Extracting the Main Path of Historic Events from Wikipedia

Benjamin Cabrera and Barbara König

Abstract The large online encyclopedia "Wikipedia" has become a valuable information resource. However, its large size and the interconnectedness of its pages can make it easy to get lost in detail and difficult to gain a good overview of a topic. As a solution we propose a procedure to extract, summarize, and visualize large categories of historic Wikipedia articles. At the heart of this procedure we apply the method of main path analysis—originally developed for citation networks—to a modified network of linked Wikipedia articles. Beside the aggregation method itself, we describe our data mining process of the Wikipedia datasets and the considerations that guided the visualization of the article networks. Finally, we present our web app that allows to experiment with the procedure on an arbitrary Wikipedia category.

Keywords Online social networks · P2P infrastructure · Dependence · Scalability · Performance · Security

1 Introduction

Wikipedia is a large free online encyclopedia founded in the year 2001. It not only allows users the access to information but also encourages them to collaboratively work on the articles. At the time of writing the English Wikipedia contained more than five million articles and over 300,000 active users. Because of its accessibility and its large size Wikipedia has become one of the most important sources for encyclopedic knowledge in the world.

However, the richness of information in Wikipedia can also be overwhelming. It can be hard to separate relevant and irrelevant information and in addition the typical audience of Wikipedia can consist of people with a varying background knowledge.

B. Cabrera (✉) · B. König
University of Duisburg-Essen, Duisburg, Germany
e-mail: benjamin.cabrera@uni-due.de; barbara_koenig@uni-due.de

R. Alhajj et al. (eds.), *Network Intelligence Meets User Centered Social Media Networks*, Lecture Notes in Social Networks,
https://doi.org/10.1007/978-3-319-90312-5_5

While some content might be important for a detailed analysis of a topic, it might lead to an information overload for users interested only in a general overview. As a result it is necessary to provide tools that empower the users to make a choice and select relevant pieces of information.

In this article we want to focus on a particular part of the content in Wikipedia, namely the articles about history. Being interested in a historical event with some extension in time, such as the *Thirty Year's War*, the *French Revolution*, or the *Vietnam War*, one can look at the overview article of the topic or at all articles in the corresponding Wikipedia category (and their subcategories). Overview articles usually give a good introduction to a topic; however, they often consist of long texts and it may be difficult to get a grasp of the flow of singular events, such as battles and peace treaties, at one glance. Looking at all articles in a category in order to obtain an overview is often infeasible due to the sheer number of articles—thousands of articles are quite common in larger categories.

In the following we will describe a novel approach of summarizing and visualizing categories in Wikipedia. From a given Wikipedia category and its subcategories we will generate a graph whose nodes correspond to dates occurring in these articles and edges linking two dates whenever the source date precedes the target date and the corresponding articles are connected by a link (in any direction). In order to help the user navigate through the network, we propose to use the method of main path analysis to emphasize the structural backbone in a large (acyclic) graph.

In order to make it easy to try out the described approach we implemented a web app capable of visualizing the graph of a category highlighting the main path. This can be seen as a tool for *visual analytics*, which helps users to obtain deeper insights into the data using a visualizations. We encourage the readers to experiment with the web app.[1]

Related Work Mining Wikipedia data and using it for data visualization are certainly not a new topic and have been done in several, also nonscientific projects.

Part of our approach is about parsing the wiki markup files of Wikipedia to structured datasets used as input for our visualization methods. The most known related projects dealing with mining Wikipedia are DBPedia [2] and WikiData [14]. These projects aim to make Wikipedia's data machine readable and provide interfaces to the structured data. While in particular dates could have been extracted from one of these projects we opted for parsing Wikipedia dumps ourselves because the parsing infrastructure was needed for the other meta-data and we had full control of the extraction process.

Concerning the specific parsing of historic events from Wikipedia there is related work [8] offering an API to access the structured data. However, their data is limited to around 150,000 events and no extraction of relationships is performed. Beside Wikipedia as a data source, in general, TimeLine extraction is an emergent research field in NLP [11, 12]. Here, approaches focus on extracting temporal relationships

[1] http://www.ti.inf.uni-due.de/research/tools/wikimainpath/.

across documents, e.g., from news media using language features. In a sense [11, 12] deal with the task of building what we call the event network from a different perspective.

