

newsLens: building and visualizing long-ranging news stories*

Philippe Laban

UC Berkeley

phillab@berkeley.edu

Marti Hearst

UC Berkeley

hearst@berkeley.edu

Abstract

We propose a method to aggregate and organize a large, multi-source dataset of news articles into a collection of major stories, and automatically name and visualize these stories in a working system. The approach is able to run online, as new articles are added, processing 4 million news articles from 20 news sources, and extracting 80000 major stories, some of which span several years. The visual interface consists of lanes of timelines, each annotated with information that is deemed important for the story, including extracted quotations. The working system allows a user to search and navigate 8 years of story information.

1 Introduction

Complex news events unfold over months, and the sequence of events over time can be thought of as forming *stories*. Our objective is to generate, from publicly available news articles, story outlines and visualizations that help readers digest and navigate complex, long-lasting stories across a large number of news articles. We attempt this construction of stories by building a dataset with multiple news sources, exploiting the overlap in coverage by different sources. Our contributions include:

1. A method for creating a dataset of articles from multiple sources across a decade from scratch,
2. A topic detection method that handles interruption in topics,
3. A novel way to name stories, and

4. A method for clustering, rating, and displaying quotations associated with the stories.

The demo is available at newslens.berkeley.edu/

The remainder of the paper is organized as follows. Section 2 presents current related research work. The aggregation of the dataset and the creation of the timelines is explained in Section 3. Section 4 presents the interface with the created timelines, as well as the extraction process for the information shown. Finally Section 5 concludes the paper and presents future work directions.

2 Related Work

Related work includes prior methods for generating stories from topics, for visualizing stories, and for summarizing news.

Topic Detection and Tracking refers to techniques to automatically process a streamable dataset of text into related groups called topics. In the context of news, the topics detected and tracked are commonly called stories.

Swan and Allan (2000) use the Topic Detection and Tracking (TDT) and TDT2 datasets, consisting of 50,000 news articles to produce 146 stories, called clusters. The clustering process is done using named entities and noun phrases, as opposed to unigrams. They report an inability to merge clusters if there are large gaps in time with no articles, and their algorithm does not group documents in an online fashion.

Pouliquen et al. (2008) build a large dataset of news articles, named the Europe Media Monitor (EMM). Their topic detection creates local clusters in each language. The monolingual stories are then linked across languages to form global stories. A reported drawback is the clustering cannot handle merging and splitting between disparate

*In the proceedings of the 2017 ACL Workshop on Events and Stories in the News

topics and cannot mend gaps between stories that last more than 8 days.

Ahmed et al. (2011) propose an online inference model named Storylines that clusters articles into storylines which are assigned to broader topics. Emphasis is put on scalability with a goal of processing one article per second.

Poghosyan and Ifrim (2016) leverage keywords in social media to generate storylines, and Vossen et al. (2015) propose to use news timelines to build storylines, structured index of events in the timeline, and event relationships.

Visualizing news stories focuses on building a user interface to present a given story to help the user digest a complex story. Using the EMM dataset, Krstajić et al. (2013) propose a visual analytics system to represent relationships between news stories and their evolution over time. Each story element is represented as a tile in a vertical list. Over time (x-axis), the placement of story elements is adjusted on the vertical axis according to the level of activity in the story.

Shahaf et al. (2012) propose a “Metro map” view for a given story. Article headlines are selected in the story corpus to maximize coverage of pieces of information. The selected items are put in different “lines” of the metro maps, showing how the story developed. Only headline information is accessible on the produced metro map.

Tannier and Vernier (2016) build timelines for journalistic use. Based on a user query, documents are retrieved and dates are extracted from sentences. A timeline is built where peaks represents important dates, and key dates are annotated with representative article headlines and an image from the article when available.

3 The newsLens pipeline

In order to build news stories over long time spans based on a variety of news sources, there are two main challenges: an organizational challenge of collecting news articles, and an algorithmic and computational challenge of building the stories. We describe our solutions to both problems.

