TimeLineCurator: Interactive Authoring of Visual Timelines from Unstructured Text

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

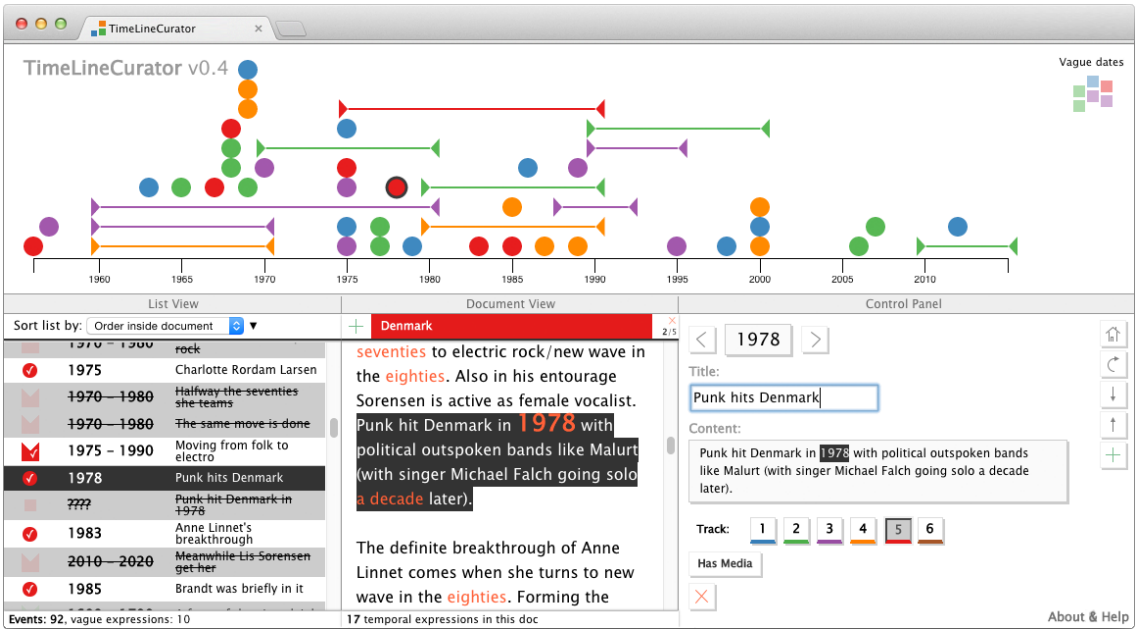


Fig. 1: The browser-based visual timeline authoring tool TimeLineCurator, showing a timeline of Scandinavian pop music, where each colour corresponds to a country; access the interactive timeline at `http://goo.gl/0bHlvA`.

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 Aug. 2015; date of current version 25 Oct. 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 [62] 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 articles[2] 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

[1] For example, see *Timeline: The Higgs, From Theory to Reality* [10]
[2] See 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, 44]. 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.

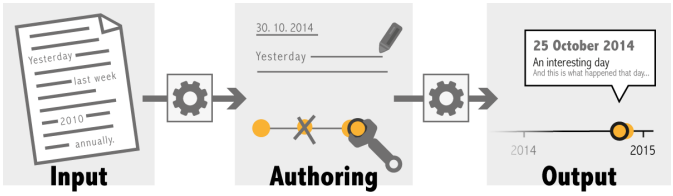


Fig. 2: An abstract representation of TimeLineCurator's pipeline: (i) unstructured text input; (ii) an authoring environment; (iii) curated timeline output.

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 [60] and ManyEyes [67] 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 [76] 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 sources and organizes them chronologically, while Evolutionary Timeline Summarization [75] 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, and 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 [68], 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 [44] 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 [44] 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

TimeLineCurator incorporates a form of Natural Language Processing (NLP) known as *information extraction*, or more specifically, *entity extraction*, a process that identifies predefined 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 [64] and TARSQI (Temporal Awareness and Reasoning Systems for Question Interpretation) [65] 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], HeidelTime [59], and TERNIP [43]. TimeLineCurator uses the Python-based TERNIP system in its natural language processing pipeline. TERNIP uses the TARSQI extraction engine [65] for recognition; TERNIP also normalizes temporal expressions using a rule engine.

