking and content change increasingly reflect problems grounded in the real world. As a result of this blending of the Web with the real world, we observe that the Web, both in its composition and use, has incorporated many of the dynamics of the real world. Many of the problems associated with searching dynamic collections are not well understood, such as defining time-sensitive relevance, understanding user query behavior over time and understanding why certain Web content changes.

To address these temporal information needs effectively, it is essential for the Web agent, such as a search engine, to model and predict Web dynamics as well as to understand the changes in user behavior over time that are caused by them. In this thesis, we presented methods for modeling time-sensitive content on the Web, and developed state-of-the-art approaches for integrating temporal signals into Web search.

We first developed methods for modeling the dynamics of the query and click behaviors seen in a large population of Web searchers. We modeled temporal characteristics that are often observed in query and URL click behavior, including trend, periodicity, and surprise disruptions. We presented several different temporal representations and learning procedures and showed how we can construct models that predict future behaviors from historical data.

We then tackled the problem of predicting significant content change on the Web. Our results shed light on how and why content changes on the Web, and how it can be predicted. We explored several information sources that can be used to predict such change – the commonly used frequency of change, the page content and related pages content.

To fully predict human agents actions, one must endow the machine the ability

to understand them and the world they live in. We present methods of leveraging this Web dynamics for a NLP task, providing evidence that such dynamics treasures information about our perception of the world. We proposed a novel approach to computing semantic relatedness with the aid of a large scale temporal corpus. We used the New York Times archive that spans over a large period of time, and which, to the best of our knowledge, have not been used before in such tasks. Specifically, we introduced two innovations over the previous words' semantic relatedness methods: first, a new method, Temporal Semantic Analysis, for representing the semantics of natural language terms, and a new method for measuring semantic relatedness of terms, using this representation. The algorithm is robust in that it can be naturally tuned to assign different weights to time periods, and can be used for studying language evolution over time. Our empirical evaluation confirms that using our algorithm leads to significant improvements in computing words relatedness.

We established the grounds for claiming that Web dynamics indeed exhibits indicators of our perception of the real world, and we proceeded to utilizing this dynamics to predict what will appear in the news. We presented a novel methodology that uses query popularity today, and historical patterns of what queries tended to follow other queries to predict terms in future news.

However, Web dynamics is not the only source of information that can be used to predict future events. We discussed how to leverage the knowledge accumulated on the Web into a large-scale AI problem of event prediction. We present a system that is trained to predict future events, using a cause event as input. Each event is represented as a tuple of one predicate and 4 general semantic roles. The event pairs used for training are extracted automatically from news headlines using simple syntactic patterns. Generalization to unseen events is achieved by:

1. Creating an abstraction tree (AT) that contains entities from observed events together with their subsuming categories extracted from available online ontologies.

2. Finding predicate paths connecting entities from cause events to entities in the effect events, where the paths are again extracted from available ontologies.

We discuss the challenges of building such a system: obtaining a large enough dataset, representing the knowledge, and developing the inference algorithms required for such a task. We perform *large-scale mining* and apply *natural language techniques* to transform the raw data of over 150 years of history archives into a structured representation of events, using a mined Web-based object hierarchy and action classes. This shows the scalability of the proposed method, scalability which is crucial to any method that requires large amounts of data to work well.

We also show that the numerous resources built by different people for different purposes (e.g., the different ontologies) can in fact be merged via a concept-graph to build a system that can work well in practice.

We perform *large-scale learning* over the large data corpus and present *novel inference* techniques. We consider both rule extraction and generalization. We propose novel methods for rule generalization using existing ontologies, which we believe can be useful for many other related tasks. Tasks such as entailment and topic tracking can benefit from the concepts of understanding sequences and their generalizations.

Our experimental evaluation showed that the predictions of the Pundit algorithm are at least as good as those of humans. We believe that our work is one of the first to harness the vast amount of information available on the Web to perform event prediction that is general purpose, knowledge based, and human-like.

We finish this thesis with presenting our holy grail, that combines both Web dynamics and knowledge, and present methods for mining chains of events from 22 years of news archives to create a system with the ability to make real-time predictions about the likelihoods of future world events. The system harnesses multiple Web resources to enrich data about sequences of reports with world knowledge to generalize the events that it learns about and predicts. The heterogeneous sources are combined into a probabilistic system that provides alerts about future event, such as disease outbreaks, riots and deaths. To demonstrate the behavior of the system, we presented the results of several evaluations and representative examples of sequences of events and proactive alerts. We believe that our system can be used as a real time system for global alerts to asses experts. The system has the ability to learn patterns from large amounts of data, it monitors large numbers of information sources, constantly learns new probabilistic associations, and can continue to do real-time monitoring, prediction, and alerting on rises in likelihoods of concerning events. Beyond knowledge that is easily discovered in studies or available from experts, new relationships and context-sensitive probabilities of outcome can be discovered by a computational system. Additionally, the system has faster and more comprehensive access to news stories that might seem less important (e.g., a story about a funeral published in a local newspaper that does not reach the main headlines), but that might provide valuable evidence in the evolution of larger, more important stories (e.g., massive riots).

We believe that we only scratched the surface of the potential that lies in Web-based predictions. As the social-media takes a prominent spot on the Web, more and more of the real-time events are reflected through it. Leveraging this constantly evolving information on Foursquare, Facebook, Twitter etc. can be a fertile ground for better events predictions. Furthermore, the ever-evolving improvements in natural language processing will enable us to perform deep analysis

of the stream of information and provide even more valuable predictions.

In this thesis, we demonstrate general, unrestricted prediction capacity and present methods based on heterogeneous Web sources to make knowledge-intensive reasoning about causality and future events, using both automatic feature extraction and novel algorithms for generalizing over historical examples. Our work is one of the first to harvest this type of knowledge to perform general predictions from the Web to the real world.

References

B. S. A. Smola, P. Bartlett and D. Schuurmans. *Advances in Large Margin Classifiers*. MIT Press, 1999.

E. Adar, D. S. Weld, B. N. Bershad, and S. D. Gribble. Why we search: visualizing and predicting user behavior. In *Proceedings of the International Conference on the World Wide Web (WWW)*, 2007.

