Constructing Timeline for News Events
EECS 486 Term Project Final Report

Guyi Chen
University of Michigan
chenguy@umich.edu

Zhiming Ruan
University of Michigan
ruanzhim@umich.edu

Yichen Yang
Univeristy of Michigan
yyichen@umich.edu

Qiyue Yao
University of Michigan
qiyueyao@umich.edu

Zhen Yu
University of Michigan
yzdsoy@umich.edu

Abstract

Our project identifies news events from the newsfeed such as CNN according to user queries, and models the development or evolution among those events as a timeline. There is only limited research focus on our project topic, but we have conducted literature review to get ideas from current research on related subfields in IR as topic summarization and topic detection and tracking. We use the preprocessed data offered by Kaggle.com to construct our database for this project and collect further data from Google search. One approach to obtain the desired timeline is to evaluate and extract all related events and then order them by timestamp; the other approach is to recursively get each historical event. Our result is visualized as a web application which takes in queries as the input and gives the user a timeline on the interface. Since no large annotated corpora related to our result is available, we perform a small experiment to compare the constructed timelines against manually constructed timelines to evaluate the effectiveness of our approach.

1 Project Description

People are often interested in knowing the development of the news event when they search certain event keywords online or when they browsing through their newsfeed such as CNN, FOX, etc, especially when the growth of information is exponential nowadays. However, most search engines only return a series of most relevant results and organize them into a hierarchical structure, and there is no development organization of these retrieved results. Our project is interested in identifying news events from news stories and model the development or evolution among these news events as a timeline.

The goal of our project is to construct a prototype news retrieval system and provide the users with a timeline of the news events related to the users query. The scope of our project is to find existing news datasets and build our database using them as the basis for our retrieval system. We performed data cleansing and wrangling to these datasets as needed. Those datasets preferably are composed of all the news that is within the last 10 years from well-known newsfeeds. Then we performed query processing on users queries. The core of our project is to retrieve the relevant results from our database and construct an event development graph which shows the development of the news events. This event development graph is transformed into a timeline of the news events in the end. The results of our project are also evaluated against chosen metrics, such as recall and precision. The end product of our project is a web page. The users are going to enter their query words in a search box, and then the web page is going to invoke the API and retrieve the relevant information from our database. The constructed timeline are displayed as the result, and specifically, we used flask API and jinja html template to create the timeline and visualize the processed data.

The novelty and core of our project is constructing the news event development network. We are assuming that every news event in a given network has a timestamp associated with it. The best way to represent this kind of development network is using a directed acyclic graph. Each event is considered as a node in the network. Event A has an outgoing edge to event B if event B is evolved from event A, and event A happens before event B. In order to evaluate if event B is evolved from

event A, we are going to check the content similarity between the two events and also the time proximity between the two events. In order to retrieve all the news events in a network, we would use different searching methods. Figure 1 displays part of the sample event development graph for the news affair of Beslan school hostage tragedy that happened in Russia in 2004.

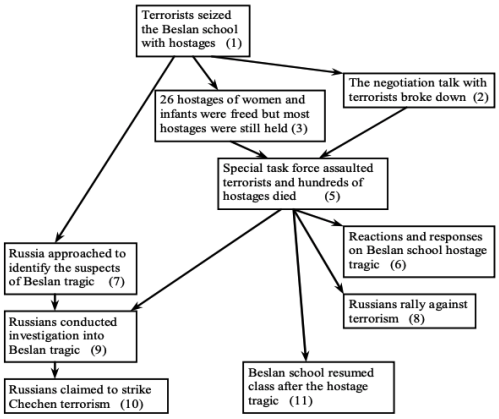

Figure 1: The partial event evolution graph for Beslan school hostage crisis. The numbers in the bracket indicates their temporal orderings.

2 Related Work

Limited research has been conducted in this particular project topic, but a fair amount of work has been done in related subfields in IR as topic summarization and topic detection and tracking (TDT). The biggest innovation of our project is the pipeline flow, which fits the natural human intuition to track an event development. Certain steps of the pipeline relates to previous works. TDT and our project differ in the direction of the pipeline, where our project traces back a given event and TDT detects a new event based on previous events. Topic summarization differs with our project in the final output as a summary rather than a development.

