Extracting and Aggregating Temporal Events from Text

Lars Döhling
doehling@informatik.hu-berlin.de

Ulf Leser
leser@informatik.hu-berlin.de

Department of Computer Science
Humboldt-Universität zu Berlin
Unter den Linden 6, 10099 Berlin, Germany

ABSTRACT

Finding reliable information about a given event from large and dynamic text collections is a topic of great interest. For instance, rescue teams and insurance companies are interested in concise facts about damages after disasters, which can be found in web blogs, newspaper articles, social networks etc. However, finding, extracting, and condensing specific facts is a highly complex undertaking: It requires identifying appropriate textual sources, recognizing relevant facts within the sources, and aggregating extracted facts into a condensed answer despite inconsistencies, uncertainty, and changes over time. In this paper, we present a three-step framework providing techniques and solutions for each of these problems. We tested the feasibility of extracting time-associated event facts using our framework in a comprehensive case study: gathering data on particular earthquakes from web data sources. Our results show that it is, under certain circumstances, possible to automatically obtain reliable and timely data on natural disasters from the web.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information filtering, Query formulation*; I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*Text analysis*

Keywords

Query expansion; Natural language processing; Information extraction; Publication date extraction; Information fusion; Curve fitting; Earthquake

1. INTRODUCTION

Everyday, millions of people use the web as an information source, browsing blogs, tweets, newspaper articles etc. Given long-lasting events like disasters, people seek for reliable and timely data describing the event and its aftermath. Today, even rescue teams use the web to gather facts about damages and casualties, especially if no on-site contact is available. Knowing these facts helps them to determine the required scale of relief operations and supports their coordination [16]. Surely, such information can be found on the web, but with diverse timeliness, quality, and granularity, imposing several challenges. First, one needs to find those documents (web pages, blog entries, tweets) potentially containing the facts of interest. Next, one has to analyze the documents and extract the desired facts. This also includes the temporal alignment of the documents and their contained facts. Knowing the temporal dimension of facts is the key to their usefulness, as facts change over time. Since the analyzed documents originate from various sources, published at different points in time, the extracted facts will contain inconsistencies. Resolving these inconsistencies requires adequate aggregation strategies, resulting in condensed views on events. Performing such complex tasks manually is cumbersome, clearly calling for automation.

Although information extraction from the web has been studied before, to our knowledge, no previous work exists providing comprehensive solutions for all of the outlined challenges at once. Query formulation, for example, has been examined by Endrullis et al. [6]. They analyzed the problem of formulating specific queries to retrieve information on consumer products from the web through textual search interfaces. Similar to our approach, they employed alternative product labels to generate multiple queries for the same product. Focusing on fact extraction, Banko et al. created the TextRunner system, capable of efficiently analyzing millions of web pages [3]. By applying a naïve Bayes classifier, they extracted millions of facts from a large collection of web pages and estimated their correctness. In contrast to our work, neither inconsistencies between these facts are examined, nor their temporal dimension. Augmenting facts by their temporal scope has been studied by Wang et al. [19]. They extracted additional year information within the same sentences containing the facts, using wikipedia articles and web pages. Talukdar et al. [18] utilized frequency changes of fact occurrences in time-stamped data to scope facts, again at a yearly granularity. Both beneficially employed temporal constraints for scoping facts, e.g. birth is before death. While the achieved temporal granularity might be sufficient for facts like *isMarriedTo*, it is insufficient for facts describing shorter-termed or current events. Exploring the temporal dimension of documents (and their contained facts) has two perspectives: (1) the time described within the document and (2) the publication date. Although there are temporal taggers available focusing on the first per-

WWW'14 Companion, April 7–11, 2014, Seoul, Korea.
ACM 978-1-4503-2745-9/14/04.
http://dx.doi.org/10.1145/2567948.2579043.