We first describe how we use the Internet Archive to recover a dataset of news articles. Given an article dataset, we propose a lightweight pipeline to process articles into topics in a streamable fashion, so that timelines can be updated as new articles are added. The pipeline we propose has the following stages: extracting key-

Table 1: Number of articles collected by source

Source name	# articles
reuters.com	1.2 million
allafrica.com	1 million
foxnews.com	475000
washingtonpost.com	440000
telegraph.co.uk	390000
france24.com	250000
nytimes.com	230000
cnn.com	140000
theguardian.com	51000
Other sources	166000

words from articles, creating *topics*: local groups of articles in time, solidifying the local topic clusters into *stories*: long-ranging sets of articles that share a common theme, and automatically naming the stories. We then present timing measurements for each step of the pipeline.

3.1 Collecting news articles

For each article in our dataset, we require some information, from which we can build the features needed for our processing. The minimum information required is: the publication date, the url, the headline, and the content of the article. Most common news sources build their news websites with specific patterns, to make their articles easier to index. For instance, CNN.com, France24.com and NYTimes.com article urls match the following regular expressions, respectively:

```
http://cnn.com/yyyy/mm/dd/*
http://france24.com/en/yyyymmdd*
http://nytimes.com/yyyy/mm/dd/*
```

We collected 20 such patterns from globally recognized English-language news sources, and collected all news articles matching these patterns through the Internet Archive’s advanced search interface. We start our collection on January 1st 2010 and collect until the present time. The publication date is extracted from the url pattern, and we access the news article’s webpage to extract the headline and content.

A somewhat unexpected complication we faced was the process of deduplicating some articles. Some news agencies publish up to 7 different versions of a news article, each with a very minor change (for instance, to the headline, or by adding or removing a single sentence to the content). Be-

cause we use counts of articles to measure importance and create stories, it is important to remove duplicate articles. We apply a simple but effective method:

1. For a given source, group articles into small ranges of time (e.g. 1 week),
2. Compute bag of word vectors for each article,
3. Transform the bag of words for each group into a tf-idf matrix,
4. If two articles are above a certain cosine similarity, they are assumed to be duplicates,
5. Retain only the most recent article, as it may have corrected information.

Roughly 10% of articles across all sources are deleted in the deduplication process. After deduplication, our dataset contains 4 million news articles in English, or an average of 1,500 articles per day. The detail of number of articles per source is given in Table 1. A study of how article duplicates are created and the types of modifications that news sources create would be interesting.

3.2 Generating the topics

3.2.1 Extracting article keywords

We use a standard method to extract keywords from an article’s content. Given a set of articles with no keywords, we represent each document as a bag of words vector. We apply a tf-idf transform on the bag of words corpus and select a word w_i in document d_j as a keyword for the document if the tf-idf score $S(w_i, d_j) > T$, where T is manually set. If we are trying to extract keywords for a large dataset, we process the articles in batches of a fixed size and randomize the order in which we take the articles. Each article is processed a single time. The keywords are lemmatized and lowercased. Although simple, this approach is effective: for the France24 news article with headline:

Battle to retake Mosul from Islamic State group has begun, says Iraqi

the keywords obtained are:

shiite, force, abadi, militia, mosul, iraq

3.2.2 Local topic graph

There is not one clear definition of when two articles are about the same “story” in news. Our goal is to cluster articles into local groups we call topics, which are then merged over time into stories. We define two articles to be in the same topic if they share several keywords, and are published in a close range of time.

We propose to group articles into common local topics by building a graph of articles. The algorithm for building the graph is:

1. For each article a_i over a small range of N days, prepare keyword set kw_i
2. Articles (a_i, a_j) are assigned an edge between them if $\|kw_i \cap kw_j\| \geq T_2$, where T_2 is a manually set threshold.

An example graph obtained over a range of 6 days is shown in Figure 1. The graph obtained is not connected and has several components. One can think at first that each component represents a story, however, it is possible for different densely connected topics to erroneously connect over a few edges. This can be seen on Figure 1, where two large components: the Ferguson and Hong Kong protests are loosely connected by a single edge. To avoid the problem of merging topics due to erroneous edges, we use a community detection algorithm, whose role is to find correct assignment of the nodes into communities that maximize a quality function on the communities obtained. We use a standard community detection algorithm, the Louvain method (Blondel et al., 2008), which is both lightweight and efficient at finding the correct clusters. It can be seen in Figure 1 that the Louvain method correctly assigns the two protests to different communities.