Previous work has considered the use of event graphs of news stories (Yang and Shi, 2006; Glavas and Snajder, 2014) where they retrieved related news events and then constructed the directed evolution relationships rather than a flat hierarchical relationship. Yang and Shi (Yang and Shi, 2006) focused on analyzing the temporal and distributional properties of news events. Their method builds on given temporal information. By calculating cosine similarity of events, choosing events with small temporal distances and balancing with document distributional proximity due to event burst, their proposed model managed to perform well without a large training dataset, yet there lacks a detailed evaluation part in their paper. They chose event pairs that are most similar by cosine similarity, but we realized that this would not provide a well spreaded event development, so we set our similarity threshold to be least similar in a certain range after careful experiment to retrieve previous news. Glava and najder (Glavas and Snajder, 2014) based their model on the contextual level and used machine learning to captured news information. They extracted anchor info by training discriminative classification model on EvExtra corpus, extracted argument info with existing patterns into (agent, target, time, location) and extracted temporal relation into (before, after, overlap, equal) by training regression models. Graph kernels are implemented to generate event graphs based on these extracted information. This machine learning method requires large labeled data to perform well, so we instead used keyword extraction (S. Rose and Cowley, 2010) on the articles for information and recursive search to retrieve previous event. Based on the professional property of news articles and adjusting keyword/phrase length, it provides enough information for article extraction. Combining temporal data, we achieved similar results for temporal relation.

Previous work on event summarization also focused on sentence similarity and temporal information similar to Yang and Shi. Chieu and Lee (Chieu and Lee, 2004) ranked interest of sentences based on sentence similarity using cardinal calculation and ranked burstiness based on date duration. After removing duplicate sentences, a summary of a certain event related to the given query was generated. Rather than removing duplicate sentences or even news, we prevented searching very similar news in the stage of choosing next queries rather than truncating later.

Work on TDT has some relevance to our project if we consider each processing step separately. Previous work (Y. Yang and Lattimere, 2000) generally used machine learning algorithms and large labeled training data similar to Glava and najder. In the preprocessing stage, it used the most commonly learned document representation - vector of weighted terms and TF-IDF scheme. Then variants of kNN are used to train models for detecting

incoming news. In an earlier version, we also used document representation to preprocess the Kaggle dataset for information retrieval. However, the performance was not satisfactory partly due to the dataset. Then we also combined temporal information into the information retrieval process for better performance. Work by Swan and Allan (Swan and Allan, 2000) used an algorithm calculating χ^2 value based on number of documents containing a feature on each day, total number of documents and df for the feature. If the χ^2 value is above a threshold, then it is marked as potential member of the cluster. A standard hierarchical agglomerative clustering method is then applied to determine the cluster feature. The big idea is similar to our per retrieval in the recursive search process. We generated several keyword phrases for each article, and for every phrase of new pairs, we use word embedding (J. Pennington and Manning, 2014) to represent the phrases for calculating similarity. If the value is above a threshold, then they are marked as related. Yet, we then choose the pair with least similarity to construct next retrieval query. Details are presented in following sections.

3 Data Methodolgy

Our datasets contain pre-processed news datasets, data collected from Google search.

Pre-processed news dataset was used in our first stage. It was used to verify whether the system could work in this situation and test the effectiveness of our model. In addition, we also use the local dataset to make test on the precision of the output.

For example, the dataset on Kaggle contains 143,000 news articles from 15 American publications, ranging from 2011 to 2017. The data are stored in CSV format and we can download it and store it locally for use. The columns we are interested in are the title, date, URL, and content.

When pre-processing the dataset, we first separated the content from other columns since we only want to use the content to extract the key words for the next step. Then we could preprocess the content into tokens. In this step, we tried different combinations of pre-processing ways: we tried pre-processing with and without remove stopwords; without stemming, stemming with Porter Stemmer or stemming with Snowball Stemmer; transfroming all the words into lowercase or not. In these different combinations, we found out that removing stopwords, using Snowball Stemmer and applying lowercase to all the words will provide the best result for information retrieval and we choose this combination as our final pre-processing model.