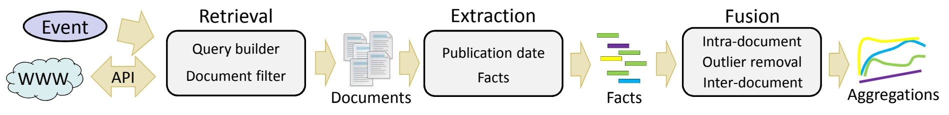


Figure 1: Framework overview.

spective [17], we observed poor accuracy for recognizing the publication date, tested on a sample set of web pages. SalahEldeen et al. studied the second perspective, utilizing temporally tagged links to the questioned document, hosted on third-party sites [14]. Hence, only web pages already linked on these sites can be temporally aligned, a crucial limitation not present in our approach, as we use the page content itself. Furthermore, applying their approach results in solely rough estimates of publication dates, since the utilized link creation dates constitute only upper bounds. Moving from web pages to user-generated content, Sakaki et al. created a system to trace earthquakes and typhoons reported on Twitter [13]. They applied particle filters to aggregate tweets into spatiotemporal trajectories describing evolutions of event locations. In contrast to web pages, tweets are already temporally tagged, easing information aggregation.

In this paper, we suggest a three-step framework providing solutions for retrieval, extraction, and aggregation. The framework's input are events – defined by type, date, and location – along with training examples of the demanded facts. The output are collected facts, aggregated and temporally aligned. We use search engines for web source identification and utilize regular expression-based heuristics for temporal alignment. For fact extraction, we apply pattern matching in dependency graphs. The specific focus of our framework is on time-dependent fact aggregation despite inconsistencies. Here, we employ curve fitting and sliding aggregates as suitable fusion strategies. Furthermore, our solutions are capable of forecasting further development of facts. We explain and evaluate the framework using a comprehensive study in the domain of natural disasters. In the study, we search for casualty numbers and are able to return their temporal evolution with satisfying high accuracy. We consider 33 earthquakes, 4752 search engine queries and analyze 61 083 documents, leading to 696 extracted facts per event on average over a period of 30 days. We also show that, for our case study, curve fitting is a proper aggregation method producing quality forecasts.

The contributions of this paper are threefold: First, we present a configurable framework for extracting event-related facts from the web. Second, we suggest suitable methods for each step: document retrieval, information extraction, and time-aware aggregation. Third, we apply our framework to a real-life case study. Altogether, we see our main contribution at the first proposal, implementation, and evaluation of a complete pipeline for solving the complex problem of extracting temporal, event-specific facts from web sources.

Section 2 gives a brief overview of our proposed framework. The three individual steps are described in Sections 3, 4, and 5, respectively. In Section 6, we describe our case study and present evaluation results. Section 7 discusses our findings and concludes.

2. FRAMEWORK OVERVIEW

Our framework is designed to find, extract, and aggregate numerical facts from the web describing long-lasting events. An example is the number of casualties during and after natural disasters. The framework especially focuses on the temporal evolution of these facts, providing a consistent view despite uncertainty and changes over time. It has three major modules, each performing one step: document retrieval, fact extraction, and information fusion (Figure 1).

(1) The retrieval module takes an event as input and returns a set of event-relevant HTML documents. As we employ search engine APIs to retrieve such documents from the web, the module contains a query builder generating appropriate queries for the event (see Section 3.1). The module is complemented by a document filter removing probably irrelevant documents (see Section 3.2). (2) The extraction module takes a set of documents as input and initially extracts for each document its structure, i.e. the title, the description and the content. Since we later require temporal aligned documents, the module contains a regular expression-based heuristic for publication date extraction (see Section 4.1). Furthermore, we apply machine learning methods to extract contained facts (see Section 4.2). The module's output are accordingly structured documents and extraction results. (3) The fusion module takes as input a set of temporal aligned facts and returns a condensed view over time. Here, we utilize a set of time-aware strategies to deal with inconsistencies in the extracted facts. This includes outlier detection (see Section 5.2) and intra-/inter-document fusion (see Section 5.1, 5.3).

3. FINDING RELEVANT DOCUMENTS

There are basically two options available for retrieving event-relevant documents from the web. Self-crawling (a subset of) all documents and applying some established retrieval model like vector space model [15] is the first. The second option is to invoke search engines via API calls. Since we do not know in advance which sites report on the desired event, option one would require extensive crawling, a laborious task. Hence, we included option two in our framework, namely adapters for Bing's and Google's search APIs. Utilizing established search providers also has the advantage of potential low latencies between the publication and the retrieval of new articles. Their high crawl frequency is of great help for analyzing current events.

3.1 Query Builder

The goal of the query builder is to generate appropriate queries based on the event, resulting in high precision and recall. Here, recall also has a temporal dimension, as we are interested in retrieving documents covering the full period of the event. In our framework, Queries are automatically generated based on the event type, location, and date. We further utilize alternative terms for each argument, known as query expansion [12]. For example, the 2012's Philippines earthquake can be queried by 'earthquake Philippines 2012' or 'quake Negros February'. As API providers limit the number of results returned, sending multiple queries helps to increase the document recall and event coverage.