Context-dependent semantics: Approaches that consider only the *syntax* of entities ignore 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.

Storyline visualization: To explain the evolution inside complex stories that have various side stories and intertwining threads, Shahaf et al. developed a methodology called "metro maps" [57]. They find salient pieces of information within a document collection and place them on a visual map. Wikipedia articles are a popular source for visualizing freeform text as well [45]. For example, LensingWikipedia attempts to visualize human history through Wikipedia's annual event summary pages over the last 2000 years [63]. Authoring, however, is not supported in any of these environments.

Entity extraction and visual analytics: Visual analytics systems such as Jigsaw [22, 58] 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.

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, though we also gathered feedback from digital humanities and policy researchers 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 online[3] 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

[3] http://www.cs.ubc.ca/group/infovis/software/TimeLineCurator/

publicity at the 2015 NICAR conference for computer-assisted reporting [11, 24, 36, 74]. We were also able to gather feedback and information about use cases from several prospective timeline authors 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 [44] 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 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.

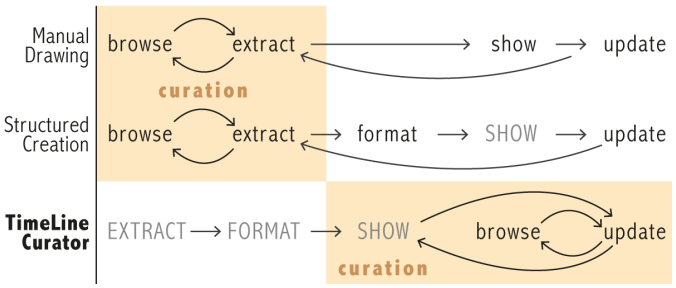


Fig. 3: Comparing the sequence of timeline authoring tasks: timeline curation (indicated by the orange shaded areas) occurs later with TimeLineCurator. Tasks in GRAY UPPER-CASE LETTERS are automated; all other tasks are performed by the author.

Manual drawing: When satisfied with the results of the *browsing* and *extracting* process, the author can manually draw a timeline using 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 [44] 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 [61], however this approach requires structured event data.

Accessible integrated system: Recent advances in natural language processing allow for the extraction and formatting of temporal references from unstructured text [43]. 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 or third-party software. We acknowledge the existence of standalone information extraction software, and that individuals acquainted with them might prefer to use these tools rather than TimeLineCurator's integrated temporal reference extraction. However, these individuals are not our target audience.

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 [44] 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 [66]. 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 [43], 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 events, including those associated with vague dates; she can add, delete, merge, edit, or change their granularity up to the level of minutes. 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.