E. Adar, J. Teevan, S. Dumais, and J. Elsas. The web changes everything: Understanding the dynamics of web content. In *Proc. of WSDM*, 2009.

E. Agichtein and V. Ganti. Mining reference tables for automatic text segmentation. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2004.

E. Agichtein and L. Gravano. Snowball: extracting relations from large plain-text collections. In *Proceedings of Joint Conference on Digital Libraries (JCDL)*, pages 85–94, 2000.

E. Agichtein, E. Brill, and S. T. Dumais. Improving web search ranking by incorporating user behavior information. In *Proceedings of the Special Interest Group on Information Retrieval (SIGIR)*, 2006.

A. Ahmed, Q. Ho, J. Eisenstein, E. P. Xing, A. J. Smola, and C. H. Teo. Unified analysis of streaming news. In *Proceedings of the International Conference on World Wide Web (WWW)*, 2011a.

A. Ahmed, Q. Ho, J. Eisenstein, E. P. Xing, A. J. Smola, and C. H. Teo. Unified analysis of streaming news. In *Proceedings of the International Conference on the World Wide Web (WWW)*, 2011b.

J. Aitken. Learning information extraction rules: An inductive logic programming approach. In *Proceedings of the 15th European Conference on Artificial Intelligence (ECAI)*, pages 355–359, 2002.

H. Ali and H. E. Williams. What's changed? measuring document change in web crawling for search engines. In *SPIRE*, pages 28–42, 2003.

J. Allan, editor. *Topic Detection and Tracking: Event-based Information Organization*, volume 12. Kluwer Academic Publishers, Norwell, MA, USA, 2002.

G. Amodeo, R. Blanco, and U. Brefeld. Hybrid models for future event prediction. In *Proceedings of the ACM Conference on Information and Knowledge Management (CIKM)*, 2011.

I. Androutsopoulos and P. Malakasiotis. A survey of paraphrasing and textual entailment methods. *Journal of Artificial Intelligence Research (JAIR)*, 38: 135–187, May 2010.

S. Asur and B. A. Huberman. Predicting the future with social media. In *ArxiV*, 2010.

G. H. Bakir, T. Hofmann, B. Schölkopf, A. J. Smola, B. Taskar, and S. V. N. Vishwanathan. *Predicting Structured Data*. MIT Press, 2007.

M. Banko, M. J. Cafarella, S. Soderl, M. Broadhead, and O. Etzioni. Open information extraction from the web. In *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI)*, 2007.

Z. Bar-Yossef and N. Kraus. Context-sensitive query auto-completion. In *Proceedings of International World Wide Web Conference (WWW)*, pages 107–116, Hyderabad, India, 2011.

L. Barbosa, A. Salgado, F. de Carvalho, J. Robin, and J. Freire. Looking at both the present and the past to efficiently update replicas of web content. In *Proc. of WIDM Workshop*, 2005.

S. M. Beitzel, E. C. Jensen, A. Chowdhury, D. Grossman, and O. Frieder. Hourly analysis of a very large topically categorized web query log. In *Proceedings of the Special Interest Group on Information Retrieval (SIGIR)*, 2004.

R. Bellman and R. Kalaba. On adaptive control processes. *IRE Transactions on Automatic Control*, 4:1–9, 1959.

P. N. Bennett, K. Svore, and S. T. Dumais. Classification-enhanced ranking. In *Proceedings of International World Wide Web Conference (WWW)*, 2010.

S. Bickel, P. Haider, and T. Scheffer. Learning to complete sentences. In *Proceedings of the European Conference on Machine Learning (ECML)*, pages 497–504. 2005.

C. Bizer and A. Schultz. The berlin sparql benchmark. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 2009.

C. Bizer, T. Heath, and T. Berners-Lee. Linked data – the story so far. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 2009.

E. Blanco, N. Castell, and D. Moldovan. Causal Relation Extraction. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, 2008.

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research (JMLR)*, 3:993–1022, Mar. 2003.

B. Bogert, M. Healy, and J. Tukey. Cepstrum pitch determination. *Journal of the Acoustical Society of America*, 41:293–309, 1967.

V. Borkar, K. Deshmukh, and S. Sarawagi. Automatic text segmentation for extracting structured records. In *Proceedings of ACM SIGMOD International Conference on Management of Data (KDD)*, 2001.

J. Bos and K. Markert. Recognising textual entailment with logical inference. In *Proceedings of the Human Language Technology Conference Conference on Empirical Methods in Natural Language Processing (HLT EMNLP)*, 2005.

L. Breiman. Bagging predictors. *ML*, 24(2):123–140, 1996.

R. Bunescu and R. Mooney. Learning to extract relations from the web using minimal supervision. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 576–583, 2007.

R. C. Bunescu and R. J. Mooney. A shortest path dependency kernel for relation extraction. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT EMNLP)*, pages 724–731, 2005.

A. Burchardt, M. Pennacchiotti, S. Thater, and M. Pinkal. Assessing the impact of frame semantics on textual entailment. *Natural Language Engineering*, 15: 527–550, 2009.

C. J. C. Burges. From RankNet to LambdaRank to LambdaMART: An Overview. Technical report, 2010. URL `http://research.microsoft.com/en-us/um/people/cburges/tech\_reports/MSR-TR-2010-82.pdf`.

M. E. Califf and R. J. Mooney. Relational learning of pattern-match rules for information extraction. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI)*, pages 328–334, 1999.

J. Carbonell, Y. Yang, J. Lafferty, R. D. Brown, T. Pierce, and X. Liu. Cmu report on tdt-2: segmentation, detection and tracking. In *Technical Report*, 2000.

A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. Hruschka, and T. Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI)*, 2010.

N. Chambers and D. Jurafsky. Template-Based Information Extraction without the Templates. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2011.

N. Chambers, S. Wang, and D. Jurafsky. Classifying temporal relations between events. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL) (Poster)*, 2007.

K. Chan and W. Lam. Extracting causation knowledge from natural language texts. *International Journal of Information Security (IJIS)*, 20:327–358, 2005.