Beside the parsing aspect there are also existing approaches improving the search of historic events in Wikipedia. The need for tools to navigate through historic Wikipedia articles was addressed by Tiwiki [1], a tool that allows filtering articles by their dates. In our web app we offer a similar feature filter where a range of years can be specified which restricts the events under consideration. Close to our approach is also the histography project by Matan Stauer[2] which visualizes historical events from Wikipedia on a timeline. However, this approach does not provide navigational guidance by using a network approach such as main path analysis. A related project, Wikigalaxy,[3] visualizes the entire English Wikipedia as a huge graph-like structure (which resembles a galaxy) and offers support for navigating this graph. Here the focus is clearly on the visualization. Aggregation of Wikipedia data for visual analytics is also described in [5], which shows how visual analytics techniques can help Wikipedia volunteers (Wikipedians) to detect vandalism and edit wars and gather information about the trustworthiness of articles.

Finally our approach can be seen as part of the emerging field of computational history. In this field data mining and machine learning approaches are applied to large historical datasets. For example, in [4] the authors extract timelines of historical figures from Wikipedia data. However, their focus is on the mining part and less on summarization and visualization.

2 Main Path Analysis

Main path analysis describes a set of methods and algorithms on graphs which aim to reduce a potentially large graph to a much smaller path through the graph. It was first introduced by Hummon and Doreian [9] in 1989 to study the flow of scientific ideas in citation networks. Since then there have been several improvements to the method and applications to citation networks of various scientific fields [3, 13]. Recently there have also been successful applications to other types of networks [6, 7] different from the citation networks the method was originally invented for. In this section we will give a short overview of the algorithms used in main path analysis. For a more detailed explanation, see, e.g., [13].

[2] http://histography.io/.

[3] http://wiki.polyfra.me/.

2.1 Preliminaries

In this article we consider networks in the form of a *graph* $G = (V, E)$ consisting of a set of *vertices* V and a set of *edges* E connecting those vertices. As we will see later, main path analysis is only applicable to *directed* and *acyclic* graphs (also known as DAGs). For a directed graph we have $E \subseteq V \times V$, i.e., edges are ordered pairs of vertices. Hence there might be an edge $(u, v) \in E$ connecting $u \in V$ and $v \in V$ but not necessarily a backward edge $(v, u) \in E$. Acyclicity of a graph means that there are no cycles in the graph, i.e., no subsets $C = \{(s_1, t_1), \ldots, (s_l, t_l)\} \subseteq E$ such that $t_i = s_{i+1}$ for $1 \leq i < l$ and $t_l = s_1$. Intuitively, this means that there is no path in the graph going back to a vertex once we have left it. Additionally, as part of the algorithms described later, we will need to assign weights to edges of the graph. To this end we consider edge weights via a function $w : E \to \mathbb{N}_0$ where $w(e)$ defines the weight on an edge $e \in E$. Finally, we define a path P in G to be a sequence of vertices $P = (v_1, \ldots, v_k)$ in which no vertex appears more than once, i.e., $v_i \neq v_j$ for all $i \neq j$. The goal of our analysis will be to compute a path P representative of G—this is the *main path*.

The hidden assumption behind main path analysis is that the underlying DAG encodes some form of causal or at least sequential relationships between entities. For citation networks these relationships represent citations from newer to older scientific articles. A citation is interpreted as a reuse of ideas taken from the cited article and used in the citing article. This intuition motivates to ask for a path representing the whole graph—instead of, say, an arbitrary subgraph. A representative path, in this context, is the chain of scientific articles that lead to the current state of the research field. Additionally this intuition explains why we are considering only DAGs in the first place. On one hand, causal relationships naturally lead to directed connections because we want to identify what was the cause and what the result of an action—as opposed to only stating that entities are in some kind of relationship without stating the direction. On the other hand, the acyclic nature of the graph—at least in the case of citation networks—comes from the fact that we cannot have citations going forward in time. An author can, in theory, only cite existing articles which in turn cannot change their references anymore. The assumptions satisfied by the citation networks have to be considered for our application as well. However, before we do so in Sect. 3 we first explain some basic algorithms for computing the main path.

2.2 Main Path Computation

Main path analysis can be broken down into three steps explained in this subsection. Performed in the following order they are:

1. adding a source and a sink vertex,
2. computing edge weights guiding the upcoming path finding algorithms,

3. computing the path through the graph.

A (finite) directed, acyclic graph is guaranteed to contain vertices without incoming edges as well as vertices without outgoing edges. The main path should be as representative of the original graph as possible, so naturally we will always start the path at vertices without incoming edges and end it at ones without outgoing edges. That is because if we start at a vertex with an incoming edge we can simply add an incoming neighbor to the path and get a longer path capturing more of the original graph. The analogue holds true if the main path ends in vertices with outgoing edges. However, while there are always these potential start and end vertices, there are often multiple of them. In order to have a unique start and end vertex, which is a requirement of our algorithms in step three, we add two vertices v_s, v_e to be used as the new source (start) respectively sink (end) vertex. The new start vertex is then connected to every already existing vertex without incoming edges and every existing vertex without outgoing edges is connected to the new end vertex.

