Storytelling Graphs

Paul C. Nichols
Department of Computer Science
University of California, San Diego
pnichols@cs.ucsd.edu

Abstract

Stream-like data sources, such as a collection of RSS feeds, can be challenging to summarize as the corpus is ever growing and the nature of the data evolves over time. This project implements a web application for managing RSS feeds that addresses this challenge. LDA topic modeling is performed at daily intervals and linkages between learned topics from different days are computed to form the edges in what can be considered a "Storytelling Graph". This storytelling graph can be explored through the web application to answer a number of contextual questions from both the perspective of a topic and a document.

1 Introduction

The modern media landscape is a chaotic, fast-moving stream of content created by authors of all size and scale. Well known examples of streaming data sources include news content from major media outlets (i.e. through RSS feeds), the collective musings of Twitter's user base (i.e. the Twitter Firehose), and the click-stream generated from surfing internet (i.e. from network logging) .

It is becoming increasingly more difficult for news consumers, media professionals, and researchers to organize, follow, and identify the importance of each story or topic. Summarizing and contextualizing information in a data stream is a natural and important problem that arises. Various solutions such as newsmap.jp, Google Word Cloud, and Google News have been developed to visually organize and give weighting to topics and news stories. These solutions present information on topics in a real-time snapshot, but do a poor job with contextualizing topics or stories over a period of time.

The most popular form of summarization on a data stream is mining for trending topics, typically through an unsupervised learning process. Trending topics provide a high level view of the prominent themes present in a data stream during a window of time. A set of trending topics, or even multiple sets from different windows of time, can provide insight in isolation. However, contextual questions about a topic's lifetime, momentum, and evolution cannot be addressed. To answer these more advanced questions, topics must be linked

beyond the window in time they were discovered. This set of time sensitive topics and the linkages between them is referred to in this report as a "Storytelling Graph".

This project implements a web application to overcome the limitations of prior solutions by visualizing data from the storytelling graph in a way that emphasizes the role of time in a story and connectedness between topics rather than documents. Data is gathered from a collection of RSS feeds over a period of 60 days for 3 categories of news: general news, business, and sports. The web application's visualizations provide an easy to explore a dataset for the following contextual features:

- **Importance** : The prominence of a topic relative to its peers for a given day.
- **Lifetime** : The beginning and ending of a topic.
- **Momentum** : The importance of a topic over time.
- **Evolution** : A timeline view that connects related topics and documents over time.

2 Background

The foundation of this project rests on the ability to extract a set of relevant topics from a corpus of text documents. Given this set of topics, the construction of a storytelling graph is only slightly more complicated than the discovery of the topics themselves. However, not all approaches to topic discovery result in representations where computing linkages between topics from different days is straight forward. Further, the topic modeling should be sufficiently powerful to accommodate complicated real-world mixtures of topics. For both of the above reasons, Latent Dirichlet Allocation (LDA) is used on the bag of word representation of documents to mine for topics.

A brief tour of alternatives to LDA topic modeling is given to motivate its usage. Unsupervised methods for discovery and summarization of a corpus have a rich history rooted in information retrieval. Before the more abstract task of a topic discovery became widely known, document clustering was a well studied problem. Document clustering is used to find document similarities and partitions of similar documents in a corpus.

The vector space representation of a document is a common place to begin document clustering. The vector space representation of a document holds term counts by assigning a unique dimension to each term that exists in the corpus. Various term weighting schemes can be applied to the document vectors such as tf-idf weighting. These schemes enhance the similarity measures by reducing the contribution of unimportant words such as stop words. Similarity between two documents in vector space can be measured with a geometric interpretation either, by the angle between them (cosign similarity) or the distance apart (euclidean distance).

A good first step towards topic discovery is the K-means algorithm applied to corpus of tf-idf weighted documents in the vector space representation. K-means attempts to find a partitioning of the corpus into K groups such that the euclidean distance of a cluster's members is minimized. Iterative refinement from randomly initialized means can be used to approximate this hard problem. The most highly weighted term indices of the resulting K mean vectors from a clustering job loosely represent the topics present within the corpus. The approach is advantageous for streaming data because the learned mean vectors can be compared across clustering jobs from different time windows easily (using either cosign similarity or euclidean distance between the means). The approach is also desirable because it is simple and fast. However, the technique lacks the richness of topic mixtures by assuming each document can belong to only one mean vector.