S. Chaudhuri and R. Kaushik. Extending autocompletion to tolerate errors. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 707–718, 2009.

S. Chien and N. Immorlica. Semantic similarity between search engine queries using temporal correlation. In *Proceedings of International World Wide Web Conference (WWW)*, 2005.

D. Childers, D. Skinner, and R. Kemerait. The cepstrum: A guide to processing. *IEEE*, 65:1428–1443, 1977.

J. Cho and H. García-Molina. The evolution of the web and implications for an incremental crawler. In *Proc. of VLDB*, 2000.

J. Cho and H. García-Molina. Effective page refresh policies for web crawlers. *TODS*, 28(4):390–426, 2003a.

J. Cho and H. García-Molina. Estimating frequency of change. *TOIT*, 3(3): 256–290, 2003b.

J. Cho and A. Ntoulas. Effective change detection using sampling. In *Proc. of VLDB*, 2002.

C. Cieri, D. Graff, M. Liberman, N. Martey, and S. Strassel. Large, multilingual, broadcast news corpora for cooperative research in topic detection and tracking: The tdt-2 and tdt-3 corpus efforts. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, 2000.

F. Ciravegna. Adaptive information extraction from text by rule induction and generalisation. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI)*, 2001.

E. Coffman, Z. Liu, and R. Weber. Optimal robot scheduling for web search engines. *Journal of Scheduling*, 1(1), 1998.

A. Culotta and J. Sorensen. Dependency tree kernels for relation extraction. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL)*, pages 423–429, 2004.

N. Dai, M. Shokouhi, and B. D. Davison. Learning to rank for freshness and relevance. In *Proceedings of the Special Interest Group on Information Retrieval (SIGIR)*, 2011.

W. Dakka, L. Gravano, and P. G. Ipeirotis. Answering general time sensitive queries. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, 2008.

H. T. Dang, M. Palmer, and J. Rosenzweig. Investigating regular sense extensions based on intersective levin classes. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, 1998.

J. J. Darragh, I. H. Witten, and M. L. James. The reactive keyboard: A predictive typing aid. *Computer*, 23:41–49, November 1990.

F. Denis. PAC learning from positive statistical queries. In *Proceedings of the International Conference on Algorithmic Learning Theory (ALT)*, pages 112–126, 1998.

F. Diaz. Integration of news content into web results. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*, 2009.

Q. Do, Y. Chan, and D. Roth. Minimally supervised event causality identification. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing (EMNLP)*, 2011.

A. Dong, Y. Chang, Z. Zheng, G. Mishne, J. Bai, R. Zhang, K. Buchner, C. Liao, and F. Diaz. Towards recency ranking in web search. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*, 2010a.

A. Dong, R. Zhang, P. Kolari, J. Bai, F. Diaz, Y. Chang, Z. Zheng, and H. Zha. Time is of the essence: improving recency ranking using twitter data. In *Proceedings of International World Wide Web Conference (WWW)*, 2010b.

P. F. Dunn. *Measurement and Data Analysis for Engineering and Science*. McGraw-Hill, 2005.

J. Durbin and S. Koopman. *Time Series Analysis by State Space Methods*. Oxford University Press, 2008.

J. Edwards, K. McCurley, and J. Tomlin. An adaptive model for optimizing performance of an incremental web crawler. In *Proc. of WWW*, 2001.

M. Efron. Linear time series models for term weighting in information retrieval. *Journal of the American Society for Information Science and Technology (JASIST)*, 6(7), 2010.

M. Efron and G. Golovchinksy. Estimation methods for ranking recent information. In *Proceedings of the Special Interest Group on Information Retrieval (SIGIR)*, 2011.

M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *PNAS*, 95:14863–14868, 1998.

J. L. Elsas and S. T. Dumais. Leveraging temporal dynamics of document content in relevance ranking. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*, 2010.

J. Fan, H. Wu, G. Li, and L. Zhou. Suggesting topic-based query terms as you type. In *Proceedings of the International Asia-Pacific Web Conference (APWeb*, pages 61–67, Washington, DC, 2010.

R. Feldman, Y. Regev, M. Finkelstein-Landau, E. Hurvitz, and B. Kogan. Mining biomedical literature using information extraction. *Current Drug Discovery*, 2002.

D. Fetterly, M. Manasse, M. Najork, and J. L. Wiener. A large-scale study of the evolution of web pages. *Software Practice and Experience*, 1(1), 2003a.

D. Fetterly, M. Manasse, M. Najork, and J. L. Wiener. A large-scale study of the evolution of web pages. In *Proc. of WWW*, 2003b.

J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2005.

L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin. Placing search in context: The concept revisited. *ACM TOIS*, 20: 116–131, 2002.

Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *JMLR*, 4:933–969, 2003.

E. Gabrilovich and S. Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJCAI*, pages 1606–1611, 2007.

D. Garcia. Coatis, an NLP system to locate expressions of actions connected by causality links. In *Proceedings of Knowledge Engineering and Knowledge Management by the Masses (EKAW)*, 1997.

M. Gerber, A. S. Gordon, and K. Sagae. Open-domain commonsense reasoning using discourse relations from a corpus of weblog stories. In *Proceedings of Formalisms and Methodology for Learning by Reading, NAACL-2010 Workshop*, 2010.

L. Getoor and L. Mihalkova. Exploiting statistical and relational information on the web and in social media. In *Proc. of WSDM*, 2011.

J. Ginsberg, M. H. Mohebbi, R. S. Patel, L. Brammer, M. S. Smolinski, and L. Brilliant. Detecting influenza epidemics using search engine query data. *Nature*, 457:1012–1014, 2009.

R. Girju and D. Moldovan. Text mining for causal relations. In *Proceedings of the Annual International Conference of the Florida Artificial Intelligence Research Society (FLAIRS)*, pages 360–364, 2002.

A.-M. Giuglea and A. Moschitti. Shallow semantic parsing based on framenet, verbnet and propbank. In *Proceedings of the the 17th European Conference on Artificial Intelligence (ECAI 2006)*, 2006.