The second step of main path analysis consists of computing weights for the edges of the graph to be used as a guide by the upcoming path finding algorithms. In standard main path analysis [13] there is only one common approach to computing these weights named the *search path count (SPC)*. Taking the "flow of ideas" intuition into account the SPC weight $w(e)$ of an edge $e \in E$ is defined as the number of paths from the source vertex v_s to the sink vertex v_e that run over e. Thus edges connecting more important articles tend to have a higher SPC weight because important articles are usually cited more often and thus tend to contribute more paths to the mentioned path count. Additionally, the whole history of predecessor articles is considered, i.e., edges citing highly cited articles lead to a higher SPC weight. Since the number of paths through the graph grows potentially exponentially in the number of vertices, it is nontrivial to compute the SPC weight efficiently. For the sake of brevity we will not explain the exact method here and point to [3] for further details. To the reader familiar with network analysis the SPC weight might seem similar to stress centrality. Note, however, that stress centrality considers only shortest paths and is defined on the vertices, not the edges.

The final step of main path analysis is the actual path finding algorithm. There are different approaches for computing the main path, but they are all based on the SPC weights computed in step two. The most simple one is a greedy approach often called *local forward main path*. It starts at v_s and then adds a successor v of v_s for which the weight $w(v_s, v)$ is maximal to the path. If there are multiple choices of the same weight we simply pick one of them. The procedure is then repeated from v until we reach v_e. The resulting sequence of vertices is the main path.

For an example of how the steps of local main path analysis are performed, see Fig. 1.

A second algorithm computes the so-called *global main path*. This approach considers all paths from v_s to v_e and picks the one with the highest sum of all weights to be the main path. That is the weight of a path $p = (v_1, \ldots, v_k)$ with $v_1 = v_s, v_k = v_e$ is $w(p) = \sum_{i=1}^{k-1} w(v_i, v_{i+1})$ and we choose a path p for which $w(p)$ is maximal.

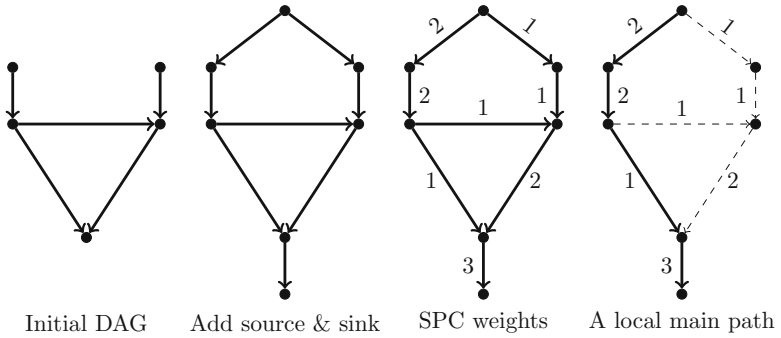

Fig. 1 A step by step main path analysis for a basic example graph. Starting with an initial directed, acyclic graph we add a source and sink vertex. We then compute the SPC weights and run a local path finding algorithm. Note that after the second edge there are two possible choices for continuing the local path finding—we simply choose the displayed one. In contrast, the global main path would deterministically choose the other path including the third edge of weight 2

In our application area, local and global main paths have the tendency to become very long, especially since there exist categories with many thousands of articles. In some cases, the user may want to navigate such a long main path, in others, a more condensed version is required. For the latter case we use a new variant of main path analysis, the α main path algorithm. In this case, given a parameter α, the modified weight of a path $p = (v_1, \ldots, v_k)$ is $w(p) - \alpha \cdot (|p| - 1)$, i.e., the weight of the path (as defined above), minus its length (number of edges), multiplied with the factor α. This formula favors shorter paths over longer ones and the search for the path with maximal weight can still be efficiently implemented. Finding the correct α value for getting a main path of a certain length is often based on trial and error. Because the length of the main path is monotone decreasing for increasing α we can use a binary search for finding the α value that leads to a main path closest to the wanted size.

There are many further variants of local and global path finding algorithms which we do not describe in this article. They are described in detail in [13]. Note, however, that while the local main path algorithms are naturally very fast, considering all paths for a global variant can lead to exponential-time algorithms. Fortunately, there are fast algorithms also for the global methods.

3 Extracting the Main Path of Historic Events

As detailed in the introduction, it is our aim to apply the main path analysis methods described in the previous section to a use case different from citation networks. To be precise, we compute the main path of historic events based on Wikipedia datasets. To this end, we first have to be able to build a meaningful graph from the Wikipedia datasets to which a main path analysis can be applied. In this section, we first describe the general ideas about how this graph can be built and then go into some implementation details.