3.2.3 From topics to stories

So far we have presented a method to group articles into topics that are local in time. However, it is not computationally tractable to process the graph for a large number of days, given that we have a total of $N \simeq 3000$ days to process. Apart from the computational complexity, we would like a streamable method where adding new articles updates already existing stories and creates new ones, while avoiding recomputing all stories from scratch. The method we propose to merge topics into long-ranging stories is two-fold: a sliding

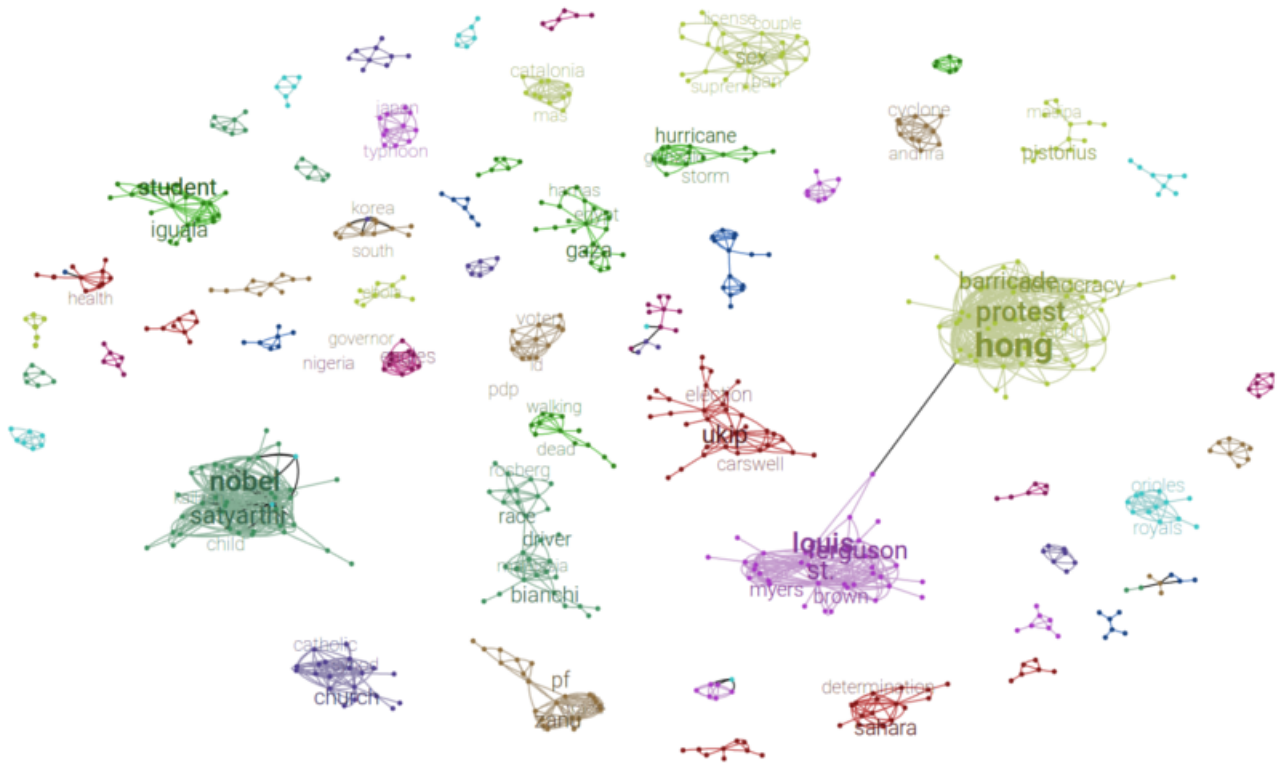


Figure 1: Local topic graph from June 10th 2014 to June 16th 2014. Nodes on the graph are news articles, edges are placed according to our method. Color of the node represents the topic assigned by community detection. Even though the Ferguson and Hong Kong protests form a single connected component, they get assigned to different communities. Keywords are placed on the display for convenience of the reader.

window to enlarge the topics, and a topic matching process for stories that might be interrupted in time.

The first step is to run the local topic assignment in chronological order using a sliding window. For instance, if we choose $N = 5$ for the number of days in a local graph, and 50% for the window overlap, the topic assignment is first run for days 1 to 6, and then run for days 3 to 8, etc. This sequence of overlapping graph clustering creates interesting dynamics. Linking, splitting, and merging are three phenomena we believe are important for story generation from topics.