O. Glickman, I. Dagan, and M. Koppel. A probabilistic classification approach for lexical textual entailment. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI)*, 2005.

A. S. Gordon, C. A. Bejan, and K. Sagae. Commonsense causal reasoning using millions of personal stories. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI)*, 2011.

K. Grabski and T. Scheffer. Sentence completion. In *Proceedings of the Special Interest Group on Information Retrieval (SIGIR)*, pages 433–439, Sheffield, United Kingdom, 2004.

A. D. Haghighi. Robust textual inference via graph matching. In *Proceedings of the Human Language Technology Conference Conference on Empirical Methods in Natural Language Processing (HLT EMNLP)*, 2005.

A. Hickl. Using discourse commitments to recognize textual entailment. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, 2008.

J. R. Hobbs, J. Bear, D. Israel, and M. Tyson. Fastus: A finite-state processor for information extraction from real-world text. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1172–1178, 1993.

R. Hoffmann, C. Zhang, and D. S. Weld. Learning 5000 relational extractors. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2010.

R. Hoffmann, C. Zhang, X. Ling, L. Zettlemoyer, and D. S. Weld. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (HLT)*, 2011.

C. C. Holt. Forecasting seasonals and trends by exponentially weighted moving averages. *International Journal of Forecasting*, 20(1):5–10, 2004.

R. Hyndman, A.Koehler, J.Ord, and R.Snyder. *Forecasting with Exponential Smoothing (The State Space Approach)*. Springer, 2008.

I. S. Idan, H. Tanev, and I. Dagan. Scaling web-based acquisition of entailment relations. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing (EMNLP)*, pages 41–48, 2004.

A. Jatowt and C. Yeung. Extracting collective expectations about the future from large text collections. In *Proceedings of the ACM Conference on Information and Knowledge Management (CIKM)*, 2011.

T. S. Jayram, R. Krishnamurthy, S. Raghavan, S. Vaithyanathan, and H. Zhu. Avatar information extraction system. *IEEE Data Engineering Bulletin*, 29: 40–48, 2006.

S. Ji, G. Li, C. Li, and J. Feng. Efficient interactive fuzzy keyword search. In *Proceedings of International World Wide Web Conference (WWW)*, pages 371–380, Madrid, Spain, 2009.

Z. G. Jian Zhang and Y. Yang. A probabilistic model for online document clustering with application to novelty detection. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, 2004.

J. Jiang and C. Zhai. A systematic exploration of the feature space for relation extraction. In *Proceedings of the Human Language Technologies and the Conference of the North American Chapter of the Association for Computational Linguistics (HLT NAACL)*, pages 113–120, 2007.

T. Joachims. Structured output prediction with support vector machines. In D.-Y. Yeung, J. Kwok, A. Fred, F. Roli, and D. de Ridder, editors, *Structural, Syntactic, and Statistical Pattern Recognition*, volume 4109 of *Lecture Notes in Computer Science*, pages 1–7. Springer Berlin / Heidelberg, 2006.

R. Jones and F. Diaz. Temporal profiles of queries. *ACM Transactions on Information Systems*, 25, July 2007.

M. Joshi, D. Das, K. Gimpel, and N. A. Smith. Movie reviews and revenues: An experiment in text regression. In *Proceedings of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL HLT)*, 2010.

Kalev. Culturomics 2.0: Forecasting large-scale human behavior using global news media tone in time and space. *First Monday*, 15(9), 2011.

N. Kambhatla. Combining lexical, syntactic and semantic features with maximum entropy models for information extraction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 178–181, 2004.

R. Kaplan and G. Berry-Rogghe. Knowledge-based acquisition of causal relationships in text. *Knowledge Acquisition*, 3:317–337, 1991.

E. Keogh and M. Pazzani. Scaling up dynamic time warping for datamining applications. In *Proc. of KDD*, 2000.

C. Khoo, S. Chan, and Y. Niu. Extracting causal knowledge from a medical database using graphical patterns. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 336–343, 2000.

J. Kim. Supervenience and mind. *Selected Philosophical Essays*, 1993.

K. Kipper. Extending verbnet with novel verb classes. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, 2006.

A. Kittur, H. Chi, and B. Suh. Crowdsourcing user studies with mechanical turk. In *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems is the premier International Conference of human-computer interaction (CHI)*, 2008.

J. Kleinberg. Bursty and hierarchical structure in streams. In *Proceedings of the Annual ACM SIGKDD Conference (KDD)*, 2002.

J. Kleinberg. Temporal dynamics of on-line information systems. *Data Stream Management: Processing High-Speed Data Streams. Springer*, 2006.

A. C. König, M. Gamon, and Q. Wu. Click-through prediction for news queries. In *Proceedings of the Special Interest Group on Information Retrieval (SIGIR)*, 2009.

Y. Koren. Collaborative filtering with temporal dynamics. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, 2009.

A. Kulkarni, J. Teevan, K. M. Svore, and S. T. Dumais. Understanding temporal query dynamics, 2011.

J. Lafferty, X. Zhu, and Y. Liu. Kernel conditional random fields: Representation and clique selection. In *The 21st International Conference on Machine Learning (ICML)*, 2004.

S. Lahiri, P. Mitra, and X. Lu. Informality judgment at sentence level and experiments with formality score. In *Proceedings of the 12th International Conference on Computational Linguistics and Intelligent Text Processing (CICLing)*, 2011.

Landis and Koch. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):74–159, 1977.

M. Lapata and A. Lascarides. Learning sentence-internal temporal relations. *Journal of Artificial Intelligence Research (JAIR)*, 27:85–117, 2006.

S. Lappin and H. Leass. An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20:535–561, 1994.

O. Lassila, R. R. Swick, W. Wide, and W. Consortium. Resource description framework (rdf) model and syntax specification, 1998.

D. Lenat and E. Feigenbaum. On the thresholds of knowledge. *Artificial Intelligence*, 47(a):185–250, 1991.

D. B. Lenat and R. V. Guha. *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project*. Addison-Wesley, 1990.

J. Leskovec, S. Dumais, and E. Horvitz. Web projections: learning from contextual subgraphs of the web. In *Proc. WWW*, 2007.

