

LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
INSTITUTE FOR INFORMATICS

Media Informatics Group
Prof. Dr. Andreas Butz



Master Thesis

TimeLineCurator: Interactive Authoring of Visual Timelines from Unstructured Text

Johanna Fulda
johanna.fulda@campus.lmu.de

Time frame: 11 November 2014 to 15 May 2015
Advisor: Simon Stusak
Professors in charge: Prof. Dr. Andreas Butz (LMU München)
Prof. Tamara Munzner (University of British Columbia)

Abstract

Interactive visualizations are popular elements inside digital news. In comparison to static infographics, they engage the reader, promote interest, and can lead to a better understanding of complex issues. Timelines are one recurring form of interactive visualization. They summarize events relating to a particular topic and give a quick overview of the temporal progress of a story.

In this thesis, I investigated how timelines are created, compared timeline authoring tools and approaches, and the challenges that timeline authors face. Drawing from interviews with journalists as well as my own experience of working in a newsroom for three years, I envisioned automating parts of the timeline authoring process. Combining methods from *Natural Language Processing* and *Information Visualization*, I developed TimeLineCurator, a browser-based authoring tool. It automatically extracts events from unstructured text and allows authors to easily and visually curate the extracted data. On the one hand, the tool is meant to facilitate the creation of a timeline for presentation; on the other hand, it can be used as a tool for analysis, since it enables a fast way to determine whether a document contains temporal information, and if so, what timeframe it relates to.

TimeLineCurator was evaluated by way of interviews and an analysis of usage scenarios emanating from the journalism community. We also received positive feedback from a broader prospective user community following the online deployment of TimeLineCurator, which included ideas for extensions as well as integration with existing tools and workflows.

Zusammenfassung

Digitale Technologien ermöglichen es interaktive Visualisierungen innerhalb Online-Nachrichtenseiten zu verwenden. Im Vergleich zu statischen Infografiken werden Leser dabei dazu motiviert selbst Einfluss auf die Darstellung zu nehmen. Das kann zum einen das Interesse des Lesers steigern, zum anderen sogar dazu führen, ein komplexes Thema besser zu verstehen. Eine immer wieder auftauchende Form einer interaktiven Visualisierung ist der Zeitstrahl. Er fasst wichtige Ereignisse eines übergeordneten Themas zusammen und gibt dadurch einen guten Überblick über den zeitlichen Verlauf des Themas.

Für meine Masterarbeit beschäftigte ich mich damit wie man einen Zeitstrahl erstellen kann und welche unterschiedlichen Werkzeuge und Vorgehensweisen es für die Erstellung gibt. Durch eigene Erfahrung, die ich während meiner Arbeit in einer Nachrichtenredaktion machte, und durch Gespräche mit Journalisten ermittelte ich auftretende Probleme bei der Erstellung und überlegt wie man den Prozess vereinfachen könnte.

Mit maschineller Sprachverarbeitung und Techniken aus der Informationsvisualisierung entwickelte ich ein browserbasiertes Programm, das frei formulierten Text automatisch nach Zeitangaben durchsucht und gefundene Daten auf einem Zeitstrahl darstellt. Zum einen ermöglicht es TimeLineCurator einfach und schnell einen Zeitstrahl zu erstellen. Zum anderen kann TimeLineCurator aber auch zur Analyse eines Textdokuments eingesetzt werden, da automatisch festgestellt wird, ob und welche zeitlichen Informationen ein Dokument beinhaltet.

Wir führten Interviews und beobachteten Testanwender, um TimeLineCurator zu evaluieren. Nachdem wir das Programm online zur Verfügung stellten, und dadurch wesentlich mehr Leute erreichten, erhielten wir neben positiven Rückmeldungen auch einige Ideen wie TimeLineCurator in Zukunft weiter entwickelt werden könnte.

Task

The creation of event timelines in a journalistic context should be facilitated. We propose to use natural language processing to detect temporal expressions inside unstructured text to generate a scaffold of a timeline that is easily adjustable by the author afterwards, in conjunction with the capability of visualizing the resulting temporal data in timeline form. The tool has to be web-based for platform independence and to avoid installation overhead. The interface should be intuitive according to current HCI principles and accept unstructured text as its input. The timeline author should be able to manually refine the created scaffold by editing, adding or removing events, and enriching them with media like pictures or videos. The output of the tool is an interactive visualization for use by an online reader that supports drill-down from the overview level to explore selections in more detail.

Preface

Prior to formulating this thesis we submitted a paper to the IEEE Conference on Visual Analytics Science and Technology (IEEE VAST 2015, <http://ieevis.org>). The paper was co-authored by Matthew Brehmer and Tamara Munzner from the InfoVis Group at the University of British Columbia (UBC) in Vancouver, Canada. Several paragraphs in the following thesis are adopted literally and will be marked with vertical lines on both sides (like this paragraph). To stay consistent in the wording I will also refer to myself as “we”, the individual contributions are clarified below. The paper in its version from March 31, 2015 can be found in appendix A.2 (page 61 ff.).

The individual contributions to this work, split between the three paper authors, are described in the following: The general idea for the concept is based on my experience as a working student in a newsroom for three years. The detailed concept originates from discussions with Tamara Munzner and the InfoVis group at UBC. The prototype was developed by me and went through three major states: the implementation of the initial idea, the deployment online, and the addition of a read-only version for presentation. Between the states we refined the design together as a group, the implementation was done by me. The informal interviews and test cases were conducted by Matthew Brehmer and me. The Timeline Authoring Model was suggested by Matthew Brehmer and elaborated together with Tamara Munzner and me. Pre-paper considerations about structure and content were suggested by me and refined together with UBC’s InfoVis group. The writing was done in alternating passes between Matthew Brehmer, Tamara Munzner and me. Illustrations and graphics were designed by me.

I hereby affirm that I have composed this thesis independently, that I have marked all quotes as such, particularly sections taken from the research paper co-authored with Matthew Brehmer and Tamara Munzner, and that I have declared all used resources and tools.

Vancouver, May 15, 2015

.....

Contents

1	Introduction	1
1.1	Computational methods in the news	1
1.2	Motivation: Enhancing accessibility	1
1.3	Method: Combining NLP and InfoVis	2
1.4	Contribution: TimeLineCurator & Authoring Model	3
1.5	Outline	4
2	The big picture of timelines	5
2.1	Transmission of Time	5
2.1.1	Sense of time to Language	5
2.1.2	Language to Data	6
2.2	Visualizing Time	8
2.2.1	Time as quantitative value	8
2.2.2	Marching along the line	9
2.2.3	Historical excursion	10
2.3	Use cases	12
2.3.1	Edgy examples	12
2.3.2	Scientific examples	13
2.3.3	Interactive general purpose examples	15
2.4	Timelines in journalism	16
2.4.1	Shiny examples	16
2.4.2	Benefits	17
2.4.3	Missed opportunities	18
3	Creating Timelines	19
3.1	Common approaches	19
3.1.1	Manual drawing	19
3.1.2	Structured creation	20
3.2	Challenges	21
3.3	New approach	22
3.4	Timeline Authoring Model	23
4	Related Work	25
4.1	Visualization Authoring Tools	25
4.2	Timeline Visualizations from Structured Event Data	27
4.3	Extracting Time Expressions from Unstructured Text	29
4.4	Visualizations from Unstructured Text	30
5	TimeLineCurator	31
5.1	Identifying TimeLineJS Limitations	31
5.2	Requirements & Goals	32
5.3	Design Process	33
5.3.1	Necessary elements	33
5.3.2	Design Iterations	34
5.3.3	Implementation	37
5.4	Architecture	39
5.5	Interface & Design Rationale	41
5.5.1	Timeline View	41
5.5.2	List View	41
5.5.3	Document View	42

5.5.4	Control Panel	42
5.5.5	View Coordination and Navigation	43
5.5.6	Presentation and Export	43
5.5.7	Exemplary walkthrough	43
6	Analysis	45
6.1	Extraction Error Benchmark	45
6.2	User Experience Comparison	47
6.3	Speculative Browsing	48
6.4	Curated Examples	49
6.5	Use Cases	50
6.5.1	Solicited potential users	50
6.5.2	Unsolicited current users	51
7	Discussion & Future Work	53
7.1	Discussion	53
7.2	Future Work	54
8	Conclusion	55
A	Appendices	61
A.1	Contents of the CD	61
A.2	VAST Paper	61

1 Introduction

Working in the newsroom is an exciting and fast-paced experience. The motivation for this thesis springs from that environment and aims to give an idea of how digital tools can and should be used more frequently. It introduces the idea and implementation of TimeLineCurator, a tool that was developed for improving the specific case of creating event timelines.

1.1 Computational methods in the news

The traditional newspaper in its analogue paper format is in a state of crisis these days. Still we all rely on solid news reporting and must not trust anything that is floating around the Internet. Thus reliable journalism is still in high demand, but has to undergo a makeover using today's technological possibilities. Using computational methods inside the newsroom is not new at all, but due to the increased prevalence of tablet computers and smartphones, the distribution channels for news has also changed. These days, news are consumed mainly digitally, which opens up many new ways (and challenges) to present and explain information.

Instead of using a big data set as mine of information and perhaps taking a selection from it to present to the reader (visualized as a bar chart or the like), a journalist could provide a bigger picture and offer more of the original data.

Either by simply linking to the origin of the data or by transforming it into a visualization, where patterns can be discovered or the readers can delve deeper into data concerning their personal interest. That not only engages readers because they get the chance to influence the visualization, but also builds trust and meets the demand for transparency of sources. However it is a slow process for journalists to become aware of those technological possibilities and often it is rather encountered with skepticism. The two worlds of sophisticated new computational methods and news reporting do not seem to overlap too much. Thus helpful tools created by computer scientists or engineers remain untapped by journalists. One reason for not keeping pace with new technology trends is its rapid speed of change. To be constantly informed about the newest trends and developments, you basically have to spend most of your time tracing them. Those who don't are quickly left behind. In today's journalistic education, computational tools for analysis as well as for presentation are part of the curriculum but there's still a long way to go, until they will actually be used to their full potential. Of course there exist exceptions to the rule and a huge community of tech-savvy journalists push forward fields like data-journalism and make use of the newest exciting trends in technology.

There are two different kinds of tools that aim to help journalists. Tools can either support analysis, thus the author's research; or they can be for presentation, where they support creating cleaned-up output for the reader, with information that is based on knowledge the author already acquired. The goal of this thesis was primarily the presentation task, but later we discovered that a tool intended for presentation can also be used as a tool for analysis.

1.2 Motivation: Enhancing accessibility

While working in a newsroom for three years, I discovered that there were attempts to include computational methods into the daily workflow, but often well-intended ideas ended up in irritation and finally rejection, where journalists reverted to old known workflows. The hectic environment doesn't allow for much time to learn new tools very often. Thus long-winded but well understood workflows win out over new unknown ones, even though these may have the potential to save time in the long run; for example, much

time was regularly spent on typing numbers into a table from another table or retyping text passages. In the individual case it often is faster that way, but being able to deal with spreadsheet operations or knowing how to use optical character recognition would be beneficial in the longer term. Another consequence of not knowing about available tools was that sometimes ideas for exciting interactive visualizations within news articles were precluded. Sometimes because ideas were too ambitious, but in many cases the realization of those ideas failed because of the lack of knowledge or experience with available tools.

When it comes to infographics, one recurring form of visualization were timelines. Event timelines aim for giving an overview over a sequence of events that all belong to a particular topic. Timelines can, for example, show happenings in the news, the biography of a person, the history of a company, the development of a certain technology, or a review of events that led to or influenced a historic event. A visual timeline for a print product is created “by hand” - in an illustration program, but composed manually; a digital version often simply builds on top of that static print graphic, perhaps adding photos or additional information. Even though we assumed that there were existing tools to create more elegant interactive timelines, that were not based on a static graphic but independent and responsive, there was never enough time during the workday to devote attention to finding out more about them. As a thesis project, I wanted to take a closer look at possible ways to improve the process of creating timelines. I knew that often the content of a timeline was either based on the article that it is accompanying, or that the information about the single events came from only a few different source documents, such as a short biography or a Wikipedia page.

1.3 Method: Combining NLP and InfoVis

After learning more about automated timeline creation I discovered that there existed several advanced tools that allow for an easy creation, so there was actually no need to revolutionize the idea of translating event data into an interactive visual timeline. Only the process of generating the underlying event data set seemed to indicate potential for improvement, since authors have to create the underlying event data set in a rather tedious process. They have to manually enter dates into a spreadsheet or in some other structured file format. It turned out that this manual data set generation appears intimidating to many authors, and was a common reason for not using those tools in the first place. I imagined if that process was easier and more visual it could attract also less technologically-inclined people to actually use it. Also, the author should not have to start with an initially empty data set but get suggestions based on the source documents he has available - which could automatically be searched for information about events. We will refer to those source documents as *unstructured text* in the following. Unstructured meaning that the text is written freely in consecutive sentences, without following any structured pattern. Since events most often have a temporal component - keywords referring to the time of occurrence - temporal expressions could be used as indicator for an event. After investigating existing methods for temporal information extraction from unstructured text, I decided to combine the two domains of *Information Visualization* (InfoVis) and *Natural Language Processing* (NLP) to approach the problem. To meet the original demand for facilitating the creation process without requiring to learn a complicated tool, it has to be easily accessible as well as easily understandable. To make it accessible the author should not be required to download or install anything (which may require admin rights and hard-drive space), but he should rather be able to access the environment from anywhere with a regular browser. To ensure a user-friendly creation process principles of current humancomputer interaction (HCI) should be considered and applied. And finally the environment should offer several options for export to enable distributing the resulting

timeline not only in one default way, but in different ways. Also the raw data should be accessible to the author.

1.4 Contribution: TimeLineCurator & Authoring Model

We analyzed common methods for creating visualizations, as well as methods for creating timelines in particular. We discovered which aspects of these tools work well and which do not. Based on personal experience and via several semi-structured interviews, we found out what timeline authors would like to have and what confuses them.

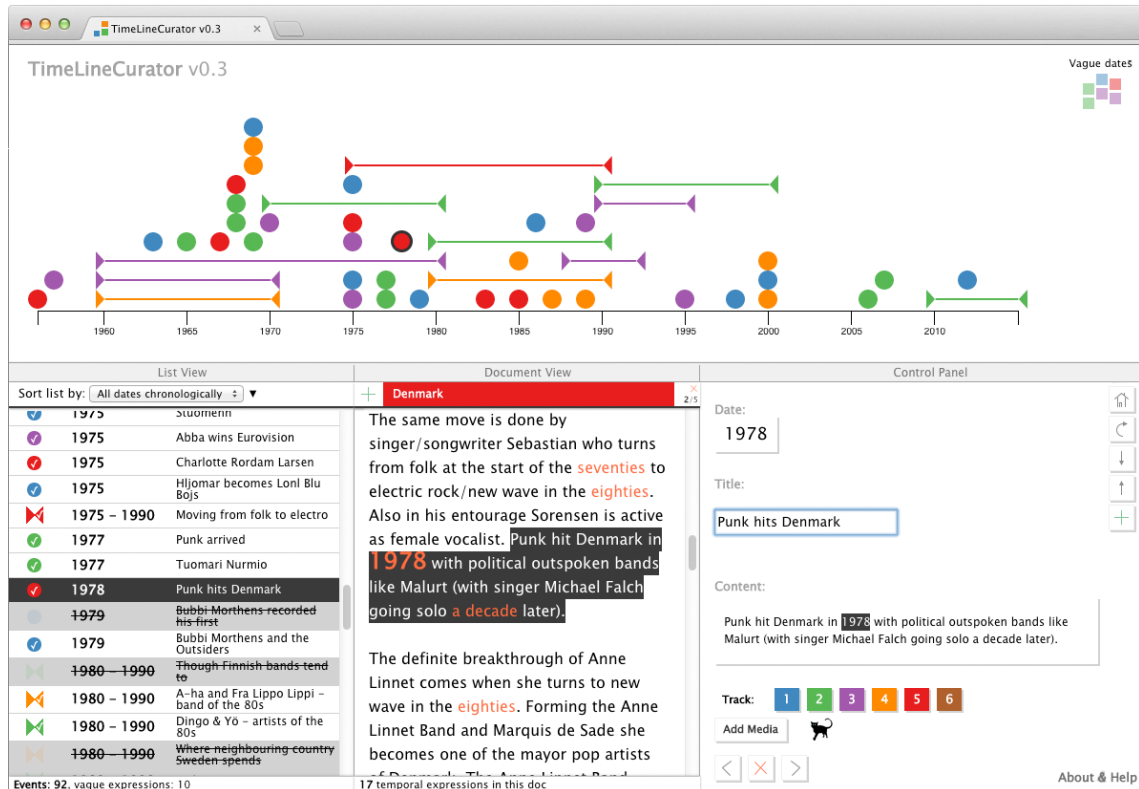


Figure 1.1: The browser-based visual timeline authoring tool TimeLineCurator, showing a timeline of Scandinavian pop music, where each color corresponds to a country; access the interactive timeline at <http://goo.gl/0bH1vA>.

Our primary contribution is TimeLineCurator, the web-based visual timeline authoring system shown in Figure 1.1. It allows for the fast and easy creation of a structured temporal event data set from unstructured document text, combining imperfect natural language processing and “human in the loop” authoring. With TimeLineCurator, an author can speculatively browse a document’s temporal structure; she can quickly rule out documents as unsuitable for timelines within seconds, or interactively curate suitable documents to refine an event set within minutes, receiving constant visual feedback throughout the curation process. Our secondary contribution is a Timeline Authoring Model, which we use to position TimeLineCurator relative to other timeline generation approaches in terms of goals and tasks.

1.5 Outline

Before we provide more detail about the TimeLineCurator project, we will consider the big picture of timelines in **Section 2**. We start with how we as humans imagine time in our heads, how we translate these thoughts into language, and how written descriptions of time can be interpreted by machines in **2.1**. We also survey how time can be visualized on paper in **2.2**. Historical as well as current examples from several domains show the benefits of visual timelines in **2.3** and lead us to today’s usage of timelines in journalism in **2.4**. In **Section 3** we describe different approaches to create timelines in 3.1 and the challenges they pose in 3.2. Based on this survey, we suggest a new approach in 3.3, which we compare to previous approaches according to our *Timeline Authoring Model*, which we introduce in 3.4.

Section 4 gives an overview of related work. It is separated into four parts concerning the different tasks that TimeLineCurator addresses. **4.1** and **4.2** consider the InfoVis part: general authoring environments for visualizations and those for timelines in particular. **4.3** relates to the NLP task of the temporal information extraction. **4.4** points out projects that already join both parts and use NLP to generate different kinds of visualization - many of them address topic extraction.

Section 5 considers the development of TimeLineCurator in detail. **5.1** outlines the limitations of current authoring systems, **5.2** then defines our goals and requirements based on that. The iterative design process and its resulting architecture are described in **5.3**, the single components and options for export are explained in **5.5**.

The analysis of TimeLineCurator will be described in **Section 6**. We conducted benchmark tests (6.1), asked people to compare TLC to current timeline authoring tools (6.2), show examples of speculative browsing (6.3) as well as curated results (6.4) and describe experiences and feedback from users and the community in 6.5.

Discussion and ideas for future work will follow in **Section 7**, the conclusion in **Section 8**.

2 The big picture of timelines

In addition to the three dimensions of space time is the 4th dimension¹, time is inseparable from space², “Time is what keeps everything from happening at once”³. Those thoughts on the abstract nature of time give an idea that time is hard to grasp but still somehow implies a spacial representation.

We won’t go deeper into matters of physical reality, psychology or behavioral science; however it is worth mentioning that time, despite its ubiquity, is something impalpable and cause for much philosophical and scientific analysis. In the following we will have a look at how temporal information is expressed in language and how we can transform the written word into a form that is interpretable by a machine. Despite its abstract nature, time can also be expressed as a quantitative value and visualized graphically in many different ways, but it has by far not always been clear of how it can be best visualized. At the end of this section, we will look at the special case of timelines in journalism, and how they can be used, for example, to explain chronological developments inside a story or to summarize historic events.

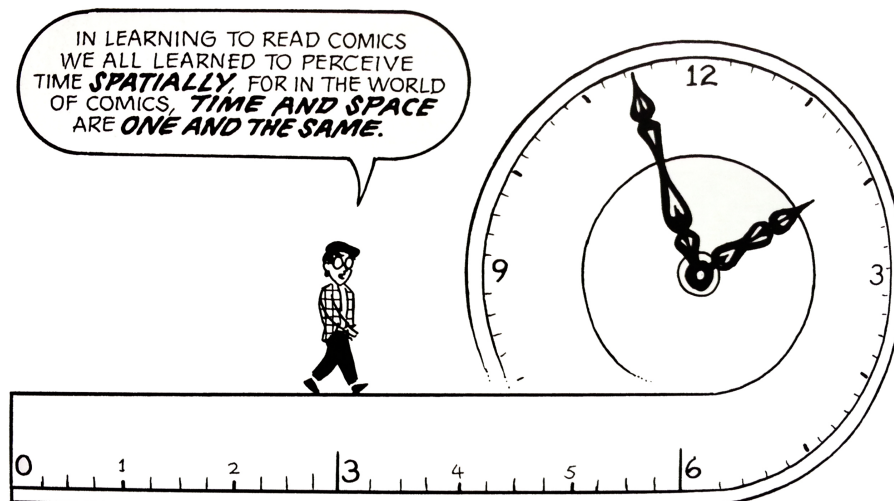


Figure 2.1: From Scott McCloud’s “Understanding Comics” [26], Chapter Four “Time Frames” (1993)

2.1 Transmission of Time

To convey the time and duration of an event we use certain grammatical rules and often spatial metaphors. These rules and metaphors allow us to understand it quite precisely, but in order to translate it into machine-readable data, so that a computer can classify an event’s time and duration, we need to search for patterns and write rules - which is one approach of natural language processing.

2.1.1 Sense of time to Language

Many expressions in our language use spatial metaphors to express time. Also, our imagination of time is closely connected to space. Often we use the metaphor of walking along a path. Everything in front of us refers to the future, and things behind us happened in

¹introduced by Sir Isaac Newton in his “Mathematical Principles of Natural Philosophy” from 1687

²That is according to how Scott McCloud explains comics [26] in Figure 2.1; and also in common language use, see Section 2.1.1

³From Ray Cummings’ science fiction novel “The Girl in the Golden Atom” from 1922

the past [27]. Because we know that time is continuous and nonrecurring, we often use one-dimensional terms to describe time. We say, for example, “I am looking *forward* to something” or “when I’m looking *back*”. Those spatial schemes that we create in our minds provide us information about how to organize events in the continuous flow of time [3].

We use temporal words to indicate when something happened, or to position it relative to something else. Most languages use similar ways to refer to time, however we will primarily focus on English. In written and spoken English, *tense* and *aspect* give information about time. *Tense* (the grammatical form of a verb: past, present, future) allows us to “localize” events or states in time, whereas *aspect* tells us more about the flow of the event. Aspect can either be *perfective* or *imperfective*. *Perfective* means a bounded and self-contained event, where the verb indicates an action that took place at one point in time, beginning and ending included (“I ate ice cream”). Whereas *imperfective* indicates a continuous or repeating event (“I was eating ice cream” or “I used to eat ice cream”) where it is implied that the event had a certain duration. In “How time is encoded” Klein [17] describes four more types of devices that encode time in language. These devices include *lexical aspect* (where the meaning of the word implies more about its temporal feature, for instance sleep vs. crack), *temporal adverbials* (temporal expressions like “soon”, “now”, “rarely”), *temporal particles* (only present in few languages, such as Chinese, where a suffix of a verb can influence its temporal meaning), and *discourse principles* (position within the whole story, assuming that the default order is chronological).

In order to place an event in a sequence, we need a reference date, which is the time of speaking or the time a document was written. With this point of reference, we are able to get information about the time of events, compared to the reference date or in relation to another event (whether it was before, at the same time, or after some other event).

2.1.2 Language to Data

When we speak or write about events, we do not always use concrete dates that we might see in a calendar, like “*on September 16, 2008 stock markets crashed*”. Of course, the style of writing or speaking varies according to different purposes, but if we don’t write formal reports, it’s rather unlikely that we annotate every event with a concrete date. We often use vague temporal references, like some point in the future or past, or a reference time, like “*at the time of the economic crisis*”. Also, the duration rarely gets indexed; often we can infer if the event was short or long. To understand the context it might be enough information and we can already order events in time and get the chronology straight in our heads. If we want to make the time understandable by a machine and automate the process of extracting temporal information from unstructured text, these vague temporal expressions present some challenges.

That’s where the field of *Natural Language Processing* (NLP) comes in. There are two different approaches to process written text: the **rule-based** approach and the **machine learning** (ML) approach. With the former approach, the machine can stubbornly follow some rules, using search strings and regular expressions to identify something specific. With the latter approach, ML algorithms can be applied to identify something based on previous “experience” gained from an annotated example document collection. (There are more approaches, such as unsupervised learning, where the example data doesn’t have to be annotated as much, but we won’t go deeper into that).

Currently, ML approaches dominate the field. However, in some cases, the heuristic rule-based approach is better suited. Recognizing temporal expressions, for example, is one such case, because we assume that there is a huge but finite amount of possibilities to express them. That’s why we can use rules, which are very powerful when it comes to pattern matching. The big advantage compared to ML is, that there is no expensive

annotation of documents and training cycles needed. The rules make sure that instructions are followed stubbornly, so it is more likely to comprehend why the system does what it does - which you can't always tell when it comes to ML. Of course, writing the rules and dictionaries is a very tedious process as well, and in general rule engineering doesn't scale too well. The more rules you have, the more possibilities of them getting into each other's way appear and it can easily result in a messed up system. The order in which the rules are run has to be determined and also what happens when a rule matches - should it look further or stop immediately. Which approach performs better depends on the number of rules or the algorithms a ML approach uses and can't be determined in general.

When raw data gets bigger, more chaotic and noisy ML becomes more suitable. ML algorithms can, for example, create decision trees to enable a decision in the case of ambiguity. Based on experience they are able to go for the more likely case. Since ML models are robust also to unfamiliar input (like typos or unknown words), they can handle huge unorganized data sets to a greater extent than a rule-based approach. To improve an ML system, the creators can provide it with additional annotated test data; this is time consuming, but much less complex than expanding rule-based systems. Still, NLP tasks such as "Named Entity Recognition" (to recognize names, such as persons or organizations) or "Morphological Segmentation" (to identify the class of a word), hand written rules and dictionaries can obtain better precision.

Once a temporal reference is detected, the machine might still not be able to find the date in a calendar, because it is probably not expressed in a standardized format. "Normalizing" these expressions is required, which means translating them into a standardized format. This can happen by reformatting it (for instance *3 May 2015* becomes *2015-05-03*), or by calculating its date relative to a reference date such as the document creation time (DCT) - which is the time when the text was written (for instance *yesterday* becomes *DCT - 1*). The machine-readable format looks like this: YYYY-MM-DD, which is the ISO 8601 standard.

With **Recognition** and **Normalization**, a machine can determine with some certainty which date a temporal reference inside unstructured text refers to [24]. In numbers, state-of-the-art techniques can achieve F-measures of around 0.9 [62, 21].

Already way earlier Priestley, who is called the inventor of the modern timeline (more in 2.2.3), describes the line as the most suitable representation of time in his book “A description of a chart of biography” [33].

Thus the abstract idea of TIME, though it be not the object of any of our senses, and no image can properly be made of it, yet because it has real quantity, and we can say a greater or less space of time, it admits of a natural and easy representation on our minds by the idea of a measurable space, and particularly that of a LINE; which, like time, may be extended in length [...] and thus a longer or shorter space of time may be most commodiously and advantageously represented by a longer or shorter line. (Joseph Priestley)

According to that quote from 1765 [33] and according to today’s overall experience the most common representation for time is a line in all variants of shape: circular, to visualize periodicity, wavy to show vacillations and so on. The most basic form when it comes to aligning events is along a horizontal straight line, where time is “running” from left to right.

2.2.2 Marching along the line

If we draw visualizations in two dimensional space, we basically have two axes along which we can align our elements. The horizontal and the vertical axis. Both often indicate increase in value, either to the top or to the right.

Vertical alignment The reasons for why a top position is interpreted bigger, more or better than a lower one comes from our ubiquitous gravity and thus how we and everything around us is oriented [40]. Physically speaking: something at the top has more potential energy than something lying on the ground. The vertical position is also prevalent in our language: “being on the top”, “feeling down”. That’s one reason why the vertical axis is convenient for displaying value, quantity or ranking. Figure 2.3 shows examples, such as weather graphs or poll results - the more quantitatively, the further at the top. Not following that idea can lead to quite some confusion as Figure 2.3 c) shows - here the vertical axis is reversed and gives the impression that gun death went down after the “stand your ground” law was passed in Florida in 2005 - even though exactly the opposite was true.