Linking consists of assigning a topic from a preceding graph to a topic in the current graph: given a cluster in the current graph, if a majority of nodes in the cluster have previously been assigned another topic (in a previous graph, because of the sliding window), no new topic is created, and the cluster is assigned to the old topic, enabling topics to span more than N days. Linking happens for instance on the story about the French elections, that lasted more than 5 days: the articles from the first 5 days formed a topic, and as later articles appear,

they are linked into this topic that already exists. The story experiences no interruption greater than 5 days (the span of the window) from January 15th to May 10th 2017, and linking combines all articles in a single topic.

Splitting occurs when one topic is later on divided into 2 distinct topics: it can happen that a topic's start, a few initial articles are clustered together, and then diverge into clusters that are detected as separate by the community detection. In this case, the smallest cluster gets assigned to a new topic. An example of splitting: the shooting of Jo Cox (Brexit story), and the Orlando Shooting occurred within a few days of each other. The first articles covering each topic were at first assigned in the same topic, due to enough common keywords (shooting, death, killing, etc). However as each story grew with new articles, the topics became more distinct, at which point the topics were split.

Merging is similar to linking: if a current cluster found contains articles that have already been assigned to two distinct old topics, both topics are merged. An example of merging: the "Olympics

in Rio” and the topic related to “Athletes worried about the Zika virus” were at first separated, but as the Athletes arrived in Rio, the stories were merged. This does not occur as often as linking and splitting.

A story is what emerges when many local topics are linked or merged. With linking, we see how local topics can be connected into stories with an unbounded time span. As long as a topic has new articles appearing continuously, all articles are linked to the same topic, and the story grows.

The assumption that a story must be uninterrupted is constraining, as some stories can have arbitrarily large gaps in time. Consider the “MH317 Malaysia Airline plane crash” story shown in Figure 4, where new evidence was found a few months after the crash, and then again years after the crash happened. The second step for creating stories is to merge topics into a common story if they do not overlap in time but are similar enough in keyword distribution. We build a vector $v(t_i)$ for topic t_i which contains the counts of keywords in all articles of topic t_i . When a new topic t_j is created, its similarity to old topics is computed using a cosine similarity:

$$\text{sim}(t_i, t_j) = \frac{v(t_i) \cdot v(t_j)}{\|v(t_i)\| \|v(t_j)\|}$$

If the similarity is above a threshold T_3 , and the two topics do not overlap in time significantly, the topics are merged. The final topics obtained after these two steps represent the stories we will display in our interface. The choice of T_2 and T_3 affect the precision and recall of the algorithm. Increasing T_2 reduces the number of edges on the graph, reducing the number of articles placed in topics. In our implementation, we choose high thresholds ($T_2 = 4$, $T_3 = 0.8$), which limits the number of errors (high precision). The drawback is that only 10% of articles of the overall dataset get assigned to topics. When setting $T_2 = 3$, the number of articles in topics raises to 20%, but we expect more incorrect topics to be created.

3.2.4 Naming stories

Finding a good name to represent the story that can encompass several thousands of articles is challenging. We propose a simple system based on observations of what makes a good title for a topic. Here are examples of good titles we want to be able to pick: “North Korea nuclear tests”, “Ukraine crisis”, “Ebola outbreak”,

“Brexit vote”, “Paris attacks”. The features these names have in common are:

1. A story name is a noun phrase,
2. It contains a proper noun (entity),
3. It contains a common noun or word, and
4. One of the words is abstract (test, crisis, outbreak, ...).

For each headline in our story, we extract all maximal noun phrases and assign a score to each. For example, in the headline below (from telegraph.co.uk), noun phrases are underlined:

Pakistan frees Taliban prisoners to help Afghan peace process

Notice that noun phrases such as “peace process” and “prisoners” are not proposed as they are enclosed in a larger (maximal) noun phrase. The highest scoring noun phrase is chosen as the name of the story. Here are the features used to score a noun phrase p :

1. $f_1(p) = 1$ if there is a proper noun else 0
2. $f_2(p) = 1$ if there is a common noun else 0
3. $f_3(p) = \log_{10}(\text{count}(p))$, where $\text{count}(p)$ is the number of occurrences of phrase p in all headlines of the story
4. $f_4(p) = \sum_{w \in p} f(w)$, w are the words in p , $f(w)$ is the frequency of w in the titles
5. $f_5(p) = \max_{w \in p} \text{abstractness}(w)$, where $\text{abstractness}(w)$ is a word abstractness measure (Kato et al., 2008)
6. $f_6(p) = \text{length}(p)$, number of words in p

The final score is then computed as a linear combination of the features:

$$\text{score}(p) = \sum_{i=1}^6 \lambda_i f_i(p)$$

We choose the λ_i manually, and $p_{\text{final}} = \text{arg max}_p \text{score}(p)$. The five titles presented above are results for some of the major stories available in the system.