B. Levin and M. R. Hovav. A preliminary analysis of causative verbs in english. *Lingua*, 92:35–77, 1994.

X. Li and W. B. Croft. Time-based language models. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, 2003.

D. Lin and P. Pantel. Dirt-discovery of inference rules from text. In *Proceedings of the Annual ACM SIGKDD Conference (KDD)*, 2001.

X. Ling and D. Weld. Temporal information extraction. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI)*, 2010.

H. Liu and P. Singh. Conceptnet: A practical commonsense reasoning toolkit. *BT Technology Journal*, 22:211–226, 2004.

L. M. Manevitz and M. Yousef. Document classification on neural networks using only positive examples. In *Proceedings of 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 304–306, 2000.

L. M. Manevitz, M. Yousef, N. Cristianini, J. Shawe-Taylor, and B. Williamson. One-class svms for document classification. *Journal of Machine Learning Research*, 2:139–154, 2001.

I. Mani, B. Schiffman, and J. Zhang. Inferring temporal ordering of events in news. In *Proceedings of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL HLT)*, 2003.

M. Marneffe, B. MacCartney, and C. Manning. Generating typed dependency parses from phrase structure parses. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, 2006.

D. Maynard, V. Tablan, C. Ursu, H. Cunningham, and Y. Wilks. Named entity recognition from diverse text types. In *Recent Advances in Natural Language Processing Conference (RANLP)*, pages 1172–1178, 2001.

D. M. Mcdonald, H. Chen, H. Su, and B. B. Marshall. Extracting gene pathway relations using a hybrid grammar: The arizona relation parser. *Bioinformatics*, 20:3370–3378, 2004.

D. Metzler, R. Jones, F. Peng, and R. Zhang. Improving search relevance for implicitly temporal queries. In *Proceedings of the Special Interest Group on Information Retrieval (SIGIR)*, 2009.

J. Michel, Y. Shen, A. Aiden, A. Veres, M. Gray, Google Books Team, J. Pickett, D. Hoiberg, D. Clancy, P. Norvig, J. Orwant, S. Pinker, M. Nowak, and E. Aiden. Cholera epidemics in bangladesh: 1985-1991. *Journal of Diarrhoeal Diseases Research (JDDR)*, 10(2):79–86, 1992.

J. Michel, Y. Shen, A. Aiden, A. Veres, M. Gray, Google Books Team, J. Pickett, D. Hoiberg, D. Clancy, P. Norvig, J. Orwant, S. Pinker, M. Nowak, and E. Aiden. Quantitative analysis of culture using millions of digitized books. *Science*, 331:176–182, 2011.

G. Miller. Wordnet: A lexical database for english. *Journal of Communications of the ACM (CACM)*, 38:39–41, 1995.

G. Mishne. Predicting movie sales from blogger sentiment. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI) Spring Symposium*, 2006.

J. Mitchell and M. Lapata. Vector-based models of semantic composition. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2008.

S. Muggleton. Learning from positive data. In *Proceedings of the Inductive Logic Programming Workshop*, pages 358–376, 1996.

R. Nagarajan. *Drought Assessment*. Springer, 2009.

A. Nandi and H. V. Jagadish. Effective phrase prediction. In *Proceedings of the Conference on Very Large Databases (VLDB)*, pages 219–230, Vienna, Austria, 2007.

T.-V. T. Nguyen and A. Moschitti. Joint distant and direct supervision for relation extraction. In *Proceedings of the The 5th International Joint Conference on Natural Language Processing (IJCNLP)*, 2011.

T.-V. T. Nguyen, A. Moschitti, and G. Riccardi. Convolution kernels on constituent, dependency and sequential structures for relation extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2009.

C. Olston and M. Najork. Web crawling. *Foundations and Trends in Information Retrieval*, 4(3):175–246, 2010.

S. Pandey and C. Olston. User-centric web crawling. In *Proc. of WWW*, pages 401–411, 2005.

J. Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2000.

M. Priestley. *Spectral Analysis and Time Series*. Academic Press, 1981.

C. Quirk, C. Brockett, and W. Dolan. Monolingual machine translation for paraphrase generation. In *Proceedings the Conference on Empirical Methods on Natural Language Processing (EMNLP)*, pages 142–149, 2004.

R. Rada, H. Mili, E. Bicknell, and M. Blettner. Development and application of a metric to semantic nets. *IEEE Transactions on Systems, Man and Cybernetics*, 19(1):17–30, 1989.

K. Radinsky, S. Davidovich, and S. Markovitch. Predicting the news of tomorrow using patterns in web search queries. In *Proceedings of the IEEE/WIC International Conference on Web Intelligence (WI)*, 2008.

K. Radinsky, E. Agichtein, E. Gabrilovich, and S. Markovitch. A word at a time: Computing word relatedness using temporal semantic analysis. In *Proceedings of International World Wide Web Conference (WWW)*, 2011.

K. Radinsky, K. Svore, S. Dumais, J. Teevan, A. Bocharov, and E. Horvitz. Modeling and predicting behavioral dynamics on the web. In *Proceedings of International World Wide Web Conference (WWW)*, 2012a.

K. Radinsky, K. Svore, S. Dumais, J. Teevan, and E. Horvitz. Time-aware ranking: Improving ranking by learning to weight user behavior. In *Under Submission to the ACM International Conference on Information and Knowledge Management (CIKM)*, 2012b.

R. Raina, A. Y. Ng, and C. D. Manning. Robust textual inference via learning and abductive reasoning. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI)*, 2005.

D. Ravichandran, A. Ittycheriah, and S. Roukos. Automatic derivation of surface text patterns for a maximum entropy based question answering system. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Short Papers (NAACL Short)*, pages 85–87, 2003.

M. Richardson, E. Dominowska, and R. Ragno. Predicting clicks: estimating the click-through rate for new ads. In *Proc. of WWW*, 2007.

E. Riloff. Automatically constructing a dictionary for information extraction tasks. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI)*, pages 811–816, 1993.

E. Riloff. Automatically Generating Extraction Patterns from Untagged Text. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI)*, 1996.