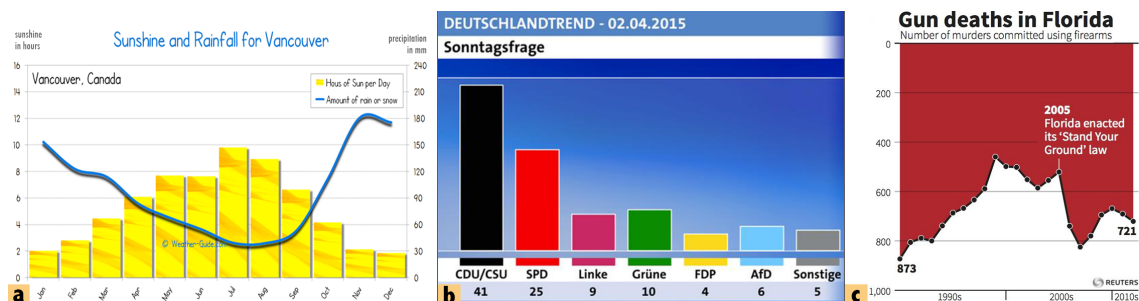


Figure 2.3: Three examples of graphs visualizing quantity along the vertical axis: a) Information about precipitation and sun in Vancouver (<http://www.weather-guide.com>), b) Regular political poll about German parties’ popularity (<http://tagesschau.de>), c) Example of a misleading chart from Reuters, showing the vertical axis upside down (<http://goo.gl/gmZwJz>)

Horizontal alignment An increase from left to right doesn't seem to have natural causes, because on the horizontal plane most things are generally pretty symmetric. Including our body. Except that most people have one dominant hand. The handedness influences not only our ability to act but according to a study from 1988 by Ladavas also our judgment of the referential term on the horizontal dimension [20]. Even more influential however are the different directions of writing. A study by Tversky et al. from 1991 tested how that influences spatial perception [44]. With around 1200 participants (children, as well as adults) from different language cultures (English, Hebrew, Arabic) they looked at how they would use space to represent relations that are non-spatial. They let their study participants place elements on a paper, that either represented temporal information, quantitative information or their preferences. The result for the temporal alignment was a strong tendency to left-to-right for English speakers, and the other way around for Arabic speakers (Hebrew speakers were less coherent - probably because they are more likely to learn European languages, also arithmetic operations are made from left to right in Hebrew). Interestingly very young participants tended to align the temporal information from top to bottom [44]. In figure 2.3 a) we already saw the progression of the year marching along the horizontal line, Figure 2.4 shows two more use cases other than bar charts.

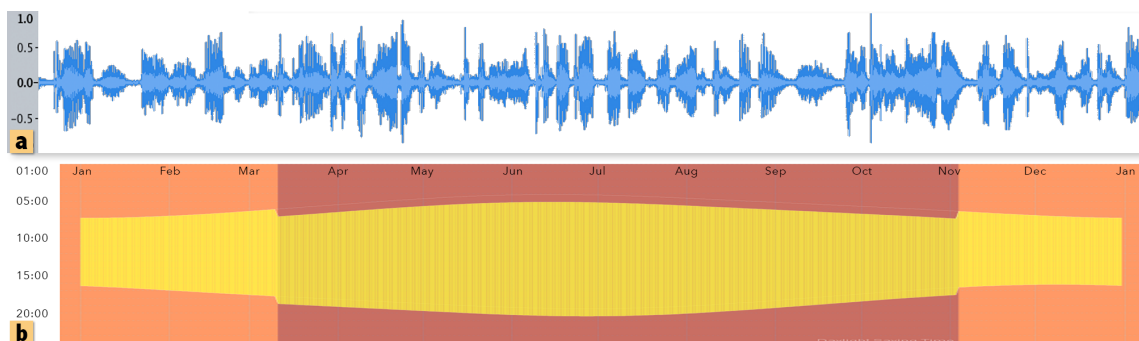


Figure 2.4: Two examples with the horizontal axis being the time floating from left to right: a) The audio track as it can, for example, be found inside an audio editing program, b) A chart plotting the times of sunrise and sunset over one year (in this case 2007 in Boston, MA, USA), based on data from <http://daylightchart.sourceforge.net>

Marking events The single events that are aligned along that line can have different properties. They can vary in duration and importance or belong to different categories. So we need to think about how to indicate these differences. In today's timeline tools we often see a dot for an instantaneous event, a line for a period of time and either coloring them accordingly to their categories or use vertical alignment to indicate their affinity. Weighting events according to their importance is not very common, but could be done by changing the radius of the circle and increasing the line width or, for example, through saturation of color.

2.2.3 Historical excursion

Distributing events along a one-dimensional line seems well accepted and easily understandable today. However Rosenberg pointed out it is "only quite recently that scholars first thought to represent chronological relationships among historical events by placing them on a measured timeline" [36]. Historically we could go back to ancient cave paintings and claim those drawings to be very early timelines, but let's start with "modern" timelines from the 17th century. ("A Timeline of Timelines" shows an elaborate listing with

representative examples, beginning with the very first discovered timeline-like records, referring back to the second century A.D. [53]).

“The timeline seems among the most inescapable metaphors we have. And yet, in its modern form, with a single axis and a regular, measured distribution of dates, it is a relatively recent invention.” (Rosenberg and Grafton)

In the book “Cartographies of Time: A History of the Timeline” [37] Rosenberg and Grafton give an in-depth review of graphic representations of time in Europe and North America from 1450 until today. They entitle Joseph Priestley, an Anglo-American theologian, to be the inventor of the modern timeline as we know it today. 1765 he published “A Chart of Biography” shown in figure 2.5, where the life spans of 2,000 famous men between 1750 A.D. and 1200 B.C. were shown.

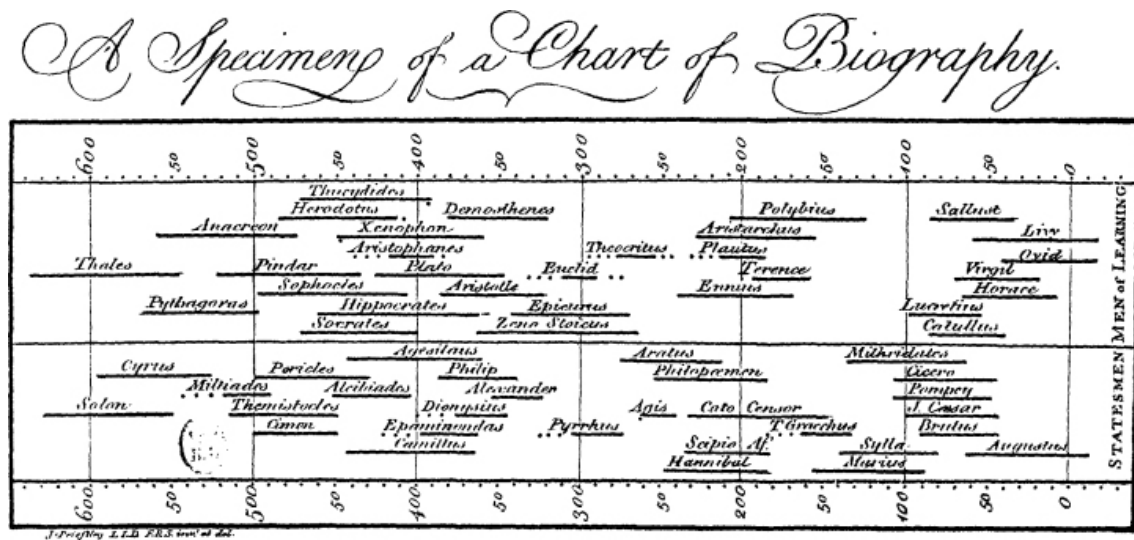


Figure 2.5: “A Specimens of a Chart of Biography” from Joseph Priestley, 1765 [37]

Because it was the first chart that included all the necessary visual vocabulary to map time, it was seen as the worthy successor of matrices - the standard representation of chronological data up to that point. Priestley established the idea of showing the lifespan of a person with a horizontal line (starting at birth date, ending at its death), to indicate dates that are uncertain he used dots. That new kind of representation needed explanation to be understood by people and also had opponents. Laurence Sterne for example, an English writer didn't like the idea of showing history in a straight line and preferred telling stories with digressions and deviations. Presenting time as something linear seemed like a construction to him. Nevertheless since the 18th century the timeline became more familiar and understood. It even played a crucial role in the “modern understanding of historical time as linear, chronological, and flowing from future to past” [30], and it enabled people to compare different events in history. The broad dissemination let new questions arise, like how can you fit all important events on a horizontal line without resulting in a 54 feet long scroll (like Dubourg's Chronological Chart from 1753) or how can you combine time data with additional information. In our modern digital times those questions can luckily be answered by using interactive methods like zooming and details-on-demand [59, 37, 30].

2.3 Use cases

Before we start focusing only on timelines in the news, we show a few selected examples of historic timelines and timelines in scientific environments and what the term “interactive” means in that context.

2.3.1 Edgy examples

To look a little outside the box of a straight horizontal timeline we show two examples of different kinds of timelines. They are both originally hand drawn and can rather be seen as artworks, more than infographics. It is probably necessary to see them in poster size to acknowledge their full beauty.

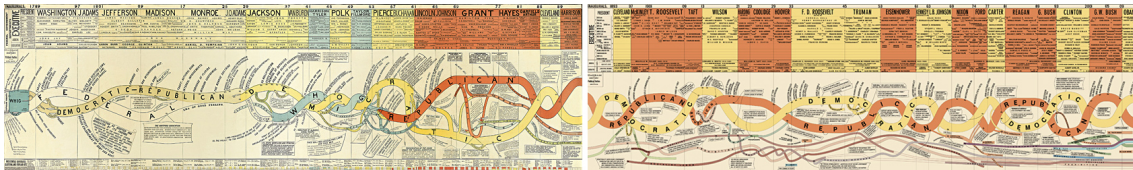


Figure 2.6: “The History of the Political Parties”, from HistoryShots <http://goo.gl/yWtMes>

The History of U.S. Political Parties The illustration shown in Figure 2.6 devotes itself to the history of political parties in the United States from the birth of the political system in 1789 until today (or the day of creation of the second part of the timeline). It is a two-part work made by designers living in two different centuries. Around 1894 the first part was created by an unknown designer, 115 years later Larry Gormley and Bill Younker continued the graphic with surely way more modern design tools, but modeled on the look and structure of the original piece. The graphic shows the serpentinous course of popularity of the single parties, combined with details about political and historical events. All the reigning presidents and their cabinet members are listed to give American history enthusiasts an artistic visual summary to hang up on the wall. The horizontal line indicating time is estranged here to also indicate the popularity, and also uses the possibilities of annotating it with summaries or headlines of events.

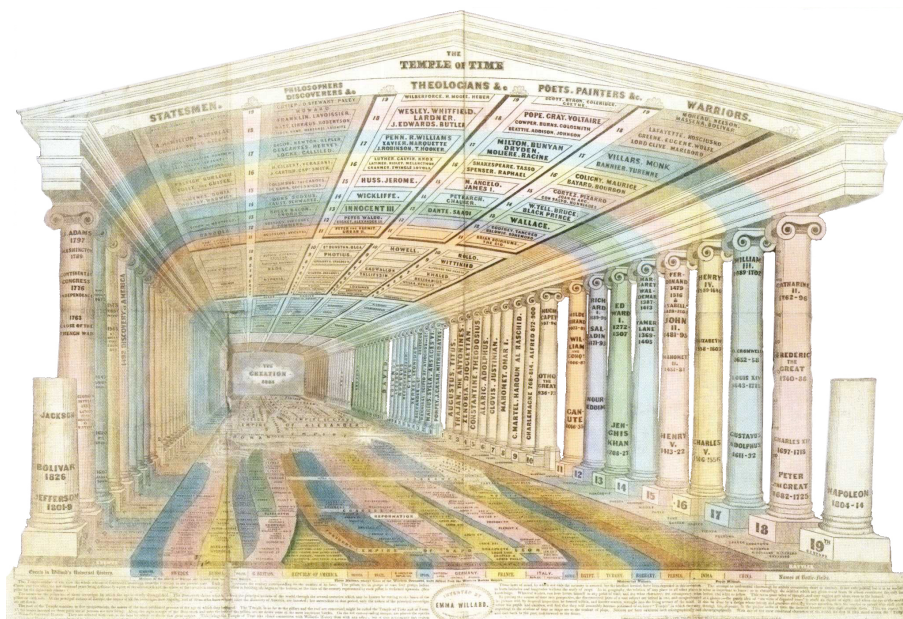


Figure 2.7: “The Temple of Time” by Emma Willard, 1846 [37]

The Temple of Time Figure 2.7 shows a graphic created by Emma Willard in 1846. It projects important figures and historical events onto a 3D temple. Columns represent the centuries, the floor shows historical events, the ceiling biographies of important people. The intention behind that representation was to facilitate remembering historical facts due to spatial imagination and architectural details [37]. As Willard wrote herself onto the chart: “The attempt to understand chronology by merely committing dates to memory, is not only painful, but [...] useless [...]. The relation which any given event bears to others constitutes the only useful knowledge.” She suggests that the viewers should (mentally) locate themselves inside the temple and look around, to see the characters of the time. To that time, this kind of illustration was very unconventional and must have been pretty mind-blowing.

2.3.2 Scientific examples

Many research areas use timelines for investigation and representation of events. Medicine, Engineering, History only being a few of them. And there certainly are more who would like to use them, but are lacking the tools or the knowledge. For example, in Computer Forensics, Olsson and Boldt claim that evidence plotted on a timeline would help investigators solve crimes faster and more intuitively [29]. We look at one example from medicine, meant to learn from the course of a disease and to improve medical treatment; and one from geology and human science, tackling the scalability issue in the history of the universe.

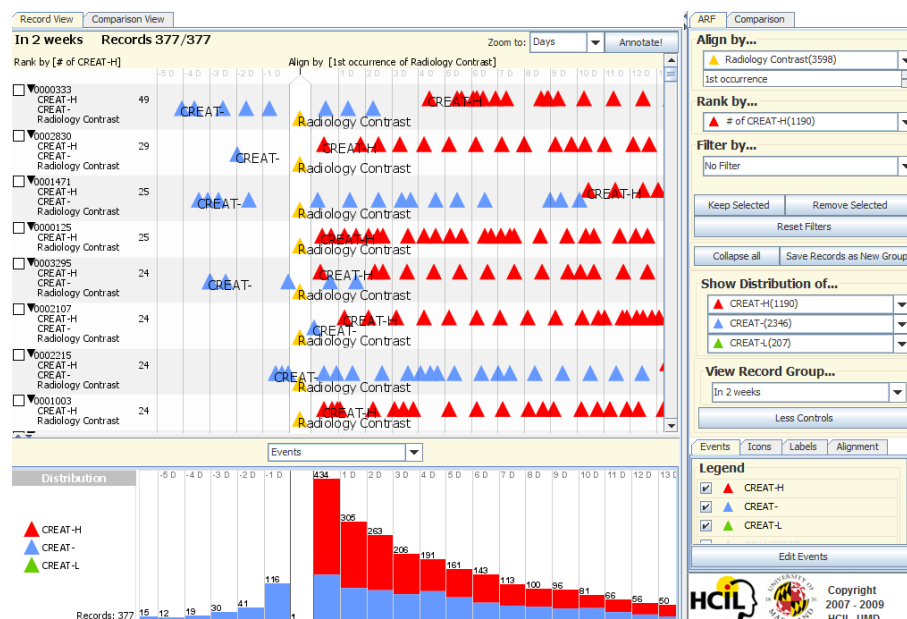


Figure 2.8: Lifelines2 to discover temporal categorical patterns across patient records. <http://www.cs.umd.edu/hcil/lifelines2>

LifeLines This project that was first presented 1996 at CHI by Plaisant et al. and is being improved and refined until today (LifeLines2, EventFlow [57]), visualizes personal medical histories generated from electronic health records as seen in Figure 2.8. Different incidents from medical records like reported problems, diagnoses, test results or medications are aligned on a timeline to enable the discovery of patterns that could give information about cause and effect. Temporal patterns can be highlighted, and it is possible to align several records from different patients one above the other according to one

central event (for instance a heart attack). That enables a comparison of symptoms and treatments between patients, because in that case the exact calendar date is less important than the elapsed time between incidents and the course of the disease itself [31]. The University of Maryland research group continued improving the tool, and now also developed EventFlow, which can not only handle point event data but interval data.

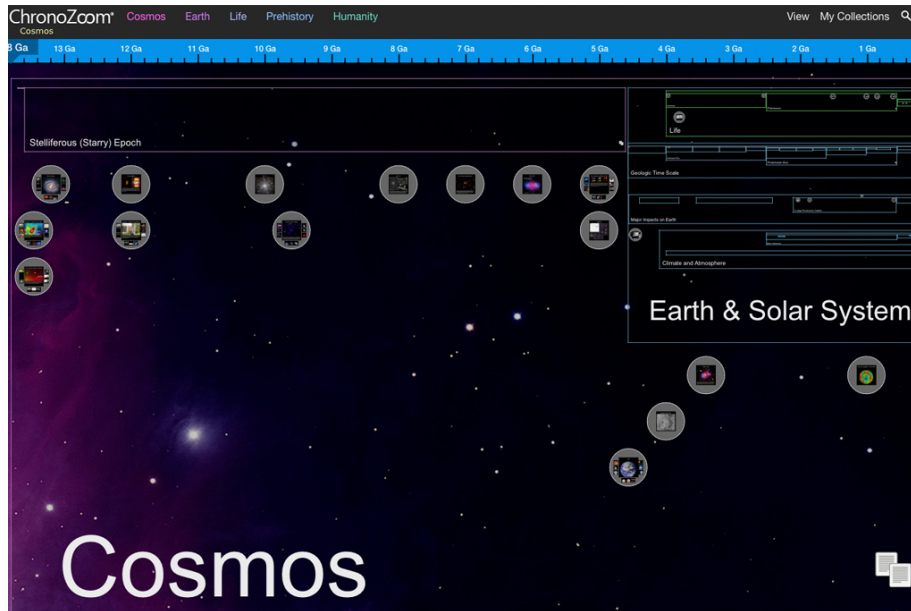


Figure 2.9: ChronoZoom project dedicated to “visualizing the history of everything” <http://www.chronozoom.com>

ChronoZoom Another example in Figure 2.9 approaches the problem of scalability in timelines. ChronoZoom started in 2009 at the University of California Berkeley. The idea was to develop something that gives an understanding of the very profound difference between the scale of humanity (which starts with the Ancient Egypt, ca. 5,000 years ago), geologic history (the earth shaped around 4.54 billion years ago), and the history of the universe (the big bang, which is supposed to be around 13.8 billion years ago). Doing the math shows: Humanity could fit 2,760,000,000 times into the time of evolution of the universe. Because it is impossible to show such scales on a paper or even a monumental poster, they assumed interactive tools could remedy this problem. The scalability challenge needed new deep zooming technologies. Microsoft Research, Moscow State University and the Outercurve Foundation joined in and (after some rather less performant attempts) an interactive, browser-based beta version of ChronoZoom was released in 2012. The tool enables the reader to navigate through 13.8 billion years, offering an all-in-one overview and interactive zooming into human history even to the level of selected single day events (which corresponds to a five trillion to one zoom ratio) - important events can be selected to get more information. Animations between the states give an idea of the distances that lie in between. All code is open-source and since 2013 there are also authoring tools available, enabling students, educators and scientists to create their own timelines, with other scientific, personal, or statistical data. According to the authors it offers the “opportunity to algorithmically generate timelines and exhibits that can help researchers in various fields, as well as the general public” [49]. A further aim of the project is now to establish some lesson plans for teaching historical thinking in schools and universities [49, 63].

2.3.3 Interactive general purpose examples

Interactivity is currently a widely used term and can mean a lot of different things. In many online applications it simply means that the reader can click on something to get more information - a hyperlink is the most basic interactive element on a website. Advanced interactive elements are those where readers can, for example, input their own data to look up things that match their personal interest or they can filter from a list, zoom into a map and so on. Yi et al. defined seven kinds of interaction inside information visualizations. Those are: Select, Explore, Reconfigure, Encode, Abstract/Elaborate, Filter, and Connect [51, 28, 5].

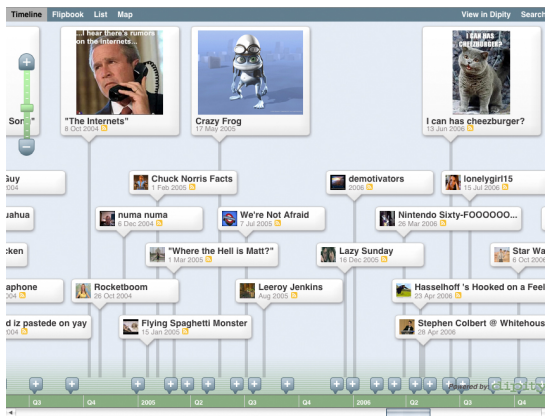


Figure 2.10: Interactive timeline about internet memes, created with Dipity, <http://www.dipity.com>

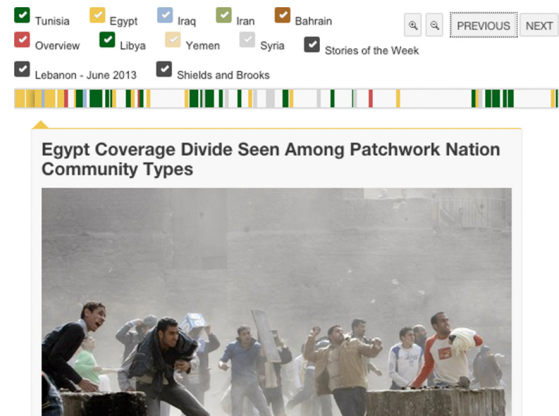


Figure 2.11: Timeline about the Arab Spring, created with TimelineSetter on PBS Newshour, <http://goo.gl/KquEJq>

In Dipity's timeline about memes, as shown in figure 2.10, the reader can *select* a meme to get more information about it, in that case a short description, a video or a picture and the chance to write a comment or share that meme on social media. Other timelines, like shown in figure 2.11 (created with ProPublica's TimelineSetter) let the reader *filter* what categories to display, and *abstract/elaborate* the scale. Some timelines use semantic zooming to provide different granularity for different zoom levels. In a zoomed out view events can for instance be aggregated to give an overview, and be layed out more detailed when zoomed in.

Interactivity can solve the issues of scalability and occlusion satisfactorily, as seen in the ChronoZoom project (Figure 2.9) and in the interactive examples above (Figure 2.10 and 2.11). The structured creation also enables a faster and more generic creation, much more information can be included compared to static timelines. Potential drawbacks are that the automatic generation allows for a sloppy selection of events, because the author doesn't have to care about space limits and they certainly are less of artworks than hand-drawn static ones, but simply serve the purpose of information transfer.

2.4 Timelines in journalism

After showing examples from several domains, we now focus on timelines used on news websites or in digital newspapers. They generally summarize a news topic, often something that is still ongoing and would benefit from being updated as soon as something new happens.

2.4.1 Shiny examples

There certainly are many examples, varying in shape and functionality. To give an impression of what's possible we picked out two quite different timelines. One minimalist example from *The New York Times*, and one colorful and loaded one from *The Guardian*.

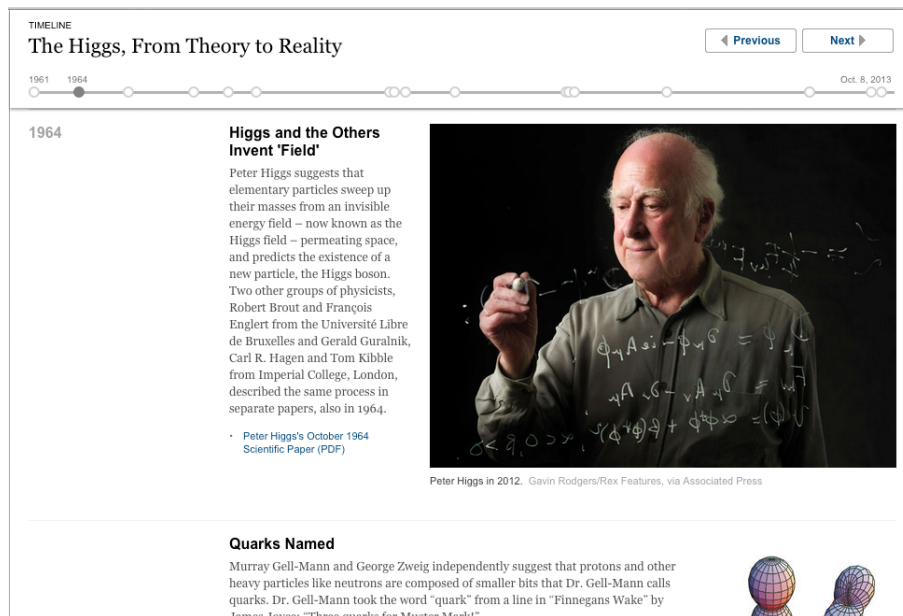


Figure 2.12: Timeline about the discovery of the Higgs boson, *The New York Times*, Mar 2013 [55]

The Higgs, From Theory to Reality Very simple but yet elegant timelines, that give a quick overview of a news story can be found at *The New York Times*' website - Figure 2.12 shows an example. Authors used that framework for several stories. It immediately gives an overview of the timeframe of the story, and thereafter accompanies the reader through the development of story. Content of the single events is composed as regular top-down text. On the one hand the timeline offers navigation through the page, by scrolling to the event's position inside the text when selected; on the other hand it always highlights the currently viewed event, when navigating through the text manually. The timeline is very simple and unobtrusive, but still offers a convenient visual assistance for the story.

The path of protest A colorful and ludic approach was done by *The Guardian* for their timeline about the Arab spring, shown in Figure 2.13. The reader can navigate through a huge data set of events. The data points are generated from all articles that covered that topic. All countries involved have their own track on a long three-dimensional uphill road towards the most recent data point. Different icons indicate different categories of events (protest, political move, regime change, and international response). Hovering over an event gives a quick summary of the data point, clicking on it guides the reader

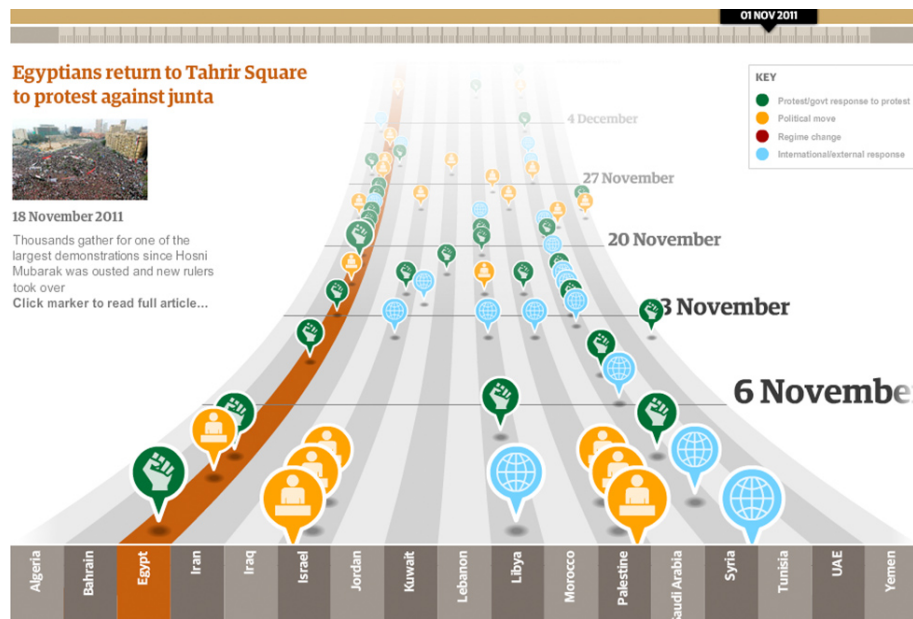


Figure 2.13: Interactive timeline about the Arab Spring, *The Guardian*, Jan 2012 [56]

to its accompanying article. A horizontal timeline at the top lets the reader keep track of the position within the whole timeframe. The timeline indicates the progress of time horizontally inside the top bar, as well as vertically (or rather along the depth axis into the third dimension) inside the main view. The interactive visualization first gives an impression of how much happened during the Arab Spring, and how it was distributed throughout different countries; second it serves as directory for stories happening at a specific date, and how they relate to others and the whole timeframe. Compared to the example in Figure 2.12 where the reader is invited to read the whole story chronologically, that timeline is rather meant for discovery and the bigger picture.

2.4.2 Benefits

As seen in previous examples, timelines can either be used as an accompanying element within a text story or they can stand alone. They can tell a story chronologically or simply invite the reader to explore dates of their interest.