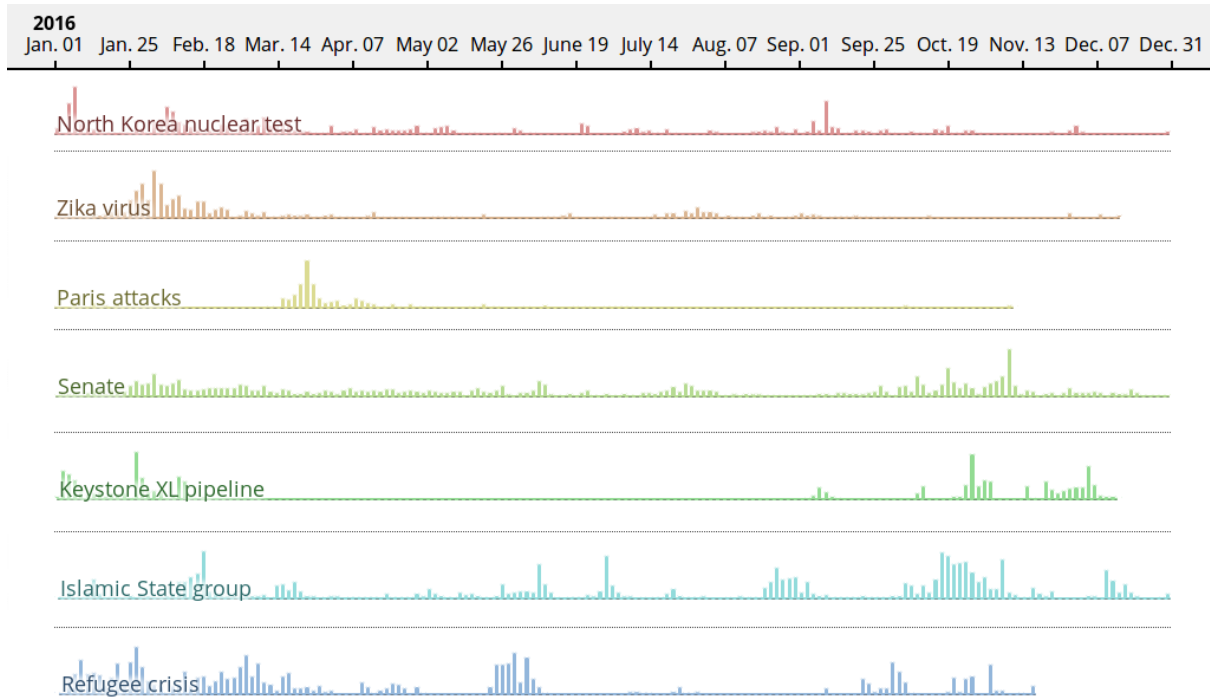


Figure 2: “lanes” interface. The 7 stories with most articles in 2016 are shown in timeline format.

Table 2: Timings of the pipeline. Time per unit, is a time per processed element. Total time is when running the pipeline on the entire dataset.

Process name	Time per unit	Total time
Internet Archive	4 min / source	80 min
Populating articles	0.05 sec / article	2.3 days
Extracting keywords	0.01 sec / article	12 hrs
Creating stories	2 sec / day	4 hours
Naming stories	0.02 sec / story	20 min

3.3 Processing Times

The processing speed determines the system’s capacity, if it is run in real-time. Table 2 presents the speed per unit for each stage of the pipeline, as well as the total time spent when processing 20 sources, with 4 million articles over 7 years.

4 Visualizing stories with lanes

We have now presented a method to retrieve 4 million news articles and organize them into more than 80000 stories. Many of these stories have hundreds or thousands of articles. We are posed with the visualization challenge of displaying content in an understandable manner. The following section introduces *lanes*, the interface we propose to represent stories. Lanes is composed of three components: a timeline, article headlines and quotes tiles. Figure 3 presents two example

lanes generated by our system.