E. Riloff and R. Jones. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI)*, 1999.

S. Robertson, H. Zaragoza, and M. Taylor. Simple bm25 extension to multiple weighted fields. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, 2004.

B. Rosenfeld and R. Feldman. Using corpus statistics on entities to improve semi-supervised relation extraction from the web. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 600–607, 2007.

D. Sack, R. Sack, G. Nair, and A. Siddique. Cholera. *Lancet*, 363(9404):223–233, 2004.

S. Sarawagi. Information extraction. *Foundations and Trends in Databases*, 1(3): 261–377, 2008.

B. Schölkopf, R. C. Williamson, A. Smola, and J. Shawe-Taylor. Sv estimation of a distribution's support. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, 1999.

B. Schölkopf, R. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt. Support vector method for novelty detection. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, pages 582–588, 2000.

L. Schubert. Can we derive general world knowledge from texts? In *Proceedings of the Second Conference on Human Language Technology (HLT)*, 2002.

G. E. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 2(6): 461–464, 1978.

D. Shahaf and C. Guestrin. Connecting the dots between news articles. In *Proceedings of the Annual ACM SIGKDD Conference (KDD)*, 2010.

D. Shahaf, Y. Azar, E. Lubetzky, and E. Horvitz. Optimal web crawling. Technical Report, April 2011.

W. Shen, A. Doan, J. F. Naughton, and R. Ramakrishnan. Declarative information extraction using datalog with embedded extraction predicates. In *Proceedings of the Conference on Very Large Data Bases (VLDB)*, pages 1033–1044, 2007.

L. Shi and R. Mihalcea. Putting pieces together: Combining framenet, verbnet and wordnet for robust semantic parsing. In *Proceedings of the Sixth International Conference on Intelligent Text Processing and Computational Linguistics (CICLing)*, pages 100–111, 2005.

Y. Shimshoni, N. Efron, and Y. Matias. On the predictability of search trends. In *Technical Report*, 2009.

Y. Shinyama and S. Sekine. Preemptive information extraction using unrestricted relation discovery. In *Proceedings of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL HLT)*, 2006.

M. Shokouhi. Detecting seasonal queries by time-series analysis. In *Proceedings of the Special Interest Group on Information Retrieval (SIGIR)*, 2011.

M. Shokouhi and K. Radinsky. Time-sensitive query auto-completion. In *Proceedings of the Special Interest Group on Information Retrieval (SIGIR)*, 2012.

A. Sil, F. Huang, and A. Yates. Extracting action and event semantics from web text. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI) Fall Symposium on Commonsense Knowledge*, 2010.

R. Snow, B. O'Connor, D. Jurafsky, and A. Ng. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2008.

J. A. Snyman. *Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms*. Springer, 2005.

S. Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34, 1999.

M. Strube and S. P. Ponzetto. Wikirelate! computing semantic relatedness using wikipedia. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI)*, 2006.

F. M. Suchanek. Combining linguistic and statistical analysis to extract relations from web documents. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 712–717, 2006.

F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *Proceedings of the International Conference on the World Wide Web (WWW)*, 2007.

Q. Tan and P. Mitra. Clustering-based incremental web crawling. *TOIS*, 28(4), 2010. Article 17.

Q. Tan, Z. Zhuang, P. Mitra, and C. Giles. A clustering-based sampling approach for refreshing search engine's database. In *Proc. of WebDB Workshop*, 2007.

B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, 2003.

M. Tatu and D. Moldovan. A semantic approach to recognizing textual entailment. In *Proceedings of the Human Language Technology Conference Conference on Empirical Methods in Natural Language Processing (HLT EMNLP)*, 2005.

M. Tatu and M. Srikanth. Experiments with reasoning for temporal relations between events. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, 2008.

D. Tax. One class classification. In *PhD thesis*, Delft University of Technology, 2001.

D. M. J. Tax and R. P. W. Duin. Support vector domain description. *Pattern Recognition Letters*, 20:1191–1199, 1991.

J. Teevan, S. T. Dumais, and E. Horvitz. Personalizing search via automated analysis of interests and activities. In *Proceedings of the Special Interest Group on Information Retrieval (SIGIR)*, 2005.

K. Todar. Todar's online textbook of bacteriology: Vibrio cholerae and asiatic cholera. `http://www.textbookofbacteriology.net/cholera.html`.

P. D. Turney. Expressing implicit semantic relations without supervision. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2006.

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, NY, USA, 1995.

H. Varian and H. Choi. Predicting the present with google trends. *Technical Report*, 2009.

A. J. Viera and J. M. Garrett. Understanding interobserver agreement: The kappa statistic. *Family Medicine*, 37(5):360–363, 2005.

M. Vlachos, C. Meek, Z. Vagena, and D. Gunopulos. Identifying similarities, periodicities and bursts for online search queries. 2004.

M. Wang. A re-examination of dependency path kernels for relation extraction. In *Proceedings of the Third International Joint Conference on Natural Language Processing (ACL IJCNLP)*, 2008.

P. Wang, M. W. Berry, and Y. Yang. Mining longitudinal web queries: trends and patterns. *Journal of the American Society for Information Science and Technology (JASIST)*, 54:743–758, 2003.

X. Wang, C. Zhai, X. Hu, and R. Sproat. Mining correlated bursty topic patterns from coordinated text streams. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, 2007.

R. W. White and G. Marchionini. Examining the effectiveness of real-time query expansion. *Information Processing and Management*, 43:685–704, May 2007.

R. W. White, P. N. Bennett, and S. T. Dumais. Predicting short-term interests using activity-based search context. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, 2010.

J. Wolf, M. Squillante, P. Yu, J.Sethuraman, and L.Ozsen. Optimal crawling strategies for web search engines. In *Proc. of WWW*, 2002.

P. Wolff. Representing causation. *Journal of Experimental Psychology General*, 137:82–111, 2007.

P. Wolff, G. Song, and D. Driscoll. Models of causation and causal verbs. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2002.

Y. Yang, T. Pierce, and J. Carbonell. A study on retrospective and online event detection. In *Proceedings of ACM SIGIR Special Interest Group on Information Retrieval (SIGIR)*, 1998.