Visual Index of a text story If used additionally to a written text, a timeline can give a quick summary of the text. Figure 2.12 shows an example from *The New York Times*, where the timeline visually defines the scope of the whole article. It enables a jump start into the topic and the reader knows immediately which timeframe will be covered. Furthermore it indicates roughly how many events there are to expect. Going through the story, the timeline will always indicate where the displayed paragraph is within the whole story. Also the distribution of events across that extent becomes visible immediately; for example, if things happened quickly after another or if there was a long time without any incidents. Spatial metaphors help giving the reader a chronological understanding.

Story told chronologically There are also many examples where the timeline *is* the article. To understand the story the reader is expected to navigate all events successively and is guided through the story like that. Again the visual timeline supports understanding where the current event is in the overall story, how much happened around the same time and so on. Especially news stories often consist of several consecutive events, so this

fragmentation can help to understand the complete story. When events evolve quickly, a timeline can support the reader staying up-to-date.

Comparison Often we want to compare things. People, countries, companies etc. Comparing them on a temporal basis can shed light on many things. Meeting points, possibilities of influence, who was first with having an idea, who met whom before the other one and so on. Figure 2.13 shows many different Middle Eastern countries that were all part of the Arab spring. The timeline displays protests, political moves, regime changes and international responses and gives a good impression of what happened simultaneously, when the big movements were and how the world reacted.

2.4.3 Missed opportunities

As pointed out, timelines seem to be helpful in understanding a news story, especially when it can be split into single events - still we don't see too many timelines. More often we see articles that list events in textual form explaining the temporal development of a story as seen in Figure 2.14 and 2.15. In this tabular form we don't see the distribution of events, we don't initially see the timeframe and often (if it's a long list) we first have to scroll to the bottom to see the scope of the story. In Section 3.2 we will go into more detail about possible reasons for those missed opportunities.

The screenshot shows a Guardian article titled "Edward Snowden and the NSA files - timeline". The article is dated Wednesday 27 August 2013 12:54 BST. It features a photo of Edward Snowden and a list of key events:

- 30 May** Edward Snowden, an employee of defence contractor Booz Allen Hamilton at the National Security Agency, arrives in Hong Kong from Hawaii. He carries four laptop computers that enable him to gain access to some of the US government's most highly-classified secrets.
- 1 June** Guardian journalists Glenn Greenwald and Ewen Macaskill and documentary maker Laura Poitras fly from New York to Hong Kong. They meet Snowden in a Kowloon hotel after he identifies himself with a Rubik's cube and begin a week of interviews with their source.
- 5 June** The Guardian publishes its first exclusive based on Snowden's leak, revealing a secret court order showing that the US government had forced the telecoms giant Verizon to hand over the phone records of millions of Americans.
- 6 June** A second story reveals the existence of the previously undisclosed programme Prism, which internal NSA documents claim gives the agency "direct access" to data held by Google, Facebook, Apple and other US internet giants. The tech companies deny that they have set up "back door access" to their systems for the US government.
- 7 June** Barack Obama defends the two programmes, saying they are overseen by the courts and Congress. Insisting that "the right balance" had been struck between security and privacy, he says: "You can't have 100% security, and also then have 100% privacy and zero inconvenience."

The Guardian reports that GCHQ has been able to see user communications data.

Figure 2.14: Non-visual timeline from *The Guardian* about the leaking process of NSA files, July 2013, <http://goo.gl/0cB9YB>

The screenshot shows a timeline article titled "TIMELINE Pat Quinn: A hockey life". The article is from The Canadian Press, posted on Nov 24, 2014 3:58 PM ET, and last updated on Nov 25, 2014 9:03 AM ET. It features a photo of Pat Quinn pointing at a whiteboard with hockey diagrams. The timeline lists key events in his life:

- Jan. 29, 1943**: John Brian Patrick Quinn is born in Hamilton.
- 1968**: Quinn makes his NHL debut with the Toronto Maple Leafs.
- 1969**: Quinn provokes a bench-clearing brawl when he delivers a blindside hit on star Boston Bruins defencemen Bobby Orr during the playoffs.
- 1970**: Quinn begins playing for the Vancouver Canucks.
- 1971**: Quinn moves to the now defunct Atlanta Flames.
- 1977**: A persistent ankle injury forces Quinn to retire as a player. He

Former NHL player, coach and executive Pat Quinn died Sunday night at the age of 71.

Here's a look at some key moments in his life:

Figure 2.15: Important events of a hockey player's life in textual form, November 2014, <http://goo.gl/5kVFNA>

3 Creating Timelines

After discussing why timelines can be useful in many cases, the following section will go into more detail about how timelines can be created, what challenges can appear along the way, and how we attempt to tackle those challenges with TimeLineCurator. Finally we will introduce our Timeline Authoring Model, that divides the authoring process into five tasks, which are: *browse, extract, format, show, and update.*

3.1 Common approaches

There are different ways to create timelines. Which one to take, mainly depends on the medium they aim for. As described in the prior section we can either have static timelines (primarily for print products) or we can make them interactive so that the reader can navigate through single events inside digital media. We do not claim that the following approaches are the only possible ones and they can certainly vary for different desks and people, still based on interviews and empirical value they are widely used.

3.1.1 Manual drawing

The most common way of creating static timelines is by using an illustration program like Adobe’s Illustrator, Corel Draw, or a free alternative like Inkscape. They are all based on vector graphics. It requires some journalistic work to define what events have to be on the timeline. The detailed knowledge about those events very likely comes from several source documents and articles. We can divide the process roughly into three phases as Figure 3.1 illustrates: First, documents that contain information about the topic have to be found and read. Second, information has to be extracted, formulated comprehensibly, and passed on to the third phase: translating the events into a cleaned-up graphic. The bigger the publisher the more likely it is that those different tasks are done by different people. The third step includes positioning dots and description texts of events manually one at a time. That means the more events a timeline has, the more complex it becomes. Especially when some events are close together and others far away, it’s getting tricky to fit them all in the predefined space. Because of the space limitation and because designers have to follow certain rules (for instance having a minimum font-size, having a minimum distance between elements and the like), titles and description texts may have to be rephrased or shortened, additional elements like images have to be handled with care - all that results in a rather time-consuming process.

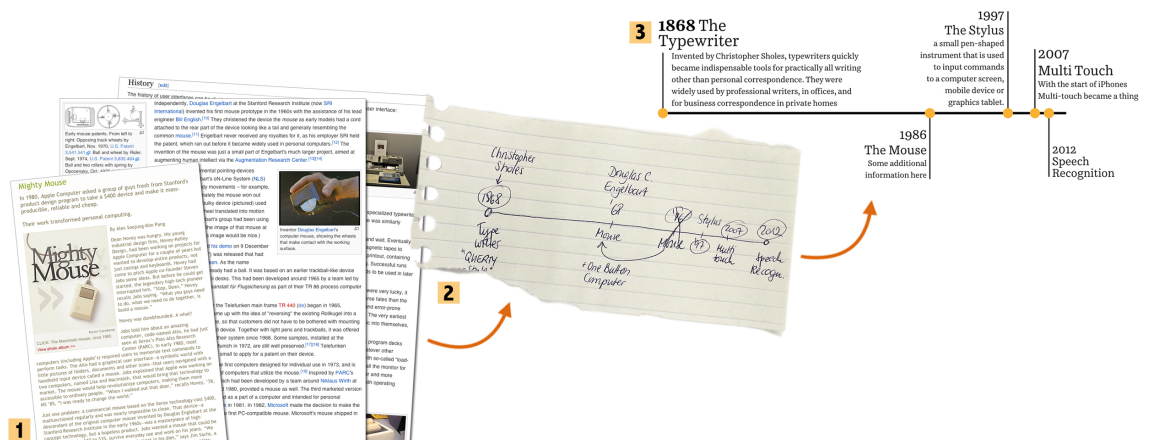


Figure 3.1: Three steps of creating a timeline manually: 1. Searching for information about events, 2. Scribbling information on paper, 3. Produce cleaned-up timeline in vector-based program.

A positive feature of this approach is that the author has a significant amount of creative license when performing this task. As a result, manual drawing can lead to intricate and engrossing timelines. Furthermore events were probably selected and formulated with great care and reduced to its essentials. Drawbacks of that approach are that the number of displayable events is limited to predefined space, thus the amount of text every event can have is limited. The initial setup and construction plus the positioning and adjusting of events takes a long time, adding or removing events later results in major restructuring work, since all events will probably have to be rearranged.

3.1.2 Structured creation

As mentioned before there are alternatives to the digital hand drawing process. As soon as we have the event data in a predefined structured form we can let the computer generate an interactive timeline. These approaches however produce timelines that cannot be customized without a basic knowledge of coding. We demonstrate the standard process with the example of the JavaScript framework TimelineJS which we already mentioned in Section 4.2. Figure 3.2 shows the three steps: The starting situation is the same as before, so the first step still is to find information about events. Second, instead of scribbling the information somewhere, events have to be translated into machine-readable data; when writing the events into the table several rules have to be followed. For instance the date has to be in one specific format (in this case the US format MM/DD/YYYY, which is rather counter-intuitive for everybody outside the US), titles should have a certain length, media can be added by inserting a valid URL. The third step is done automatically by a script. Advanced authors have the chance to change the style or add more JavaScript functions, but in general no coding is needed to get a basic timeline.

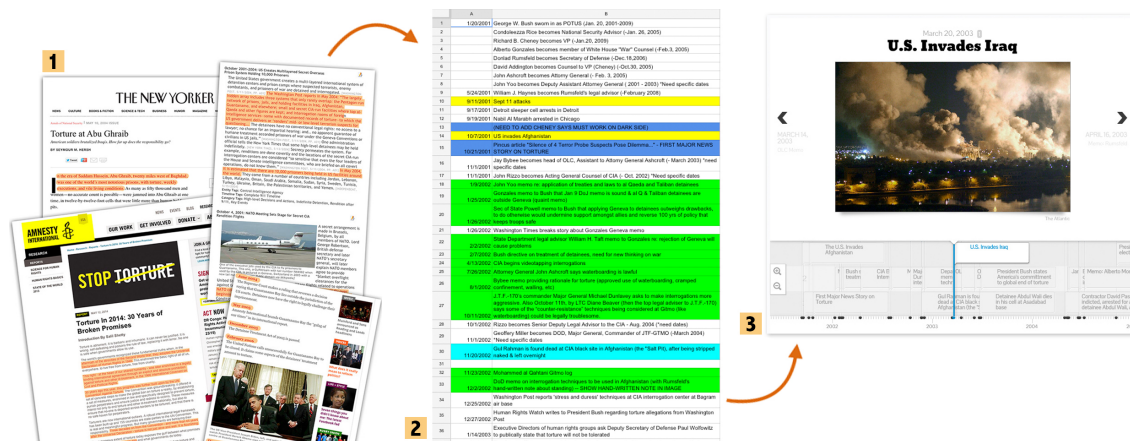


Figure 3.2: Three steps of generating a timeline with a structured data set: 1. Searching for information about events, 2. Writing the information into a table, 3. Automatically generating the visualization.

Drawbacks here are that the author doesn't get feedback if the format entered in the spreadsheet is correct (and will only find out when the visualization is not working in the end), neither does the author have visual feedback until the very last step of the creation process. Section 6) goes into more detail about issues users had when using that approach. Nonetheless advantages are that the author doesn't have to care about space limitations and can simply add all events that he thinks are important for the story and even enrich them with media. No shuffling around dots or fiddling with texts is needed and it will still result in an acceptable looking timeline. Especially for evolving news stories that approach is much more viable than manual drawing, because events can be added or updated at any time.

3.2 Challenges

Even though we pointed out many advantages of timelines, especially in the news, we don't see timelines as often. The idea of summarizing important events that concern a certain topic in chronological order is common, but it is more likely to find lists instead of visualization (like in figure 2.14 and 2.15). These don't give the reader the chance to quickly understand the amount and distribution of events, and the timeframe it is covering. We found several reasons and challenges that could be crucial points for journalists not making usage of visual timelines more often.

Time In newsrooms especially, but in many other surroundings as well, time is scarce. Tight deadlines prevent eagerness to experiment. Also it is not very common that writing journalists think a lot about how they could use graphical elements or visualizations for their stories. And those who do usually don't see it as a primary task. That results in tackling that concern pretty late in the publishing process, which leads to not having too much time for it. In big publishing companies it is also not common that journalists make the graphics themselves. Usually there is a graphical department that creates all kinds of graphical elements and visualizations for the journalists - often based on their scribbles or notes. However the very short time that is calculated for that process, leads to sometimes deciding against using graphical elements in the end.

Tools As mentioned before, tools attempting to facilitate a journalist's life are available, but not used to their full potential. That springs from either not knowing about them at all or not knowing enough about them to be able to comfortably work with those tools. Even if they were keen on getting to know new tools and possibilities it comes back to the first challenge - that there is just not enough time during the daily production to spend much time on learning to use them. In the special case of timelines it may also not be clear in the first place if a timeline is suitable for a certain topic. So unless the author is convinced of the suitability, he will probably not bother putting much work into finding a suitable tool.

Integration News sites work with huge and complex Content Management Systems (CMS). For traditional print authors it often is already a big commitment to occupy themselves with learning how to use those systems. CMS offer different layouts for different formats of articles - news article, report, commentary, image gallery and so on. However they are not made for people going creative. Integrating interactive graphics is often solved by including an iFrames that calls an external source. The integration however is often not very sophisticated; for example, the style of the external element does not match the style of the website (different colors, fonts, shapes) or the width of the article is different to the width of the graphic, which leads to either cutting off the graphic or having scrollbars, which essentially reduce readability. Furthermore today's websites should be accessible from any kinds of devices, whether it is a small smartphone or a huge 84-inch screen. High traffic news sites (like *The New York Times* or *The Guardian*) are well equipped for all occasions, but smaller ones still seem to struggle with formats that are different to a standard desktop screen size. So if we want to integrate a timeline into a news article, first an integration feature has to be available within the CMS, second the style has to adjust to the general styleguide of the website, and third the timeline has to be responsive and adjust to different screen sizes - at least as much as the website itself - to be enjoyable on as many devices as possible.

3.3 New approach

After this extensive introduction into timelines, their usage, the building process and the challenges we suggest an alternative that facilitates the creation of interactive timelines to make them more accessible for a less technologically-inclined audience.

Idea We observed that the whole process starts with finding information about the time and the details of an event from source documents. We imagine that this process could already be facilitated, if there was a way to initially indicate whether a document contains temporal information at all. This could be done by using natural language processing to automatically detect temporal information within unstructured text. Available libraries and methods enable this functionality, and are able to at least discover most temporal expressions. We know that the results are not perfect, but they are still good enough to provide scaffolding for a visual timeline. Through curation and selection the timeline can then be adjusted according to the author's needs in an easy and quick way; that means events can be edited, deleted, and moved around, as well as new events that are not part of the source documents but known by the author can be added. We think the visual curation process can not only speed up the process of creating a timeline, but also make it much more enjoyable.

Processing Pipeline Figure 3.3 shows the three major steps of our idea. First, unstructured text is entered as input - that can be a news article, a Wikipedia page, a biography, a historic text and the like. Inside the gearbox temporal information will be found and extracted, and visualized on a timeline in step two. The visualization's environment allows for curation. The next gearbox takes the resulting curated data set and transforms it into a non-editable and presentable timeline for publishing in step three.



Figure 3.3: An abstract representation of TimeLineCurator's pipeline: (i) unstructured text input; (ii) an authoring environment; (iii) curated timeline output.

3.4 Timeline Authoring Model

Based on the different approaches described above we introduce five timeline authoring tasks and compare how these tasks are accomplished using the three different approaches *manual drawing*, *structured creation* and the usage of TimeLineCurator. Figure 3.4 summarizes the differences according to the descriptions made in Section 3.1 and to according to previously introduced ideas and upcoming descriptions of TimeLineCurator.

The timeline generation process begins with **browsing** source documents, where the author looks for event information. *Browsing* is defined as a form of search in which the locations of potential search targets are known, but the identity of the search targets may not be known a priori [5]. During this period, the author might identify and **extract** events by highlighting or annotating relevant passages in documents, adding events to a list or sketching a timeline on paper. To transfer these events to a digital medium, the author must decide how to **format** the events, and determine how to **show** or encode them. Finally, in some instances, an author **updates** the timeline: events may be added, edited, or deleted to reflect new information, such as in the case of an evolving news story.













	Browse	Extract	Format	Show	Update
Manual Drawing	 high	 high	—	 high	 high
Structured Creation	 high	 high	 high	 low	 low
TimeLineCurator	 low	—	—	 low	 low

Figure 3.4: Comparing the human time and effort required to perform the five tasks encompassed by our Timeline Authoring Model with common approaches and with TimeLineCurator.

Both common approaches are connected with high time exposure for the *browsing* and *extracting* process as documents first have to be found and then read. The *formatting* is obsolete for the manual drawing, but has to be done very carefully for the structured creation, since the data format has to match specific guidelines. *Showing* as well as *updating* in turn can be achieved quickly for the structured creation but takes high time exposure when created manually.

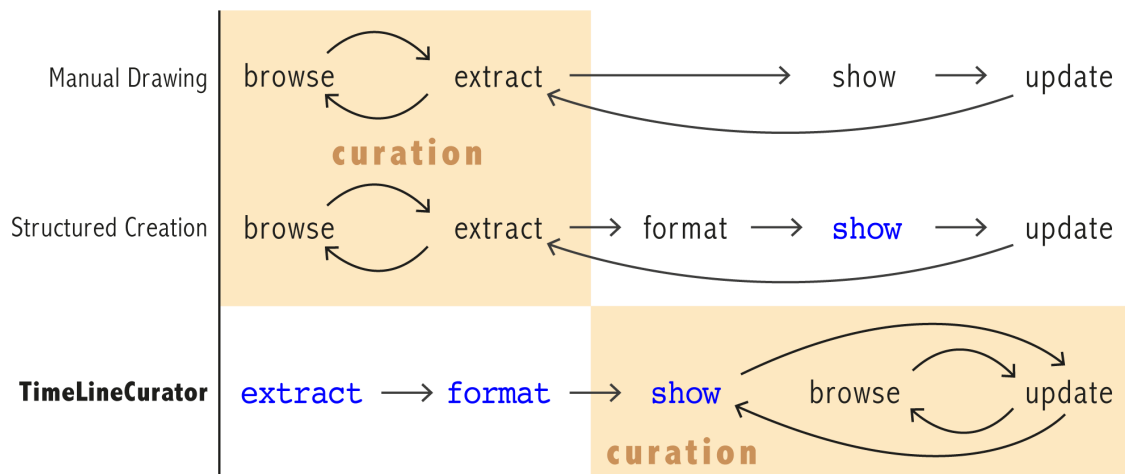


Figure 3.5: Comparing the sequence of timeline authoring tasks: timeline curation (indicated by the orange shaded areas) occurs later with TimeLineCurator. Tasks in **blue fixed-width** font are automated; all other tasks are performed by the author.

Our new tool, TimeLineCurator, was developed to overcome these difficulties. With manual drawing and structured creation approaches, timeline curation was accomplished by iterating between the *browse* and *extract* tasks; with TimeLineCurator, timeline curation is a visual process, swapping the order of the *browse* and *show* tasks while automating the *extract* and *format* tasks, as indicated in Figure 3.5. TimeLineCurator also explicitly supports the *browsing* of events from multiple documents simultaneously, allowing, for instance, the author to compare multiple sources discussing the same subject or comparing subjects that do not obviously relate but might have influenced one another. Finally, updating a timeline with TimeLineCurator is easy, and does not require editing the source documents.

4 Related Work

TimeLineCurator touches several fields, therefore we split the discussion of relevant previous work into four parts. Figure 4.1 illustrates the three states our data undergoes and which parts each chapter covers. First we look at general visualization authoring environments, for general analytical as well as for journalistic purposes (**red**). Then we show tools that generate timelines in several domains that use structured data as their input (**yellow**). To give an overview of how structured temporal data could be generated from unstructured text, we look at work that has been done for *Entity Extraction* as used in natural language processing (**green**). And finally there are tools already that use NLP to create visualizations and we will have a look at those as well (**blue**).

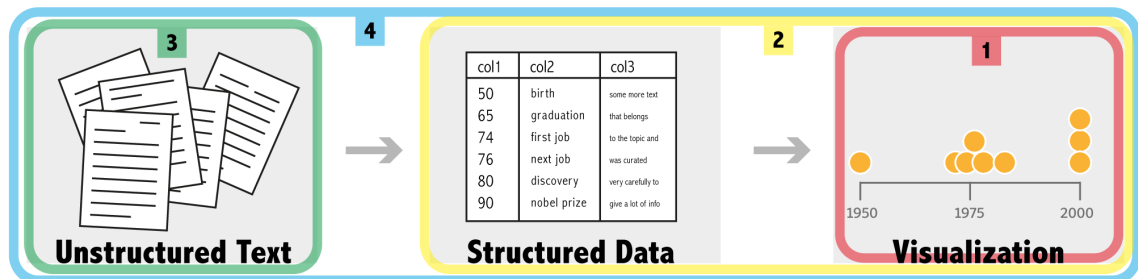


Figure 4.1: Illustrating the four parts we consider in our related work. From unstructured text processing, over structured data sets to visualization environments and authoring tools.

4.1 Visualization Authoring Tools

Many tools are available that help creating all different kinds of visualizations. There is almost one for every level of expertise. More options for customization however still require more complex tools and more time for the author to familiarize with it. First we show a few tools that can create a set of standard visualizations very easily and could be used by anybody. Then we show those which require a higher level of technical expertise and finally a few that were created especially for journalistic use.

General purpose tools for visualization presentation Popular and accessible tools such as Tableau (<http://tableau.com>) and ManyEyes [48] provide the means to generate, share, and publish visualizations without having to write any code. However, these tools expect structured data; it is difficult to generate visualizations from unstructured text data without wrangling the data into a structured form. In addition, these tools do not explicitly support the generation of visual event timelines. For example, ManyEyes offers a set of general-purpose visualizations and there is no visualization for event-based data within its repertory. Although Tableau is sufficiently customizable that the visual appearance of a timeline can be achieved with elaborate data transformations, this task is clearly not one of its primary design targets.

Custom visualization authoring environments Visual authoring tools such as Lyra [39] and iVisDesigner [35] are more expressive, allowing the author to compose visualizations with multiple layers and annotations. It is thus feasible to produce a custom visual timeline, once again assuming that the event data is already in a structured form. Since environments like Lyra and iVisDesigner provide more options

for customization and typically require more time to learn, they are less suitable for fast and easy authoring than a specialized tool, such as those that are specific to timeline authoring.

Authoring tools for journalists narrative visualization authoring environments such as Ellipsis [38] and VisJockey [19] specifically target journalists. With these tools, journalists can compose narrative sequences of common visualizations depicting structured quantitative data; visual event timelines are not explicitly supported. Narratives authored with VisJockey [19] further allow readers to trigger visualization transitions with inline links in an accompanying text article, similar to the linking between *The New York Times*' interactive timelines and corresponding sections of their accompanying articles. TimeLineCurator also relies on a linking between visualization elements and corresponding sections of a text document, but these links are established via natural language processing, whereas with VisJockey, these links are established manually by the author.

4.2 Timeline Visualizations from Structured Event Data

Now focusing only on creating timelines, there are examples from different domains, where they can automatically be created if an accordingly structured data set is available. Those timelines either have the purpose of analyzing or of summarizing. And then there are already several environments that enable a more or less comfortable creation of an event timeline.

Tools for timeline analysis Though we focus primarily on timelines as a presentation tool, timeline visualizations are also often used for data analysis. TimeSlice [52] is a domain-agnostic analysis tool that affords the faceted browsing of timelines containing many events; these timelines are generated from structured event data. In the medical domain, LifeLines [32] and its descendants are also used for analysis, wherein an analyst can summarize and compare patient treatment timelines comprised of event types specific to the treatment context; these events are recorded via manual data entry by medical staff.

Law enforcement tools such as Criminal Activities Network [9] are used for data analysis such as identifying crime patterns and discovering criminal associations, and are once again suitable only for structured domain-specific data. Social media analysts also use timelines for detecting events, trends, and anomalies, relying on structured social media data [7]. TimeLineCurator does not require structured event data and is portable across application domains.

News timelines In an ephemeral online news environment, timelines are a popular way to convey an evolving story or to provide context. For example, Google News Timeline (<http://news.google.com>) automatically aggregates news stories from several thousand news sources and organizes them chronologically, while Evolutionary Timeline Summarization [50] generates timelines based on a user query and identifies the “relevance, coverage, coherence, and diversity” of that query inside many time-stamped articles. However, both of these approaches return lists of events rather than visual timelines. Moreover, they treat an entire document as a single entity characterized by the document creation time; finer-grained temporal information from within the document is ignored.

Timeline authoring tools Many simple and accessible timeline authoring tools exist. Examples include TimeRime (<http://timerime.com>), Dipity (<http://dipity.com>), Tiki-Toki (<http://tiki-toki.com>), and Timeglider (<http://timeglider.com>). Some of these tools allow an author to add single events to an initially empty timeline one at a time, while others provide the ability to connect to RSS, Twitter, or other services that provide structured time-stamped data. Some of these tools are easy to use, but not at all customizable.

The customizable tools most relevant to our current work are SIMILE’s Timeline (<http://simile-widgets.org/timeline>), ProPublica’s TimelineSetter (<http://propublica.github.io/timeline-setter>), WNYC’s Vertical Timeline (<http://github.com/jkeefe/Timeline>), and TimelineJS [61] from the Northwestern University Knight Lab. These tools require structured event data as input; they generate timelines that can be embedded in websites. Advanced users can also make changes to the underlying code and adjust it to suit their needs. However, the author must first assemble and format a spreadsheet, JSON data set, or a correctly-formatted

CSV file containing event data. TimelineJS is perhaps the most widely-used timeline authoring tool in newsrooms today. The timeline creation process is straightforward: beginning with a Google Spreadsheet template, an author can fill in this spreadsheet with events, each of which requires a date or date span, a title, a description of the event, and, optionally, a link to an image, video, or other form of embeddable media. Publishing the spreadsheet generates a visual timeline automatically. We compare the experience of assembling and generating timelines using TimelineJS to that of TimeLineCurator in Section 6.1.

4.3 Extracting Time Expressions from Unstructured Text

The principals of how natural language processing (NLP) works was already discussed in Section 2.1.2. The technique we need for extracting temporal information from unstructured text is called *Entity Extraction*. It identifies for instance names, locations, organizations, or dates inside unstructured text. To annotate dates in a standardized form, the TimeML specification language for temporal information extraction [34] was defined in 2003. It defines how to annotate events and temporal expressions inside unstructured text. It became the international standard in 2009 (ISO-TimeML) and is used within almost all current approaches.

Syntax-based recognition Environments such as Tango [46] and TARSQI (Temporal Awareness and Reasoning Systems for Question Interpretation) [47] offer environments that automatically add TimeML markup to news articles. Temporal entity extraction is typically accomplished with hand-engineered deterministic rules that use regular expressions and pattern interpretation to detect signal words referring to anything temporal. Further improvements to these *recognition* approaches enable *normalization* of the recognized temporal expressions with respect to a Document Creation Time (DCT). For instance, the value of *yesterday* can be resolved to one day before the DCT. Examples include TempEx Tagger [25], SUTime [8], HeidelTime [42], and TERNIP [62]. TimeLineCurator uses the Python-based TERNIP system in its natural language processing pipeline. TERNIP uses the TARSQI extraction engine [47] for recognition; TERNIP also normalizes temporal expressions using a rule engine.

Context-dependent semantics Approaches that consider only the *syntax* of entities ignores the surrounding context and can lead to misinterpretation or ambiguities. Newer approaches that incorporate machine learning use context-dependent *semantic* parsing for entity extraction; examples include learning contextual rules from question-answer pairs [18] or the use of various forms of weak supervision [2]. In contrast to these general-purpose systems, UWTime [21] is the first context-dependent model for semantic parsing that handles the special case of temporal expressions, where the additional step of normalization is required. Using the combination of hand-engineered and trained rules, it considers the tense of a governing verb to determine if the temporal expression refers to the future or the past, and it determines if a four-digit number refers to a year depending on the context. Incorporating the Java-based UWTime system into TimeLineCurator as an alternative to TERNIP would be interesting future work.

4.4 Visualizations from Unstructured Text

In our project we combine visual timeline authoring and natural language processing. There were several projects taking that road already, even though exclusively focusing on temporal expression to create visualizations has not been done yet, as far as we know.

Topic discovery and analysis Thematic analysis of many text documents is a popular area of research. Tools such as Serendip [1] leverage natural language processing to permit thematic analysis for documents at different scales, from individual passages to documents to entire corpora. Meanwhile, a number of tools [10, 15, 22, 23] (those only being a selection) extract topics and keywords while also considering each document’s creation time, allowing the analyst to observe topic changes over time. These tools do not extract temporal information in the unstructured text of documents; rather, they use bag-of-words models or more complex algorithms to determine the importance of words, word combinations, or topics. Furthermore, these tools are intended for data analysis rather than authoring or presentation.