4.1 Story timeline

The overall interface is framed on the x-axis representing time, each element added has a given x-position representing its occurrence within the story. We use a timeline as the main visual representation of the topic. The x-axis represents time, and the y-axis represents the number of articles in a given short period of time. This timeline creates a shape the user can identify the story with. Figure 2 shows the timelines of the 7 stories in year 2016 with most news articles. The assumption we follow is that major events in a topic lead to more news articles in a following short period of time, which can be made prominent in the timeline of the overall topic by a peak. For example, in Figure 2, it appears that the most active periods for the story “Keystone XL pipeline” are in February, October and November 2016.

The timelines of Figure 2 help the user see “when” action occurred in a given story. The following two subsections present the annotations added when a user clicks on a chosen timeline. The annotations help understand the “what” and the “who” of the timeline, respectively.

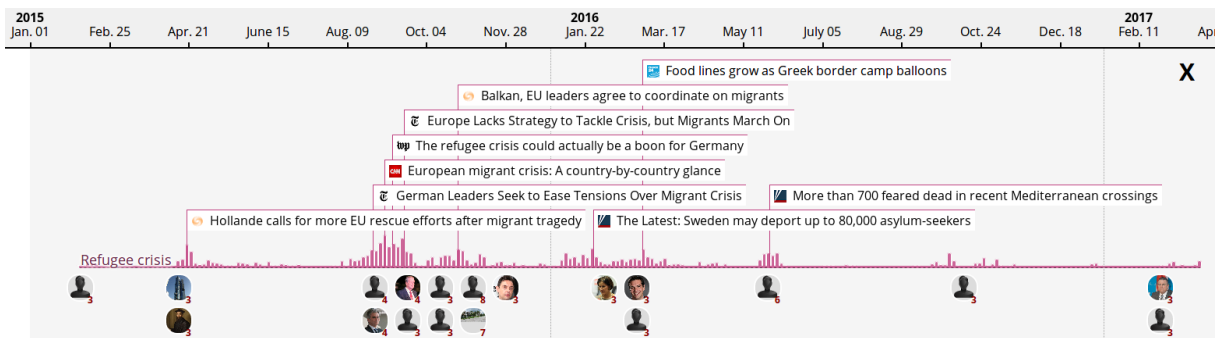


Figure 3: “Refugee crisis” story. Top to bottom: time legend, article headlines, timeline, and quote tiles.

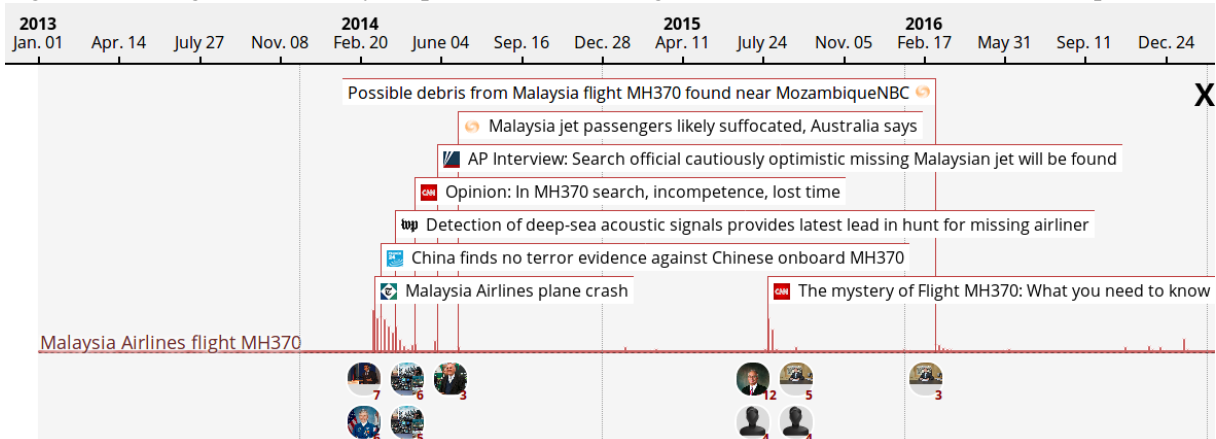


Figure 4: Timeline of the “Malaysia Airline flight MH370”, it has large time gaps with no articles.