3.2 Document Filter

To increase the precision of the returned results, we included several event type-independent document filters into our framework. These filters are configurable. Examples are (1) a regular expression-based URL filter and (2) a document size filter. In our study, we focus on news-like articles, therefore dropping documents whose

(1) URL consists of the domain name only, assuming that news articles have identifiers as path component. So `http://www.example.com/` will be removed, `http://www.example.com/articleId=123` not.
(2) size exceeds 500 kB, a limit determined by page sizes found in an external news articles collection.

4. FACT EXTRACTION

To enable fact extraction from retrieved documents, we first convert the HTML code into text strings, distinguishing between title, description, and content. The first two are defined by their respective HTML tag, whereas the actual page content is extracted by Boilerpipe [7], a boilerplate removal library. Boilerplate here denotes non-content elements on a web page like advertisements or navigation bars. The text extraction process implicitly discards non-HMTL documents. We also remove documents whose text do not contain one of the desired type strings like 'quake' for earthquakes. The reasons for this are two general observations: (1) Search engines also return results not containing all query terms and (2) event-relevant documents contain at least one type-specific key word. For example, there is hardly any article thinkable of reporting about an earthquake, without including the string 'quake'.

When describing events by facts, their correctness highly depends on the considered point in time as truth evolves over time. In other words, extracted facts like "4081 people are involved" are nearly rendered useless, if the temporal dimension is unknown. Hence, our framework uses a regular expression-based heuristic for temporal alignment of news-like documents and their contained facts, outlined in Section 4.1. In Section 4.2, we present the machine learning methods utilized for fact extraction.

4.1 Publication date extraction

Typically, news-like documents contain one dedicated line reporting on the publication date. Examples are 'February 6, 2012 -- Updated 2315 GMT (0715 HKT)' or 'Posted at 02/06/2012 4:40 PM | Updated as of 02/06/2012 5:10 PM'. Although we suspect a limited number of possible line formats in general, specifying a regular expression for each would be a cumbersome and error-prone task. Instead we apply date expression-specific stemming as a preprocessing step before the actual publication date extraction. This rule-based stemming removes all characters in text lines which do not belong to possible date expressions, reducing the number of required extraction patterns substantially. Stemming the two examples results in 'February 6 2012 2315 GMT 0715 HKT' and '02/06/2012 4:40 PM 02/06/2012 5:10 PM'. After stemming, regular expressions are applied for the final date extraction, constructed from external data.

In general, we select the first date found as the publication date of a document, excluding dates before 1995 (acting as pre-WWW boundary) and after the fetch date. If we detect two date expressions in one line, we suspect a combination of creation and modification time, propagating the most current one. With more than two hits, we ignore this line completely. Consequently, all documents without publication date are excluded from further processing.

We also try to detect the correct time zone if contained in the date expression. While terms like '+1300' or 'GMT' uniquely identify time zone offsets, there are many abbreviations which do not. For example PST denotes both, Pacific Standard Time (UTC−08) and Philippine Standard Time (UTC+08), differing in 16 h. Depending on the time-sensitivity of the examined facts, this bias might have significant influence on the final results. To minimize the effect, we apply the average offset for non-unique abbreviations, as listed on Wikipedia[1]. If we cannot identify any time zone, we use the event's time zone derived from the location.

Furthermore, we included a configurable publication date-based document filter in our framework. It allows to remove documents created probably too early to be valid, targeting at the time delay between the event and possible mentions on the web. For processing earthquake events, we apply a 15 min filter, since we do not expect informative articles earlier after the event.

4.2 Fact extraction

For fact extraction, we utilize the method presented in [5]. This method allows to extract arbitrary facts from texts, formalized as complex n-ary relationships, a technique called relationship extraction [2]. Here, n denotes the number of entities – single words or word groups – used to express the fact. Complex indicates that not all n entities are required to form a valid relationship tuple. Taking reports on casualties as example, the sentence 'The death toll is now at least 32, with 467 injuries' contains two facts: $\geq$ 32 killed and 467 injured. These facts can be formalized as 3-tuples [modifier, quantity, type], resulting in [at least, 32, death toll] and [–, 467, injuries].