Entity extraction and visual analytics Visual analytics systems such as Jigsaw [13, 41] integrate entity extraction with visualization to show detected entities such as dates from unstructured text documents in several ways. However, the use of Jigsaw entails a high learning curve [12, 16], requires desktop installation, and is again intended for data analysis rather than presentation.

Visualizing Wikipedia articles Date entity extraction has also been applied to the generation of timeline and topic visualizations based on Wikipedia articles. For example, LensingWikipedia [45] attempts to visualize human history through Wikipedia’s annual event summary pages over the last 2000 years. It extracts temporal and spatial information to find out “who did what to whom, when, and where” [45]. It is a discovery environment restricted to those specific pages; users cannot insert their own data and there is no support for authoring. Another example is WikiChanges (<http://sergionunes.com/p/wikichanges>).

Date entity extraction is more accessible in TimeLineCurator than in previous work, since our tool is browser-based, is intended for fast timeline authoring rather than data analysis, and can ingest any unstructured text.

5 TimeLineCurator

Before we started coding the prototype, we analyzed available tools, and decided to use TimelineJS [61] as the state-of-the-art tool for comparison. In the following we point out limitations of TimelineJS, and set up a list of requirements and goals for TimeLineCurator based on that analysis. Afterwards we show selected steps from the iterative design process, and finally present the architecture of TimeLineCurator.

5.1 Identifying TimeLineJS Limitations

Even though TimelineJS [61] seems to be the most popular tool for creating interactive timelines, we learned about several limitations and drawbacks when talking to current users of TimelineJS - Section 6.5.1 will go into more detail about the user feedback. The authoring process with TimelineJS falls into *structured creation* as defined in Section 3.1.2. That approach requires the author to spend a lot of time and effort on extracting and formatting structured event data. We use that approach as primary comparison, and describe the different experiences during the authoring process in Section 6.1.

We identified several drawbacks to how TimelineJS presents a timeline to the reader (as shown in Figure 5.9f), which informed the design of presentation-ready timelines exported from TimeLineCurator, described in Section 5.5.6. A TimelineJS widget presents a zoomable and scrollable interactive timeline that invites the reader to progress through the timeline with linear navigation from one event to another, beginning with the first event in the timeline. TimelineJS does not provide an initial overview of the temporal distribution of events: on opening, the horizontal timeline view is centered on a specific date and only a small region is visible. By default this first date corresponds to the earliest event in the timeline; while the user can explicitly navigate by zooming out, it is not possible to simply set the start view to show the entire timeline. Moreover, clutter and occlusion is a significant issue: glyphs representing individual events are displayed along a narrow axis spanning the bottom of the timeline, and the event labels placed above this axis overlap in regions where multiple events occur.

5.2 Requirements & Goals

Based on the analysis of current timeline authoring tools and approaches we define several requirements and goals, that a system should fulfill to offer a valuable alternative to the existing approaches:

Automate event extraction As table 3.4 shows, the *browsing*, *extracting* and *formatting* of event data takes up much time in common approaches. Thus a new approach should strive to reduce, if not eliminate those tasks. As soon as an event description within unstructured text contains information about its time, the NLP should be able to detect and extract it. That process saves the time that is usually needed to go through a text manually; it also eliminates the uncertainty about whether a document contains temporal information or not.

Accessible As seen in Section 4.3, recent advances in NLP enable a decent detection and normalization of temporal references from unstructured text [62]. However those packages and tools are not very accessible and require installation and often the ability to handle code. Our timeline authoring tool aims to be accessible for everybody. It should not require any installation - therefore it has to be browser-based. It should be easy to understand - therefore match basic human-computer interaction principles. Considering the basic principles should enable an intuitive handling also to authors without a highly developed technical skill set. Any need for coding should be avoided.

Flexibility in input and formatting Since the formatting of the underlying data set for current tools is one of the major issues for not using them, we want to create an environment where the author doesn't have to care about formatting and can just type in free form text. Therefore the input, as well as the textareas in the event editing area, have to accept any form of unstructured text. Even if the user accidentally enters something incomprehensible (for example in the date input field, only digits make sense), he should get immediate feedback with an advice to correct it.

Visual feedback at all times Often it can be disadvantageous when no visual feedback is available during the creation process, as Victor points out in a Stanford HCI seminar about "Drawing Dynamic Visualizations" [66]. Our timeline authoring system should thus provide visual feedback as early and immediate as possible. After inserting a text, temporal information is automatically *extracted*, *formatted*, and visualized immediately (*show*). Also during further *browsing* and *updating* event data will constantly be updated visually, as indicated in Figure 3.5. That immediate visual feedback is missing when programming a timeline from scratch, or when using existing timeline authoring tools, such as TimelineJS and others mentioned in Section 4.2. Without that intermediate visual support, it sometimes might be difficult to tell whether creating a visual timeline is worth the effort.

Allow for speculative browsing Finally, the ideal tool not only accelerates the authoring process, but also helps discovering suitable documents. Documents that don't contain interesting temporal information about events could be ruled out immediately as unsuitable for a timeline.

5.3 Design Process

TimeLineCurator passed several design cycles. To give an impression of the iterative process we outline important steps on the way to its current state in the following.

5.3.1 Necessary elements

To fulfill above mentioned requirements we defined several elements, that an environment needs to have. Those elements have to be easy to spot and immediately communicate their functionality:

- (a) **Upload:** A space where you can insert or upload text (start with having an input field to copy in unstructured text)
- (b) **Document View:** The original text with the temporal information highlighted (if more than one document was inserted, ability to switch between documents needed)
- (c) **List View:** A list with all dates that were found inside the document(s), sortable by several criteria
- (d) **Timeline View:** The centrepiece where events are displayed on the timeline. Events are displayed as glyphs (distinguishing between date/duration, category, state of curation)
- (e) **Editing Area:** Area where a selected event can be modified (changing the date, title, description text)
- (f) **Add single date:** The possibility to add a single new date that is not connected to a document
- (g) **Export:** The possibility to export the timeline into a “cleaned-up” timeline

Additionally all views have to be connected with linked highlighting. Events have to be selectable in all views (list, document, timeline) and simultaneously update the other views (including the editing area). Figure 5.1 shows the very first scribble with annotations for mentioned elements.

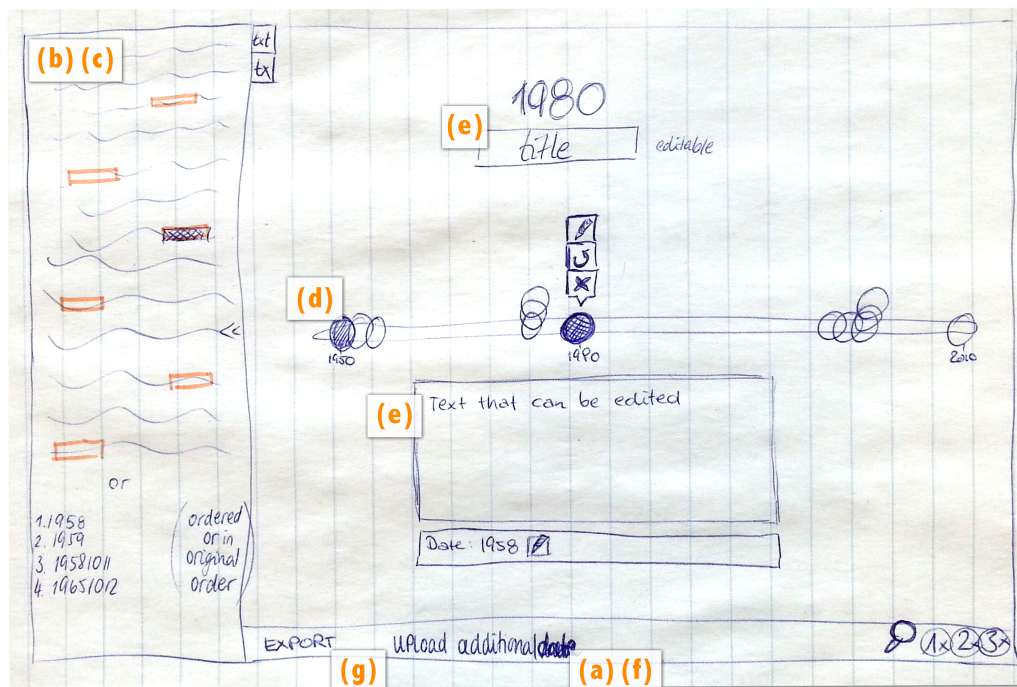


Figure 5.1: First scribble of a timeline generating tool on paper - with marks for list of needed elements

5.3.2 Design Iterations

On the basis of screenshots of different selected states of TimeLineCurator, we will explain the development during the building process.

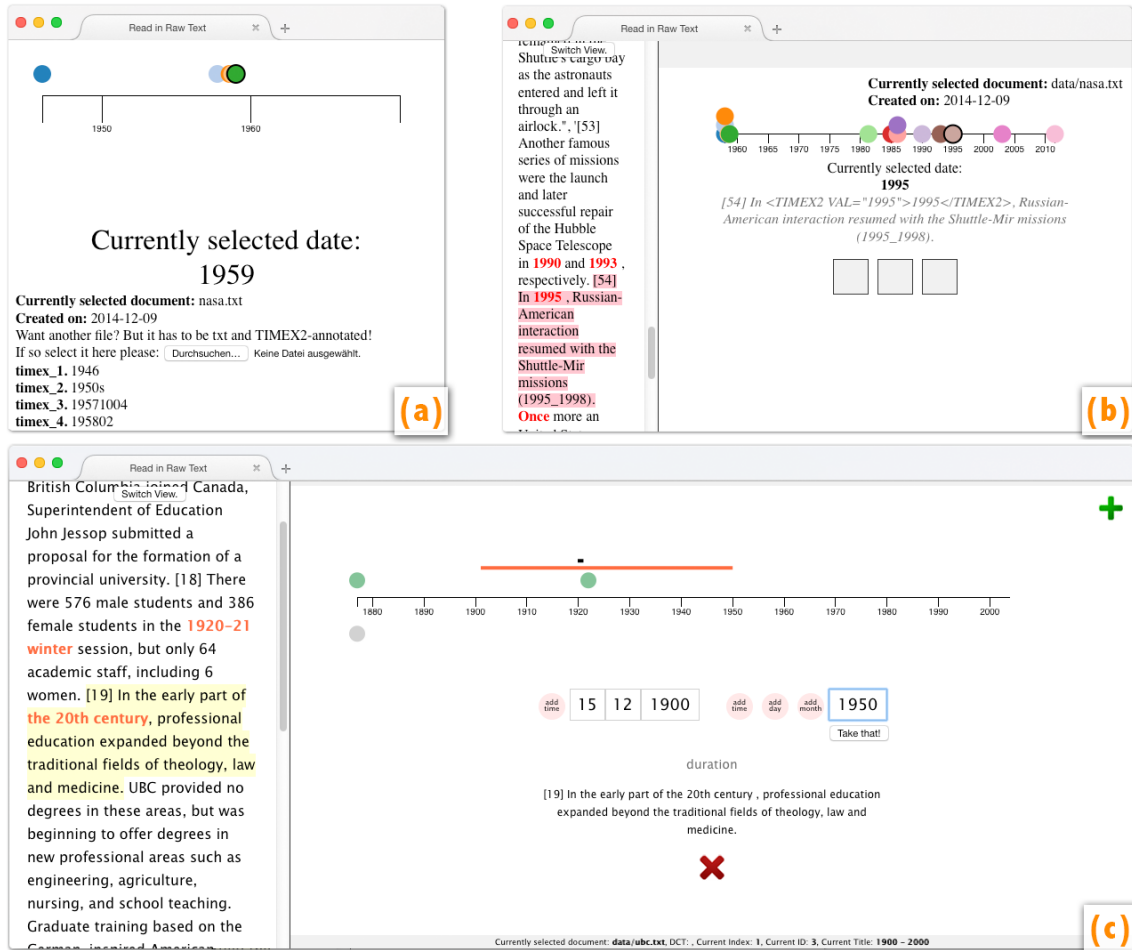


Figure 5.2: First steps with pre-processed files as input. a) Placing dates on timeline, b) Document view added, c) Time periods and editing functionality for events added

First tentative steps to implement it digitally are shown in Figure 5.2. Primary goal was to manage the alignment of dates on the timeline. We started with two separate processes for the back-end and the front-end side. Text processing was done by running a Python script locally, producing an output file that was then called by the JavaScript code. The generated file was provided with *TIMEX*-tags which had the date as their value (for instance, *In <TIMEX2 VAL="1958">1958 </TIMEX2> Michael Jackson was born.*). Client-sided the values were extracted and translated into a Unix timestamp. With the maximum and minimum values the scale of the timeline was calculated and single events positioned accordingly. When selecting one date an automatic scrolling inside the document view was triggered, to highlight the sentence they originate from. Therefore the original text had to be available inside a separate scrollable view. The details of the selected date had to be shown in yet another view. With AngularJS the selected date was made editable, and the changes were immediately applied within the data set. The state shown in Figure 5.2 (c) introduced the functionality to change the granularity of the date of an event; for instance, 1958 could be extended with a month and a day to 29.08.1958 - even the time of day could be added. We started off only displaying dates (as circles), time periods were added in the third step as well (as lines).

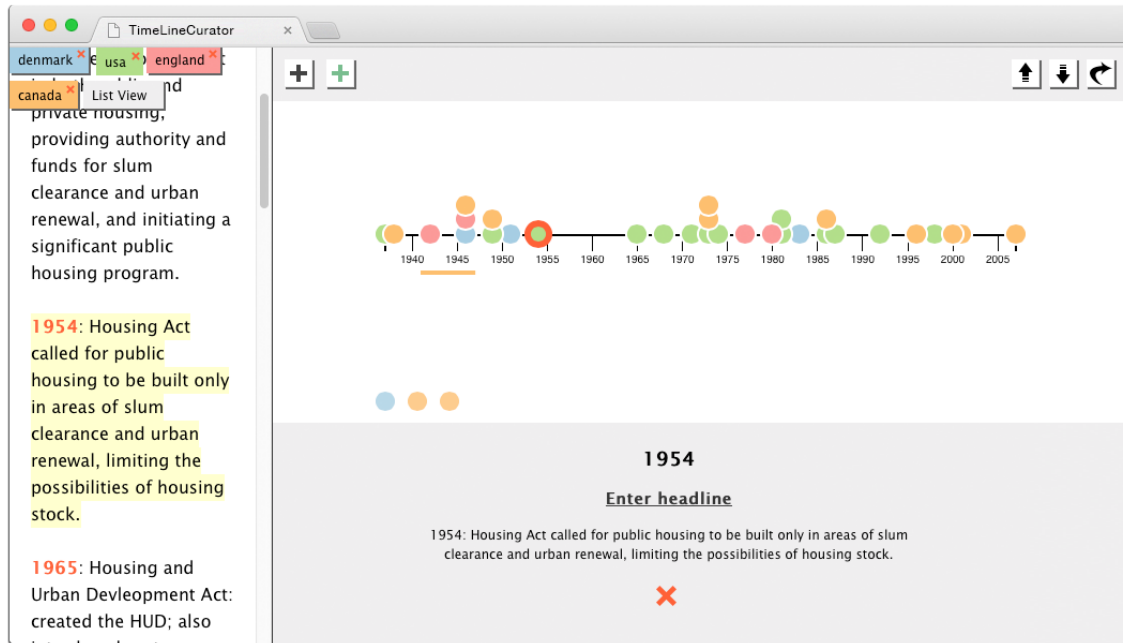


Figure 5.3: First cleaned-up version of TLC, where pre-processed documents could be selected as input. Events were editable, the current state could be saved and downloaded as a “.tl”-file.

A first tidied up version is displaying in Figure 5.3. On the left side the author could switch between the documents and the list view. A TLC-specific file format with the ending “.tl” was introduced, to enable saving the current state of curation. The file could then be uploaded in the next session to reproduce the state from before (Additionally the current state was stored inside the browser’s local storage automatically to prevent accidental data loss). The controls for Save/Export/Upload, as well as to add a new document, were gathered in a bar above the timeline. The content of one date (initially the sentence surrounding the temporal expression in the original document) could be edited by clicking on it - the text would change into a textarea. The headline (initially set to “Enter headline”) was left to the author to formulate. Vague dates that could not be placed on the timeline were gathered on the bottom of the timeline view.

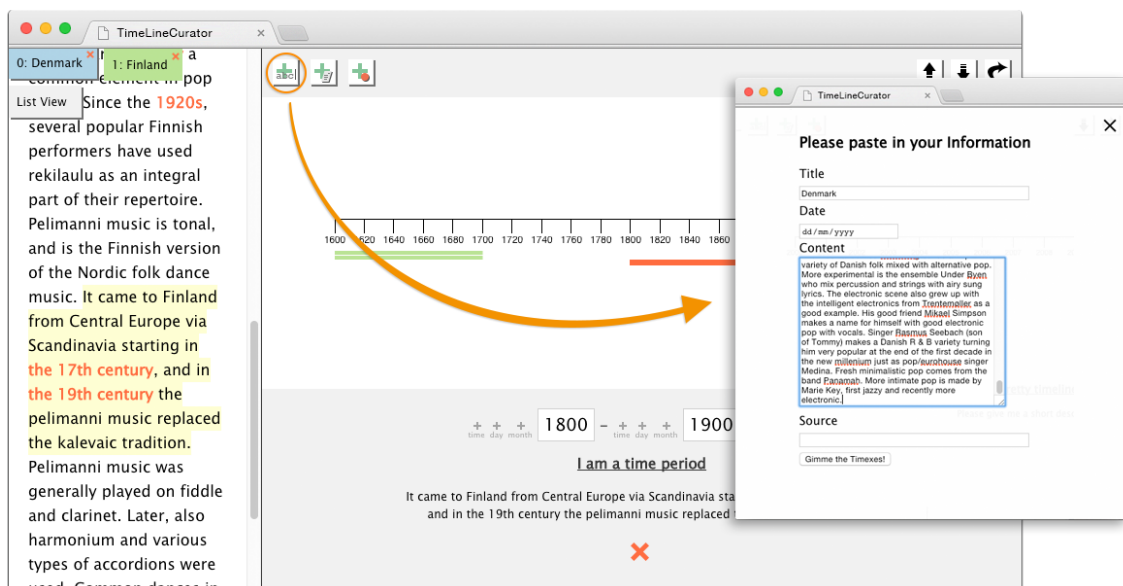


Figure 5.4: Design process step 2: Including possibility of granularity change of date

After the visualization and curation part was running, the next step was to include the event extraction part into the browser environment. With microframework Flask for Python it is possible to transfer data between JavaScript and Python code and make it run inside the browser. This enabled to upload and process unstructured text. Figure 5.4 shows the dialog box that opened up and let the author copy in any unstructured text. The creation date of that text (if it was not the current date) had to be specified, to be taken into account when calculating the values of the dates. With that version we started asking people to have a look at the tool and thus got real user feedback, which again led to several changes.

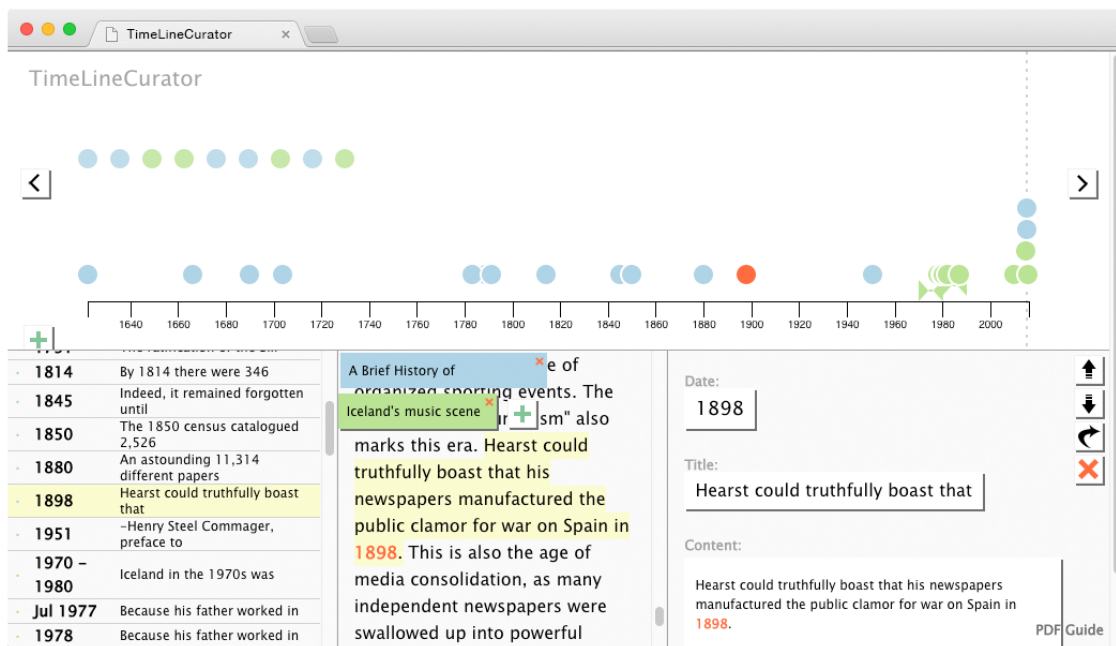


Figure 5.5: Rearrangement of the whole layout after evaluating first user feedback.

The next step, shown in Figure 5.5, introduced a completely new layout, where document and list view were separated into two views. The timeline itself got the full width of the browser window. Time spans were now part of the timeline and not aligned underneath it anymore. Their glyph changed into having a bounding triangle on either side. That way they seemed equally important than the circles, since they were taking up roughly the same amount of space. Instead of waiting for the author to provide a headline for a date, we decided taking the first five words of the sentence to at least give a suggestion that can then easily be changed.

Figure 5.6 shows further changes based on user feedback as well as on own experience. The color of each individual date could now be changed independent from its origin to connect it to a track. The list view could be sorted according to several criteria. The document selection inside the document view was solved more elegantly by showing a drop-down menu when clicked. Both list and document view were enriched with a footer bar, that summarizes the total number of events inside the current view. Keyboard shortcuts were implemented (Arrow-keys to select previous/next date, Delete and Return key to delete a date, Shift for selecting several dates). Dates could now be augmented with media. The color of the date's glyph became more saturated, as soon as it was edited - that way it was easier to spot which dates were already curated and which stayed

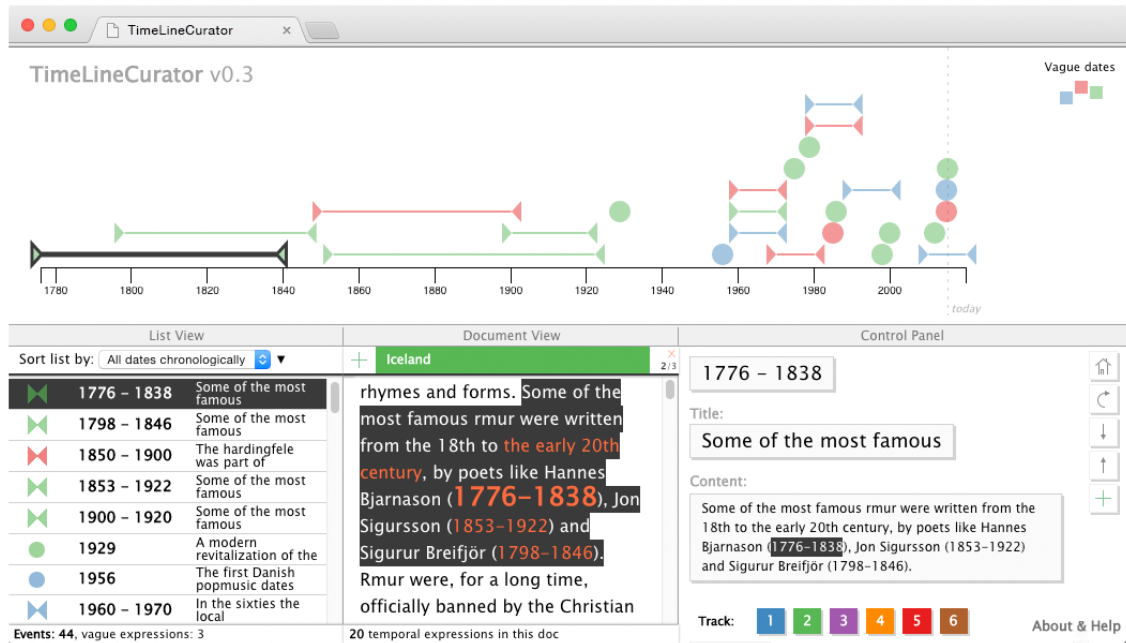


Figure 5.6: Current state of TimeLineCurator in its version 0.3

untouched. The vertical distribution of the glyphs posed a problem up to this point, because they were only stacked vertically, when they had exactly the same date. This led to difficulties when selecting a date, some dates were not visible at all, when they were too close together. The new layout started stacking dates to the top as soon as they would overlap on the horizontal line. That loosened the whole view but introduced the new problem, that more vertical space was necessary. So vertical scrolling had to be added as soon as the dates pulled out of the viewport.

As last step we decided to add an alternative to the TimelineJS export. One that looked more similar to the editing environment itself and also offers access to the original data. We chose to provide a non-editable version of TLC, where the Control Panel became the read-only display, showing event details. Section 5.5.6 explains the different export options in more detail.

5.3.3 Implementation

The back end of the system that provides the data handling (*extract* and *format*) is implemented in Python. The TERNIP framework [62] together with the NLTK toolkit (<http://www.nltk.org>) first split the document into sentences and then words (“tokenizing”). “Part-of-speech tagging” is applied and a list of tokens handed to the recognizer, which identifies the temporal expressions. Those are sent to the normaliser which tries to determine a concrete date or duration. The result is written as attribute (“VAL”) into a TIMEX tag (based on TimeML [34]). The whole text is recomposed, enriched with the TIMEX tags.

The micro web application framework Flask (<http://flask.pocoo.org>) enables calling variables from Python script and using them inside our front-end environment, which supports the *show*, *curate*, *update*, and *present* tasks. It is implemented in AngularJS (<http://angularjs.org>). AngularJS offers comfortable functionality for dynamic updating and editing of elements inside the Document Object Model. The visualization

is implemented with D3.js [4], a library that binds data to SVG elements and enables comfortable updating and manipulation of the data. The whole system is hosted on the Heroku cloud application platform (<http://heroku.com>), which runs the Python code on the server side.

If the author exports the timeline into TLC's own presentation mode, we store the curated timeline data on Amazon's Simple Storage Service (<http://aws.amazon.com/s3>). Via Python script data is sent to the service, an Ajax request calls it from the TLC presentation view. Storing the data in the cloud, enables us to generate a unique URL, that can be shared and accessed from everywhere, not only the author's working environment.

5.4 Architecture

The processing pipeline of TimeLineCurator can be explained in more detail based on walking through the steps of the curation process. Figure 5.7 shows all the stages.

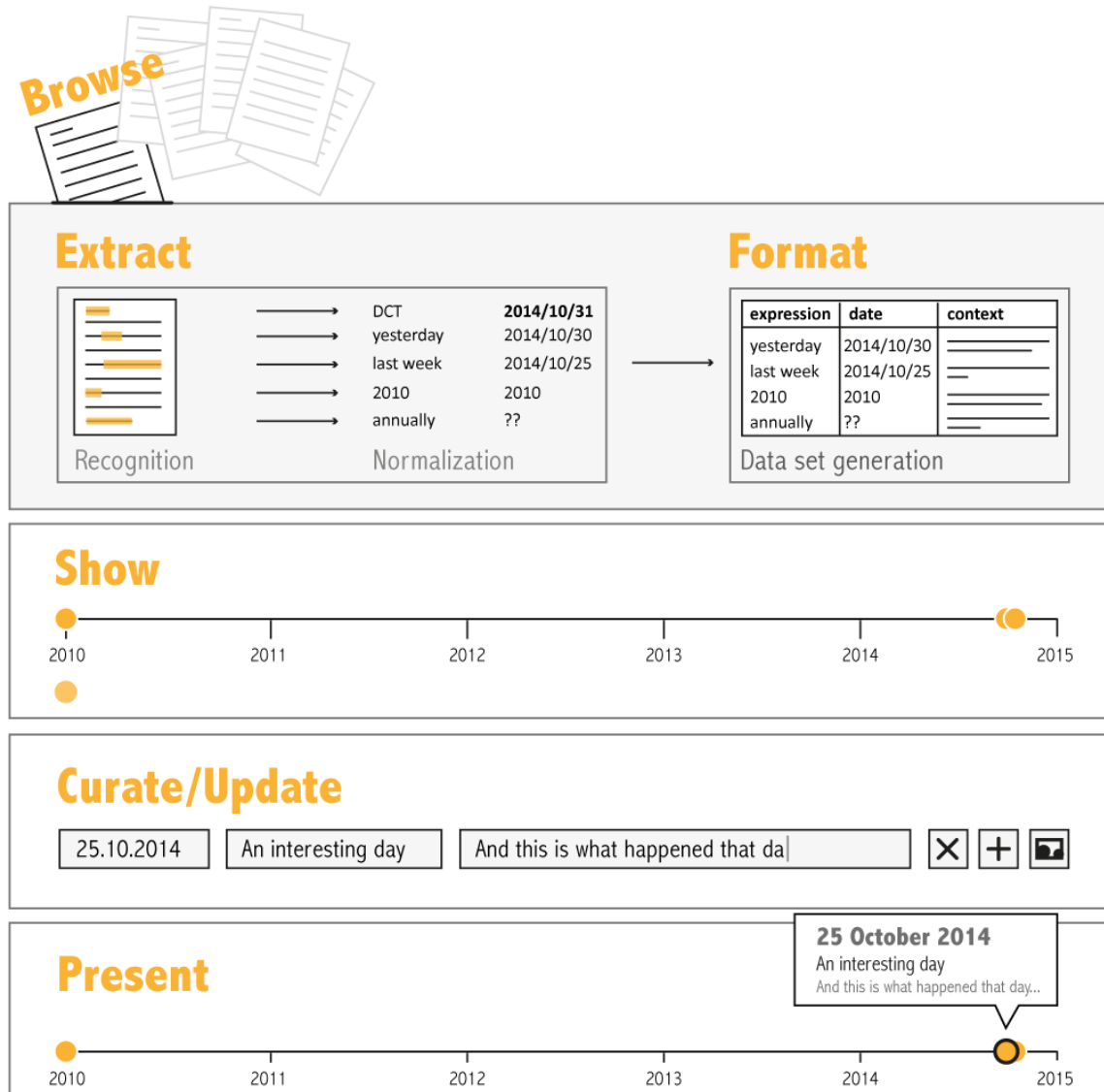


Figure 5.7: Processing pipeline for TimeLineCurator.