4.2 Headline selection

Because we assume that peaks in the timeline of the story correspond to key times in the story, we propose to annotate these points for the reader. We sample news articles from peak periods of the story and add their headlines as annotation to the timeline. This allows the user to get an idea of what occurred during that period of the topic. The headline is clickable and takes the user to the article’s original URL. This enables the user to access articles about a topic that can be several years old. When selecting which article to display for a given peak, we randomly sample an article. Added to the article headline is an image icon representing the logo of the news source, which helps the user know the source of the headline at a glance.

There can be stories where many peaks happen in a short period of time, in which case the visualization would become cluttered. We impose a hard constraint in the visualization: headline annotations cannot overlap, and they are placed on a number of “rows” above the timeline. A maximum number of headline rows is allowed, and if a headline cannot be placed because of a lack of

space, it is not displayed. Headlines are placed in decreasing order of their peak heights, so that more “more important” peaks get placed first.

4.3 Quote ranking and selection

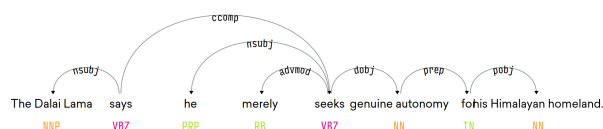


Figure 5: Dependency tree of a quote sentence illustrating how extraction process. This figure was generated using a modified version of displaCy.

We assume showing headlines annotations on the timeline helps the news reader answer the “what” of the story. We are experimenting with adding additional kinds of information to the interface. The first of these is quotations extracted from the article that are assumed to be important. Quote extraction is an active field of research (Pouliquen et al., 2007; O’Keefe et al., 2012). Our objective is to build a simple system to experiment with ranking and displaying the quotes. This process is done in 3 steps: entities are extracted, quotes for these

entities are extracted and then grouped and scored for importance.

We extract entities from all articles using an NLP library named spaCy. In order to reduce entity duplication, we proceed with a simple entity linking process leveraging Wikidata (Vrandečić and Krötzsch, 2014). Each entity string is searched through Wikidata’s search interface. Wikidata provides unique identifiers that match the search query. The first identifier in the query result is associated with the entity string. This allows us to merge entities such as: “Obama”, “Barack Obama”, “Mr. Obama”, etc

Entity disambiguation is a complex task, and although Wikidata is a first step in resolving entities, it also introduces errors. For instance, many news articles mention “Washington” as the author of a quote. When searching for Washington in Wikidata, the first entry that appears is “George Washington” instead of the city of Washington D.C. Additional patterns verifying the span of life and entity types could be put in place, but overall, this is a complex task and we will introduce more sophisticated entity recognition in future work.

Once entities are extracted, the next step is to attribute quotes to the entities. To extract quotes, we look at each individual sentence in our corpus and determine whether it is a quote by a known entity. The method for quote extraction is the following:

1. The sentence is parsed into a dependency tree
2. Check if the subject (NSUBJ) of the root verb of the sentence is a known entity
3. Check if the lemma of the root verb is in a predefined list (say, tell, state, ...)
4. Check if the root has a complementary clause
5. If all checks are validated, extract the pair (entity, quote)

For example given the sentence from a Reuters article:

The self-exiled Dalai Lama says he merely seeks genuine autonomy for his Himalayan homeland.

The dependency tree for this sentence is shown in Figure 5. We can see that for this sentence, all three conditions are met and the quote pair extracted is: (Dalai Lama, “he merely seeks genuine

autonomy for his Himalayan homeland.”). The dependency parsing is also achieved with the spaCy library.

This process does not extract all quotes as the pattern recognition we propose is fairly rigid. For now, we accept the low recall for a high precision in the quotes extracted, as we assume users would react more negatively to erroneous quotes than missing quotes. This produces on average 2 quotes per news article, which can represent thousands of quotes for a single story, which is too much to show to users. We propose a simple way to cluster quotes together to find important quotes.

Quotes are transformed into bag of words vectors, and the tf-idf transform is applied to the quote vector corpus. Quotes can then be compared using a cosine similarity measure. Two quotes are judged to be in the same “quote cluster” if they are from articles that are close in time, and they meet a minimum cosine similarity.

Once quote clusters are obtained, the size of the cluster is our measure for the quote cluster’s importance. This assumes that a quote that is mentioned by several journalists from various sources has more importance in the story.