The automatic fact extraction consists of three steps: First, we recognize all entities (relevant words or word groups) by evaluating them against a lexicon and regular expressions. Second, we infer semantic relationships between pairs of entities by matching patterns in dependency graphs [8]. Here, patterns consist of shortest paths [4] between two entities. The results of this pattern matching step are entity graphs with edges between all pairs of related entities. By finding maximal cliques [9] in these entity graphs, the binary relationships are finally merged into tuples of the desired n-ary relationship. The required entity lexicon and dependency patterns are automatically learned from annotated training texts, provided by the user. Hence, configuring the framework to extract different facts just requires to provide different training data.

In our case study, we extract casualty numbers, encoded as 3-tuples [modifier, quantity, type]. We utilized previously annotated articles for training [5]. We skip documents containing more than 25 fact tuples, a configurable limit derived from the same training articles. This filter is intended to separate event-specific articles from compilation-like ones, describing more than one event[2].

[1] `https://en.wikipedia.org/w/index.php?title=List_of_time_zone_abbreviations&oldid=516840413`
[2] `http://news.bbc.co.uk/2/hi/2059330.stm`

5. MERGING EXTRACTED FACTS WITH RESPECT TO THE TIMELINE

Extracted facts are inherently inconsistent as they were sourced from multiple documents, which were retrieved from different web sites and were published at various points in time. Furthermore, facts might be additionally distorted due to irrelevant documents (see Section 3), wrong temporal alignment (Section 4.1) or erroneous fact extraction (Section 4.2). Our framework contains a set of time-aware strategies to deal with these inconsistencies adequately, providing a consistent view on events. First, we reach consensus on the document level by a frequency-based aggregation (see Section 5.1). This is complemented by applying a median-based outlier detector to discard unlikely values for the examined facts, explained in Section 5.2. Finally, we fuse all remaining facts coming from different documents by curve fitting, outlined in Section 5.3. The result is a time-dependent function describing the evolution of the requested facts.

5.1 Intra-Document Fusion

The first step in our fusion strategy is to aggregate for each document all extracted facts of similar type into one value, called intra-document fusion. The underlying hypothesis is that each document, even if it reports different values for the same fact type, can be reduced to one most-likely value. For example, articles reporting on disasters might contain the official casualty number, an estimate from on-site units and some historic numbers from previous disasters in this region[3]. All numbers are equally extracted, but in most cases the desired information is the official count. We observed that generally key information in news-like articles are already contained in the title, the description or the first sentence/paragraph of the content [20]. Also, key facts are often repeated across these article elements. Hence, we use the most frequent value or, if ambiguous, the first value as the fusion result on the document level.

5.2 Outlier Detection & Removal

The next step is to detect among fused facts those which are hardly correct, called outliers, which will be ignored in later processing. We do this by comparing them with facts previous in time, resulting in an adapted version of an online median filter [10]. Median filters in general have the advantage of not assuming any specific probability distribution of the values, supporting even discontinuities.

Ordering the documents according to their publication date creates a temporal sequence of fused facts, more precisely a sequence of numerical values attached to facts. For each value we calculate the median of the last n values (including this value) and mark the value as outlier if its below a fraction or above a multiple of that median.

In our case study, we determined the outlier detector parameters from our training set (see Section 6). We found a window size of $n = 9$ to be appropriate and label all values outside the interval $[0.5 * median, 2.0 * median]$ as outliers.

5.3 Inter-Document Fusion

The final step in our fusion strategy is to combine the facts of all remaining documents – the inliers – into functions describing the fact evolution over time. For this inter-document fusion, we employ curve fitting with specific functions, based on the type of facts. We assume that by encoding typical evolutions of facts, i.e. linear increase, as event type-specific functions, this a priori knowledge later helps to fuse noisy inliers into a consistent view on events. Our framework supports various function families, each defined by an expression containing a number of parameters, e.g. the affine linear family $f(x) = a \cdot x + b$ with parameters a, b. The parameter values are optimized based on the inlier values, utilizing a least-squares approach. The optimization process can be influenced by assigning weights to inliers affecting the fitting error calculation. For example, confident values might be assigned weights above the default and doubtful values weights below. The resulting parameterized functions are directly applicable to return exactly one value per point in time as requested by the fusion process. Furthermore, they allow extrapolation into the future. Additionally, the framework includes configurable sliding window aggregates like mean, e.g. applicable as baseline. Here, the aggregate of the last n inliers (window size) form a step function, returning exactly one value per point in time as well.