E. Yeh, D. Ramage, C. D. Manning, E. Agirre, and A. Soroa. Wikiwalk: Random walks on wikipedia for semantic relatedness. In *TextGraphs Workshop*, pages 41–49, 2009.

C. Yeung and A. Jatowt. Studying how the past is remembered: Towards computational history through large scale text mining. In *Proceedings of the ACM Conference on Information and Knowledge Management (CIKM)*, 2011.

K. Yoshikawa, S. Riedel, M. Asahara, and Y. Matsumoto. Jointly identifying temporal relations with markov logic. In *Proceedings of the Third International Joint Conference on Natural Language Processing (ACL IJCNLP)*, 2009.

Y. Yue, R. Patel, and H. Roehrig. Beyond position bias: Examining result attractiveness as a source of presentation bias in clickthrough data. In *Proceedings of International World Wide Web Conference (Proceedings of International World Wide Web Conference (WWW)*, 2010.

F. M. Zanzotto, M. Pennacchiotti, and A. Moschitti. A machine learning approach to textual entailment recognition. *Natural Language Engineering*, 15:551–582, 2009.

D. Zelenko, C. Aone, and A. Richardella. Kernel methods for relation extraction. *Journal of Machine Learning Research*, 3:1083–1106, 2003.

T. Zesch and I. Gurevych. Wisdom of crowds versus wisdom of linguists - measuring the semantic relatedness of words. *Journal of Natural Language Engineering.*, 16(01):25–59, 2010.

T. Zesch, C.Müller, and I. Gurevych. Using wiktionary for computing semantic relatedness. In *AAAI*, 2008.

M. Zhang, J. Zhang, J. Su, and G. Zhou. A composite kernel to extract relations between entities with both flat and structured features. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 825–832, 2006.

S. Zhao and R. Grishman. Extracting relations with integrated information using kernel methods. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 419–426, 2005.

ללמוד לחזות את העתיד בעזרת הדינמיקה והידע ברשת האינטרנט

קירה רדינסקי

ללמוד לחזות את העתיד בעזרת הדינמיקה והידע ברשת האינטרנט

חיבור על מחקר

לשם מילוי חלקי של הדרישות לקבלת התואר

דוקטור לפילוסופיה

קירה רדינסקי

הוגש לסנט הטכניון — מכון טכנולוגי לישראל

כסלו תשע"ג חיפה דצמבר 2012

המחקר נעשה בהנחיית פרופ'/ח' שאול מרקוביץ וד"ר ניר אילון
בפקולטה למדעי המחשב

אני מודה לטכניון על התמיכה הכספית הנדיבה בהשתלמותי

לבעלי, שגיא דוידוביץ,
ולאמי ולדודתי, נטליה ומילנה רדינסקי

אתה רואה דברים, ושואל "למה?", ואילו אני חולם על דברים שמעולם לא היו ושואל "למה לא?"

-- ג'ורג' ברנרד שו

תקציר

כאשר סוכן הניצב בסביבה מורכבת מתכנן את פעולותיו, הוא בעצם מסיק לגבי שינוי-ים עתידיים בסביבה זו. חלק מהשינויים הללו הם תוצאה של פעולותיו, וחלק הם תוצאה של שרשראות אירועים, שחלקן נובעות מפעולות של סוכנים אחרים באותה סביבה. בעבר, סוכנים ממוחשבים לא יכלו לתפקד בסביבות מורכבות כאלו בשל יכולות תפיסה מוגבלות. ההתפשטות המהירה של רשת האינטרנט שינתה כל זאת. סוכן מושכל יכול עתה לפעול בעולם הווירטואלי על ידי צריכת מידע על המצב הנוכחי של העולם על ידי מקורות רבים של מידע כתוב, הכולל: עמודי אינטרנט, טוויטים, דיווחי חדשות ואינציקלופדיות ממוחשבות. הסוכן עתה גם יכול לבצע משימות רבות בעזרת מידע זה, כגון: חיפוש, סידור וייצור מידע.

רשת האינטרנט יכולה להיתפס כסביבה ענקית, בה מיליארדי סוכנים פועלים על מנת להשיג את מטרותיהם. חלק מהסוכנים הללו הם אנושיים, אשר צורכים ומייצר-ים מידע, וחלק מהסוכנים הם ממוחשבים, כמו מנועי חיפוש, זחלני אינטרנט ושרתי אינטרנט. על מנת לפעול בצורה מחושבת בסביבה הוירטואלית של רשת האינטרנט, הסוכנים צריכים להיות מסוגלים לקבל מידע על המצב הנוכחי ולהסיק לגבי מצבים עתידיים על ידי חיזוי הפעולות של הסוכנים האחרים. סוכנים ממוחשבים יכולים להפיק תועלת מחיזוי על פעולות הסוכנים האנושיים. לדוגמא: ומצסועבפו

לצפות את התדירות של שאילתא ביום מסוים. היכולת לצפות מאורע מסוג זה יכולה לשפר את הביצועים של מנגנוני השלמת השאילתות של מנועי החיפוש.

בהינתן שאילתא, היכולת לצפות את ההסתברות של לחיצה על תוצאת חיפוש יכולה לשפר את מנגנוני הדירוג של מנועי החיפוש.

היכולת לצפות את הביקוש לעמוד אינטרנט ביום מסוים יכולה לעזור לזחלן אינטרנט לקבוע את סדר הזחילה שלו.

היכולת לצפות את ההסתברות שעמוד אינטרנט ישתנה ביום מסוים יכולה, כמו קודם, לשפר את תהליך הזחילה באינטרנט.