We can now rank quotes in order of importance and show a limited number of quotes in the “lanes interface”. Each quote cluster is represented by an image tile of the entity speaking. When clicking on a tile, a frame showing the list of quotes in the cluster opens. Figure 6 shows one result of opening a quote tile: four quotes from the cluster are displayed, as well as the source from which the quote is extracted. Clicking on the quote opens the article from which the quote was extracted. In this






Iran said: its nuclear program is for peaceful energy purposes only	
Iran said: it only wants to harness nuclear energy for peaceful purposes	
Iran said: its nuclear program is for peaceful energy purposes only	
Iran said: its nuclear program is for peaceful civilian energy purposes only	
Iran said: its nuclear program is for civilian use ,	

Figure 6: Interface that opens upon a user’s click on a quote. Quotes shown were assigned to a common cluster in the story named “Iran nuclear talks” example, we can see that the quote cluster contains

quotes from Reuters, CNN and the NYTimes. The phrasing of each quote is slightly different, showing that sources modify and specify detail in their quote.

The lanes interface presents the stories as timelines annotated both with headlines at key times, as well as quotes representing main actors within the story.

5 Conclusion and Future Work

We have presented a method to build a dataset of news articles over a long range of time from several sources and an efficient, novel algorithm for organizing millions of articles into stories that span long time ranges, despite gaps in coverage. These stories are named with a simple but effective algorithm and visualized using a lanes metaphor, providing the user with a way to view each story in more detail.

Future work includes an assessment of the accuracy of the story creation algorithm: both the accuracy within stories, verifying that articles within a given story are related, and across stories, verifying that story humans would agree with the stories we propose. We also plan to continue refining the user interface and assess it with journalists, media analysts and other relevant end users: we will compare our interface with other news aggregator systems such as Google News, to assess the usability of this approach.

Future work will also leverage the considerable related work on event detection and event pattern understanding, and incorporating that into the story creation process.

Finally, source bias and information validity are important, in the context of alternative news sources and social media. An interface that presents the facts with the source of the information in a transparent way, as well as the results of calculating biases of news sources from a computational perspective is a future direction of interest.

References

- Amr Ahmed, Qirong Ho, Jacob Eisenstein, Eric Xing, Alexander J Smola, and Choon Hui Teo. 2011. Unified analysis of streaming news. In *Proceedings of the 20th international conference on World wide web*. ACM, pages 267–276.
- Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment* 2008(10):P10008.
- Makoto P. Kato, Hiroaki Ohshima, Satoshi Oyama, and Katsumi Tanaka. 2008. Can social tagging improve web image search? In *International Conference on Web Information Systems Engineering (WISE)*. Springer, pages 235–249.
- Miloš Krstajić, Mohammad Najm-Araghi, Florian Mansmann, and Daniel A Keim. 2013. Story tracker: Incremental visual text analytics of news story development. *Information Visualization* 12(3-4):308–323.
- Tim O’Keefe, Silvia Pareti, James R Curran, Irena Koprinska, and Matthew Honnibal. 2012. A sequence labelling approach to quote attribution. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, pages 790–799.
- Gevorg Poghosyan and Georgiana Ifrim. 2016. Real time news story detection and tracking with hashtags. In *Computing News Storylines Workshop at EMNLP 2016, Austin, Texas*.
- Bruno Pouliquen, Ralf Steinberger, and Clive Best. 2007. Automatic detection of quotations in multilingual news. In *Proceedings of Recent Advances in Natural Language Processing*. pages 487–492.
- Bruno Pouliquen, Ralf Steinberger, and Olivier Deguernel. 2008. Story tracking: linking similar news over time and across languages. In *Proceedings of the workshop on Multi-source Multilingual Information Extraction and Summarization*. Association for Computational Linguistics, pages 49–56.
- Dafna Shahaf, Carlos Guestrin, and Eric Horvitz. 2012. Trains of thought: Generating information maps. In *Proceedings of the 21st international conference on World Wide Web*. ACM, pages 899–908.
- Russell Swan and James Allan. 2000. Automatic generation of overview timelines. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, pages 49–56.
- Xavier Tannier and Frdric Vernier. 2016. Creation, Visualization and Edition of Timelines for Journalistic Use. In *Proceedings of Natural Language meets Journalism Workshop at IJCAI 2016*. New York, USA.
- Piek Vossen, Tommaso Caselli, and Yiota Kontzopoulou. 2015. Storylines for structuring massive streams of news. In *Proceedings of the First Workshop on Computing News Storylines*. pages 40–49.
- Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM* 57(10):78–85.