An author begins with an empty timeline, and can populate the timeline by uploading unstructured document text. TimeLineCurator **extracts** events from this text using natural language processing techniques; it first *recognizes* absolute temporal references such as “October 30, 2014” or “2010” using the Python library TERNIP [62], which is based on a large set of regular expressions and rules. In addition to single dates, durations are also extracted, such as the reference “from 2 Sept 2014 to 31 Mar 2015”. TERNIP also *normalizes* all relative temporal references such as “yesterday”, “since Tuesday” or “next year”, giving them a value relative to the document creation time. When this normalization does not result in a concrete date or span, the expression is categorized as a *vague date* and assigned the value “????”. In many cases these are

genuinely non-specific temporal expression like a duration (“99 days”) or an interval (“monthly”) that do not belong on a timeline; in other cases, these are expressions that TERNIP failed to extract correctly but can be curated by the author to a meaningful date or span. Next, TimeLineCurator **formats** the set of extracted dates into structured JSON, which also includes the sentence containing each temporal reference and its location within the source document.

Given this structured format, TimeLineCurator then **shows** the timeline, encoding individual events as well as event spans along the timeline axis; vague dates are not shown on the timeline, but are presented to the author separately. At this point, the author can **update** the timeline; she can add, delete, merge, or edit events until satisfied, including events associated with vague dates. This entire process can be repeated any number of times with additional unstructured text. When ready to **present**, the author can export the timeline, and at any time, the author can save the state of an edited timeline to resume editing later.

5.5 Interface & Design Rationale

To give a better understanding of the interface’s functionality we will explain the specific tasks and contents of the single views in more detail and explain the visual encoding.

5.5.1 Timeline View

The core of the tool is the timeline visualization itself. It provides the global overview of all events. Information-dense areas are spread out so that there is no occlusion and a simple navigation provided - an approach similar to Ferstay et al.’s Variant View [11].

Figures 1.1 and 5.9d show examples with many stacked and dodged glyphs, providing an overview where the temporal distribution of events is visible even in densely populated areas of the timeline. There is no zooming or horizontal scrolling: the size of the discrete events is fixed and the entire horizontal axis is shown at all times. As a result, the author always has an overview of the full time range. Vertical scrollbars appear when the events overflow the available vertical space, as a backstop solution to ensure that arbitrarily dense time distributions can be curated. Typically, the final curated version of the timeline exported for presentation does not require vertical scrolling.

The horizontal time axis is scaled automatically to the range of time encompassed by the active events, and will update if any addition, removal, or editing of an event changes that range. The document creation time is indicated on the axis as a vertical dashed line labeled ‘today’.


An event corresponding to a single date is encoded as a circle ●, while an event span with a beginning date and an end date is encoded as a connecting bar of variable length flanked by triangles ▶◀. Vague dates corresponding to possible events, based on temporal references like “the day after” or “summer” are encoded as a square ■ and shown outside the horizontal range of the timeline axis, in the upper right corner of this view, as in Figure 5.9c. Events are coloured by hue according the six possible *tracks* (●●●●●●), and this base univariate colour palette was selected from ColorBrewer [14]. Glyphs corresponding to events that have already been edited are more saturated than those corresponding to unedited events (● vs. ●), for a bivariate palette with 12 colors in total. By default, events from each successive document text pasted into TimeLineCurator are assigned to a different track, but the author can override this behaviour by explicitly selecting a colour track when loading a new document (Figure 5.9b). Having multiple colour tracks can assist the author in comparing timelines from multiple documents.

5.5.2 List View

The list view combines all events and can be sorted by several criteria. By default the list is sorted according to the appearance of events within the original document - we found that to be the preferred order to initially detect the document’s structure. The list can additionally be sorted chronologically, by type of event (date, duration or vague), by state of curation (was it edited yet or not?), by document or by track. Each list entry contains the event glyph, its date and its title. An event that was deleted and thus is not displayed on the timeline anymore will still be retrievable from the list view, but grayed and crossed out - it can be recovered in the Control Panel. In the list view as well as in the document view, a bar at the bottom shows a summary of the total number of events and vague dates.

5.5.3 Document View

The document view supports the growing trend in journalism of linking original source documents to online news media, as with tools such as DocumentCloud (<http://documentcloud.org>), following the demands for more transparency and involvement of the readers [58]. In addition to supporting the curation process for authors, the document view allows readers of the curated timeline to see the relationships between events and corresponding sentences in source documents. This panel displays original unstructured document text, where all recognized temporal references are **highlighted** in orange.

The green “plus”-button  in the top bar lets the author add a new document. When clicking on it the input dialog as in Figure 5.9b opens up. If there is more than one document, clicking on the document’s title which is colored according to its track, opens up a list where the author can select another document. The displayed document will automatically switch when an event from another document is selected over the timeline or the list view.

5.5.4 Control Panel

The Control Panel on the bottom right allows the author to edit an event selected in any of the other three views, as shown in Figure 5.9d. She can modify the date of an event, turn a single event into a span, or vice versa; she can also edit the title and description for an event by clicking on either of these fields. By default, the event description is the sentence from which the event was extracted. When a vague date is given a concrete date, its corresponding glyph is moved to its appropriate place in the timeline visualization and becomes more saturated. The author can also delete the event, reassign the event to another color track, or add media such as image to it.

The toolbar on the right side of the Control Panel offers several options:

- (a) Leads the author back “home”, where the title and the description of the whole timeline can be edited; also tracks can be renamed here
- (b) Export options to display the curated timeline: either as a read-only version of TLC, or as TimelineJS project. More about the different options in Section 5.5.6
- (c) Download the curated data: either as TLC’s own “.tl”-file, as JSON file or as Zip folder that contains the JSON file and a index.html. With the Zip folder the author could host the TimelineJS-project on another server - script and stylesheets are with absolute paths to their server of origin
- (d) Upload previously curated data: to continue working on a previous state, the author can either restore the local storage data from the previous session, or upload a “.tl”-file
- (e) Create a new date, independent from any document
- (f) *(Only visible when more than one date selected)* Merge selected dates to one date (new date has value of primarily selected date)
- (g) *(Only visible when more than one date selected)* Delete all selected elements



Figure 5.8: Toolbox

5.5.5 View Coordination and Navigation

Event selection is propagated as linked highlighting across all views, with selected events highlighted in black, as shown in Figure 1.1. In the document view, events can be selected by clicking on any sentence that includes a temporal reference. Navigation is also linked across the views; when clicking on an event in the Timeline Visualization View, the list view and document view will scroll to the corresponding sections of the list and document, respectively. Keyboard arrow keys and paging buttons in the Control Panel will iterate through events using the current sort order of the list view.

5.5.6 Presentation and Export

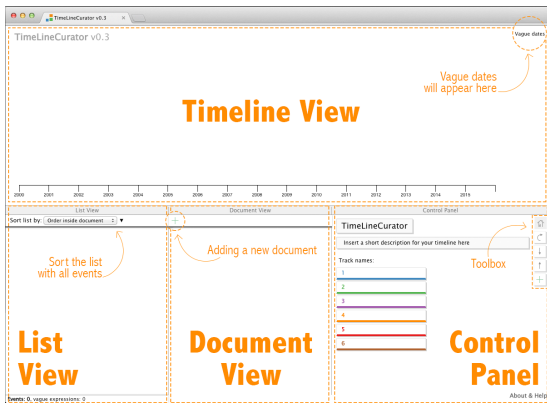
When the author is done with curating, he can export the timeline in two different ways. Vague events that were not provided with a date will not be exported.

The **TimeLineCurator presentation view** is a read-only version very similar to the editing interface, as shown in Figure 5.9e. The timeline is hosted on a shareable unique URL. Coordinated navigation and selection across the views remain the same; the Control Panel is replaced with an Event Details panel, in which any image media associated with an event is shown.

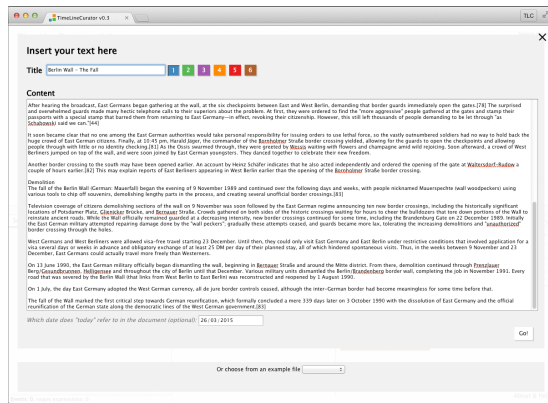
A timeline can also be exported as a **TimelineJS widget** that can be downloaded and embedded on the author's site, as shown in Figure 5.9f. We provide TimelineJS export capability because of its popularity, despite the drawbacks discussed in Section 5.1.

5.5.7 Exemplary walkthrough

Figures 5.9a to 5.9f show several steps through the curation process of one example data set. Figure 5.9a is the tool when opened up in the browser. It is empty with only an arrow in the document view indicating where to start. To clarify we annotated the figure with the names of the different views. When following the instruction to upload a document an overlay appears as in Figure 5.9b. Here the author can insert text - in this case we took the Wikipedia article about the "Berlin Wall" (http://en.wikipedia.org/wiki/Berlin_Wall), copied the section "The Fall" and inserted it in here. We have to give it a title and optionally determine a color. Because it is a Wikipedia article there is no specific Document Creation Time, so we leave that as it is and press "Go!". Figure 5.9c shows the timeline immediately after the temporal extraction process is finished. It contains some vague and falsely identified dates that can now be edited or deleted. In Figure 5.9d the timeline has been cleaned up and the dates have been divided into two different categories indicated by the two different colors. Figures 5.9e and 5.9f show the two different options for export: TLC's own export and TimelineJS's.



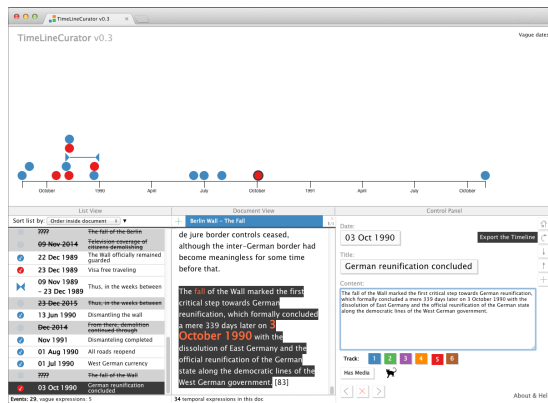
(a) Initially, the timeline is empty. Orange annotations show the four views: timeline view, list view, document view, and control panel.



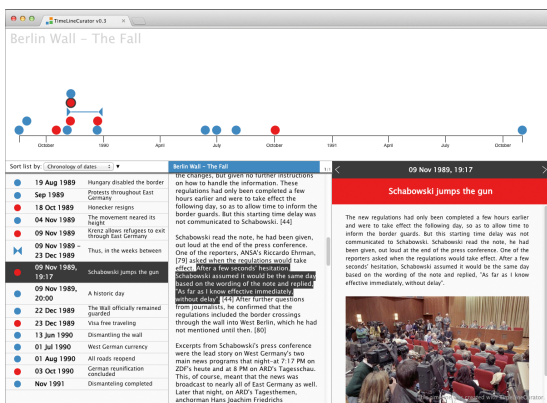
(b) Unstructured text is added via a popup dialog. Optionally, the document creation time can be specified below the input field.



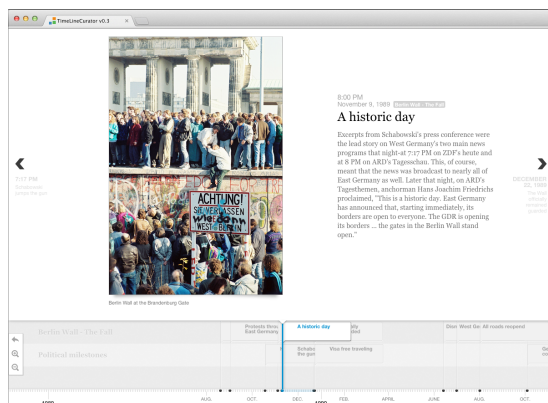
(c) A timeline immediately after importing text, with many vague and uncurated dates. General timeline information can be modified when no event is selected.



(d) Event dates, title, and description can be adjusted when an event is selected, it can also be assigned to another track, enriched with images, or deleted.



(e) Export option 1: a slightly modified read-only version of TLC.



(f) Export option 2: the open-source tool TimelineJS [61].

Figure 5.9: A walkthrough of the TimLineCurator curation process. We demonstrate this process using unstructured document text from the “The Fall” section of the Wikipedia article on the Berlin Wall (http://en.wikipedia.org/wiki/Berlin_Wall). The resulting timeline can be accessed at <http://goo.gl/SU1faP>.

6 Analysis

We evaluate TimeLineCurator in several ways. We benchmark its correctness in terms of text extraction quality. We also compare its user experience to the structured creation approach. We present instances where TimeLineCurator is used to rule out documents that contain little or no interesting temporal information, and we present examples of curated timelines and provide before and after images to show the changes made in the curation process. Finally, we discuss preliminary feedback from target users.

6.1 Extraction Error Benchmark

First we want to demonstrate how well the temporal information extraction works. Therefore we did benchmark tests and compared the automated with the manual extraction. The tests have been conducted by one of the authors of the VAST paper, thus familiarity with the system was given. We are aware of the fact that this approach does not represent the way a user would address the problem, but it gives an idea of the error rate and the speed-up potential, which is hard to test during a real creation process, as every person approaches the curation process very differently.

The automated extraction process involved uploading unstructured document text into TimeLineCurator and systematically checking every extracted event to verify that it was recognized correctly; we also determined if incorrectly extracted dates required editing or deletion. The manual extraction process involved reading the original document text and performing manual data entry, copying all temporal references and their surrounding sentences into a spreadsheet in the structured format required for TimelineJS input. In this initial benchmark, the author’s judgement was restricted to simply judging whether the expression correctly indicated a single event or a date range. No judgement was used about whether an event was interesting enough to merit inclusion on the timeline. Event titles or descriptions were not edited.

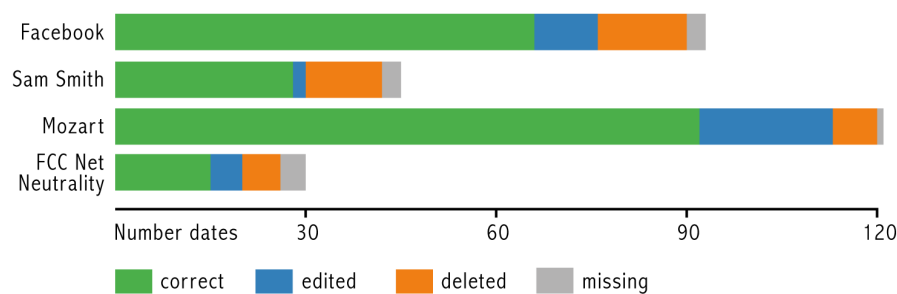


Figure 6.1: The results of the benchmark tests, which compares the gold standard manual creation of an event set with the automated event extraction of TimeLineCurator.

The benchmark data sets were three Wikipedia articles^a and two recent news articles^b; the two news articles were added to a single timeline. Figure 6.1 shows the quality assessments of TimeLineCurator’s temporal expression extraction compared against the gold standard of manual extraction. These results indicate that most of the dates were identified correctly (an average of 65%), though some needed curation via editing

or deletion (an average of 29%), and a small fraction were not extracted (an average of 6%). These results confirm that automatic extraction is a good match with our expectations: the true positive rate is reasonable but far from perfect, and the false negative rate is low. Thus, we deem that scaffolded curation is a viable approach to timeline authoring.

^aThe history of Facebook (<http://goo.gl/aKRKvr>), the biography of pop musician Sam Smith (<http://goo.gl/dF4Gzm>), and the biography W. A. Mozart (<http://goo.gl/VTmwAF>).

^bBoth pertained to the topic of net neutrality <http://goo.gl/wFS0Jf> and <http://goo.gl/9cD2V2>.

This benchmark also yielded qualitative insights on the kinds of expressions that were incorrectly extracted. Incorrectly identified dates often were time spans, which can be expressed in many different ways in prose. For example, in *“The family again went to Vienna in late 1767 and remained there until December 1768”*, two separate dates were automatically extracted, but the author combined them into one time span during manual curation. Another reason for incorrectly extracted events were temporal expressions that implicitly refer to a previously named date rather than explicitly containing a year. The natural language processing misses these expressions because it only considers the immediate context and incorrectly ties them to the document’s creation date. The result is that historical texts incorrectly have many dates assigned to “today” despite only containing dates from the distant past. Another source of false positives are temporal expressions that are used as names and do not refer to a specific event, such as Taylor Swift’s album title *“1989”* or the TV Show *“Last Week Tonight”*.

Events that were missed by the automatic extraction were often those which referred to another event, such as *“six days after the site launched”* or possessive statements, such as *“last week’s vote”*. In some cases these were extracted as vague dates, and in others they were missed completely. Currently, the year recognition is limited to Anno Domini years with four digits; references such as *“13,000-12,000 BC”* are not handled.

The speedup when comparing the manual to the automated extraction benchmark tests were that the automated extraction is between 2x and 3x faster. However as mentioned above the benchmark test were an artificial scenario so those numbers don’t represent end-user behavior for now.

Moreover, this benchmark scenario focused solely on the verification and correction of event dates and did not involve any editorial judgment, such as deciding which events to include in the timeline and how to embellish these dates with interesting event titles and descriptions. However, we conjecture that the complete curation process with TimeLineCurator is easier and preferable to the tedious manual structured creation approach.

Nevertheless the tests revealed some usability issues as well as little problems with the TERNIP rules. So after the first tests we conducted some changes for the curation environment (such as the ability to navigate through dates using arrow keys and auto-selecting the next event after event deletion), as well as tweaking and expanding the Python based rules within TERNIP.

6.2 User Experience Comparison

To also get feedback from a more realistic timeline authoring process, we conducted a second benchmark, where we asked fellow grad students to go through the authoring process.

We recruited six arms-length participants from our department who were unaffiliated with the project and asked them to create coherent timelines. We provided them with short text articles and asked them to make editorial judgements about each event they encountered; they were also asked to curate event titles. Each author curated two timelines: first, one using manual structured data entry as required by TimeLineJS [61] and second, one using TimeLineCurator. They were directed to curate the timeline until they were fully satisfied and felt that it was ready to be exported. All participants strongly preferred TimeLineCurator’s visual authoring environment to the structured data entry required by TimelineJS, and they found working with TimeLineCurator to be highly engaging. Every user encountered at least some difficulties with the structured editing approach despite having a strong technical background. One participant even abandoned the structured editing approach completely after a few minutes because it was so tedious. The curation time from start to finish across participants is not directly comparable because the scope of the editorial judgment performed during the curation process varied considerably between them. This informal comparison of user experience provided encouraging qualitative evidence that the design goals of our authoring system were met.

6.3 Speculative Browsing

As mentioned previously we think TimeLineCurator is also a valuable tool for quickly ruling out unsuitable documents. Figure 6.2 shows three examples of timelines where it was possible to decide within a few seconds that the document is not a suitable source for an engaging timeline. As soon as the text is processed by the script and the author sees the visual timeline, the decision can basically be made immediately.

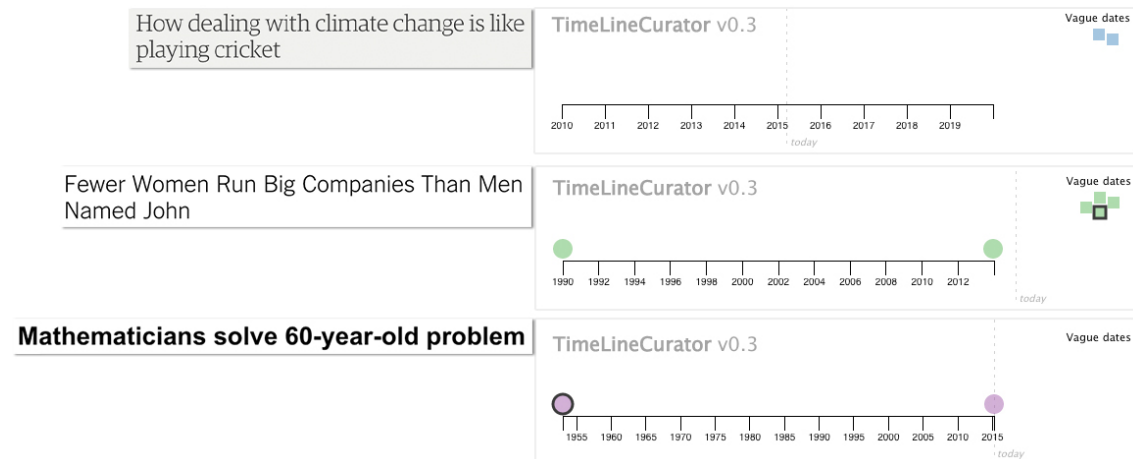


Figure 6.2: Timelines extracted from two news articles (<http://goo.gl/YLkmSx>, from Mar. 23, 2015 and <http://goo.gl/W8qrtT>, from Mar. 2, 2015) and a report from a science press release site (<http://goo.gl/jKRwM0>, from Mar. 23, 2015). All three do not contain much temporal information and thus can quickly be ruled out as a suitable basis for an interesting timeline.

6.4 Curated Examples

While working on the project, we curated many different timelines, for instance the history of the fall of the Berlin Wall as documented in Figure 5.9 and the biography of W. A. Mozart, as shown in Figure 6.3. Here the case of having several false results around “today” comes into the picture (as mentioned in Section 6.1). Those are based on missing context for events - an expression like “In August of that year” would be resolved to August of the document’s creation year.

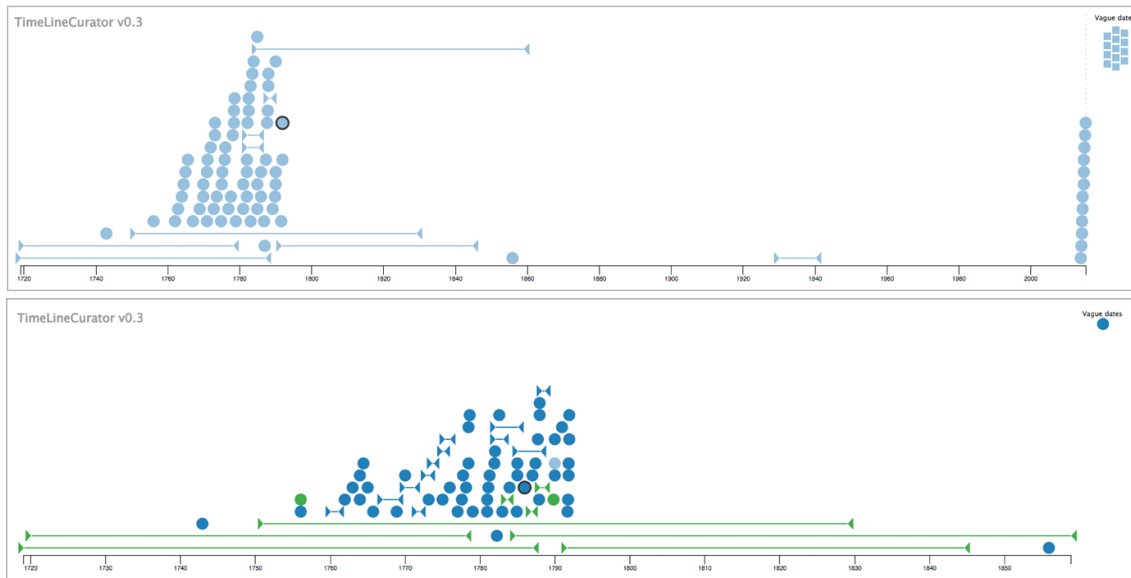


Figure 6.3: A timeline of composer W. A. Mozart’s biography <http://en.wikipedia.org/wiki/Mozart>, both before and after curation. The resulting timeline can be accessed at <http://goo.gl/2JikND>.

Many more examples are available in a gallery on the project page: <http://cs.ubc.ca/group/infovis/software/TimeLineCurator/#examples>. All examples were exported with both TimelineJS and TimeLineCurator’s presentation view.

6.5 Use Cases

In addition to those artificial scenarios, that we conducted in our immediate vicinity and where we defined the usage scenario, we also got feedback from real use cases. On the one hand we interviewed people who were currently working on timeline projects or have had experience creating timelines before, on the other hand we received voluntary feedback from interested people who caught on to the online deployment of TimeLineCurator.

6.5.1 Solicited potential users

We conducted semi-structured interviews with eight people: seven journalists and one policy researcher. Four of these individuals already had experience creating interactive timelines and provided us with feedback about the strengths and limitations of currently available timeline tools. Two of these individuals had pre-existing plans to use a timeline authoring tool in an upcoming project.

Most journalists agreed on the point that using data visualizations within an article traditionally is an afterthought and not an idea to start with. Most often it is something they might think about at the end to add a graphical element to the story or to support a statement with some numbers. Some also mentioned that they sometimes could see their story as a timeline but wouldn't know about available tools that enable a straightforward and code-free creation. So if journalists decide for using supplemental material, such as data visualizations they *“either have to throw money at people to do it for them or they decide for a non-perfect way to show information”*.

Even though having the dates in that format would only require transferring them into a spreadsheet, that step seemed to be a barrier for journalists that were not too established in the world of digital tools and reported that they felt intimidated when having to deal with new applications. Even though journalistic education includes more and more tools for computer-assisted reporting and data journalism, some of the interviewees admitted that they still weren't too comfortable with using digital tools.

When we presented TimeLineCurator to these individuals and asked them to try it out, their reaction was very positive and they remarked that it was very easy to use. They enjoyed the approach of extracting temporal event data from unstructured document text, and that they no longer had to start start with an empty spreadsheet and add every event manually one at a time. The immediate visual feedback during the authoring process was also highly appreciated.

One journalist said: *“For the less geeky journalists who might be scared of timelines, this is a brilliant super-easy way to see what it might look like”* and that TimeLineCurator might be a good way to *“break the barrier between the artiste writer and the data journalist”*.

We asked these individuals to speculate about possible kinds of stories that might benefit from accompanying timelines: these included the unfolding of political scandals, how amendment bills proceed in government, and biographies. They also proposed several use cases that we had not previously considered, such as using TimeLineCurator for data analysis rather than timeline authoring for presentation. One idea involved using TimeLineCurator with court documents when reporting on a trial to better understand the context of a criminal or legal case. Another possible use case is fact-checking during investigative analysis. Typically, details are verified through two reliable sources before publication. A journalist that we spoke to imagined that TimeLineCurator might accelerate fact-checking for temporal events and finding mis-

matches between sources. Finally, a third use case involved using TimeLineCurator to prepare for interviews, to quickly catch up the subject's biography or background.

6.5.2 Unsolicited current users

In contrast to the ideas above that are potential use cases for prospective users of TimeLineCurator, we can also report on use cases from people in different communities who already used TimeLineCurator for their own projects after it was deployed and publicized. One author was a digital humanities researcher who created a timeline to see the historical development of deaf churches in England. Another author was a user experience professional who created a timeline to accompany the profile of his company.