During the dataset pre-processing, we found out that using words as tokens directly will cost huge amount of time in the next steps. To fix this problem, we chose to use index to represent the tokens rather than use words. Therefore, we performed the word-index transformation process and stored the two index-word and word-index dictionaries for later use. After using index represeting the words, we found out that the system was much faster than the former one. It will cost 80% less time than using words directly.

When using the local dataset for testing, we found out that local dataset was too small to contain enough serial events. Therefore, we chose to construct the serial news event by ourselves. We used Wikipedia to find different serial news event. Then we extracted the serial news from the Wikipedia and used the search engine to find the corresponding news content. Finally, we put the raw material of the news into our local dataset and massed up the order of these news so that we could use these news for precision testing.

After our model is validated and tested in the first stage, in order to enable our program to deal with more recent news and try to make it respond in real-time, we extended our datasets to CNN search results and ordinary Google search results.

For ordinary Google search, we were using the *Custom Search JSON API*. We went through the web page with BeautifulSoup in python to extract useful information. By focusing on news pages only, we only interested in the news title, dates, and contents. In the final product demo, we actually restricted our search range into only several main news agencies in order to give more time efficient result.

In the data from Google search API, we found out some interesting data which were deeply related with each other rather than directly related. For example, when we search Iraq War, we could also retrieve the news related to ISIS which is closely and deeply related to Iraq War. We thought this kind of data was also important for constructing timeline and we chose to keep it into our news event.

4 Method Description

We applied two approaches to get the time line of the development of the given event. The first approach is **searching and ordering**. The second one is **recursive searching**. In both approaches, we are given an event keyword such as the *Iraq War*, and we are required to give preceding events related to the keyword event ordered by time or cause and effect.

4.1 Searching and Ordering

For the searching part, we crawled all the articles that are related to the keyword event by our CNN API. Then we filtered out duplicated search results. This is done by a divide and re-rank pattern. First, we generated several time ranges by time division. Denote N to be the predefined number of articles we want to show in the search result. Let's say the articles crawled fall into a time region $[t_{min}, t_{max}]$. The length of each new time region would be $\Delta_t = \frac{t_{max}-t_{min}}{N}$. The set of time regions resulted from time division is then

$$T := \{[t_{min}+i\Delta_t, t_{min}+(i+1)\Delta_t]\}_{i=0,1,\cdots,N-1}$$

For each time range $T_i \in T$, we got a set of articles $D_i := \{d|t_d \in T_i\}$. For each $D_i, i = 0, 1, \cdots, N-1$, we ranked all the articles in the set by a weighted score of page ranking and $tf - idf$ similarity. The article with the top score was chosen as the representative of articles in the time range. Pseudo-code for this method is

1. Input: N, query Q
2. Initialization: $R = \{\}$
3. Crawl all the articles related to Q use CNN API which gives us an article set D.
4. Get the whole time range $[t_{min}, t_{max}]$ from D.
5. Divide the time range into a set of small time ranges

   $$T := \{[t_{min}+i\Delta_t, t_{min}+(i+1)\Delta_t]\}_{i=0,1,\cdots,N-1}$$

6. For each time range $T_i \in T$, get a representative article r_i, and append into R
7. Output R

4.2 Recursive Searching

In this approach, we got each historical event recursively. Once an event query is entered, we crawled all the historical events related to and happened before it. Then we extracted keywords in each website article and put all of them into a keywords pool. To choose next query keyword that indicate the event happened before the last query keyword, we first used *GloVe* word embedding framework (Jeffrey and Richard and etc. 2014) to embed each keywords. After that, we calculated the cosine similarity between each embedded vector representation of keyword and the vector representation of the query. The one with the smallest cosine similarity will be the keyword that with which the event associated is most likely to be the one happened before the last query event in our assumption. The recursive searching will be terminated if some criteria are satisfied. The pipeline for this algorithm is in Figure 2.