3.1 A Graph of Historic Events in Wikipedia

In order to build a graph, the first step is to select a group of articles that are of relevance for a particular topic. Wikipedia has a system for grouping articles on the same topic into categories. For example, there are categories on *World War II* or the *French Revolution* containing even finer subcategories such as *Battles and operations of World War II* or the *French First Republic*. A detailed overview can sometimes only be obtained by recursively searching for all subcategories and assembling all pages within subcategories. In our approach one can choose whether to perform this recursive descent (or not).

Since it is our goal to build a graph from Wikipedia articles, the first thing that comes to mind is to use the articles as nodes and hyperlinks between articles as edges. It can be argued that the links encode a meaningful relation—not only between the articles but also the article's topics. If an article about topic A references topic B, we can assume that the two topics are related. Furthermore, if an article is referenced often by articles of a certain topic, it also seems reasonable to assume that this article is more important to the topic than articles not referenced as often.

However, articles do not necessarily correspond to events. Most of the Wikipedia articles about historic events cannot be reduced to only one date. Instead, big historic events often take place in several steps so that we extract multiple dates from one article. We deal with this problem by considering each date extracted from an article as its own event labeled by the context (i.e., key, see following section) in which the date appeared in the article.

Beside that, we often extract date ranges stating that some event started at some point and carried on until a later point in time. For example looking at a topic such as the *Thirty Years' War* we have events that range over a substantial period of time while others such as the *September 11 attacks* took place on only one day. Since it is unclear how to order overlapping date ranges, we split such ranges into atomic start and end dates. Hence, the nodes of our graphs are not the articles, but the various dates. Edges are supposed to model causality and should hence point "forward in time," i.e., from an older to a newer event. More concretely, we add an edge from event a to event b whenever a precedes b and a, b are listed either in the same article or in articles which are related by a link (in either direction!). A schematic depiction can be found in Fig. 2.

We argue that an edge in the described graph encodes a causal relation between the two historic events. On the one hand, a connection in the link graph introduces some form of relation between the events that was at least important enough for the article's author to mention it. On the other hand, because the edge is pointing from the older to the more recent event, we capture that the influence of an event towards the other can only be in this direction. Together this makes it reasonable to interpret an edge from event A to B as "event A influenced—or even directly caused—event B."

The resulting graph is necessarily acyclic, making it amenable to main path analysis. A path in this graph represents a sequence of historic events from the

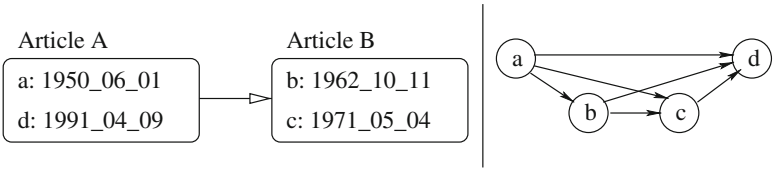

Fig. 2 Extraction of the graph of historic events (schematic depiction)

oldest to the most recent one. A main path is such a sequence summarizing the historic development in the whole graph.

Our implementation is split into two main components: the parser component and the visualization component. Both use relatively different technologies and we describe them separately in the next two sections.

3.2 Implementation of the Parser Component

As can be inferred from the previous section we need three types of structured data to execute our method:

1. the set of all Wikipedia articles with extractable dates,
2. a mapping from any category to the articles it contains (taking into account nested categories, see below),
3. the links which exist between articles.

Naturally, we want to have an automatic gathering and extraction process instead of a tedious manual extraction of articles and dates. What is difficult about this approach, however, is that Wikipedia datasets are not available as structured data. The building blocks of Wikipedia are pages, text files in a specific Wiki markup[4] that are behind each visible web page of Wikipedia. While the Wiki markup allows to format dates, positions, etc. in a specific way it does not separate them into something resembling a relational database. Thus, in order to access specific data from the pages, one has to parse the text files first. This parser component of our implementation runs on full Wikipedia XML dumps,[5] extracting the needed datasets. This enables users to analyze any category in Wikipedia containing any article with an extractable date.

The parser component consists of several C++ programs parsing the large Wikipedia XML dumps files. These programs represent the different steps necessary for extracting the required entities and serializing them to structured files where they can be used by the visualization component. The first step extracts all articles

[4] https://en.wikipedia.org/wiki/Wiki_markup.

[5] https://dumps.wikimedia.org/enwiki/.

with dates and all categories in Wikipedia. A category is easily recognized because its corresponding Wikipedia page title always starts with the string `Category:`, followed by the category's title. We can thus check for this string at the beginning of each page and save all the category titles to a file.