For natural disasters, we observed in the reference set of the training data (see Section 6) that evolutions of casualty numbers are very well approximated by saturation functions [11]. In our experience, the best results are achieved with Monod's hyperbola $f(t) = a \cdot t \cdot (b + t)^{-1}$. Here, t is the time elapsed since the event and a, b are the parameters to be optimized. Since we presume that reported facts are becoming more trustful with time elapsed, we assign for each value a weight proportional to its age, penalizing fitting errors in the early stages of events less than at the end. To determine if curve fitting is a suitable inter-document fusion strategy, we utilized a sliding window average ($n = 5$) as baseline. The window size n is tuned on the training set.

6. EVALUATION

We evaluated our framework in a comprehensive case study from the domain of natural disasters, tracking casualty numbers after earthquakes. Using Wikipedia articles as reference, we compared evolutions of casualty numbers with extracted numbers from Bing search results, processed by the framework. Here, we focused on two scenarios: The real-time scenario covers the quality of current fact values returned by the framework whereas the forecast scenario deals with the quality of future values.

6.1 Data sets

We used Wikipedia articles as reference as their stored revisions allow to access previous article versions. These revisions represent descriptions of the same event at different points in time, forming an excellent gold standard for analyses of time-dependent facts. Focusing on earthquake events between 2006 and 2012, we retrieved all article revisions and automatically extracted the casualty numbers contained in the infoboxes. These infoboxes contain semi-structured key-value pairs, allowing exact parsing. After that, we semi-automatically removed obviously incorrect values, e.g. due to vandalism. From all fetched articles, we selected 33 earthquakes as data set. The selection criteria were: ≥ 10 casualties and at least five revisions. Ten earthquakes were randomly assigned to the training set and all remaining events to the evaluation set. Based on the training set, we deter-

[3]http://www.bbc.co.uk/news/world-asia-16901385

mined a suitable framework configuration (see Section 6.2), later applied in the evaluation.

We processed each of the events with our framework. The query builder generated up to 100 queries per event. The framework invoked the Bing Search API and fetched the top 100 result per query. Both searching and downloading were executed in April 2013, resulting in 1851 documents per event on average.

6.2 Framework configuration

We used the training set – consisting of Wikipedia references and downloaded documents – to adjust the framework according to our case study. Here, the goal was to minimize the difference between the framework's output and the Wikipedia reference. By testing several configurations, we determined

- appropriate limits for the median-based outlier detector (Section 5.2),
- a suitable function class for approximating casualty number evolutions (Section 5.3),
- the baseline for inter-document fusion (Section 5.3).

Figure 2 plots the framework's output for one training event, demonstrating close approximation of the Wikipedia reference by curve-fitted inliers. The concluded configuration was then used to process the evaluation set.

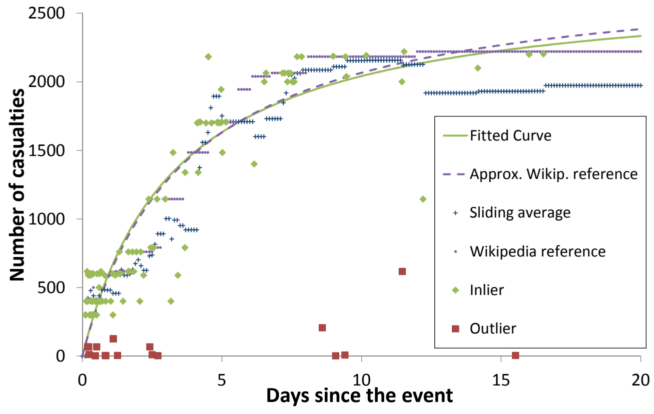


Figure 2: The framework's output for the 2010 Yushu earthquake. Single points are the result of the intra-document fusion (Section 5.1), classified by the outlier remover (Section 5.2). The purple step function is the Wikipedia reference, the purple dashed line the approximated Wikipedia reference. The blue step function (crosses) is the sliding average baseline, calculated from the inliers (green). The solid green line is the result of the inter-document fusion, the fitted curve of the inliers (green).