Another early approach that improves upon the vector space representation is Latent Semantic Analysis (LSA). LSA computes the singular value decomposition of a document-term matrix to find low-rank approximations of the latent document and term matrices. The resulting matrix factorization has the property that when a pair of documents or a pair of terms are similar, they will be near (using cosign similarity or euclidean distance) in their respective vector spaces. LSA is good step towards finding a representation that can capture synonymy and polysemy, but is not suitable in the streaming paradigm where the process repeats and regular intervals. The resulting matrix factorizations are unique to the document-term matrix they were derived from. There is no straight forward way to compare either the learned term or the document representations from different time windows.

Probabilistic Latent Semantic Analysis (PLSA) addresses the limitations above with LSA by stepping away from the vector space representation of LSA to a probabilistic representation of terms and document. PLSA represents documents in the bag-of-word format where term frequency is maintained but not order and no geometric interpretation is applied. PLSA introduces a hidden random variable to model topics and then learns document-topic and topic-term distributions through the Expectation Maximization (EM) algorithm. PLSA can also overcome issues such as synonymy and polysemy through the topic, and has the advantage of allowing mixtures of topic (or cluster) assignments. Lastly, PLSA is advantageous in the streaming paradigm because the resulting topic distributions are interpretable and comparable across different time windows.

PLSA is the predecessor to LDA, the model used in this project. LDA improves upon PLSA by modeling the document probabilities and word distributions as dependent on Dirichlet prior distributions. This dependence on prior distributions enhances the generality of the distributions learned. An LDA model is more complicated and the distributions are commonly estimated using Gibbs Sampling.

3 Method

Queries on a storytelling graph may start and end from both the perspective of a topic and a document. Starting from a topic, the storytelling graph can be traced to another topic or down to a document through the topic weights. Likewise, starting from a document, the storytelling graph can be traced through the topic weights to another topic or again down to a neighboring document. Features of a trace through the graph, such as the length in time covered or the number of topics or documents traversed, serve as the basis for the answers to the contextual questions posed above.

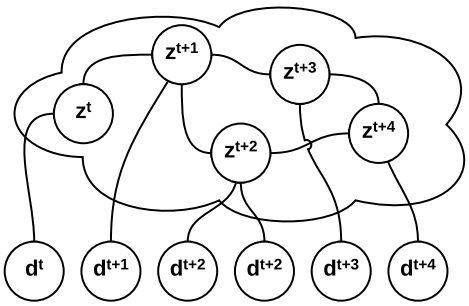

Figure 1: Storytelling Graph: An example graph over t to $t + 4$ days

3.1 Graph Creation

The starting point of graph creation is corpus of documents partitioned by a time interval over an indefinite period. Formally, we may represent this by the following notation where the maximum interval T is ever increasing.

$$\mathbf{C} = \{\mathbf{C}_0, \ldots, \mathbf{C}_T\} \tag{1}$$

$$\mathbf{C}_t = \{\mathbf{d}_0^t, \ldots, \mathbf{d}_N^t\} \tag{2}$$

$$\mathbf{d}_i^t = \{w_{i,0}, \ldots, w_{i,M}\} \tag{3}$$

For the purposes of this project, the partitions $\mathbf{C}_t$ for $t \leq T$ represent the documents partitioned by day. Each individual day's sub-corpus $\mathbf{C}_t$ is a collection documents where each document $\mathbf{d}_i^t$ is in the bag-of-word format. Superscripts are used to denote documents, topics, and distributions from a given day t except when describing running times. In this project LDA topic modeling is

Summary of LDA Variables	
$i < N$	Number of Documents
$j < M$	Number of Terms
$k < K$	Number of Topics
d_i	Document i
$w_{i,j}$	Count of Term=j in Document=i
z_k	Topic k of K
α_k	Document-Topic Prior Parameter
β_j	Topic-Term Prior Parameter
$\theta_{i,k}$	P(Topic=z_k | Document=i)
$\phi_{k,j}$	P(Word=j | Topic=z_k)

Table 1: Notation related to LDA topic modeling.

used to estimate from the daily corpus $\mathbf{C}_t$ the probability distributions of the the document-topic distribution, $\theta^t_{i,k}$, and topic-term distribution, $\phi^t_{j,k}$.