Extracting dates from articles, however, is the most challenging part of the parsing process. The reason for this is twofold. On one hand, we first have to locate a date on a page. Often dates are simply part of the text of an article or they appear sporadically on tables summarizing certain events. Fortunately, there is also a relatively reliable source of extractable dates in the form of infoboxes. Infoboxes appear on many Wikipedia articles, summarizing its content into a few key bullet points. For historic articles these often contain important dates about the event. In the Wiki Markup infoboxes are initialized by an {{`infobox` string, followed by a set of key-value pairs consisting of a label (key) and the content (value) of a bullet point. Our parsers search for infoboxes on a page and then scan the key-value pairs for keys which implicate a date in the value (e.g., `started`, `ended`, `date`, etc.).

Once we identified strings possibly containing dates we have to parse them into a common structured format. However, here lies the second problem because dates are formatted in many different ways in Wikipedia. For example, many authors will simply use a format where day is followed by month and year, while others prefer month followed by day and then year and so on. The date extraction process is made even more complicated by the fact that Wiki Markup provides special templates for dates that lead to them being displayed in a particular fashion. These also need to be parsed correctly. To incorporate all these variations we used an approach based on grammars specifying the different formats. Using the *boost.spirit*[6] library for turning the grammars into parsers we were able to reliably extract dates for most articles at a speed that allowed us to parse full Wikipedia dumps in a reasonable amount of time.[7] A schematic visualization of our date extraction scheme can be found in Fig. 3.

So far we extracted categories that can group articles into related topics and articles with detectable dates. The next parsing step now deals with getting the links needed for the edges of the graph we want to build. Links in the Wiki markup start with `[[` followed by the name of the Wiki page to which the link points and are closed by another `]]`. While in reality there are some slight variations to this format it is relatively easy to extract all links from a Wikipedia page. We call the Wikipedia page where the link occurred the source and the page it is pointing to the target. Each link is of one of the following four types:

1. source and target are articles with extractable dates,
2. source is an article with extractable date and the target is a category page,
3. source and target are category pages,
4. none of the above is true.

[6] http://boost-spirit.com/.

[7] Around 2 h for extracting the dates in ∼130 GB of Wikipedia pages.

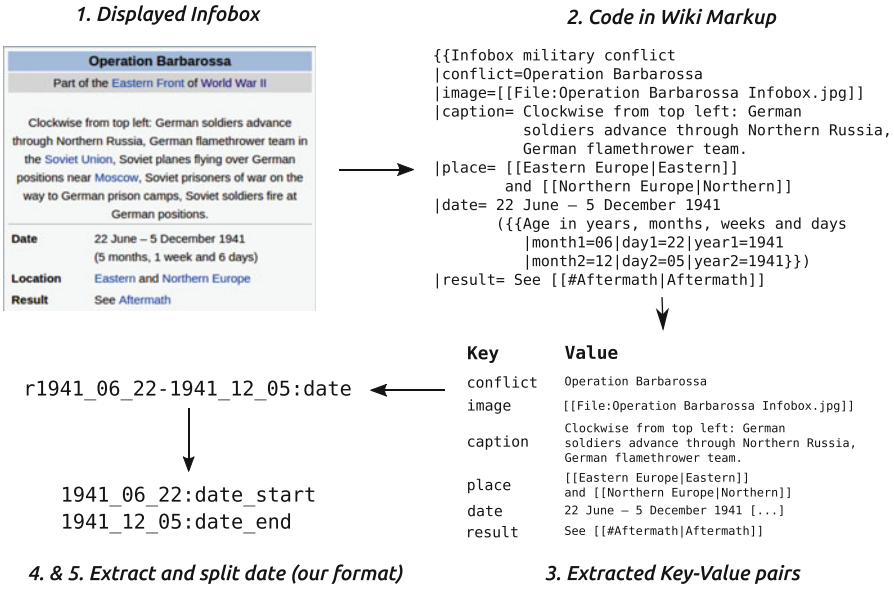

Fig. 3 The date extraction process used for extracting events with dates from a historic article. Most historic articles contain infoboxes summarizing important information including the date. Our parser detects key-value pairs likely containing dates and parses the different formats in which the date could be encoded

For the first type of links, we add an undirected link between source and target to an article network dataset. Note that we do not already split the articles into events and that we do not build a directed network yet. This will happen online for requested categories in the visualization component. For the second type of links, recall that we are selecting articles relevant for a topic by choosing a category. Links from an article to a category imply membership of the article in the category. Thus we can use the second type of links to save which articles are part of which category. The third type of links is used to solve a different problem related to categories and their containing articles. Wikipedia often uses subcategories to split large categories into smaller ones. For example, a category such as *World War II* might have subcategories such as *Battles and operations of World War II* or *World War II resistance movements*. However, if we want to analyze the full *World War II* category we have to consider all articles that are part of a subcategory of it. To this end, we use the third type of links to build a hierarchy of categories which is later used in the visualization component to recursively build the set of articles in a category from its subcategories. Finally, the remaining links of the fourth type are not relevant for our approach and can be ignored.