7 Discussion & Future Work

7.1 Discussion

TimeLineCurator offers a new way of exploring the temporal structure of a document in order to make the process of creating timelines enjoyable rather than arduous. We designed the system under the assumption that entity extraction through natural language processing is decent but not perfect, and can serve to support human-in-the-loop curation. Moreover, even if the extraction were perfect and all date events and spans were extracted correctly, there are still many subtasks involved in timeline curation that will need nuanced human judgement for quite some time.

One of those subtasks is the decision which information is relevant to inform about the desired level of detail. Interactivity potentially allows an unlimited number of events, but that doesn't mean the reader should receive an unlimited number - it's rather the opposite: in many cases less is more and it's the journalist's job to decide what information is relevant. The reader ideally trusts the journalist's ability to make that decision - to be on the safe side original documents and sources could be made accessible, enabling fact-checking or further investigations if required.

Figuring out the filtering remains an issue in modern AI in general. In a recent interview with Ken Goldberg (a computer engineer, roboticist, and artist) he points out that for computers the "ability to distinguish, to filter out what's interesting, that's still elusive" [60]. In that spirit tools like TimeLineCurator are far away from replacing the creative and considered human labor but they are rather meant to support and foster creativity. When talking about creative tasks like film or music production Goldberg says: "All the new tools for making movies and making music have been enormously beneficial for creativity. And computers and robots are relieving us of tedious tasks like handling documents and filing. That allows us to spend more of our time being creative" [60].

One more general cause for thought especially in the journalistic context: stories are meant to be told in an interesting and challenging way. And even if authors were familiar with all the tools to create different visualizations, data graphs and other interactive elements they still should only be considered when meaningful for the purpose and not only because it is possible. We could find many examples where media elements seem to be thrown into a story without contributing any value and are rather distracting than helpful. That was especially the case when a new "multimedia storytelling" movement tried to reach for the Pulitzer Prize winning *Snow Fall* project from *The New York Times* in 2012 [64]. But people tend to overlook that *Snow Fall* was a "six-month sixteen-person multimedia project" [54] with an intriguing story - and huge projects like that are not meant for the daily news reporting. Many stories can be told in words compellingly, saying it with a catchphrase: "Text isn't broken" [54]. Along these lines it also makes sense to first think about if a timeline is suitable for a story - and as we explained before TimeLineCurator can offer support for that decision as well.

During the creation process of TLC we learned that besides filtering and curating timeline events to create cleaned-up presentations, TimeLineCurator may as well serve for analysis tasks such as fact-checking, where researches have to compare information throughout different documents. Again that task involves elaborate human judgement, which will probably not be undertaken by machines too quickly.

7.2 Future Work

Generally the idea of TimeLineCurator was approved by many people from the broader community. Still some requests for refinements were made, as well as suggestions for elaborating or integration with existing tools and workflows.

Upgrading NLP Several requests concerned the improvement of the automated event extraction. We decided to use existing tools to handle the NLP task and they are known to be working fairly well, but not perfectly. Even though we did some manual corrections and additions inside the framework we used, it would make sense to replace the NLP part with newer tools as they are emerging. For example, more elaborate tools that consider context-dependent semantics become more widely available [21] and could be implemented. Also, moving to a toolkit that supports multiple languages would be worthwhile to allow the use of TimeLineCurator outside of English-speaking countries.

More choices for export TLC creates a structured event data set so it would be straightforward to offer more options for export. All tools mentioned in Section 4.2 that require structured event data as input could basically be added. Also, more options for customization concerning the style of a resulting timelines were asked for. Right now the author is restricted to the default style of the export. He could only download the project and changes the stylesheets - but that involves a fair amount of understanding the structure of web projects as well as some coding experience. Even though that was not part of the original plan, more options for customizing the output could be an interesting future implementation.

Integration with existing tools and workflows There are several tools available that approach not only one type of visualization but a broader field. Therefore it would be interesting to integrate TLC's approach into one of those existing tools. Systems that do textual analysis of documents would be a good opportunity for integration. One system in particular is *Overview* [6], an open-source tool for investigative journalism that supports the analysis of large collections of documents by sorting them according to topics and tags. *Overview* even offers an API which handles the document engine and allows the creation of own visualizations. Building a plugin for *Overview* will very likely be the next step to further distribute TLC.

8 Conclusion

In this thesis we gave an overview of how temporal data and events are expressed and displayed - historically as well as currently. Many use cases across several domains showed our literacy with the metaphor of time marching along a horizontal line. We pointed out benefits of their usage within journalistic environments and analyzed common approaches to create them, which are static manually created timelines, as well as digitally-generated timelines that are based on structured data sets. We introduced a model that divided the timeline creation process into five different tasks, which are: Browse, Extract, Format, Show, and Update. Based on this model, we introduced our new approach to this process, by applying techniques from natural language processing and integrating the creation process into a visual environment.

We presented TimeLineCurator, a visual timeline authoring system that recognizes temporal expressions within unstructured document text. It accelerates the event-extraction process and fulfills two broader tasks. First, it enables authors to create polished timelines from interesting documents within only a few minutes. Second, it enables speculative browsing, which lets authors eliminate temporally uninteresting documents from consideration within seconds. TimeLineCurator can be used by a broad community of authors including those without a strong technical background, because it is easily accessible, has a simple user interface, and does not require any programming. It lowers the access barrier for timeline creation for a broad set of potential authors, including journalists, who would like to work visually rather than via manual data entry into spreadsheets. TimeLineCurator can directly create two forms of curated timeline: the popular TimelineJS and our own presentation format that provides an information-dense overview. Moreover, the resulting set of curated events can be exported as a structured data set, opening up further possibilities beyond these two currently-supported presentation formats. Interviews and community feedback provided evidence that the TimeLineCurator approach of scaffolded curation built on top of imperfect automatic entity extraction provides useful and appealing functionality in several application domains.

References

- [1] E. Alexander et al. “Serendip: Topic model-driven visual exploration of text corpora”. In: *Proc. IEEE Conf. Visual Analytics Science and Technology (VAST)*. 2014, pp. 173–182.
- [2] Y. Artzi and L. Zettlemoyer. “Weakly supervised learning of semantic parsers for mapping instructions to actions”. In: *Trans. Assoc. Computational Linguistics* 1 (2013), pp. 49–62.
- [3] L. Boroditsky. “Metaphoric structuring: understanding time through spatial metaphors”. In: *Cognition* 75.1 (2000), pp. 1–28.
- [4] M. Bostock, V. Ogievetsky, and J. Heer. “D³: Data-driven documents”. In: *IEEE Trans. Visualization and Computer Graphics (Proc. InfoVis)* 17.12 (2011), pp. 2301–2309.
- [5] M. Brehmer and T. Munzner. “A multi-level typology of abstract visualization tasks”. In: *IEEE Trans. Visualization and Computer Graphics (Proc. InfoVis)* 19.12 (2013), pp. 2376–2385.
- [6] M. Brehmer et al. “Overview: The design, adoption, and analysis of a visual document mining tool for investigative journalists”. In: *IEEE Trans. Visualization and Computer Graphics (Proc. InfoVis)* 20.12 (2014), pp. 2271–2280.
- [7] J. Chae et al. “Spatiotemporal social media analytics for abnormal event detection and examination using seasonal-trend decomposition”. In: *Proc. IEEE Conf. Visual Analytics Science and Technology (VAST)*. 2012, pp. 143–152.
- [8] A. X. Chang and C. D. Manning. “SUTime: A library for recognizing and normalizing time expressions”. In: *Proc. Intl. Conf. Language Resources and Evaluation (LREC)*. 2012, pp. 3735–3740.
- [9] H. Chen et al. “Visualization in law enforcement”. In: *Proc. Extended Abstracts ACM SIGCHI Conf. Human Factors in Computing Systems (CHI)* (2005), pp. 1268–1271.
- [10] W. Dou et al. “HierarchicalTopics: Visually exploring large text collections using topic hierarchies”. In: *IEEE Trans. Visualization and Computer Graphics (Proc. VAST)* 19.12 (2013), pp. 2002–2011.
- [11] J. A. Ferstay, C. B. Nielsen, and T. Munzner. “Variant View: Visualizing sequence variants in their gene context”. In: *Trans. IEEE Visualization and Computer Graphics (Proc. InfoVis)* 19.12 (2013), pp. 2546–2555.
- [12] C. Görg, Z. Liu, and J. Stasko. “Reflections on the evolution of the Jigsaw visual analytics system”. In: *Information Visualization* 13 (2014), pp. 336–345.
- [13] C. Görg et al. “Combining computational analyses and interactive visualization for document exploration and sensemaking in Jigsaw”. In: *IEEE Trans. Visualization and Computer Graphics (TVCG)* 19.10 (2013), pp. 1646–1663.
- [14] M. Harrower and C. A. Brewer. “ColorBrewer.org: An online tool for selecting colour schemes for maps”. In: *Cartographic Journal* 40.1 (2003), pp. 27–37.
- [15] S. Havre et al. “ThemeRiver: Visualizing thematic changes in large document collections”. In: *IEEE Trans. Visualization and Computer Graphics (TVCG)* 8.1 (2002), pp. 9–20.
- [16] Y. A. Kang and J. T. Stasko. “Examining the use of a visual analytics system for sensemaking tasks: Case studies with domain experts”. In: *IEEE Trans. Visualization and Computer Graphics (Proc. VAST)* 18.12 (2012), pp. 2869–2878.

- [17] W. Klein. “How time is encoded”. In: *The expression of time 3* (2009), pp. 1–43.
- [18] T. Kwiatkowski et al. “Scaling semantic parsers with on-the-fly ontology matching”. In: *Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP)*. 2013, pp. 1545–1556.
- [19] B. C. Kwon et al. “VisJockey: Enriching data stories through orchestrated visualization”. In: *Proc. Computation + Journalism Posters*. 2014.
- [20] E. L adavas. “Asymmetries in processing horizontal and vertical dimensions”. In: *Memory & cognition* 16.4 (1988), pp. 377–382.
- [21] K. Lee et al. “Context-dependent semantic parsing for time expressions”. In: *Proc. Conf. Assoc. Computational Linguistics (ACL)*. 2014, pp. 1437–1447.
- [22] Y. Liu et al. “Evaluating exploratory visualization systems: A user study on how clustering-based visualization systems support information seeking from large document collections”. In: *Information Visualization* 12.1 (2013), pp. 25–43.
- [23] D. Luo et al. “EventRiver: Visually exploring text collections with temporal references”. In: *IEEE Trans. Visualization and Computer Graphics (TVCG)* 18.1 (2012), pp. 93–105.
- [24] I. Mani. “Computational Modeling of Narrative”. In: *Synthesis Lectures on Human Language Technologies* 5.3 (2012), pp. 1–142.
- [25] I. Mani and G. Wilson. “Robust temporal processing of news”. In: *Proc. Conf. Assoc. Computational Linguistics (ACL)*. 2000, pp. 69–76.
- [26] Scott McCloud. *Understanding comics*. William Morrow Paperbacks, 1993.
- [27] M. Mitchell. “The Visual Representation Of Time In Timelines, Graphs, And Charts”. In: *Conference paper delivered to the Australian & New Zealand Communication Association Conference 2004*. (2004). URL: http://epublications.bond.edu.au/hss%5C_pubs/107/.
- [28] T. Munzner. *Visualization Analysis and Design*. CRC Press, 2014.
- [29] J. Olsson and M. Boldt. “Computer forensic timeline visualization tool”. In: *Digital Investigation* 6 (2009), pp. 78–87.
- [30] R. Pasts. “Why Time is so Now”. In: *Rethinking History: The Journal of Theory and Practice* 16.2 (2012), pp. 303–317.
- [31] C. Plaisant, R. Mushlin, and A. Snyder. “LifeLines: using visualization to enhance navigation and analysis of patient records.” In: *Proceedings of the AMIA Symposium. American Medical Informatics Association* (1998).
- [32] C. Plaisant et al. “LifeLines: Visualizing personal histories”. In: *Proc. ACM SIGCHI Conf. Human Factors in Computing Systems (CHI)*. 1996, pp. 221–227.
- [33] J. Priestley. *A description of a chart of biography*. Printed at Warrington, 1765, p. 5. URL: <http://books.google.com/books?hl=en%5C&lr=%5C&id=w5QBAAAAQAAJ%5C&pgis=1>.
- [34] J. Pustejovsky et al. “TimeML: Robust specification of event and temporal expressions in text”. In: *Fifth Intl. Wkshp. Computational Semantics (IWCS)*. 2003.
- [35] D. Ren, T. Hollerer, and X. Yuan. “iVisDesigner: Expressive interactive design of information visualizations”. In: *IEEE Trans. Visualization and Computer Graphics (Proc. InfoVis)* 20.12 (2014), pp. 2092–2101.
- [36] D. Rosenberg. “Joseph Priestley and the Graphic Invention of Modern Time”. In: *Studies in Eighteenth Century Culture* 36 (2007), pp. 55–103.

- [37] D. Rosenberg and A. Grafton. *Cartographies of time: A history of the timeline*. Princeton Architectural Press, 2013.
- [38] A. Satyanarayan and J. Heer. “Authoring narrative visualizations with Ellipsis”. In: *Computer Graphics Forum (Proc. EuroVis)* 33.3 (2014).
- [39] A. Satyanarayan and J. Heer. “Lyra: An interactive visualization design environment”. In: *Computer Graphics Forum (Proc. EuroVis)* 33.3 (2014).
- [40] R. N. Shepard and S. Hurwitz. “Upward direction, mental rotation, and discrimination of left and right turns in maps”. In: *Cognition* 18.1-3 (1984), pp. 161–193.
- [41] J. Stasko, C. Görg, and R. Spence. “Jigsaw: Supporting investigative analysis through interactive visualization”. In: *Information Visualization* 7.2 (2008), pp. 118–132.
- [42] J. Strötgen and M. Gertz. “Multilingual and cross-domain temporal tagging”. In: *Language Resources and Evaluation* 47.2 (2013), pp. 269–298.
- [43] Edward R Tufte. *The Visual Display Of Quantitative Information*. Vol. 2. Graphics press Cheshire, CT, 1983, p. 28.
- [44] B. Tversky, S. Kugelmass, and A. Winter. “Cross-Cultural and Developmental Trends in Graphic Productions”. In: *Cognitive Psychology* 23.4 (1991), pp. 515–557.
- [45] R. Vadlapudi et al. “LensingWikipedia: Parsing text for the interactive visualization of human history”. In: *Proc. IEEE Conf. Visual Analytics Science and Technology (VAST) Poster Compendium*. 2012, pp. 247–248.
- [46] M. Verhagen et al. “Annotation of temporal relations with Tango”. In: *Proc. Intl. Conf. Language Resources and Evaluation (LREC)*. 2006.
- [47] M. Verhagen et al. “Automating temporal annotation with TARSQI”. In: *Conf. Assoc. Computational Linguistics Poster Proceedings*. 2005, pp. 81–84.
- [48] F. B. Viégas et al. “ManyEyes: A site for visualization at internet scale”. In: *IEEE Trans. Visualization and Computer Graphics (TVCG)* 13.6 (2007), pp. 1121–1128.
- [49] R. L. Walter, S. Berezin, and A. Teredesai. “ChronoZoom: Travel Through Time for Education, Exploration, and Information Technology Research”. In: *Proceedings of the 2nd annual conference on Research in information technology* (2013), pp. 31–36.
- [50] R. Yan et al. “Evolutionary timeline summarization: A balanced optimization framework via iterative substitution”. In: *Proc. ACM SIGIR Conf. Information Retrieval*. 2011, pp. 745–754.
- [51] J. S. Yi et al. “Toward a deeper understanding of the role of interaction in information visualization”. In: *Visualization and Computer Graphics, IEEE Transactions on* 13.6 (2007), pp. 1224–1231.
- [52] J. Zhao et al. “TimeSlice: Interactive faceted browsing of timeline data”. In: *Proc. ACM Conf. Advanced Visual Interfaces (AVI)*. 2012, pp. 433–436.

Web References

- [53] S. Archibald and D. Rosenberg (Cabinet Magazine). *A Timeline of Timelines*. 03/2004. <http://goo.gl/h6yX3p>. (Last accessed: 15/05/2015).
- [54] D. Thompson (The Atlantic). *'Snow Fall' Isn't the Future of Journalism - And that's not a bad thing*. 21/12/2012. <http://goo.gl/KtZ3xy>. (Last accessed: 15/05/2015).
- [55] J. DelViscio and D. Overbye (The New York Times). *The Higgs, From Theory to Reality*. 10/2013. <http://goo.gl/39DW8Z>. (Last accessed: 15/05/2015).
- [56] S. Pulham G. Blight and P. Torpey (The Guardian). *Arab spring: an interactive timeline of Middle East protests*. 05/01/2012. <http://goo.gl/V9UHgR>. (Last accessed: 20/04/2015).
- [57] University of Maryland Human Computer Interaction Lab. *EventFlow: Visual Analysis of Temporal Event Sequences and Advanced Strategies for Healthcare Discovery*. Last update: 2014. <http://www.cs.umd.edu/hcil/eventflow>. (Last accessed: 15/05/2015).
- [58] S. Rogers (Mother Jones). *Hey wonk reporters, liberate your data!* 24/04/2014. <http://goo.gl/cbhgtk>. (Last accessed: 15/05/2015).
- [59] D. Rosenberg (Cabinet Magazine). *The Trouble with Timelines*. 03/2004. <http://goo.gl/HDqrv3>. (Last accessed: 15/05/2015).
- [60] J. Carstensen (Nautilus Magazine). *Robots Can't Dance - Why the singularity is greatly exaggerated*. 22/01/2015. <http://nautil.us/issue/20/creativity/robots-cant-dance>. (Last accessed: 15/05/2015).
- [61] Northwestern University Knight Lab. *TimelineJS - Beautifully crafted timelines that are easy and intuitive to use*. 2013. <http://timeline.knightlab.com>. (Last accessed: 15/05/2015).
- [62] C. Northwood. *TERNIP: Temporal Expression Recognition and Normalisation in Python*. 2010. <http://github.com/cnorthwood/ternip>. (Last accessed: 15/05/2015).
- [63] Outercurve Foundation. *ChronoZoom - Zoom through all of time*. 2013. <http://chronozoom.tumblr.com/about>. (Last accessed: 15/05/2015).
- [64] J. Branch (The New York Times). *Snow Fall - The Avalanche at Tunnel Creek*. 22/12/2012. <http://goo.gl/Q2WXqv>. (Last accessed: 15/05/2015).
- [65] C. Tominski and W. Aigner. *The TimeViz Browser - A Visual Survey of Visualization Techniques for Time-Oriented Data*. Last update: 28/11/2013. <http://www.timeviz.net>. (Last accessed: 15/05/2015).
- [66] B. Victor. *Drawing Dynamic Visualizations, Stanford HCI Seminar*. 01/02/2013. <http://vimeo.com/66085662>. (Last accessed: 15/05/2015).

A Appendices

A.1 Contents of the CD

1. **Thesis:** PDF file and L^AT_EX-source file of this document

2. **Code**

TimeLineCurator To run TimeLineCurator locally, a virtual environment has to be set up (*How-to-setup-venv.txt* included in folder). The currently running version can be accessed online <http://tl-generator.herokuapp.com/>; The code is also available on Github: <http://github.com/jo-fu/TimeLineCurator>

TLC Export The export view calls data from Amazon's Simple Storage Service, but is initially set to a fallback dataset. It is available online: <http://www.cs.ubc.ca/group/infovis/software/TimeLineCurator/tlcExport/>; the code is also on Github: <http://github.com/jo-fu/TLC-Export>

3. **Figures** used in this thesis, named according to their Figure number inside the text
4. **References:** papers and PDF prints of the web pages cited in this thesis, named according to their reference number
5. **Video:** explaining idea and functionality of TimeLineCurator (can also be accessed here: <http://vimeo.com/jofu/tlc>)
6. **Presentation:** PDF file with slides of the final presentation

A.2 VAST Paper

The following paper has been written prior to this work and was submitted to the IEEE Conference on Visual Analytics Science and Technology (IEEE VAST 2015, <http://ieevis.org>) on March 31, 2015. Notification about results of the first review cycle will be sent out June 6, 2015 - so after this thesis was handed in.

TimeLineCurator: Interactive Authoring of Visual Timelines from Unstructured Text

Johanna Fulda, Matthew Brehmer, and Tamara Munzner *Member, IEEE*

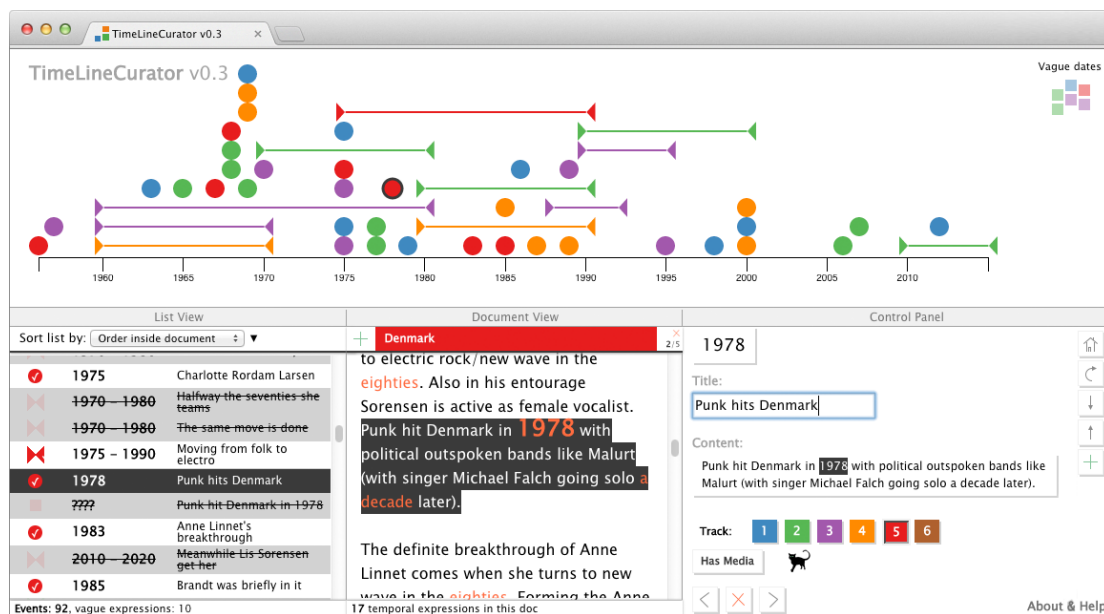


Fig. 1: The browser-based visual timeline authoring tool TimeLineCurator, showing a timeline of Scandinavian pop music, where each colour corresponds to a country; access the interactive timeline at <http://goo.gl/0bH1vA>.

Abstract— We present TimeLineCurator, a browser-based authoring tool that automatically extracts event data from temporal references in unstructured text documents using natural language processing and encodes them along a visual timeline. Our goal is to facilitate the timeline creation process for journalists and others who tell temporal stories online. Current solutions involve manually extracting and formatting event data from source documents, a process that tends to be tedious and error prone. With TimeLineCurator, a prospective timeline author can quickly identify the extent of time encompassed by a document, as well as the distribution of events occurring along this timeline. Authors can speculatively browse possible documents to quickly determine whether they are appropriate sources of timeline material. TimeLineCurator provides controls for curating and editing events on a timeline, the ability to combine timelines from multiple source documents, and export curated timelines for online deployment. We evaluate TimeLineCurator through a benchmark comparison of entity extraction error against a manual timeline curation process, a preliminary evaluation of the user experience of timeline authoring, a brief qualitative analysis of its visual output, and a discussion of prospective use cases suggested by members of the target author communities following its deployment.

Index Terms—System, timelines, authoring environment, time-oriented data, journalism.

1 INTRODUCTION

Event timelines are an effective way to present stories and provide context to an audience. The initial motivation for our work was the use of timelines by journalists for presentation, but they are common in many other domains including medicine, history, education, and law enforcement.

- Johanna Fulda is with the University of Munich (LMU). Email: mail@johannafulda.de.
- Johanna Fulda, Matthew Brehmer, and Tamara Munzner are with the University of British Columbia. E-mail: {jfulda,brehmer,tmm}@cs.ubc.ca.

Manuscript received 31 Mar. 2015; accepted 1 Aug. 2015; date of publication xx xxx 2015; date of current version xx xxx 2015.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

When presented alongside an accompanying text, a timeline provides a succinct overview for the article in the form of a temporal index that indicates the chronological extent of the article, as well as the number and distribution of events across this extent; a chronological understanding is achieved through the use of a spatial metaphor. Interactive visual timelines such as those employed by the Timeline iOS application [61] or by the *New York Times*¹ offer an immediate overview of an article's chronology and a means for the reader to orient herself within this chronology as she reads.

Despite the prevalence of stories with a fundamentally temporal structure, visual timelines are scarce; there are many articles² that simply list events in a chronological order without providing any visual overview of their chronology or the temporal distribution of events.

Why are visual timelines so uncommon? Based on the first author's experience working in the graphics department of a major German news publication, as well as interviews with journalists, we know that

¹For example, see *Timeline: The Higgs, From Theory to Reality* [10]

²See these timelines about Edward Snowden [19] or flight MH370 [39].

the timeline authoring process is too difficult: it is tedious, error-prone, and time-consuming.

Journalists are accustomed to working with daily or weekly deadlines; this constraint is not conducive to the time-consuming manual creation of visual timelines using illustration tools, or to the creation of formatted event lists required by template-based timeline generation tools [29, 43]. Furthermore, there is often little guarantee that a timeline generated via either means will be visually compelling or of benefit to the reader. As this benefit can only be gauged after the timeline is created, the significant time investment is often deemed to not be worth it. Finally, another use of a timeline is to provide additional background context for a story, including events that may not appear in the accompanying text article; locating and browsing additional source documents for these timelines can be very time-consuming.

For prospective authors willing to devote time to timeline generation, the creation process can be highly unsatisfying. They may be unaware of appropriate tools, or these tools may be difficult to integrate into an existing work environment; for instance, many journalists cannot install software on their computers without support from a central IT authority. Even browser-based tools may deliver results that are not simple to incorporate into the newsroom’s content management system, or results that do not adhere to the publication’s style guidelines, leading to issues that cannot be resolved without coding experience.

We propose an alternative to manual illustration or tools that require structured event data: the TimeLineCurator approach is illustrated in Figure 2. We use natural language processing to automatically extract temporal information from unstructured text input. We explicitly assume that this extraction provides results that are not perfect, but are good enough to provide scaffolding for interactive visual curation to accelerate the timeline authoring process. The output is a curated timeline.



Fig. 2: An abstract representation of TimeLineCurator’s pipeline: (i) unstructured text input; (ii) an authoring environment; (iii) curated timeline output.

Contributions: Our primary contribution is TimeLineCurator, the web-based visual timeline authoring system shown in Figure 1. It allows for the fast and easy creation of a structured temporal event dataset from unstructured document text, combining imperfect natural language processing and “human in the loop” authoring. With TimeLineCurator, an author can speculatively browse a document’s temporal structure; she can quickly rule out documents as unsuitable for timelines within seconds, or interactively curate suitable documents to refine an event set within minutes, receiving constant visual feedback throughout the curation process. Our secondary contribution is a *Timeline Authoring Model*, which we use to position TimeLineCurator relative to other timeline generation approaches in terms of goals and tasks.

Outline: We begin by discussing related work in Section 2 and our design process in Section 3. In Section 4 we present our Timeline Authoring Model and the architecture and processing pipeline of TimeLineCurator. Section 5 contains an overview of the interface and rationale for our design choices. We evaluate TimeLineCurator in five ways in Section 6. We discuss our results in Section 7 and present possible directions for future work. Section 8 summarizes our contributions.

2 RELATED WORK

Our discussion of relevant previous work includes visualization authoring tools, tools for generating visual timelines from structured event data, and techniques that leverage natural language processing, entity extraction, and metadata extraction from text documents.

2.1 Visualization Authoring Tools

For almost every level of expertise there exist ways to create visualizations. Visualization authoring tools that require higher levels of technical expertise provide more options for customization.

General purpose tools for visualization presentation: Popular and accessible tools such as Tableau [59] and ManyEyes [66] provide the means to generate, share, and publish visualizations without having to write any code. However, these tools expect structured data; it is difficult to generate visualizations from unstructured text data without wrangling the data into a structured form. In addition, these tools do not explicitly support the generation of visual event timelines. For example, ManyEyes offers a set of general-purpose visualizations and there is no visualization for event-based data within its repertory. Although Tableau is sufficiently customizable that the visual appearance of a timeline can be achieved with elaborate data transformations, this task is clearly not one of its primary design targets.