The storytelling graph constructed from the learned distributions follows the standard form: $\mathbf{G} = (\mathbf{V}, \mathbf{E})$. A unique feature of the storytelling graph is that the vertices in $\mathbf{V}$ can include both the documents and the topics. The graph structure follows directly from the conditional dependence graph of the LDA model, but includes additional grafts between topics across different days. The edges in $\mathbf{E}$ are undirected, as opposed to directed when representing conditional independence.

Edges are weighted by floating point values ranging between between $[0, 1]$. Edges between documents are always 0 (i.e. are not allowed). Topics may form edges with documents of the same day using the document-topic distribution. Edges between topics across different days are formed by computing the cosign similarity on the vector space interpretation of the topic-word distributions. Though blatantly not probabilistic, the cosign similarity between two probability distributions will at least always return a value between $[0, 1]$. These topic-topic edges are precomputed between the current days topics and all previous days topic within some limit (in this project, 15 days). The formal rules for constructing edge weights are as follows:

$$e(\mathbf{d}^t_i, \mathbf{d}^{t'}_{i'}) = \quad 0 \qquad \forall \quad i, t, i', t' \tag{4}$$

$$e(\mathbf{z}^t_k, \mathbf{z}^{t'}_{k'}) = \begin{cases} 0 & \text{if } t = t' \\ \frac{\phi^t_k \cdot \phi^{t'}_{k'}}{||\phi^t_k|| \, ||\phi^{t'}_{k'}||} & \text{if } t \neq t' \end{cases} \tag{5}$$

$$e(\mathbf{d}^t_i, \mathbf{z}^{t'}_k) = \begin{cases} \theta^t_{i,k} & \text{if } t = t' \\ 0 & \text{if } t \neq t' \end{cases} \tag{6}$$

3.2 Query by Topic

Starting from a topic $\mathbf{z_k^t}$, a natural query is to find the sub-graph of related topics across all time. This sub-graph can be thought of as the story a topic belongs to. This query performs a breadth first search of the neighboring topics from different days while the product of the edge weights along each path from starting to ending topic is above a threshold between [0,1].

```
Define: Subgraph()
Input:  Starting topic z_k^t,
        Topic threshold thresh_z
Output: Related topic set Z
Begin:
  Initialize Z = {},
             V = {z_k^t},
             F = {< z_k^t, 1 >}
  While F ≠ {}:
     Remove z from F
     For z' ∈ TopicNeighbors(z):
       Let w = e(z, z')*weight(z)
       If z' ∉ V and w > thresh_z:
          Z = Z ⋃ < z', w >
          F = F ⋃ < z', w >
     V = V ⋃ z
  return Z
End
```

Algorithm topic sub-graph of topic z_k^t

Querying the graph for a topic's sub-graph can, in the worse case, be somewhat expensive for long-lived stories. Breadth first search has a running time in the general case of $O(|E|)$. For a given day, there will be K^2W edges where K is the number of topics and W is the number of days compared. A long lived story can trace through T days. Therefore, the number of edges is $O(TK^2W)$, which is less than the possible $O(T^2K^2)$ edges if windowing were not applied. Better yet, the edges are pre-computed and only edges above the threshold are added. Typically much less than K^2W edges per day are used in this query.

3.3 Query by Document

Another useful query on the storytelling graph is the more classic problem of finding relevant neighboring documents given a starting document. Again a breadth first search is required. Recall that documents are connected to the graph only through their topic weights and that the interior of the graph is entirely edges between topics of different days. Therefore, the traversal of the

graph from one document to another may reuse the `Subgraph()` routine presented in the previous subsection.