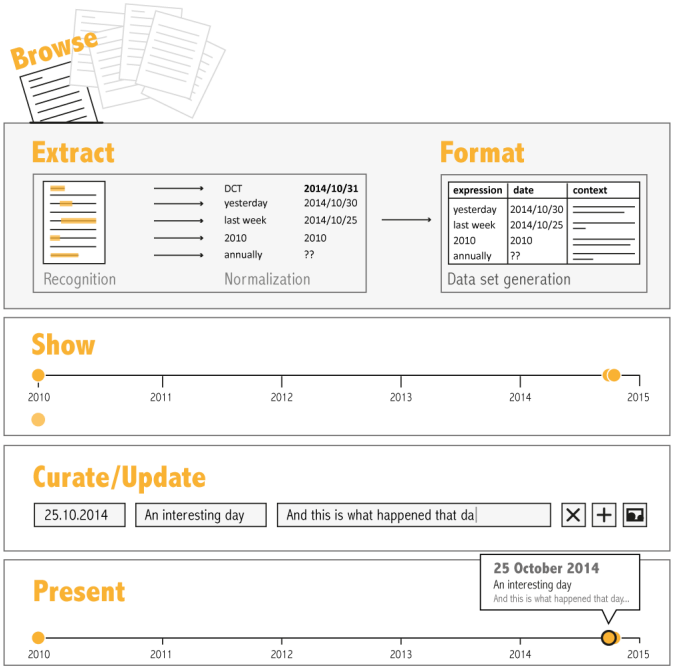


Fig. 4: Processing pipeline for TimeLineCurator.

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 lower right. These panels are initially empty, as in Figure 5a. Figure 5b shows the dialog window where the author pastes unstructured text, or a URL to it, 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 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. Finally, hovering over an event reveals its title.

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 ✔, where the ✔ 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 the list; their deleted status is represented by crossing out the list item, changing the row background colour to 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 [44] 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 articles[4] and two recent news articles[5]; 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.

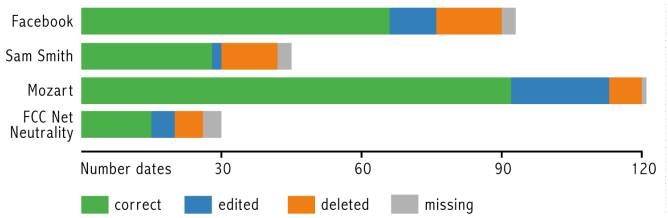
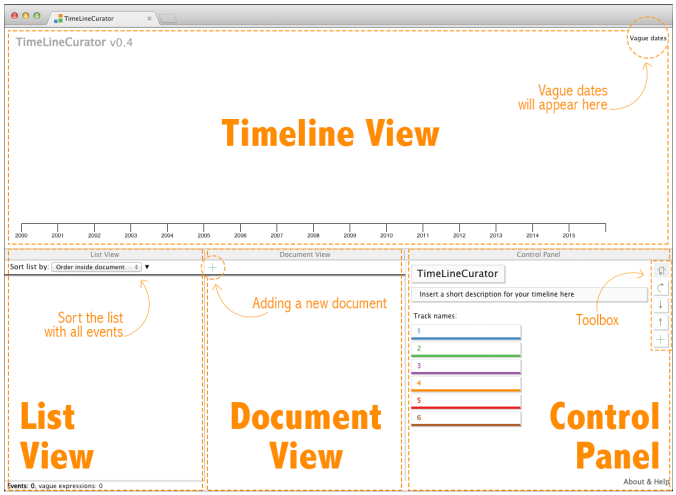


Fig. 6: The results of the benchmark tests, which compares the gold standard manual creation of an event set with the automated event extraction of TimeLineCurator.

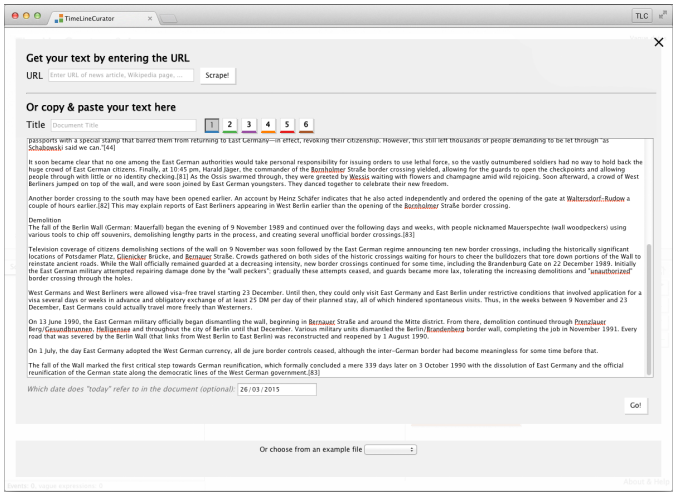
This benchmark also yielded qualitative insights on the kinds of expressions that were incorrectly extracted. Incorrectly identified dates often were time spans, which can be expressed in many different ways in prose. For example, in *"The family again went to Vienna in late 1767 and remained there until December 1768"* [72], two separate dates were 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

[4]The history of Facebook [70], the biography of pop musician Sam Smith [71], and the biography W. A. Mozart [72].