3.3 Implementation of the Visualization Component

The second large part of our implementation is the visualization component. It takes the structured data from the parsing step, computes the article network with its main path, and displays the result to the user. It is implemented as a web app that can be used from any web browser and consists of a HTTP server back-end written in `C++` and a `JavaScript` client side front-end.

Using the web app, a user is first presented with a search input which allows to search for Wikipedia categories containing certain keywords. Having extracted a list of all categories in Wikipedia the implementation of this search feature is a simple full-text search scheme we will not go into detail about. After choosing a category, every upcoming step is performed on this category.

The first task performed by the server is to gather all articles that are part of the category. To this end we have to traverse the tree-like structure describing the hierarchy of categories. We start at the selected category and add all articles belonging directly to this category to a list. Then we move to all subcategories and do the same recursively. This continues until we reach all leaves. Note that because articles might belong to multiple subcategories, we use a sorted structure to only collect distinct articles. After having a list of all articles in the category we build a list of all atomic events (compare Sect. 3.1) that are part of these articles. To this end we look at all dates we extracted for the articles and turn all of them into events. At this point we can apply some filters to the event list to throw out outliers or unfitting events. Filters provided in the app include the ability to exclude events outside of specified range of dates, excluding events related to persons or excluding events containing certain keywords.

Next, we build the event network from events in the list using the procedure described in Sect. 3.1 (cf. Fig. 2). Following the construction of the network we can apply one of the main path algorithms. Finally, the back-end creates a response to the request and sends all the above-mentioned data to the client side web app. There the network is layouted according to the dates of the events (x-coordinate) and a random y-coordinate. The main path is highlighted and some statistics on the category are displayed on the side.

4 Results

The following section contains details about the dataset parsed from a current Wikipedia dump as well as representative examples from categories with their respective main path.

The dataset used in the upcoming computations was extracted from a XML dump of the whole Wikipedia from 1st May 2017.[8] In total the XML dump consisted of 40 M Wikipedia pages at a size of 130 GB. It took all programs of the parser

[8] https://dumps.wikimedia.org/enwiki/20170501/.

Table 1 Categories with their respective sizes and the lengths of their local main path

Category	Timespan	#Events	#Links	Local MP
Ukrainian Crisis [NR]	2011–2017	24	118	16
Events of the French Revolution by Year [NP]	1789–1804	41	189	24
Norman Conquest of England	1001–1112	42	271	26
Thirty Years War [NP]	1618–1648	81	218	10
French Revolution [NP]	1785–1849	1261	12,264	276
European theatre of World War II [NP]	1939–1945	1712	13,867	265
World War II [NP]	1939–1945	6250	60,009	329

[NR] stands for "Not Recursive" meaning that we did not consider subcategories; [NP] abbreviates "No Persons" meaning that events about persons (birth, death, ...) were filtered out. The timespan is built from the earliest and latest events in the category and can deviate from the actual historic period

component around 4 h to run on a server with 4 x Intel(R) Xeon(R) @ 3.50 GHz (16 cores in total) and 64 GB RAM. We were able to parse around 1.7 M articles that contain at least one date leading to 2.7 M atomic historic events in our final dataset. Furthermore, we extracted 1.6 M categories and their respective containing articles. Finally, we collected around 27 M links between articles used in the construction of the event network for a requested category. The computations for one category in the visualization component never exceeded three seconds although displaying very large components in the browser can result in slower response times of the app.

In order to evaluate our method we looked at several historic categories of varying sizes for which we build the event network and computed the main paths. Table 1 shows a list of the categories, their sizes, and the number of vertices in the local main path. We can observe that categories vary considerably in size, ranging from those containing only a few events to categories such as *World War II* with thousands of events. Furthermore, there are great differences in structure. This is for example visible when comparing categories *Thirty Years War* and *Ukrainian Crisis*. Although the former has more than three times the links of the latter its local main path is significantly shorter. The reason is that *Thirty Years War* is not as well connected and consists to a larger extent of parallel paths while *Ukrainian Crisis* behaves much more like one connected component.

Figure 4 shows the small but prototypical event network of the *Events of the French Revolution by Year* category.[9] One can observe that the category is well connected—not breaking up in distinct paths. As a result the local main path runs through 24 of the 41 events. The main path contains important events such as the *Storming of the Bastille* and the establishing of the *National Convention*. However, there are also important events missing. For example, the article about the *Women's March on Versailles* is not in our dataset because it contains no infobox with a date and thus no date could be extracted. Note also that the main path does not only consist of nodes with high degree—although there is a high correlation for this small category. This argues in favor of the main path method because simply showing

[9]Note that in the paper we use a different layout than for the web app.