A similar approach is taken to querying by document as by topic in that again the paths from document to document will be pruned by the product of the edge weights. To make the algorithm more efficient, a modified version of the `Subgraph()` routine is assumed where a set of starting topics can be supplied instead of only a single one. This modification does not change the algorithm or the running time.

```
Define: Similar()
Input:  Starting document d_i^t,
        Topic-topic threshold thresh_z,
        Document-topic threshold thresh_d
Output: Related document set D
Begin:
  Initialize D = {},
             F = {}
  For z ∈ TopicNeighbors(d_i^t):
    If e(d_i^t, z) > thresh_d:
      F = F ⋃ < z, 1 >

  For z ∈ Subgraph(F, thresh_z):
    For d ∈ DocumentNeighbors(z):
      Let w = e(d, z)*weight(z)
      If w > thresh_d:
        D = D ⋃ < d, z, w >
  Return D
End
```

Algorithm for relevant documents d_i^t

An interesting consequence of querying by document is that a document may be similar to another document indirectly through multiple topics. The indirect nature of the query makes the running time of the algorithm slower than querying by document. For a given day t, the number of documents retrieved by any topic can be at most the number of documents that day $|\mathbf{C}^t|$. Therefore, over all time and all topic-topic edges, a factor of $|\mathbf{C}|$ extra edges between documents need be explored. The total running time is $O(TK^2W|C|)$. Again, thresholding can be performed prior to updating the graph so many of the total possible edges do not exist.

3.4 Graph Metrics

Given a topic's sub-graph, there are a number of interesting metrics that can be used to analyze the story. The first query of interest is the lifetime of a

story. The lifetime of a story is simply the maximum topic time window less the minimum. Knowing the total time a story has existed can then be used to estimate what can be interpreted as density or velocity. Story density or velocity is defined as the number of topics in the subgraph divided by a time range. This time range can be the lifetime if examining a story in isolation or an arbitrary range if comparing multiple stories.

The number of documents a given topic connects to in the sub-graph provides a notion of size. A threshold is required to ensure documents are truly relevant to the topic, which is described in the next section. The number of relevant documents over an entire subgraph can be thought of as the story's mass. Putting the velocity and mass together, the notion of momentum can be borrowed from physics to assess how enduring a story is.

An alternative measure of size of a topic is the hyper-parameter α_k. In previous versions of LDA this parameter had to supplied, but now can be estimated from the corpus using a burn-in period. The topic modeling software used in the project performs this estimation.

For notation, let $\mathbf{S}$ be a topic's sub-graph, let $\mathbf{D_S}$ be the number of unique documents connected to the sub-graph with a path product above the threshold used in `Similar`(), and let Δ represent a time window. When Δ appears in a superscript it restricts items in a set to only those within that window. The various metrics are stated below in these terms.

$$\texttt{Lifetime}(\mathbf{S}) = \max_{z \in \mathbf{S}} \texttt{time}(z) - \min_{z \in \mathbf{S}} \texttt{time}(z) \tag{7}$$

$$\texttt{Velocity}(\mathbf{S}, \Delta) = \frac{|\mathbf{S}^{\Delta}|}{|\Delta|} \tag{8}$$

$$\texttt{Mass}(\mathbf{S}, \Delta) = |\mathbf{D}_{\mathbf{S}^{\Delta}}| \tag{9}$$

$$\texttt{Mass2}(\mathbf{S}, \Delta) = \sum_{z \in \mathbf{S}^{\Delta}} \alpha_z \tag{10}$$

$$\texttt{Momentum}(\mathbf{S}, \Delta) = \texttt{Mass}(\mathbf{S}, \Delta) \times \texttt{Velocity}(\mathbf{S}, \Delta) \tag{11}$$

4 Implementation

4.1 Backend Process

Processing the document stream is accomplished by a nightly process that downloads content from a collection of RSS feeds for the previous day. The entire process is composed a series of stages that happen sequentially. While the production environment schedules the process to handle a single day's worth of data from a cron job, the process can also be run in batch mode to process all data over a provided time range. Importantly, each stage in the processing pipeline is dependent only on the output from the previous stage.