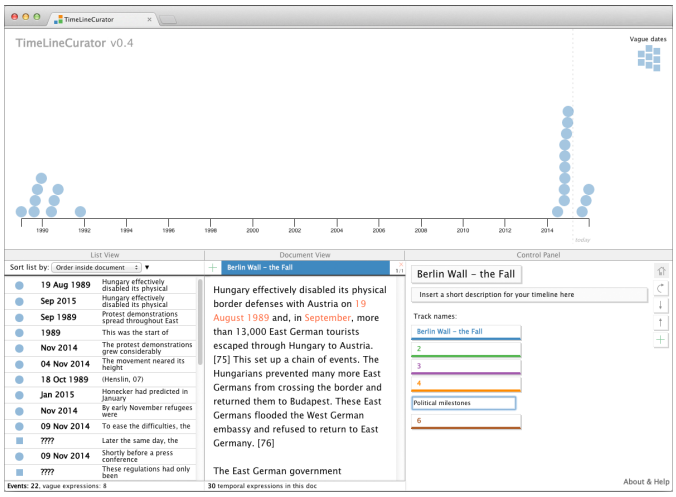
[5]Both pertained to the topic of net neutrality [38, 53].



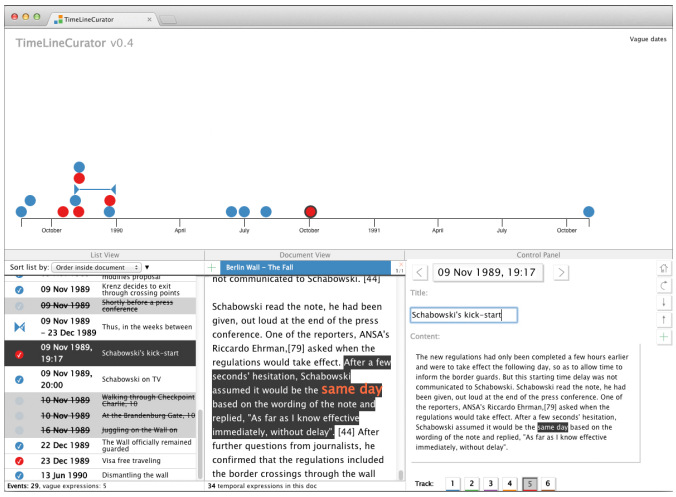
(a) Initially, the timeline is empty. Annotations in orange demarcate the four main views: Timeline View, List View, Document View, and Control Panel.



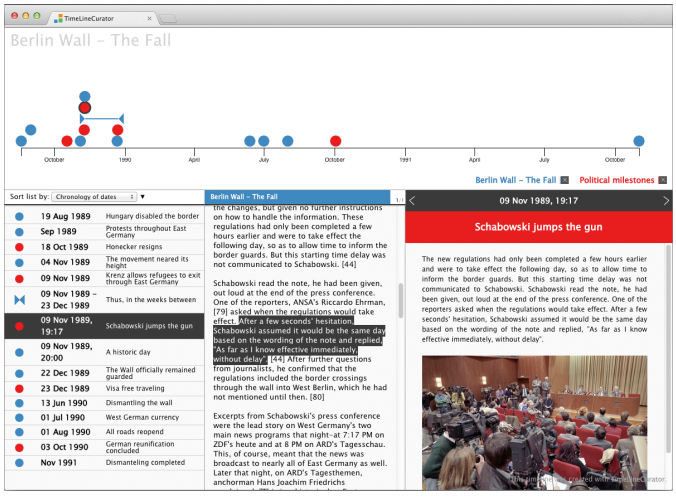
(b) Unstructured text is added via a popup dialog. Optionally, the document creation time can be specified below the input field.