Custom visualization authoring environments: Visual authoring tools such as Lyra [55] and iVisDesigner [50] are more expressive, allowing the author to compose visualizations with multiple layers and annotations. It is thus feasible to produce a custom visual timeline, once again assuming that the event data is already in a structured form. Since environments like Lyra and iVisDesigner provide more options for customization and typically require more time to learn, they are less suitable for fast and easy authoring than a specialized tool, such as those that are specific to timeline authoring.

Authoring tools for journalists: narrative visualization authoring environments such as Ellipsis [54] and VisJockey [32] specifically target journalists. With these tools, journalists can compose narrative sequences of common visualizations depicting structured quantitative data; visual event timelines are not explicitly supported. Narratives authored with VisJockey [32] further allow readers to trigger visualization transitions with inline links in an accompanying text article, similar to the linking between the *New York Times*’ interactive timelines and corresponding sections of their accompanying articles. TimeLineCurator also relies on a linking between visualization elements and corresponding sections of a text document, but these links are established via natural language processing, whereas with VisJockey, these links are established manually by the author.

2.2 Timeline Visualizations from Structured Event Data

Assuming the data is already available in a structured form, there are several tools for generating timelines; some of these target specific application domains, while others are domain-agnostic.

Tools for timeline analysis: Though we focus primarily on timelines as a presentation tool, timeline visualizations are also often used for data analysis. TimeSlice [75] is a domain-agnostic analysis tool that affords the faceted browsing of timelines containing many events; these timelines are generated from structured event data. In the medical domain, LifeLines [47] and its descendants are also used for analysis, wherein an analyst can summarize and compare patient treatment timelines comprised of event types specific to the treatment context; these events are recorded via manual data entry by medical staff. Law enforcement tools such as Criminal Activities Network [9] are used for data analysis such as identifying crime patterns and discovering criminal associations, and are once again suitable only for structured domain-specific data. Social media analysts also use timelines for detecting events, trends, and anomalies, relying on structured social media data [7]. TimeLineCurator does not require structured event data and is portable across application domains.

News timelines: In an ephemeral online news environment, timelines are a popular way to convey an evolving story or to provide context. For example, Google News Timeline [21] automatically aggregates news stories from several thousand news sources and organizes them chronologically, while Evolutionary Timeline Summarization [74] generates timelines based on a user query and identifies the “relevance, coverage, coherence, and diversity” of that query inside many time-stamped articles. However, both of these approaches return

lists of events rather than visual timelines. Moreover, they treat an entire document as a single entity characterized by the document creation time; finer-grained temporal information from within the document is ignored.

Timeline authoring tools: Many simple and accessible timeline authoring tools exist. Examples include TimeRime [28], Dipity [12], Tiki-Toki [67], and Timeglider [40]. Some of these tools allow an author to add single events to an initially empty timeline one at a time, while others provide the ability to connect to RSS, Twitter, or other services that provide structured time-stamped data. Some of these tools are easy to use, but not at all customizable.

The customizable tools most relevant to our current work are SIMILE’s Timeline [29], ProPublicas’s TimelineSetter [48], WNYC’s Vertical Timeline [3], and TimelineJS [43] from the Northwestern University Knight Lab. These tools require structured event data as input; they generate timelines that can be embedded in websites. Advanced users can also make changes to the underlying code and adjust it to suit their needs. However, the author must first assemble and format a spreadsheet, JSON dataset, or a correctly-formatted CSV file containing event data. TimelineJS [43] is perhaps the most widely-used timeline authoring tool used in newsrooms today. The timeline creation process is straightforward: beginning with a Google Spreadsheet template, an author can fill in this spreadsheet with events, each of which requires a date or date span, a title, a description of the event, and, optionally, a link to an image, video, or other form of embeddable media. Publishing the spreadsheet generates a visual timeline automatically. We compare the experience of assembling and generating timelines using TimelineJS to that of TimeLineCurator in Section 6.1.

2.3 Extracting Time Expressions from Unstructured Text

The relevant Natural Language Processing (NLP) technique for extracting temporal information from unstructured text is *entity extraction*: identifying words or phrases inside unstructured text that represent names, locations, organizations, and dates. In particular, we focus on dates. The TimeML specification language for temporal information extraction [49] defines how to annotate events and temporal expressions inside unstructured text. It became the international standard in 2009 (ISO-TimeML) and is used by most current approaches.

Syntax-based recognition: Environments such as Tango [63] and TARSQI (Temporal Awareness and Reasoning Systems for Question Interpretation) [64] offer environments that automatically add TimeML markup to news articles. Temporal entity extraction is typically accomplished with hand-engineered deterministic rules that use regular expressions and pattern interpretation to detect signal words referring to anything temporal. Further improvements to these *recognition* approaches enable *normalization* of the recognized temporal expressions with respect to a Document Creation Time (DCT). For instance, the value of *yesterday* can be resolved to one day before the DCT. Examples include TempEx Tagger [37], SUTime [8], Heidelberg [58], and TERNIP [44]. TimeLineCurator uses the Python-based TERNIP system in its natural language processing pipeline. TERNIP uses the TARSQI extraction engine [64] for recognition; TERNIP also normalizes temporal expressions using a rule engine.

Context-dependent semantics: Approaches that consider only the *syntax* of entities ignores the surrounding context and can lead to misinterpretation or ambiguities. Newer approaches that incorporate machine learning use context-dependent *semantic* parsing for entity extraction; examples include learning contextual rules from question-answer pairs [31] or the use of various forms of weak supervision [2]. In contrast to these general-purpose systems, UWTime [33] is the first context-dependent model for semantic parsing that handles the special case of temporal expressions, where the additional step of normalization is required. Using the combination of hand-engineered and trained rules, it considers the tense of a governing verb to determine if the temporal expression refers to the future or the past, and it determines if a four-digit number refers to a year depending on the context. Incorporating the Java-based UWTime system into TimeLineCurator as an alternative to TERNIP would be interesting future work.

2.4 Visualizations from Unstructured Text

TimeLineCurator brings together visual timeline authoring with natural language processing. This section discusses previous projects that similarly combine visualization with natural language processing.

Topic discovery and analysis: Thematic analysis of many text documents is a popular area of research. Tools such as Serendip [1] leverage natural language processing to permit thematic analysis for documents at different scales, from individual passages to documents to entire corpora. Meanwhile, a number of tools [14, 15, 16, 26, 34, 35] extract topics and keywords while also considering each document’s creation time, allowing the analyst to observe topic changes over time. These tools do not extract temporal information in the unstructured text of documents; rather, they use bag-of-words models or more complex algorithms to determine the importance of words, word combinations, or topics. Furthermore, these tools are intended for data analysis rather than authoring or presentation.

Entity extraction and visual analytics: Visual analytics systems such as Jigsaw [22, 57] integrate entity extraction with visualization to show detected entities such as dates from unstructured text documents in several ways. However, the use of Jigsaw entails a high learning curve [23, 30], requires desktop installation, and is again intended for data analysis rather than presentation.

Visualizing Wikipedia articles: date entity extraction has also been applied to the generation of timeline and topic visualizations based on Wikipedia articles [62, 45]. For example, LensingWikipedia attempts to visualize human history through Wikipedia’s annual event summary pages over the last 2000 years. It extracts temporal and spatial information to find out “who did what to whom, when, and where” [62]. It is a discovery environment restricted to those specific pages; users cannot insert their own data and there is no support for authoring.

Date entity extraction is more accessible in TimeLineCurator than in previous work, since our tool is browser-based, is intended for fast timeline authoring rather than data analysis, and can ingest any unstructured text.

3 PROCESS

TimeLineCurator was created through an iterative refinement process with multiple rounds of requirements gathering, designing, prototyping, and deployment, following standard practice in visualization. TimeLineCurator is an authoring system that targets a broad set of user communities, rather than a very focused set of target users as in a typical visualization design study [56]. We identified journalists as one obvious potential user community, but also solicited feedback from other communities throughout this design cycle.

3.1 Initial Requirements and Prototyping

Our initial requirements gathering was primarily based on the first author’s experience working in the graphics department in a major German newspaper, and our assessment of existing systems as discussed in Section 2. We quickly built an initial prototype in order to test our ideas, and steadily refined it based on feedback from potential users.

3.2 Deployment and Collecting Community Feedback

We first demonstrated an early version of TimeLineCurator to a journalism professor and a policy researcher; both had a need to present timeline data to readers and were familiar with TimelineJS. Shortly after, we deployed TimeLineCurator online³ and publicized it locally to faculty at the University of British Columbia Journalism School and to members of a local Hacks/Hackers Meetup group. We also publicized it more broadly to our extended professional network via email and Twitter. Interest in TimeLineCurator then grew following publicity at the 2015 NICAR conference for computer-assisted reporting [11, 24, 36, 73]. We were also able to gather feedback and information about use cases from several prospective timeline authors

³<http://www.cs.ubc.ca/group/infovis/software/TimeLineCurator/>

who contacted us with feature requests and questions. Section 6.5 discusses the full set of use cases that we learned about from all of these prospective constituencies. In addition to these direct contacts, we also could indirectly gauge interest based on increasing traffic to the TimeLineCurator site, with several thousand visits and many hundreds of unique users trying out the freely available tool.

3.3 Identifying TimeLineJS Limitations

TimelineJS [43] is perhaps the most popular tool for creating and presenting interactive timelines online. Despite its popularity, we identified several limitations by gathering feedback from several current users of TimelineJS who we came into contact with as part of the deployment process described above. We refer to the authoring process with TimelineJS as *structured creation*, which involves a significant amount of human time and effort while extracting and formatting structured event data. We discuss this process further in Section 4, and we compare the experience of authoring timelines using TimelineJS to that of TimeLineCurator in Section 6.1.

We identified several drawbacks to how TimelineJS presents a timeline to the reader (as shown in Figure 5f), which informed the design of presentation-ready timelines exported from TimeLineCurator, described in Section 5.6. A TimelineJS widget presents a zoomable and scrollable interactive timeline that invites the reader to progress through the timeline with linear navigation from one event to another, beginning with the first event in the timeline. TimelineJS does not provide an initial overview of the temporal distribution of events: on opening, the horizontal timeline view is centered on a specific date and only a small region is visible. By default this first date corresponds to the earliest event in the timeline; while the user can explicitly navigate by zooming out, it is not possible to simply set the start view to show the entire timeline. Moreover, clutter and occlusion is a significant issue: glyphs representing individual events are displayed along a narrow axis spanning the bottom of the timeline, and the event labels placed above this axis overlap in regions where multiple events occur.

4 TIMELINE AUTHORING MODEL

In this section, we introduce several timeline authoring tasks, and we compare how these tasks are accomplished using existing *manual drawing* and *structured creation* approaches to how these tasks are carried out using TimeLineCurator. These differences are summarized in Table 1. We also define several goals that a timeline authoring system should address.

	Browse	Extract	Format	Show	Update
Manual Drawing	high	high	none	high	high
Structured Creation	high	high	high	low	low
TimeLineCurator	low	none	none	low	low

Table 1: Comparing the human time and effort required to perform the five tasks encompassed by our Timeline Authoring Model with previous approaches and with TimeLineCurator.

4.1 Timeline Authoring Tasks

The timeline generation process begins with **browsing** source documents, where the author looks for event information. *Browsing* is defined as a form of search in which the locations of potential search targets are known, but the identity of the search targets may not be known a priori [6]. During this period, the author might identify and **extract** events by highlighting or annotating relevant passages in documents, adding events to a list, sketching a timeline on paper or with Post-it notes on a wall. To transfer these events to a digital medium, the author must decide how to **format** the events, and determine how to **show** or encode them. Finally, in some instances, an author **updates** the timeline: events may be added, edited, or deleted to reflect new information, such as in the case of an evolving news story.

Manual drawing: When satisfied with the results of the *browsing* and *extracting* process, the author can manually draw a timeline us-

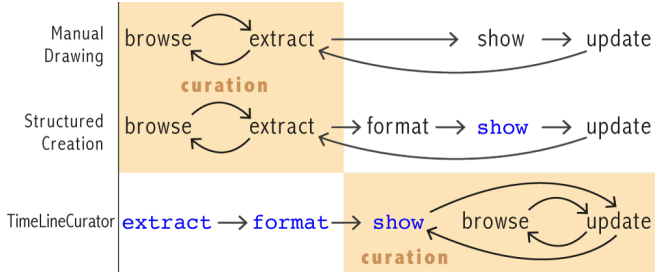


Fig. 3: Comparing the sequence of timeline authoring tasks: timeline curation (indicated by the orange shaded areas) occurs later with TimeLineCurator. Tasks in blue fixed-width font are automated; all other tasks are performed by the author.

ing an illustration program: event *formatting* is not required. *Showing* the timeline can be very time-consuming. While standard graphic design tools can be used for building a temporal scaffold, events must be added to the timeline manually one at a time. A positive feature of this approach is that the author has a significant amount of creative license when performing this task. As a result, manual drawing can lead to intricate and engrossing timelines, such as xkcd’s “Movie Narrative Charts” [41]. However, the manual illustration approach to timeline generation is clearly inappropriate for evolving stories, as *updating* the timeline with additional events may require rescaling the whole timeline, or readjusting and redrawing significant portions of it. The result of the manual drawing process is most likely a static graphic, used for print products or as a graphical element in a digital medium.

Structured creation: Several alternatives to manual timeline illustration exist. However, these approaches produce timelines that cannot be easily customized, or require a programming ability beyond a typical author’s skill set. Structured timeline generation tools like TimelineSetter [48] and TimelineJS [43] require that event items are *formatted* in a structured table of dates with event descriptions. Provided with structured event data, *showing* the timeline is performed quickly, as timeline rendering is performed by the program or tool. *Updating* the timeline is also straightforward, as the author only needs to add more formatted events to the structured event dataset and the timeline will be updated automatically. For evolving news stories, structured creation is a much more viable approach than manual drawing.

4.2 Requirements for a Visual Timeline Authoring System

Automate extraction and formatting: A new approach to timeline authoring should strive to reduce or eliminate the need to manually *extract* and *format* event data. Randall Munroe, the author of xkcd, has remarked that he drew his “Movie Narrative” timelines [41] manually not out of preference, but because no existing tool could automatically extract event timelines from movie scripts [42]; automatic generation of these timeline visualizations is now possible [60], however this approach requires structured event data.

Accessible: Recent advances in natural language processing allow for the extraction and formatting of temporal references from unstructured text [44]. However, natural language processing packages and tools require installation and programming ability; furthermore, they do not visualize their results. A timeline authoring tool should therefore be *accessible*: it should be browser-based to avoid the need to install any software, and it should provide a flexible means to import unstructured text. It should also be easy to learn and use, appealing to authors without a highly developed technical skill set; in other words, it should require no programming.

Visual feedback during curation: A timeline authoring tool should provide intermediate visual feedback when *browsing*, *showing*, and *updating* event data, as indicated in Figure 3. When programming a timeline from scratch, or when using an existing timeline authoring tool such as TimelineJS [43] or others mentioned in Section 2.2, there is no intermediate visual feedback during the authoring process; the hazards of delayed feedback have been noted previously [65]. Without

intermediate visual support, it is difficult to determine whether creating a timeline is worth the effort.

Accelerate process: Finally, an ideal tool should accelerate the authoring process: an author should be able to curate events from suitable documents in minutes, and rule out unsuitable documents in seconds.

Summary: Our new tool, TimeLineCurator, was developed to overcome these difficulties. With manual drawing and structured creation approaches, timeline curation was accomplished by iterating between the *browse* and *extract* tasks; with TimeLineCurator, timeline curation is a visual process, swapping the order of the *browse* and *show* tasks while automating the *extract* and *format* tasks, as indicated in Figure 3. TimeLineCurator also explicitly supports the *browsing* of events from multiple documents simultaneously, allowing, for instance, the author to compare multiple sources discussing the same subject or comparing subjects that do not obviously relate but might have influenced one another. Finally, updating a timeline with TimeLineCurator is easy, and does not require editing the source documents.

4.3 Architectural Instantiation

We now discuss the concrete instantiation of this authoring model through the data processing pipeline of TimeLineCurator, as illustrated in Figure 4.

An author begins with an empty timeline, and can populate the timeline by uploading unstructured document text. TimeLineCurator **extracts** events from this text using natural language processing techniques; it first *recognizes* absolute temporal references such as “October 30, 2014” or “2010” using the Python library TERNIP [44], which is based on a large set of regular expressions. In addition to single dates, durations are also extracted, such as the reference “from 2 Sept 2014 to 31 Mar 2015”. TERNIP also *normalizes* all relative temporal references such as “yesterday”, “since Tuesday” or “next year”, giving them a value relative to the document creation time. When this normalization does not result in a concrete date or span, the expression is categorized as a *vague date* and assigned the value “????”. In many cases these are genuinely non-specific temporal expression like a duration (“99 days”) or an interval (“monthly”) that do not belong on a timeline; in other cases, these are expressions that TERNIP failed to extract correctly but can be curated by the author to a meaningful date or span. Next, TimeLineCurator **formats** the set of extracted dates into structured JSON, which also includes the sentence containing each temporal reference and its location within the source document.

Given this structured format, TimeLineCurator then **shows** the timeline, encoding individual events as well as event spans along the timeline axis; vague dates are not shown on the timeline, but are presented to the author separately. At this point, the author can **update** the timeline; she can add, delete, merge, or edit events until satisfied, including events associated with vague dates. This entire process can be repeated any number of times with additional unstructured text. When ready to **present**, the author can export the timeline, and at any time, the author can save the state of an edited timeline to resume editing later.

Implementation: The back end of the pipeline that provides the data handling for the *extract* and *format* tasks is implemented in Python. The front end that supports the *show*, *curate*, *update*, and *present* tasks is implemented in D3.js [4] and AngularJS [20]. The system is hosted on the Heroku cloud application platform [27], which runs the Python code on the server side. The micro web application framework Flask [18] links together the server-side Python script with the client-side HTML, JavaScript and CSS code.

5 INTERFACE AND DESIGN RATIONALE

TimeLineCurator is a web-based single-page multiple-view authoring application that can be used to produce and export embeddable visual timeline widgets. The interface has four panels coordinated through linked highlighting and navigation, depicted in Figures 1 and 5: the Timeline Visualization at the top, the List View on the lower left, the Document View in the lower middle, and the Control Panel on the

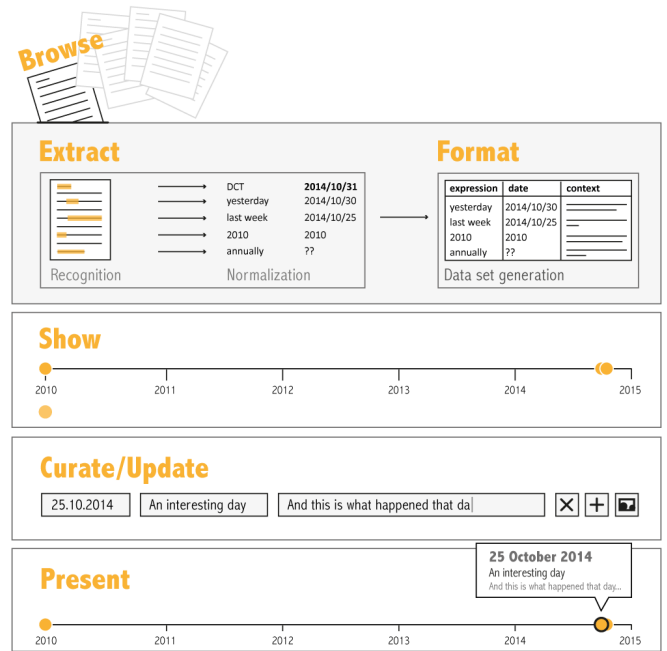


Fig. 4: Processing pipeline for TimeLineCurator.

lower right. These panels are initially empty, as in Figure 5a. Figure 5b shows the dialog window where the author pastes unstructured text and sets the date corresponding to “today” in the document; if left unspecified, the current date is used as the document creation time. The initial set of automatically extracted events then populates the interface, as shown in Figure 5c.

5.1 Timeline Visualization View

The Timeline Visualization view provides an information-dense global view with no occlusion and minimal navigation, an approach similar in spirit to the previous work of Variant View [17]. Figures 1 and 5d show examples with many stacked and dodged glyphs, providing an overview where the temporal distribution of events is visible even in densely populated areas of the timeline. There is no zooming or horizontal scrolling: the size of the discrete events is fixed and the entire horizontal axis is shown at all times. As a result, the author always has an overview of the full time range. Vertical scrollbars appear when the events overflow the available vertical space, as a backstop solution to ensure that arbitrarily dense time distributions can be curated. Typically, the final curated version of the timeline exported for presentation does not require vertical scrolling.

The horizontal time axis is scaled automatically to the range of time encompassed by the active events, and will update if any addition, removal, or editing of an event changes that range. The document creation time is indicated on the axis as a vertical dashed line labeled ‘today’.

An event corresponding to a single date is encoded as a circle ●, while an event span with a beginning date and an end date is encoded as a connecting bar of variable length flanked by triangles ▶◀. Vague dates corresponding to possible events, based on temporal references like “the day after” or “summer” are encoded as a square ■ and shown outside the horizontal range of the timeline axis, in the upper right corner of this view, as in Figure 5c. Events are coloured by hue according to the six possible *tracks* (●●●●●●), and this base univariate colour palette was selected from ColorBrewer [25]. Glyphs corresponding to events that have already been edited are more saturated than those corresponding to unedited events (● vs. ●), for a bivariate palette with 12 colors in total. By default, events from each successive document text pasted into TimeLineCurator are assigned to a different track, but the author can override this behaviour by explicitly selecting a colour track when loading a new document (Figure 5b). Having multiple

colour tracks can assist the author in comparing timelines from multiple documents.

5.2 List View

Fast scanning across many events is supported through the List View. Multiple sort options support browsing and linear navigation according to multiple different criteria. This view lists all of the events and vague dates; each list entry is comprised of an event glyph, a date, and an event title. Initially, the first five words of the sentence from which the event was extracted is assigned as the event’s title.

Events can be sorted according to the location within each document, by event type (●, >←, or ■), by event status (● or ● with a checkmark, where the checkmark in addition to saturation redundantly encodes that an event has been edited), by track (●●●●●), by date, or by event title. Events deleted from the timeline remain in this list, and their deleted status is represented by crossing out the date and title text, changing the row background colour to a darker grey, and reducing the glyph’s alpha value.

5.3 Document View

The Document View supports the growing trend in journalism of linking original source documents to online news media, as with tools such as DocumentCloud [13], following the demands for more transparency and involvement of the readers [51]. In addition to supporting the curation process for authors, the Document View allows readers of the curated timeline to see the relationships between events and corresponding sentences in source documents. This panel displays original unstructured document text, where all recognized temporal references are highlighted in orange. The control bar at the top is coloured according to the assigned track and allows the author to toggle between which document is shown, while the + button adds a new document.

5.4 Control Panel

The Control Panel on the bottom right allows the author to edit an event selected in any of the other three views, as shown in Figure 5d. She can modify the date of an event, turn a single event into a span, or vice versa; she can also edit the title and description for an event by clicking on either of these fields. By default, the event description is the sentence from which the event was extracted. When a vague date is given a concrete date, its corresponding glyph is moved to its appropriate place in the timeline visualization and becomes more saturated. The author can also delete the event, reassign the event to another colour track, or add media such as image to it. Finally, the author can add new single events manually.

5.5 View Coordination and Navigation

Event selection is propagated as linked highlighting across all views, with selected events highlighted in black, as shown in Figure 1. In the Document View, events can be selected by clicking on any sentence that includes a temporal reference. Navigation is also linked across the views; when clicking on an event in the Timeline Visualization View, the List View and Document View will scroll to the corresponding sections of the list and document, respectively. Keyboard arrow keys and paging buttons in the Control Panel will iterate through events using the current sort order of the List View.

5.6 Presentation and Export

When the author is satisfied with her curated timeline, she can export the timeline so that it can be shared online. Vague events are not exported. We provide two ways for an author to present their timeline. The TimeLineCurator presentation view is a read-only version very similar to the editing interface, as shown in Figure 5e. The timeline is hosted on a shareable unique URL. Coordinated navigation and selection across the views remain the same; the Control Panel is replaced with an Event Details panel, in which any image media associated with an event is shown.

A timeline can also be exported as a TimelineJS [43] widget that can be downloaded and embedded on the author’s site, as shown in

Figure 5f. We provide TimelineJS export capability because of its popularity, despite the drawbacks discussed in Section 3.3.

6 RESULTS

We evaluate TimeLineCurator in several ways. We benchmark its correctness in terms of text extraction quality. We also compare its user experience to the structured creation approach. We present instances where TimeLineCurator is used to rule out documents that contain little or no interesting temporal information, and we present examples of curated timelines and provide before and after images to show the changes made in the curation process. Finally, we discuss preliminary feedback from target users.

6.1 Extraction Error Benchmark

Our first benchmark is primarily intended to gauge the quality of the automatic extraction compared to manual extraction of temporal information from unstructured text, and is narrow in scope. The automated extraction process involved uploading unstructured document text into TimeLineCurator and systematically checking every extracted event to verify that it was recognized correctly; we also determined if incorrectly extracted dates required editing or deletion. The manual extraction process involved reading the original document text and performing manual data entry, copying all temporal references and their surrounding sentences into a spreadsheet in the structured format required for TimelineJS input. In this initial benchmark, the author’s judgement was restricted to simply judging whether the expression correctly indicated a single event or a date range. No judgement was used about whether an event was interesting enough to merit inclusion on the timeline, and event titles or descriptions were not edited.

The benchmark datasets were three Wikipedia articles⁴ and two recent news articles⁵; the two news articles were added to a single timeline. Figure 6 shows the quality assessments of TimeLineCurator’s temporal expression extraction compared against the gold standard of manual extraction. These results indicate that most of the dates were identified correctly (an average of 65%), though some needed curation via editing or deletion (an average of 29%), and a small fraction were not extracted (an average of 6%). These results confirm that automatic extraction is a good match with our expectations: the true positive rate is reasonable but far from perfect, and the false negative rate is low. Thus, we deem that scaffolded curation is a viable approach to timeline authoring.

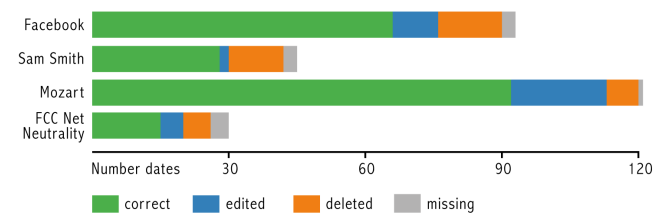
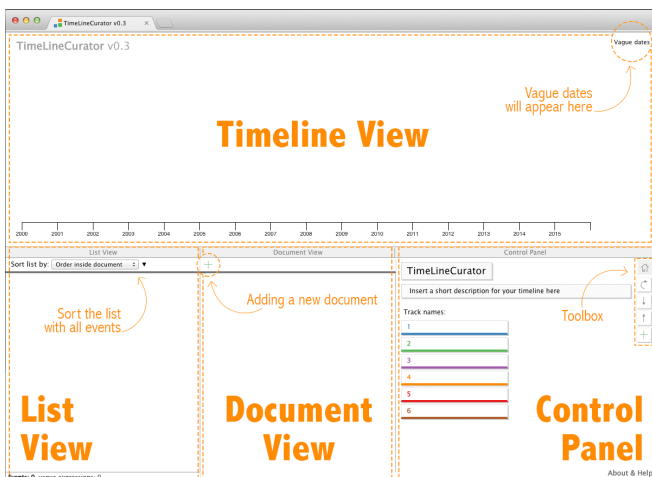


Fig. 6: The results of the benchmark tests, which compares the gold standard manual creation of an event set with the automated event extraction of TimeLineCurator.

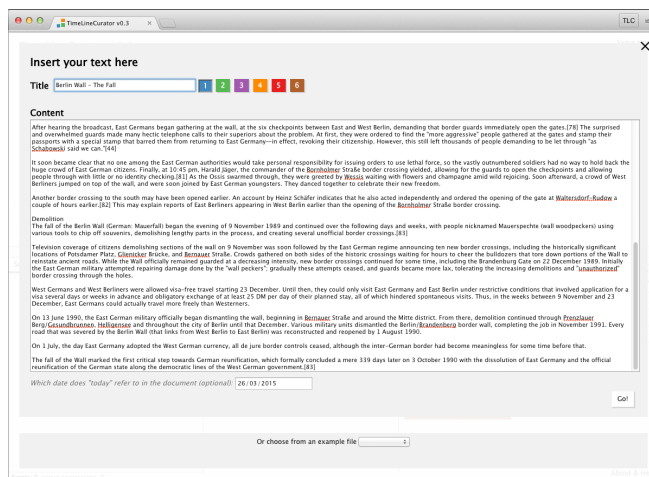
This benchmark also yielded qualitative insights on the kinds of expressions that were incorrectly extracted. Incorrectly identified dates often were time spans, which can be expressed in many different ways in prose. For example, in “*The family again went to Vienna in late 1767 and remained there until December 1768*” [71], two separate dates were automatically extracted, but the author combined them into one time span during manual curation. Another reason for incorrectly extracted events were temporal expressions that implicitly refer to a previously named date rather than explicitly containing a year. The natural language processing misses these expressions because it only considers the immediate context and incorrectly ties them to the document’s creation date. The result is that historical texts incorrectly have

⁴The history of Facebook [69], the biography of pop musician Sam Smith [70], and the biography W. A. Mozart [71].