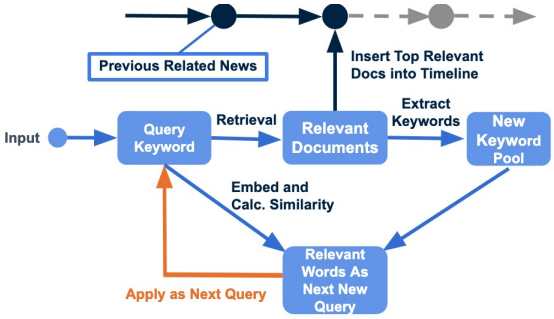

Figure 2: Pipeline for recursive searching

This approach integrates two models as its component. They are *keyword extraction* and *word embedding*.

4.2.1 Keyword Extraction

Since our pipeline should aim to be used for arbitrary news queries and crawled news data, we used document-oriented keyword extraction regardless of the corpora. The algorithm is based on rapid automatic keyword extraction (Swan and Allan, 2000). Keywords here means either a word or a phrase.

Major properties of keywords are content bearing as opposed to stopwords and frequent co-occurrence. Preprocessing of keyword extraction was removing the stopwords and tokenizing based on word delimiters. From this list of word tokens, we generated sequences of words based on phrase delimiters, which would be candidate keywords. Then for each word, calculated each word

score using the metric $degree/frequency$ favoring words in frequent co-occurrence. With the score for each word, we then calculated the total score for each phrase, simply by performing a summation. A sorted list of keywords was generated based on this score.

In the implementation of the pipeline, we limited keyword length to be 1-4 word phrases for title and 2-4 word phrases for the article content. When processing news in the Kaggle dataset, we performed keyword extraction on the title and the content separately. Keeping all keywords from the title and only keywords with scores greater than 10 from the content, which results in around ten keywords per news article.

4.2.2 Word Embedding

Word embedding in our project is done by the *GloVe* model. The statistics of words co-occurrences in a corpus provides the primary source of information to almost all unsupervised methods for learning word representations. How effectively and meaningfully these information is make use of distinguishes the existing word embedding models.

The *GloVe* model is based on the assumption that the co-occurrence of words w_1 and w_2 in the corpus of the third word w_3 is more indicative than the co-occurrence of w_1 and w_2 by themselves in terms of the relationship between words w_1 and w_2. Based on this assumption, the model uses a weighted least squares regression mode to learn the vector representation of each word with a cost function that measures the distance of the resulting similarity calculated by vector representations of two words and the similarity calculated by their co-occurrence with the third words in training documents (Jeffrey and Richard and etc. 2014).

5 Result and Evaluation

For both method described in the previous section, we built as user interface as a web app with jinja template as the front end. The results were visualized as timestamps with article titles and url links attach to a timeline. (Figure 3)

To the best of our current knowledge, we could not find any large annotated corpora that is specific to our purpose of producing a query-based timeline summary. In order to make an evaluation on this system, we perform a small experiment to compare the constructed timelines against manually constructed timelines from Wikipedia and various news sites. Note that we only consider the results within the time range that the evaluators cover, which means the articles of the comments and argument after the event are not included. Although those articles shown in the result are great reading materials of the users of our system, we exclude them in the evaluation part due to their subjectivity, which makes us hard to decide which results are relevant.

Figure 3: Result Demo for the Query "Yingying Zhang Disappearance"

In our scenario, the False Positive is the number of news in the timeline which are fetched by the program but not fetched by evaluators and the False Negative is the number of news in the timeline which is fetched by the evaluators but not the program. We evaluate our system based on the accuracy, precision, recall, and F1-Score.

No.	fptp	tp	tptn	Prec	Rec	F1
1	8	6	6	0.75	1	0.85
2	9	7	17	0.78	0.41	0.54
3	7	6	18	0.85	0.33	0.48
4	6	6	15	1	0.4	0.57
Ave	7.5	6.25	14	0.84	0.53	0.61

Table 1: Sample Query Performed Against Manually Constructed Result

Query:
1. Yingying Zhang Disappearance
2. U.S. China Trade War
3. Champion League Ronaldo
4. Trump meet Kim Jong Un

As is shown in Table 1, the precision of the queries are generally high, but the recall varies from query to query. For the queries describing a specific event such as "Yingying Zhang", recall gives better result because the keywords are rare in the database and our system can simply pick out them since almost all articles with the query words appearing are relevant. However, for global news such as U.S.-China Trade war, our system tends to miss a lot of relevant events.