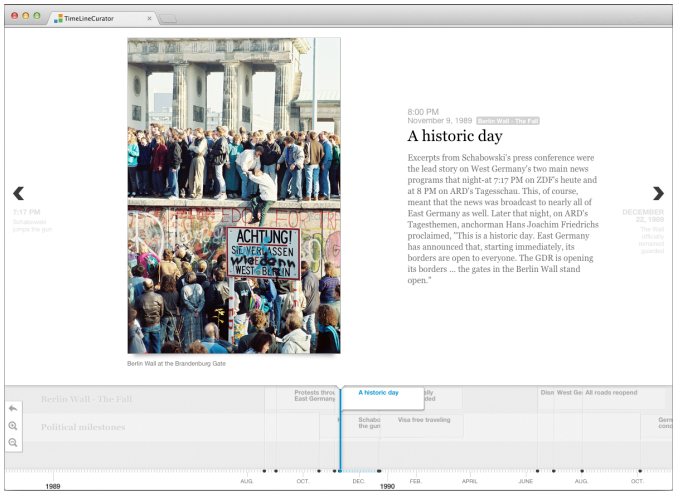
(c) A timeline immediately after importing text, with many vague and uncurated dates. General timeline information can be modified when no event is selected.



(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 [44].

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 [69]. The resulting timeline can be accessed at `http://goo.gl/SU1faP`.

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.

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 [44] 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

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 timelines[6], exported with both TimelineJS and with TimeLineCurator's presentation view.

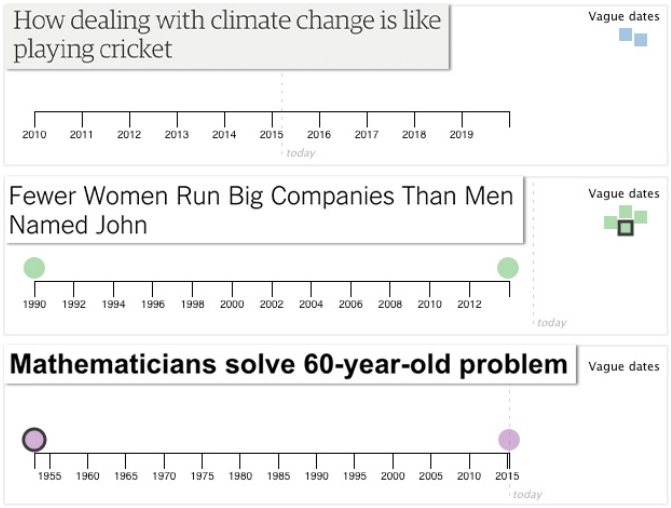


Fig. 7: Timelines extracted from two news articles [52, 73] and a report from a science press release site [46]. All three do not contain much temporal information and thus can quickly be ruled out as a suitable basis for an interesting timeline.

6.5 Use Cases

In addition to evaluation conducted in our lab where 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 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 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.

[6] http://cs.ubc.ca/group/infovis/software/TimeLineCurator/#examples

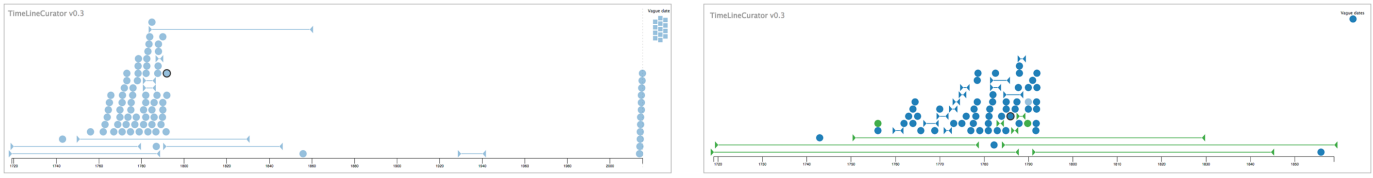


Fig. 8: A timeline of composer W. A. Mozart's biography [72], both before and after curation. The resulting timeline can be accessed at `http://goo.gl/2JikND`.