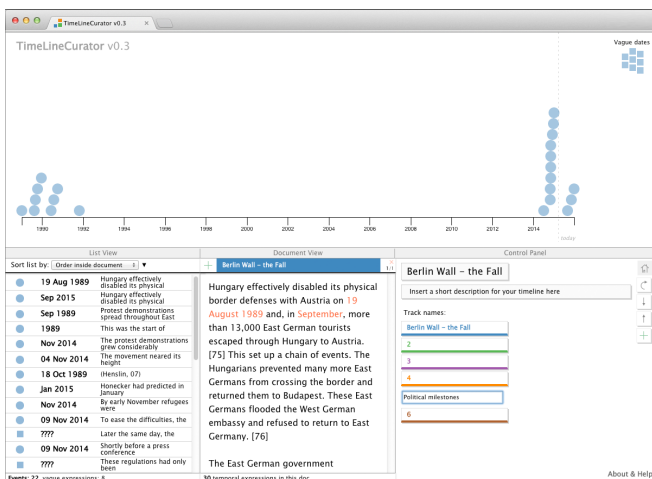
⁵Both pertained to the topic of net neutrality [38, 53].



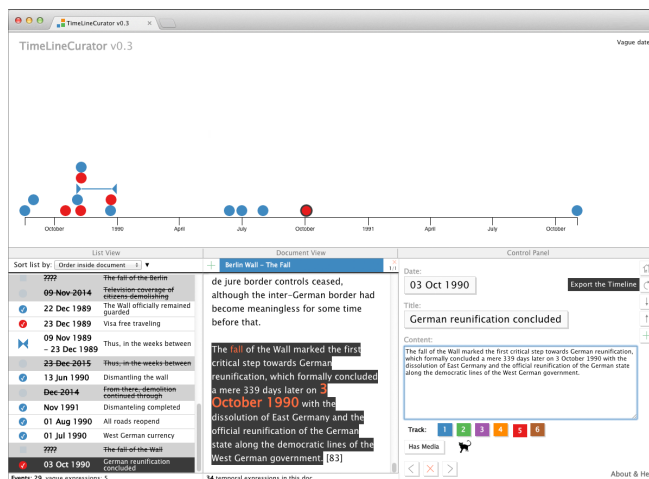
(a) Initially, the timeline is empty. Annotations in orange demarcate the four main views: Timeline View, List View, Document View, and Control Panel.



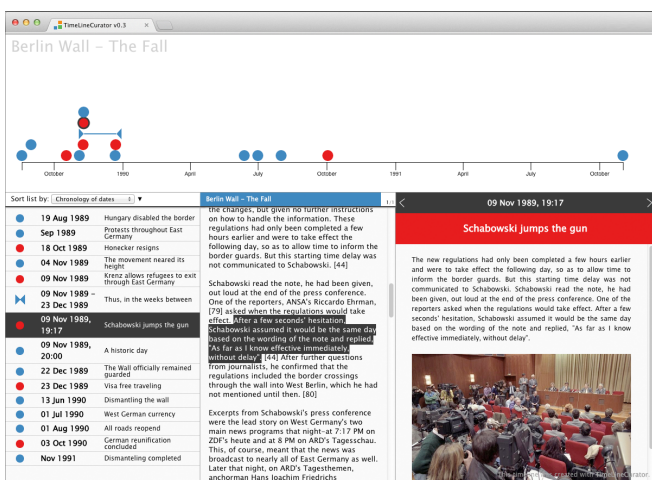
(b) Unstructured text is added via a popup dialog. Optionally, the document creation time can be specified below the input field.



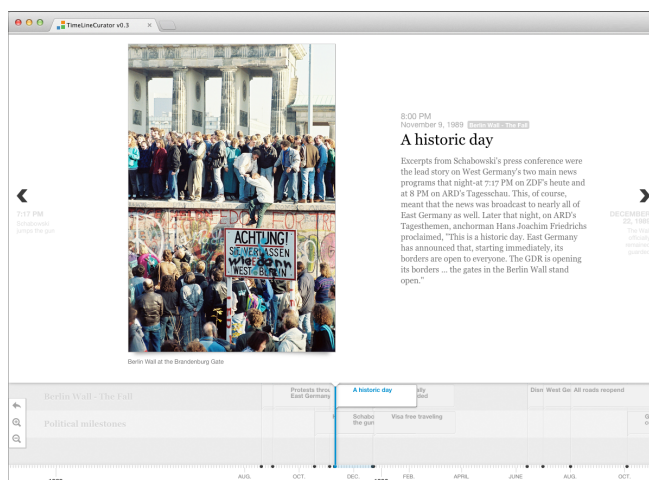
(c) A timeline immediately after importing text, with many vague and uncurated dates. General timeline information can be modified when no event is selected.



(d) Event dates, title, and description can be adjusted when an event is selected, it can also be assigned to another track, enriched with images, or deleted.



(e) The curated timeline can be exported; the presentation view is a read-only version of the editing interface.



(f) The curated timeline can also be exported using the open-source tool TimeLineJS [43].

Fig. 5: A walkthrough of the TimeLineCurator curation process. We demonstrate this process using unstructured document text from the "The Fall" section of the Wikipedia article on the Berlin Wall [68]. The resulting timeline can be accessed at <http://goo.gl/SU1faP>.

many dates assigned to “today” despite only containing dates from the distant past. Another source of false positives are temporal expressions that are used as names and do not refer to a specific event, such as Taylor Swift’s album title “1989” or the TV Show “Last Week Tonight”.

Events that were missed by the automatic extraction were often those which referred to another event, such as “six days after the site launched” or possessive statements, such as “last week’s vote”. In some cases these were extracted as vague dates, and in others they were missed completely. Currently, the year recognition is limited to Anno Domini years with four digits; references such as “13,000-12,000 BC” are not handled.

This benchmark was conducted by one of the authors who was very familiar with the system. We chose this approach because this benchmark scenario required a meticulous comparison between automatic and manual extraction that does not occur during the actual timeline authoring process. Moreover, this benchmark scenario focused solely on the verification and correction of event dates and did not involve any editorial judgment, such as deciding which events to include in the timeline and how to embellish these dates with interesting event titles and descriptions. However, we conjecture that the complete curation process with TimeLineCurator is easier and preferable to the tedious manual structured creation approach. To address this conjecture, we conducted a second benchmark with a more realistic approximation of the authoring process and an arms-length group of participants.

6.2 User Experience Comparison

The second form of evaluation involved the observation of behaviour that more closely approximates a real timeline authoring process. We recruited six arms-length participants from our department who were unaffiliated with the project and asked them to create coherent timelines. We provided them with short text articles and asked them to make editorial judgements about each event they encountered; they were also asked to curate event titles. Each author curated two timelines: first, one using manual structured data entry as required by TimeLineJS [43] and second, one using TimeLineCurator. They were directed to curate the timeline until they were fully satisfied and felt that it was ready to be exported. All participants strongly preferred TimeLineCurator’s visual authoring environment to the structured data entry required by TimelineJS, and they found working with TimeLineCurator to be highly engaging. Every user encountered at least some difficulties with the structured editing approach despite having a strong technical background. One participant even abandoned the structured editing approach completely after a few minutes because it was so tedious. The curation time from start to finish across participants is not directly comparable because the the scope of the editorial judgment performed during the curation process varied considerably between them. This informal comparison of user experience provided encouraging qualitative evidence that the design goals of our authoring system were met.

6.3 Speculative Browsing

The ability to quickly rule out unsuitable documents using TimeLineCurator is a major strength of the system. Figure 7 shows three examples of timelines where the author was able to quickly decide that the document is not a suitable source for an engaging timeline. This decision was made in under 15 seconds in all of these cases, with most of that time devoted to copying, pasting, and waiting for extraction; once the timeline is visible, the decision is essentially immediate.

6.4 Curated Examples

We generated and curated many timelines during the course of this project, including the Berlin Wall timeline documented in Figure 5 and the timeline of W. A. Mozart’s biography shown in Figure 8. We also created a gallery of curated timelines⁶, exported with both TimelineJS and with TimeLineCurator’s presentation view.

⁶<http://cs.ubc.ca/group/infovis/software/TimeLineCurator/#examples>

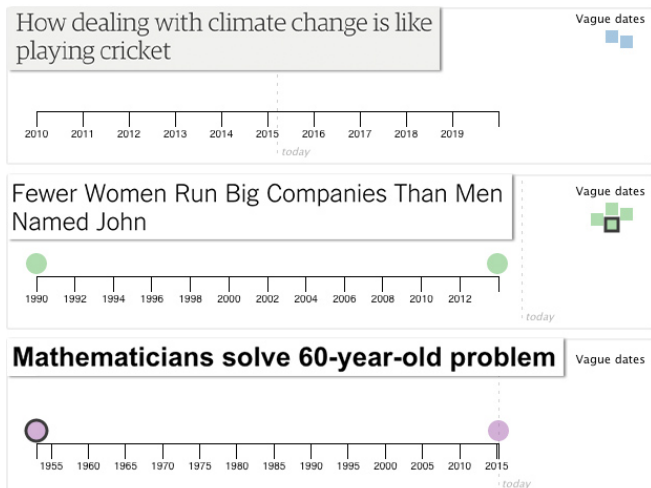


Fig. 7: Timelines extracted from two news articles [52, 72] and a report from a science press release site [46]. All three do not contain much temporal information and thus can quickly be ruled out as a suitable basis for an interesting timeline.

6.5 Use Cases

In addition to evaluation conducted in our lab where we the usage scenario was specified a priori, we also gathered feedback based on real use cases from current and prospective timeline authors from several user communities including journalism.

Solicited potential users: We conducted semi-structured interviews with eight people: seven journalists and one policy researcher. Four of these individuals already had experience creating interactive timelines and provided us with feedback about the strengths and limitations of currently available timeline tools. Two of these individuals had pre-existing plans to use a timeline authoring tool in an upcoming project.

When we presented TimeLineCurator to these individuals and asked them to try it out, their reaction was very positive and they remarked that it was very easy to use. They enjoyed the approach of extracting temporal event data from unstructured document text, and that they no longer had to start with an empty spreadsheet and add every event manually one at a time. The immediate visual feedback during the authoring process was also highly appreciated.

One journalist said: “For the less geeky journalists who might be scared of timelines, this is a brilliant super-easy way to see what it might look like” and that TimeLineCurator might be a good way to “break the barrier between the artistic writer and the data journalist”.

We asked these individuals to speculate about possible kinds of stories that might benefit from accompanying timelines: these included the unfolding of political scandals, how amendment bills proceed in government, and biographies. They also proposed several use cases that we had not previously considered, such as using TimeLineCurator for data analysis rather than timeline authoring for presentation. One idea involved using TimeLineCurator with court documents when reporting on a trial to better understand the context of a criminal or legal case. Another possible use case is fact-checking during investigative analysis. Typically, details are verified through two reliable sources before publication. A journalist that we spoke to imagined that TimeLineCurator might accelerate fact-checking for temporal events and finding mismatches between sources. Finally, a third use case involved using TimeLineCurator to prepare for interviews, to quickly catch up the subject’s biography or background.

Unsolicited current users: In contrast to the ideas above that are potential use cases for prospective users of TimeLineCurator, we can also report on use cases from people in different communities who already used TimeLineCurator for their own projects after it was deployed and publicized. One author was a digital humanities researcher who created a timeline to see the historical development of deaf churches in

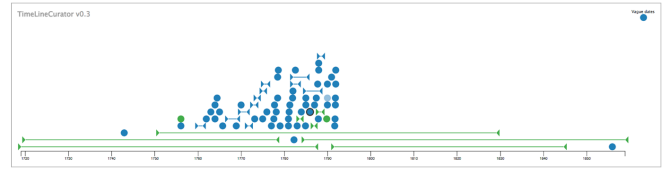
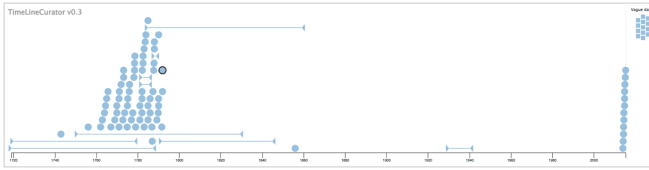


Fig. 8: A timeline of composer W. A. Mozart's biography [71], both before and after curation. The resulting timeline can be accessed at <http://goo.gl/2JikND>.

England. Another author was a user experience professional who created a timeline to accompany the profile of his company.

7 DISCUSSION & FUTURE WORK

TimeLineCurator offers a new way of exploring the temporal structure of a document in order to make the process of creating timelines enjoyable rather than arduous. We designed the system under the assumption that entity extraction through natural language processing is decent but not perfect, and can serve to support human-in-the-loop curation. Moreover, even if the extraction were perfect and all date events and spans were extracted correctly, there are still many subtasks involved in timeline curation that will need nuanced human judgement for quite some time. In addition to the core question of selecting which events are interesting to tell a particular story, there are many editorial choices in writing the title and description text that accompanies the event. Deciding whether to add media and finding relevant imagery is also a very nuanced question that benefits from human judgement, at least in the near future. Although we originally designed it to help authors create presentations, it may well serve for analysis tasks such as fact-checking, which also involves the exercise of human judgement.

The vast majority of feedback we received from interviews and from the broader community approved the general idea of TimeLineCurator. Many requests for improvement pertained to the automated event extraction. Our design goal was to use existing tools that known to be imperfect, but it would be both useful and straightforward to incorporate newer tools such as context-dependent semantics as toolkits become more widely available [33]. Also, moving to a natural language processing toolkit that supports multiple languages would allow the use of TimeLineCurator outside of English-speaking countries.

Integrating TimeLineCurator into Overview [5], an open-source system for investigative journalism that supports the analysis of large collections of documents, would open up further use cases for both analysis and presentation. Overview integration would also provide DocumentCloud [13] support for accessing online document repositories, for further utility to the journalism community.

8 CONCLUSION

We presented TimeLineCurator, a visual timeline authoring system that recognizes temporal expressions within unstructured document text. It accelerates the event-extraction process and fulfills two broader tasks. First, it enables authors to create polished timelines from interesting documents within only a few minutes. Second, it enables speculative browsing, which lets authors eliminate temporally uninteresting documents from consideration within seconds. TimeLineCurator can be used by a broad community of authors including those without a strong technical background, because it is easily accessible, has a simple user interface, and does not require any programming. It lowers the access barrier for timeline creation for a broad set of potential authors, including journalists, who would like to work visually rather than via manual data entry into spreadsheets. TimeLineCurator can directly create two forms of curated timeline: the popular TimelineJS [43] and our own presentation format that provides an information-dense overview. Moreover, the resulting set of curated events can be exported as a structured dataset, opening up further possibilities beyond these two currently-supported presentation formats. Interviews and community feedback provided evidence that the TimeLineCurator approach of scaffolded curation built on top of

imperfect automatic entity extraction provides useful and appealing functionality in several application domains.

ACKNOWLEDGMENTS

We thank the journalists and students who provided feedback. Thanks to Francisco (Pax) Escalona for development assistance. Thanks to Chad Skelton and Nick Diakopoulos for publicizing TimeLineCurator within the journalism community. We also thank Michelle Borkin, Anamaria Crisan, Enamul Hoque, Sung-Hee Kim, and Narges Mahyar for their feedback on the paper.

REFERENCES

- [1] E. Alexander, J. Kohlmann, R. Valenza, M. Witmore, and M. Gleicher. Serendip: Topic model-driven visual exploration of text corpora. In *Proc. IEEE Conf. Visual Analytics Science and Technology (VAST)*, pages 173–182, 2014.
- [2] Y. Artzi and L. Zettlemoyer. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Trans. Assoc. Computational Linguistics*, 1:49–62, 2013.
- [3] Balance Media and WNYC / J. Keefe. Vertical Timeline. <http://github.com/jkeefe/Timeline>.
- [4] M. Bostock, V. Ogievetsky, and J. Heer. D³: Data-driven documents. *IEEE Trans. Visualization and Computer Graphics (Proc. InfoVis)*, 17(12):2301–2309, 2011.
- [5] M. Brehmer, S. Ingram, J. Stray, and T. Munzner. Overview: The design, adoption, and analysis of a visual document mining tool for investigative journalists. *IEEE Trans. Visualization and Computer Graphics (Proc. InfoVis)*, 20(12):2271–2280, 2014.
- [6] M. Brehmer and T. Munzner. A multi-level typology of abstract visualization tasks. *IEEE Trans. Visualization and Computer Graphics (Proc. InfoVis)*, 19(12):2376–2385, 2013.
- [7] J. Chae, D. Thom, H. Bosch, Y. Jang, R. Maciejewski, D. S. Ebert, and T. Ertl. Spatiotemporal social media analytics for abnormal event detection and examination using seasonal-trend decomposition. In *Proc. IEEE Conf. Visual Analytics Science and Technology (VAST)*, pages 143–152, 2012.
- [8] A. X. Chang and C. D. Manning. SUTime: A library for recognizing and normalizing time expressions. In *Proc. Intl. Conf. Language Resources and Evaluation (LREC)*, pages 3735–3740, 2012.
- [9] H. Chen, H. Atabakhsh, C. Tseng, B. Marshall, S. Kaza, S. Eggers, H. Gowda, A. Shah, T. Petersen, and C. Violette. Visualization in law enforcement. *Proc. Extended Abstracts ACM SIGCHI Conf. Human Factors in Computing Systems (CHI)*, pages 1268–1271, 2005.
- [10] J. DelViscio and D. Overbye. The Higgs, from theory to reality. *The New York Times*, Mar. 4, 2013. <http://goo.gl/Y8MaKC>.
- [11] N. Diakopoulos. From words to pictures: Text analysis and visualization, Mar. 5 2015. Presentation at NICAR 2015: <http://t.co/kfqbTBzj16>.
- [12] Dipity. <http://dipity.com/>.
- [13] DocumentCloud. <http://documentcloud.org/>.
- [14] W. Dou, X. Wang, R. Chang, and W. Ribarsky. ParallelTopics: A probabilistic approach to exploring document collections. In *Proc. IEEE Conf. Visual Analytics Science and Technology (VAST)*, pages 231–240, 2011.
- [15] W. Dou, X. Wang, D. Skau, W. Ribarsky, and M. X. Zhou. LeadLine: Interactive visual analysis of text data through event identification and exploration. In *Proc. IEEE Conf. Visual Analytics Science and Technology (VAST)*, pages 93–102, 2012.
- [16] W. Dou, L. Yu, X. Wang, Z. Ma, and W. Ribarsky. HierarchicalTopics: Visually exploring large text collections using topic hierarchies. *IEEE Trans. Visualization and Computer Graphics (Proc. VAST)*, 19(12):2002–2011, 2013.

- [17] J. A. Ferstay, C. B. Nielsen, and T. Munzner. Variant View: Visualizing sequence variants in their gene context. *Trans. IEEE Visualization and Computer Graphics (Proc. InfoVis)*, 19(12):2546–2555, 2013.
- [18] Flask Microframework for Python. <http://flask.pocoo.org/>.
- [19] M. Gidda. Edward Snowden and the NSA files – timeline. *The Guardian*, Aug. 21, 2013. <http://goo.gl/hdj2PY>.
- [20] Google, Inc. AngularJS. <https://angularjs.org/>.
- [21] Google News Timeline. <http://news.google.com/>.
- [22] C. Görg, Z. Liu, J. Kihm, J. Choo, H. Park, and J. T. Stasko. Combining computational analyses and interactive visualization for document exploration and sensemaking in Jigsaw. *IEEE Trans. Visualization and Computer Graphics (TVCG)*, 19(10):1646–1663, 2013.
- [23] C. Görg, Z. Liu, and J. Stasko. Reflections on the evolution of the Jigsaw visual analytics system. *Information Visualization*, 13:336–345, 2014.
- [24] L. Groeger. Making timelines, Mar. 2015. Blog post: <http://goo.gl/AIfGCu>.
- [25] M. Harrower and C. A. Brewer. Colorbrewer.org: An online tool for selecting colour schemes for maps. *Cartographic Journal*, 40(1):27–37, 2003.
- [26] S. Havre, E. Hetzler, P. Whitney, and L. Nowell. ThemeRiver: Visualizing thematic changes in large document collections. *IEEE Trans. Visualization and Computer Graphics (TVCG)*, 8(1):9–20, 2002.
- [27] Heroku Cloud Application Platform. <http://heroku.com/>.
- [28] Hoppinger BV. TimeRime. <http://timerime.com/>.
- [29] D. F. Huynh. SIMILE Widgets: Timeline. <http://simile-widgets.org/timeline/>.
- [30] Y. A. Kang and J. T. Stasko. Examining the use of a visual analytics system for sensemaking tasks: Case studies with domain experts. *IEEE Trans. Visualization and Computer Graphics (Proc. VAST)*, 18(12):2869–2878, 2012.
- [31] T. Kwiatkowski, E. Choi, Y. Artzi, and L. Zettlemoyer. Scaling semantic parsers with on-the-fly ontology matching. In *Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP)*, pages 1545–1556, 2013.
- [32] B. C. Kwon, F. Stoffel, D. Jäckle, B. Lee, and D. A. Keim. VisJockey: Enriching data stories through orchestrated visualization. In *Proc. Computation + Journalism Posters*, 2014.
- [33] K. Lee, Y. Artzi, J. Dodge, and L. Zettlemoyer. Context-dependent semantic parsing for time expressions. In *Proc. Conf. Assoc. Computational Linguistics (ACL)*, pages 1437–1447, 2014.
- [34] Y. Liu, S. Barlowe, Y. Feng, J. Yang, and M. Jiang. Evaluating exploratory visualization systems: A user study on how clustering-based visualization systems support information seeking from large document collections. *Information Visualization*, 12(1):25–43, 2013.
- [35] D. Luo, J. Yang, M. Krstajic, W. Ribarsky, and D. Keim. EventRiver: Visually exploring text collections with temporal references. *IEEE Trans. Visualization and Computer Graphics (TVCG)*, 18(1):93–105, 2012.
- [36] S. Machlis. Tools & tutorials from NICAR15, Mar. 2015. Blog post: <http://goo.gl/JHV0vJ>.
- [37] I. Mani and G. Wilson. Robust temporal processing of news. In *Proc. Conf. Assoc. Computational Linguistics (ACL)*, pages 69–76, 2000.
- [38] G. A. Manne. Opinion: The FCC’s net neutrality victory is anything but. *Wired*, Mar. 3, 2015. <http://goo.gl/wF5OJf>.
- [39] M. Martinez. Timeline: Leads in the hunt for Malaysia Airlines Flight 370 weave drama. *CNN U.S. Edition*, Apr. 7, 2014. <http://goo.gl/XJcQBv>.
- [40] Mnemograph LLC. Timeglider. <http://timeglider.com/>.
- [41] R. Munroe. xkcd: Movie narrative charts. <http://xkcd.com/657/>.
- [42] R. Munroe. Lecture at “See, Think, Design, Produce”, Aug 7. 2014. Seattle, WA.
- [43] Northwestern University Knight Lab. TimelineJS. <http://timeline.knightlab.com/>.
- [44] C. Northwood. TERNIP: Temporal Expression Recognition and Normalisation in Python, 2010. <https://github.com/cnorthwood/ternip>.
- [45] S. Nunes. WikiChanges, 2008. <http://sergionunes.com/p/wikichanges/>.
- [46] Phys.org. Mathematicians solve 60-year-old problem, Mar. 23, 2015. <http://goo.gl/jKRwM0>.
- [47] C. Plaisant, B. Milash, A. Rose, S. Widoff, and B. Shneiderman. LifeLines: Visualizing personal histories. In *Proc. ACM SIGCHI Conf. Human Factors in Computing Systems (CHI)*, pages 221–227, 1996.
- [48] ProPublica. TimelineSetter. <http://propublica.github.io/timeline-setter/>.
- [49] J. Pustejovsky, J. M. Castano, R. Ingria, R. Sauri, R. J. Gaizauskas, A. Setzer, and G. Katz. TimeML: Robust specification of event and temporal expressions in text. In *Fifth Intl. Wkshp. Computational Semantics (IWCS)*, 2003.
- [50] D. Ren, T. Hollerer, and X. Yuan. iVisDesigner: Expressive interactive design of information visualizations. *IEEE Trans. Visualization and Computer Graphics (Proc. InfoVis)*, 20(12):2092–2101, 2014.
- [51] S. Rogers. Hey wonk reporters, liberate your data! *Mother Jones*, Apr. 24, 2014. <http://goo.gl/cbhgk>.
- [52] C. Rumaitis del Rio and A. Brown. How dealing with climate change is like playing cricket. *The Guardian*, Mar. 23, 2015. <http://goo.gl/YLkmSx>.
- [53] D. Rushe. Net neutrality activists score landmark victory in fight to govern the internet. *The Guardian*, Feb. 26, 2015. <http://goo.gl/9cD2V2>.
- [54] A. Satyanarayan and J. Heer. Authoring narrative visualizations with Ellipsis. *Computer Graphics Forum (Proc. EuroVis)*, 33(3), 2014.
- [55] A. Satyanarayan and J. Heer. Lyra: An interactive visualization design environment. *Computer Graphics Forum (Proc. EuroVis)*, 33(3), 2014.
- [56] M. Sedlmair, M. Meyer, and T. Munzner. Design study methodology: Reflections from the trenches and the stacks. *IEEE Trans. Visualization and Computer Graphics (Proc. InfoVis)*, 18(12):2431–2440, 2012.
- [57] J. Stasko, C. Görg, and R. Spence. Jigsaw: Supporting investigative analysis through interactive visualization. *Information Visualization*, 7(2):118–132, 2008.
- [58] J. Strötgen and M. Gertz. Multilingual and cross-domain temporal tagging. *Language Resources and Evaluation*, 47(2):269–298, 2013.
- [59] Tableau. <http://tableau.com>.
- [60] Y. Tanahashi and K.-L. Ma. Design considerations for optimizing storyline visualizations. *IEEE Trans. Visualization and Computer Graphics (Proc. InfoVis)*, 18(12):2679–2688, 2012.
- [61] Timeline for iOS, web. <https://timeline.com/>.
- [62] R. Vadlapudi, M. Siabani, A. Sarkar, and J. Dill. LensingWikipedia: Parsing text for the interactive visualization of human history. In *Proc. IEEE Conf. Visual Analytics Science and Technology (VAST) Poster Compendium*, pages 247–248, 2012.
- [63] M. Verhagen, R. Knippen, I. Mani, and J. Pustejovsky. Annotation of temporal relations with Tango. In *Proc. Intl. Conf. Language Resources and Evaluation (LREC)*, 2006.
- [64] M. Verhagen, I. Mani, R. Sauri, R. Knippen, S. B. Jang, J. Littman, A. Rumshisky, J. Phillips, and J. Pustejovsky. Automating temporal annotation with TARSQ. In *Conf. Assoc. Computational Linguistics Poster Proceedings*, pages 81–84, 2005.
- [65] B. Victor. Drawing dynamic visualizations, Feb. 2013. Lecture at Stanford University. <http://vimeo.com/66085662>.
- [66] F. Viégas, M. Wattenberg, F. van Ham, J. Kriss, and M. McKeon. ManyEyes: A site for visualization at internet scale. *IEEE Trans. Visualization and Computer Graphics (TVCG)*, 13(6):1121–1128, 2007.
- [67] Webalon. Tiki-Toki. <http://tiki-toki.com/>.
- [68] Wikipedia. Berlin Wall. <http://goo.gl/GtHKW7>.
- [69] Wikipedia. Facebook: History. <http://goo.gl/aKRKvr>.
- [70] Wikipedia. Sam Smith (singer). <http://goo.gl/dF4Gzm>.
- [71] Wikipedia. Wolfgang Amadeus Mozart. <http://goo.gl/BEKzXw>.
- [72] J. Wolfers. Fewer women run big companies than men named John. *The New York Times*, Mar. 2, 2015. <http://goo.gl/W8qrT>.
- [73] C. Wu. NICAR 2015 slides, links & tutorials, Mar. 2015. Blog post: <http://goo.gl/MFbXXF>.
- [74] R. Yan, X. Wan, J. Otterbacher, L. Kong, X. Li, and Y. Zhang. Evolutionary timeline summarization: A balanced optimization framework via iterative substitution. In *Proc. ACM SIGIR Conf. Information Retrieval*, pages 745–754, 2011.
- [75] J. Zhao, S. M. Drucker, D. Fisher, and D. Brinkman. TimeSlice: Interactive faceted browsing of timeline data. In *Proc. ACM Conf. Advanced Visual Interfaces (AVI)*, pages 433–436, 2012.