6 Conclusion

For this project, we have developed a web interface that takes the user query input and constructs a timeline of the news event development as the output. The key components of this project are a retrieval system, a keyword extraction model, a word embedding model, and a recursive searching model.

6.1 Main Contributions of Our Project

When the amount of information is growing exponentially nowadays, it is crucial to have a system that filters out the irrelevant events and provides the user with most relevant and comprehensive information in a minimalist style. Our project does exactly that. The scope of our project is currently on news event; however, this could easily expand to other topics as well such as twitter feeds, etc. Our project's framework can help identify the key stakeholders/events in an event series. This is important and applicable in our life, such as in computer security when the researchers are interested in knowing who are the key stakeholders in the events and the development of the events.

6.2 Project Challenges and Accomplishments

One of the biggest challenges in this project is to design an evaluation system that effectively and accurately measures the performance of our system such as recall and precision rates at a large scale. Because it is extremely hard to find labeled data, assessing the performance of the system at a large scale is also difficult.

We also found that the performance of our system depended on the events. Performance is better for one targeted person query, around whom there is only one simple event development. In this case, we generated timeline from clustered news articles. For more broadly effective events like Iraq War, the performance is influenced by the recursive searching model where deviation is accumulated. The complex causes of the event also provides larger possibilities to be included in a simple timeline.

The initial idea was to develop a prototype Python program that will output a text file as the end product. One of our team member, Zhen Yu, developed a web interface that allows users to interact with our system and visualizes the timeline.

6.3 Considerations for Future Work

For this project, we only considered the most relevant documents among all the retrieved documents. However, in an event series where a cluster of events happened in a short amount of time, i.e. their timestamp is close to one another. This method might not be the most effective way. The consideration for future work could be designing a weighting system when news events are close together in terms of timestamps.

In addition, our system does not consider the evolvement of the words semantic. For example, when the user searches for 911, the user might be referring to the actual event not events where people call for help. The program could incorporate more NLP components in the future to better understand the query or the meaning of it.

Lastly, the web interface can include more features in the future for a more mature product. For example, the web interface could include the breaking news of the day from major news publishers, such as CNN, Fox News.

7 Individual Contributions

Zhiming Ruan	Implemented word embedding for the pipeline. Designed methods and the pipeline of the project. Active participation and completion of group discussions, meetings and poster presentation.
Yichen Yang	Analyzed and decided the data storing method for retrieval system. Implemented the data pre-processing system which uses indexes to represent the words to increase the speed of the system. Active participation and completion of group discussions, meetings and poster presentation.
Zhen Yu	Implemented the searching pipeline and visualized results on website. Evaluated the pipeline results. Active participation and completion of group discussions, meetings and poster presentation.
Qiyue Yao	Implemented keyword extraction for the pipeline. Researched and analyzed related work. Active participation and completion of group discussions, meetings and poster presentation.
Guyi Chen	Implemented the information retrieval system with three different weighting schemes and reweighting scheme. Active participation and completion of group discussions, meetings and poster presentation.

References

H. L. Chieu and Y. K. Lee. 2004. Query based event extraction along a timeline. pages 425–432. Proceedings of the 27th annual international ACM SIGIR Conference on Research and Development in Information Retrieval.

G. Glavas and J Snajder. 2014. Event graphs for information retrieval and multidocument summarization. *Expert Systems With Application*, 41(15):6904–6916.

R. Socher J. Pennington and C.D. Manning. 2014. Glove: Global vectors for word representation.

N. Cramer S. Rose, D. Engel and W. Cowley. 2010. Automatic keyword extraction from individual documents. 10(1).

R. Swan and J. Allan. 2000. Automatic generation of overview timelines. pages 49–56. Proceedings of the 23rd annual international ACM SIGIR Conference on Research and Development in Information Retrieval.

T. Pierce Y. Yang, T. Ault and C. W. Lattimere. 2000. Improving text categorization methods for event tracking. pages 65–72. Proceedings of the 23rd annual international ACM SIGIR Conference on Research and Development in Information Retrieval.

C.C. Yang and X Shi. 2006. Discovering event evolution graphs from newswires. *WWW06 conference proceedings*, pages 945–946.