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

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. To support fact-checking, one possible extension to the Timeline and List Views involves encoding the certainty or uncertainty of events.

In designing the Timeline View, we opted for simplicity over expressiveness. Other visual encodings of time may be more appropriate for highlighting periodic events or for summarizing uneven distributions of events spanning centuries, millennia, or longer. The ability to toggle between alternative encodings could be beneficial.

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 are known to be imperfect, but it would be both useful and straightforward to incorporate newer techniques 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 for 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 requiring 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 timelines: the popular TimelineJS [44] 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 F. Escalona and J. Romero for development assistance. Thanks to C. Skelton and N. Diakopoulos for publicizing TimeLineCurator within the journalism community. We also thank M. Borkin, A. Crişan, E. Hoque, S.-H. Kim, and N. 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. D^3: 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/kfqbTBzjI6.

[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 Gaurdian*, Aug. 21, 2013. http://goo.gl/hdj2PY.

[20] Google. 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. Stasko. Combining computational analyses and interactive visualization for document exploration and sensemaking in Jigsaw. *IEEE Trans. Visaulization 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. *The 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 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/JHVOvJ.

[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/wFSOJf.

[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/XJcQBu.

[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] C. Northwood. TERNIP: Temporal Expression Recognition and Normalisation in Python, 2010. https://github.com/cnorthwood/ternip.

[44] NU Knight Lab. TimelineJS. http://timeline.knightlab.com/.

[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/cbhgtk.

[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] D. Shahaf, C. Guestrin, and E. Horvitz. Trains of thought: Generating information maps. In *Proceedings of the 21st international conference on World Wide Web*, pages 899–908. ACM, 2012.

[58] J. Stasko, C. Görg, and Z. Liu. Jigsaw: supporting investigative analysis through interactive visualization. *Information visualization*, 7(2):118–132, 2008.

[59] J. Strötgen and M. Gertz. Multilingual and cross-domain temporal tagging. *Language Resources and Evaluation*, 47(2):269–298, 2013.

[60] Tableau. http://tableau.com.

[61] 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.

[62] Timeline for iOS, web. https://timeline.com/.

[63] R. Vadlapudi, M. Siahbani, 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.

[64] 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.

[65] M. Verhagen, I. Mani, R. Sauri, R. Knippen, S. B. Jang, J. Littman, A. Rumshisky, J. Phillips, and J. Pustejovsky. Automating temporal annotation with TARSQI. In *Conf. Assoc. Computational Linguistics Poster Proceedings*, pages 81–84, 2005.

[66] B. Victor. Drawing dynamic visualizations, Feb. 2013. Lecture at Stanford University. http://vimeo.com/66085662.

[67] 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.

[68] Webalon. Tiki-Toki. http://tiki-toki.com/.

[69] Wikipedia. Berlin Wall. http://goo.gl/GtHKW7.

[70] Wikipedia. Facebook: History. http://goo.gl/aKRKvr.

[71] Wikipedia. Sam Smith (singer). http://goo.gl/dF4Gzm.

[72] Wikipedia. Wolfgang Amadeus Mozart. http://goo.gl/BEKzXw.

[73] J. Wolfers. Fewer women run big companies than men named John. *The New York Times*, Mar. 2, 2015. http://goo.gl/W8qrtT.

[74] C. Wu. NICAR 2015 slides, links & tutorials, Mar. 2015. Blog post: http://goo.gl/MFbXXF.

[75] 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.

[76] 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.