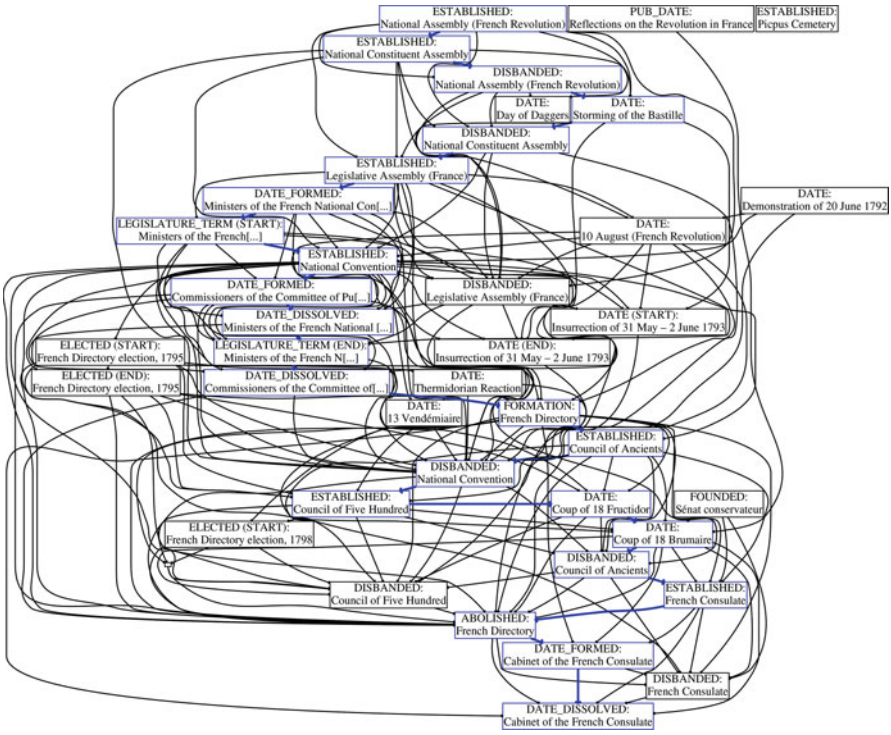

Fig. 4 Event network of the *Events of the French Revolution by Year* category. The local main path is highlighted in blue

nodes with the highest degrees would rank events differently, not considering their sequential relations. As an example, the 24 nodes with highest degree do not contain the Storming of the Bastille. This deviation between high degree and inclusion in the main path is more pronounced as the category gets larger.

Figure 5 shows the extracted local main paths for the categories *Thirty Years War* and *Ukrainian Crisis*. The one for *Thirty Years War* contains the important events *Peace of Prague*, *Battle of Nördlingen*, and the *Peace of Westphalia* that ended the war. However, the battles of Lens and Oldendorf are rather unimportant events. Also note that the article *Thirty Years War*—called like its containing category—is a summary article about the whole war. As a result it is referenced by almost every other article of the category and thus a hub of the event network. Hubs improve the connectivity of the network but, on the other hand, they usually appear on the main path which could be considered redundant because the main path itself tries to summarize a category. Another interesting observation can be made at the end of the *Thirty Years War* main path. The war is considered to end with the signing of the Peace of Westphalia. However, the summary article dates the war's end to before the end of signing of the peace treaties, with the Battle of Lens in between. The reason for this is that the treaties were signed over months and the summary

Fig. 5 Extracted local main paths from the categories *Thirty Years War* and *Ukrainian Crisis*

article's ending date is the start of the signing process while the last event of the main path marks the end of it.

We included the *Ukrainian Crisis* category to show that our method can also be applied not only to classic but also to modern history. The main path correctly starts with the Euromaidan event that started the uprising in the Ukraine, followed by the Russian intervention attempts and several treaties trying to solve the conflict.

So far the analyzed example categories were of relatively small size—although possibly still too large to be analyzed by hand. However, especially very important historic categories often contain a lot more articles. This is, in particular, the case because we aggregate all subcategories into the main category as well. As an example we picked the *European theatre of World War II* category containing over 1000 events and 13,000 links even after removing persons and events not between 1939 and 1945. The local main path contains 265 nodes which is only 15% of the original nodes but is still long in absolute terms. As a solution we can apply the mentioned α main path algorithm explained earlier where α is a parameter that can be used to influence the length of the path. Figure 6 shows α main paths for *European theatre of World War II* with varying values for α. As described in Sect. 2.2 we used a binary search to find α values for maximal path lengths $l = 10, 15, 25$. Overall we observe that some important events like the Invasion of Poland appear in all three main paths. With growing path length less important events are added. However, for instance the Normany landing appears only in the longest main path although it could be argued that this event would be important enough to already appear in the smallest one.