6.3 Real-time scenario

The fist scenario we evaluated in our case study is the real-time scenario: How well does the framework report the current fact value, given all facts prior in time? In general: Given all documents/extracted facts until a point in time t, what quality can be expected for $f(t)$, the framework's output for t? We measure this quality at t by calculating $e(t)$, the absolute value of the relative error between our Wikipedia reference value $r(t)$ and the framework's output $f(t)$, i.e. $e(t) = |r(t) - f(t)| \cdot r(t)^{-1}$. The reference values $r(t)$ are derived from the extracted casualty numbers from the Wikipedia revisions, continuously approximated by a Monod hyperbola (see Section 5.3). This approximation has the advantage of better representing possible intermediate values not mentioned in Wikipedia.

Figure 3 depicts the real-time quality of our framework, comparing curve fitting and sliding average (baseline). It shows that both fusion strategies are capable of returning current casualty numbers with a difference of less than 20 % compared to the Wikipedia reference. Furthermore, curve fitting significantly outperforms the baseline in the analyzed interval [0.5, 5] in terms of median errors ($p = 9.5 \times 10^{-5}$; Wilcoxon signed-rank test).

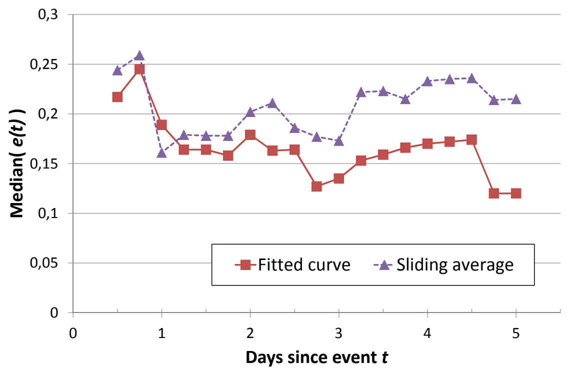


Figure 3: Real-time differences for curve fitting and sliding average compared to the Wikipedia reference. Both were calculated on the evaluation set, aggregated across events by median.

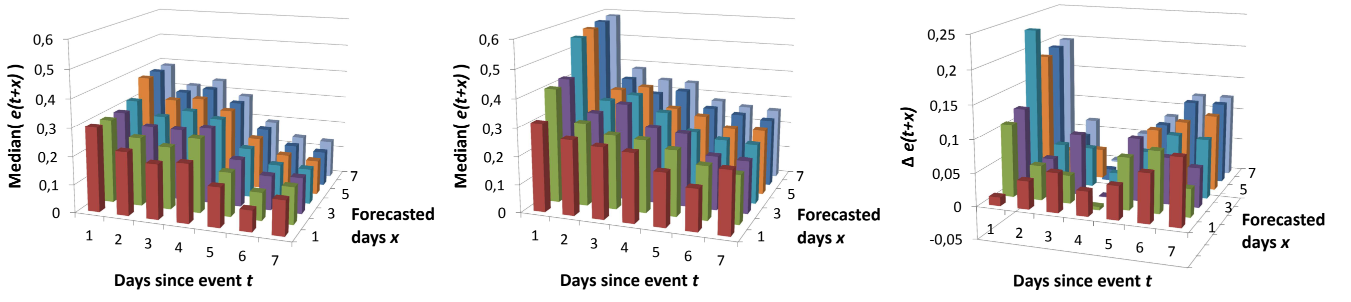
6.4 Forecast scenario

The second scenario we evaluated is the forecast scenario: How well does the framework predict future fact values, given all facts until now? In general: Given all documents/extracted facts until a point in time t, what quality can be expected for $f(t + x)$ with $x > 0$, the framework's output for $t + x$? We measure this quality at $t + x$ by calculating the unsigned relative error $e(t+x)$ as defined in Section 6.3.

Figure 4 illustrates the forecast quality of our framework, comparing fitted curves with sliding averages. Taking the forecast error at $(1, 1)$ as starting point, it shows that both fusion strategies are able to return forecasts with a difference of about 30 % compared to the Wikipedia reference. Here, (t, x) indicates the forecast error at day $(t + x)$ since the event, taking the facts until day t as known. Increasing both t and x, one can observe two trends: (1) Increasing the number of days to forecast increases the error as well and (2) increasing the number of days since the event decreases the error. Although both strategies follow these trends in general, curve fitting produces a smaller overall error than the baseline (Figure 4c). The average difference between both is 0.063 in favor of curve fitting, based on the data points $(2..7, 1..7)$. Especially the forecasts after the first day $(1, 2..7)$ expose large advantages of fitted curves resulting in error differences up to 0.234.