The process begins with the sync stage by gathering the list of URLs for unprocessed data in the stream using the Google Reader undocumented API

(GRAPI). The GRAPI normalizes the many quirks and formatting issues present in a large set of RSS/Atom feeds and exposes a standard representation of a document stream. The output of the Sync stage is a mapping from URL to path on local storage the content should be downloaded.

The fetch stage of the processing pipeline is run after the Sync stage generates output. This stage is responsible for downloading HTML content, extracting the relevant article content from the raw HTML, and saving the result to local disk. Content downloading is managed by through the Apache Commons libraries. Article extraction was originally handled by extracting text content with an HTML parser. However, the resulting saved documents included too many artifacts of the hosting site such as advertisements, comments, and boiler plate content. The artifacts present in the document made the topic modeling perform poorly. To remedy this problem, a library known as "boilerpipe‘" was used to separate article content from artifact with an in-built decision tree classifier.

The next stage of the pipeline performs topic modeling on the daily corpus that was created locally from the Fetch stage. Topic modeling is handled by the ParallelTopicModel in the Mallet library. Documents are transformed into the bag-of-word format by converting characters to lower case, removing punctuation, and splitting on whitespace. A burn-in phase of 10 iterations is used to estimate the Dirichlet hyper-parameters. The learned topics improve qualitatively when the content is boosted by appending the a document's title 5 times. The number of topics, K, is chosen to be half the corpus size. This allows space for redundant themes to cluster on a single topic, but also the common situation where a topic has no history and is new or one-time event.

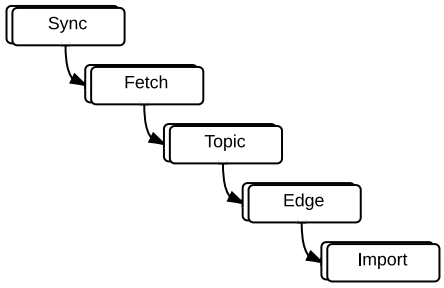

Figure 2: Sequential Stages of Data Collection/Processing

After the topic modeling has completed for the current day, edges are computed between topics of the previous days within a given window of time. As

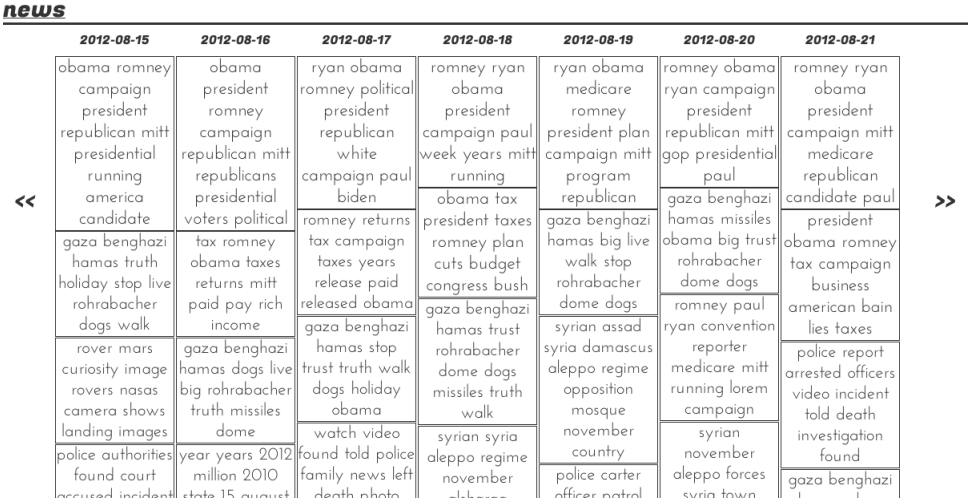

Figure 3: Screen shot of web application in browsing mode. A summary of the terms for prominent topics are displayed for each day. Topics with the most connections between other topics are displayed higher.

time progresses, the total time frame a similar topic can be detected is the window of days before and after the topic date. Similarity between topics is computed by treating the topic-term distribution as a vector space representation and computing the cosign-similarity.