Fig. 6 Main paths extracted from the *European theatre of World War II* category using the alpha main path algorithm to enforce certain maximal path lengths l

5 Conclusion

We have presented a procedure to mine data from Wikipedia in order to provide a visual user aid in form of a main path of historic events. Applying the method to various categories shows that meaningful main paths are extracted containing the main events of the category.

Naturally, our approach has some limitations. For instance, it is unclear whether links between articles encode causality in all cases. It is something we assumed for our analysis and which is intuitively what a main path is describing. Another problem can, in a few instances, be the nesting of categories within Wikipedia. Some categories contain subcategories that are either completely unrelated or only

remotely related to the topic. For instance, the *Vietnam War* category contains a subcategory *Counterculture of the 1960s* which contains articles such as *Abortion law* and *Psychedelic rock*.

On another note it is not always clear which main path algorithm yields the best results. The local algorithm might, due to its local decision, not always be robust with respect to small changes in the underlying graph, which favors the—still very efficient—global algorithm. Since main paths in our examples have a tendency of being quite long, the α main path algorithm helps to create short paths. In that case the question of choosing the "right" α value remains.

Future improvements could be applied to the visualization component's graph layouting algorithm. So far events are plotted on a time x-axis and randomly placed on the y-axis. As an improvement one could use a variant of a force-directed layouting algorithm [10] that places events by minimizing overlap. It would also be possible to perform an extended evaluation of our approach by letting historians judge the main paths and visualizations and their representative quality.

Acknowledgements We would like to thank Ulrich Hoppe and Stephanie Große for interesting and stimulating discussions on the topics of the paper. In addition we would like to thank Issai Zaks for his help with the servers and overall technical support. Finally, we have to thank our anonymous reviewers providing some very valuable feedback and missed references to related work. This work was supported by the Deutsche Forschungsgemeinschaft (DFG) under grant GRK 2167, Research Training Group "User-Centred Social Media."

References

1. Agarwal, P., Strötgen, J.: Tiwiki: searching wikipedia with temporal constraints. In: Proceedings of the 26th International Conference on World Wide Web Companion, Perth, April 3–7, 2017, pp. 1595–1600 (2017)
2. Auer, S., et al.: DBpedia: a nucleus for a web of open data. In: The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, November 11–15, 2007, pp. 722–735 (2007)
3. Batagelj, V.: Efficient algorithms for citation network analysis. In: CoRR cs.DL/0309023 (2003)
4. Bauer, S., Clark, S., Graepel, T.: Learning to identify historical figures for timeline creation from wikipedia articles. In: Social Informatics - SocInfo 2014 International Workshops, Barcelona, November 11, 2014, Revised Selected Papers, pp. 234–243 (2014)
5. Boukhelifa, N., Chevalier, F., Fekete, J.-D.: Real-time aggregation of wikipedia data for visual analytics. In: Proceedings of VAST '10 (Visual Analytics Science and Technology), pp. 147–154. IEEE, New York (2010)
6. Chuang, T.C., et al.: The main paths of medical tourism: from transplantation to beautification. Tour. Manag. **45**, 49–58 (2014)
7. Halatchliyski, I., et al.: Analyzing the flow of ideas and profiles of contributors in an open learning community. In: Prof. of LAK '13 (Conference on Learning Analytics and Knowledge), pp. 66–74 (2013)
8. Hienert, D., Luciano, F.: Extraction of historical events from wikipedia. In: The Semantic Web: ESWC 2012 Satellite Events - ESWC 2012 Satellite Events, Heraklion, Crete, May 27–31, 2012. Revised Selected Papers, pp. 16–28 (2012)

9. Hummon, N.P., Doreian, P.: Connectivity in a citation network: the development of DNA theory. Soc. Netw. **11**(1), 39–63 (1989)
10. Kobourov, S.G.: Spring embedders and force directed graph drawing algorithms. In: CoRR abs/1201.3011 (2012)
11. Kolomiyets, O., Bethard, S., Moens, M.-F.: Extracting narrative timelines as temporal dependency structures. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1, ACL '12, pp. 88–97. Association for Computational Linguistics Jeju Island (2012)
12. Laparra, E., et al.: Multilingual and cross-lingual timeline extraction. In: CoRR abs/1702.00700 (2017)
13. Liu, J.S., Lu, L.Y.Y.: An integrated approach for main path analysis: development of the Hirsch index as an example. J. Am. Soc. Inf. Sci. Technol. **63**(3), 528–542 (2012)
14. Vrandecic, D., Krötzsch, M.: Wikidata: a free collaborative knowledgebase. Commun. ACM **57**(10), 78–85 (2014)