7. DISCUSSION & CONCLUSION

Extraction and aggregation of event data from the web is a complex task. It includes source identification, fact extraction, and time-aware aggregation of these facts. The



(a) Fitted curve

(b) Sliding average

(c) Difference between (b) and (a); positive values indicate less forecast error by curve fitting than sliding average

Figure 4: Forecast differences for curve fitting and sliding average compared to the Wikipedia reference. Both were calculated on the evaluation set, aggregated across events by median. The value at (x, y) means: Taking all facts until the end of day x, what is forecast error at the end of day $(x + y)$?

results of our case study show that our framework is capable of automating this task. We measured an overall error rate between 10 and 40 %, even holding for forecasting seven days ahead. As expected, predicting current fact values produces more precise results than forecasting with a maximum error rate of 26 %. These are surprising low error rates given the large number of – probably imperfect – processing steps within the framework and their complex interaction. We think that for many applications this error rate is acceptable, taking the costs of manual information gathering into account. For example after natural disasters, knowing the scale of casualties is more important for determining the required extend of relief operations than knowing the exact numbers.

Taking the current framework as a starting point, we see different future directions. We plan to apply our framework to other natural disaster like floods, assuming generalizable patterns in casualty number evolutions. Following the idea of generalization, it would be very interesting to include domain-independent fact extraction methods, moving towards Open Information Extraction [3]. Additionally, incorporating user-generated content like tweets as novel information source might be beneficial for the framework's applicability [1] and output quality.

8. ACKNOWLEDGEMENTS

We thank Jirka Lewandowski for contributing to the curve fitting implementation. We thank him and Philippe Thomas for providing valuable feedback on drafts of this article.

9. REFERENCES

[1] S. Asur and B. Huberman. Predicting the future with social media. In *WI-IAT '10*, 2010.

[2] N. Bach and S. Badaskar. A survey on relation extraction. *LTI, Carnegie Mellon University*, 2007.

[3] M. Banko, M. Cafarella, et al. Open information extraction from the web. In *IJCAI*, volume 7, 2007.

[4] R. Bunescu and R. Mooney. A shortest path dependency kernel for relation extraction. In *HLT'05*, 2005.

[5] L. Döhling and U. Leser. Equatornlp: Pattern-based information extraction for disaster response. In *Terra Cognita 2011*, 2011.

[6] S. Endrullis, A. Thor, and E. Rahm. Entity search strategies for mashup applications. In *ICDE '12*, 2012.

[7] C. Kohlschütter, P. Fankhauser, and W. Nejdl. Boilerplate detection using shallow text features. In *WSDM '10*, 2010.

[8] C. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT, 1999.

[9] R. McDonald, F. Pereira, et al. Simple algorithms for complex relation extraction with applications to biomedical ie. In *ACL '05*, 2005.

[10] P. Menold, R. Pearson, and F. Allgower. Online outlier detection and removal. In *MED99*, 1999.

[11] D. Ratkowsky. *Handbook of nonlinear regression models*. New York : M. Dekker, 1990.

[12] J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*. Prentice-Hall, 1971.

[13] T. Sakaki, M. Okazaki, and Y. Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In *WWW2010*, 2010.

[14] H. SalahEldeen and M. Nelson. Carbon dating the web: estimating the age of web resources. In *WWW2013*, 2013.

[15] G. Salton, A. Wong, and C. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11), 1975.

[16] J.-B. Sheu. An emergency logistics distribution approach for quick response to urgent relief demand in disasters. *Transportation Research Part E: Logistics and Transportation Review*, 43(6), 2007.

[17] J. Strötgen and M. Gertz. Heideltime: High quality rule-based extraction and normalization of temporal expressions. In *SemEval '10*, 2010.

[18] P. Talukdar, D. Wijaya, and T. Mitchell. Coupled temporal scoping of relational facts. In *WSDM'12*, 2012.

[19] Y. Wang, B. Yang, et al. Harvesting facts from textual web sources by constrained label propagation. In *CIKM'11*, 2011.

[20] Y. Watanabe, Y. Okada, et al. Aligning articles in tv newscasts and newspapers. In *ACL '98*, 1998.