Lastly, after all the previous stages have completed, the downloaded document data, topic distributions, and similarity measures are imported into a database. After the import process completes the max date is updated so that queries may be performed agains the new, fully imported day's worth of information. The database contains all information required by the web portal to let a user explore the data stream and answer relevant story graph queries.

4.2 Web Application

To emphasize the importance of time, the web application renders a tabular view of the top 10 terms for each topic over a 7 day period. Topics are sorted by top to bottom by a (non-scientific) weighted sum of the connection strength with neighboring topics by mass. The top 10 words of each distribution is shown to give the user an understanding of the topic. Users may "drill-down" into a topic that is interesting to view the subgraph of that topic. This "drill-down" view is another tabular view by day of the both the topics, and the linked titles of the articles belonging to the topic.

The web application is built around the standard Model-View-Controller paradigm whereby the business logic to query the database is separated from the presentation logic. The technology stack consists of popular HTML5 and

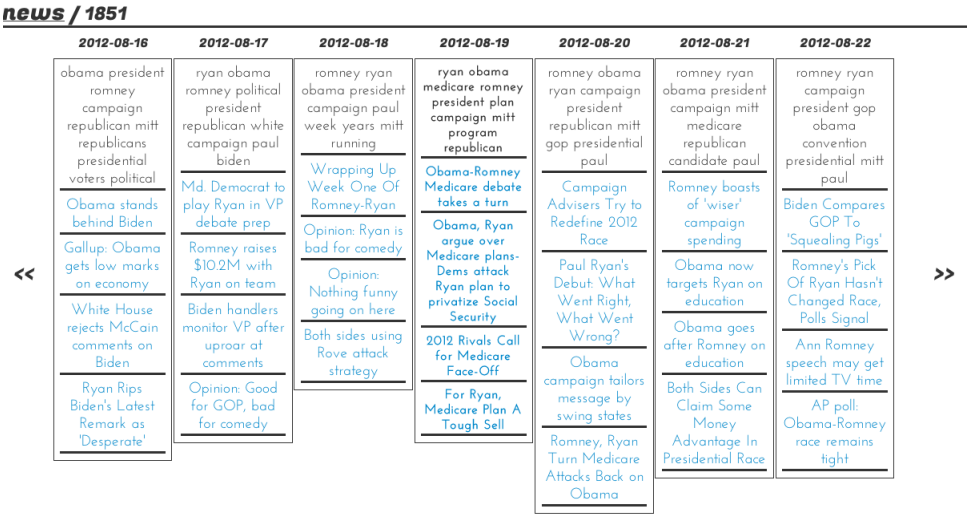

Figure 4: Screen shot of web application in "drill-down" mode. The links and titles of the relevant documents for a story are shown by day. The number 1851 is the topic ID.

JavaScript frameworks such as Bootstrap, JQuery in the View, a MySQL database as the Model, and the Dancer framework in Perl providing the Controller. To make rendering information fast, a caching layer provided by Memcache is placed between the View and Model to save results for common queries such as the weighted topics by day and popular sub-graph requests.

5 Evaluation

5.1 Popular Topics

The first item to confirm is that the topic modeling is effective at summarizing the relevant topics for a given day. Because LDA is an unsupervised method, the learned topics are already the best fit possible according to the optimization criteria (marginal likelihood). Therefore, this evaluation is more qualitative than quantitative and a judgement call must be made.

As mentioned previously, the first article extraction simply removed HTML content and returned text. This naive approach left many artifacts present in the document and the learned topics matched the raw content better than the thematic content. Examples of "artifact" topics include topics dominated by numbers, topics dominated by terms that related to the hosting sites (such as "CNN", "Fox" or "ABC"), topics related to advertisements, and topics related to comments sections.