רשת האינטרנט היא דינמית ומשתנה בהתאם למצב הפיזי והחברתי של העולם האמיתי. השינויים בעולם הוירטואלי באים לידי ביטוי בצריכת מידע ואינטראקציות תלויות זמן הנובעות מהסוכנים האנושיים. לכן, על מנת להבין לעומק את הסוכנים האנושיים, הסוכן הממוחשב חייב להיות מסוגל לצרוך מידע לגבי המצב הנוכחי בעולם האמיתי, וכן להיות מסוגל להסיק מסקנות לגבי מצבו העתידי. חיזוי מצבים אלו משליך ישירות לגבי היכולת לצפות מצבים עתידיים בעולם הווירטואלי. לדוגמא, היכולת לצפות הסתברות של כותרת בחדשות יכולה להיות שימושית לזיהוי אי-יציבות פוליטית, זיהוי ומעקב אחר תופעות חברתיות ותמיכה בקבלת החלטות. יכולת חיזוי זו יכולה לעזור בתורה גם לסוכנים וירטואליים לצפות פעולות עתידיות של הסוכנים האנושיים עם העולם הווירטואלי, כגון חיפוש מידע רלוונטי.

בעבודה זו אנחנו מציגים גישה חדשה לביצוע משימות חיזוי אלו. גישתנו ייחודית בכך שהיא משתמשת בשני מקורות מידע ייחודיים:

1. דינמיקת הרשת: רשת האינטרנט היא יצור דינמי, המתאר את המידע המשתנה של עולמנו. הדינמיקה של מיליוני הסוכנים האנושיים ברשת הוא מקור מידע חיוני לחיזוי מצבים עתידיים של הרשת, אולם מעטים המחקרים אשר חקרו דינמיקה זו. בעבודה זו אנו מציגים אלגוריתמי חיזוי, שמשתמשים בדינמיקת הזו. לדוגמא, אנו משתמשים בהיסטוריית השינויים של עמוד אינטרנט על מנת לצפות מתי הוא ישתנה ומשתמשים במידע זה על מנת לשפר אלגוריתמי זחילה

באינטרנט.

2. הידע ברשת: פיתרון בעיות מורכבות דורש שימוש נרחב בידע המכיל היגיון אנושי. היגיון זה דרוש בבעיות חיזוי הדורשות שימוש במידע כתוב, כמו זה הנ-מצא ברשת. למזלנו, הרשת היא מקור מהימן גם למידע זה. הרשת היא אוצר מידע, המתגלם באינציקלופדיות וארכיוני חדשות. עדכונים דינמיים לגבי מצב העולם מתקבלים דרך אתרי החדשות כל דקה. הרשת מכילה גם אנתולוגיות המשקפות את העולם שלנו, כגון ויקיפידיה. כל המידע הזה יכול לשמש בסיס חזק לביצוע חיזוי בצורה הדומה לזו המבוצעת על ידי אנשים. לדוגמא, האלגור-יתמים, שנציג בעבודה זו, משלבים מידע מארכיוני חדשות והאנתולוגיות על מנת לבצע חיזוי של אירועים חדשותיים.

בתיזה זו אנו מציגים מגוון אלגוריתמים לביצוע משימות חיזוי מורכבות המתבססות על שני המקורות שתיארנו.

בחלק הראשון של התיזה, אנו דנים במשימות חיזוי, המשויכות להתנהגות הסוכנים האנושיים בעולם ווירטואלי. אנו מציגים אלגוריתמים לחיזוי התנהגות המשתמשים בהתבסס על דינמיקות ואניטרקציות היסטוריות, כגון: חיזוי פופולריות של שאילתות, חיזוי לחיצות על תוצאות חיפוש וחיזוי מתי סוכן אנושי ישנה עמוד באינטרנט. אנו מייצגים התנהגויות אלו בעזרת מודלים של ניתוח סדרות עיתיות, ומראים שיפור-ים ממשיים לעומת שיטות קיימות. אנו מדגימים שימוש בשיטות אלו עבור יישומי חיפוש שונים, כגון: השלמת שאילתות תלוית-זמן, דירוג תוצאות חיפוש תלוי-זמן ושי-פור במנגנוני זחלנים. בכל יישום אני מדגימים שיפור על גבי השיטות הקיימות כיום.

בחלק השני של התיזה, אנו מציגים אלגוריתמים לחיזוי אירועים עתידיים בעזרת הדינמיקה של הרשת הווירטואלית. אנו תחילה מציגים אלגוריתם המשתמש בדינמיקה זו על מנת לצפות איך אנשים תופסים את העולם האמיתי. אנו מדגימים זאת על ידי חיזוי דמיון של מושגים מהעולם האמיתי. לאחר מכן, אנו מציגים אלגוריתם, שלומד

את דינמיקת רשת האינטרנט על מנת לחזות אירועים חדשותיים בעולם האמיתי.

בחלק השלישי של התיזה, אנו מציגים אלגוריתם לחיזוי אירועים עתידיים בעולם האמיתי בהתבסס על הידע ברשת הווירטואלית. מכיוון שאירועים ספציפיים הם נדירים, אלגוריתם החיזוי חייב להיות מסוגל ללמוד ולהכליל מאירועי עבר. אנו דולים מידע מארכיון חדשותי של 150 השנים האחרונות, מכלילים אותם בעזרת אנתולוגיות ברשת, ומציגים מערכת, המייצרת תצפיות בשפה אנושית.

בחלק האחרון של התיזה, אנו מציגים אלגוריתם לחיזוי אירועים עתידיים בהתב-סס על דינמיקת הרשת וגם בהתבסס על הידע הנמצא ברשת. האלגוריתם דולה מידע מהרשת לגבי שרשראות אירועים, ומכליל אותם בעזרת אנתולוגיות. לדוגמא, האל-גוריתם צופה כי לאחר שרשרת אירועים המתחילה בבצורת, ששנה לאחריה שיטפונות מכים את האיזור, ההסתברות להתפרצות מחלת הכולירה במדינות עולם שלישי, הנ-מצאות רחוק ממקורות מים, היא גבוהה מהרגיל. אנו מציגים מספר משימות חיזוי, הכוללות חיזוי של מהומות, מחלות ומוות, ומדגימים את ביצועי האלגוריתם במשימות הללו.

העבודה שלנו היא בין הראשונות בבינה מלאכותית, המדגימות יכולת חיזוי כללית. אנו מציגים שיטות חיזוי ממגוון רחב של מקורות אינטרנטיים על מנת לבצע תצפיות על בסיס ידע רחב לגבי סיבתיות בעולם.