These "artifact" topics can be removed for the most part with more data

Prominent "Artifact" Topics				Ambiguous Topics	
1	comments	fox	page	video	external
5	usa	news	search	watch	internet
3	today	twitter	close	2012	sites
2	news	facebook	missing	1	2012
4	life	report	found	july	network
6	comment	home	center	full	provided
7	blog	world	report	images	content
8	travel	friends	visit	photo	purposes
9	sports	air	links	aug	responsible
romney	contact	email	find	size	copyright
10	stories	article	public	news	ventures

Top terms for "artifact" topics using naive article extraction (left 3 columns). Top terms for ambiguous topics even after improved article extraction (right 3 columns). Both highlight situations where the topic modeling yields topics that are useless.

cleaning using the "boilerpipe" article extraction library. However, content problems persist for new reasons such as expired links and sites hosting video content with very little content. These ambiguous topics remain somewhat prominent. However, in the middle range of prominence the topics learned are in fact informative and effective at summarizing a topic. A few such examples of informative topics for a day are provided.

One basic approach to further improve the learned topics is filtering by manual inspection. The ambiguous topics reappear from day to day regularly and share almost identical distributions. By identifying a sample of individual topics that should be excluded from display and edge creation, the same method of similarity comparison used to link topics across days can be used to suppress future topics. This approach is used in the web application with good success.

5.2 Query by Topic

The next component to evaluate is the effectiveness of computing topic similarities. Querying the storytelling graph by topic requires a threshold for similarity that not straightforward to set. Again, because the method is unsupervised there are no explicit labels for when two topics should be joined that would allow the threshold to be optimized for accuracy. Instead the threshold must be set by a qualitative feel.

Plotting the density of the cosign similarity values for one day's topics with the neighboring topics provides some clues about where to search for a threshold. The density plot of the all topic similarities for one day reveal that the vast majority are near 0. Zooming in further to only similarities above .3 reveals a slightly more interesting structure. A surprising number of similarities are very

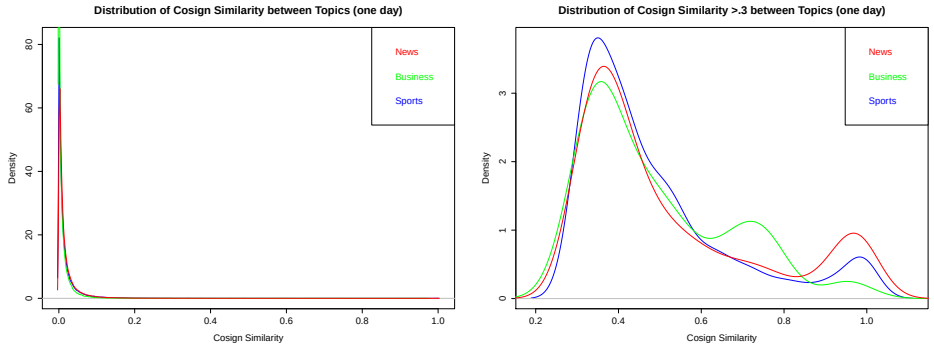

Figure 5: Density of topic similarity (left), and density of topic similarity greater than .3 (right). Nearly all cosign similarity values between topics are near 0, suggesting that even a low threshold of .3 has a large margin before it becomes too noisy.

near 1, a perfect match. Upon inspection these topics were found to be further "artifact" topics and will be suppressed manually as described above.

To get a sense of how topic distributions become less similar, this zoomed in threshold space is sampled by comparing a starting prominent topic to a neighboring topic for each similarity range: [.9, 1], [.8, .9], [.6, .8], [.4, .6], and [.3, .4]. The sampled distributions of increasingly lower similarity show qualitatively that a good threshold is likely somewhere between .6 and .4. Though the example with Hurricane Isaac is but a single instance, similar behavior is observed generally. At the lowest value in this range the neighboring topic begins to really diverge. However, tolerance to noise can be left as a parameter to the user.

5.3 Query by Document

Querying by document involves first gathering the prominent topics for a given document and then tracing through the topic-topic edges back down to another document. The topic in this case is used as a proxy to the document's actual content. This leads to the unusual problem where two documents can be similar topic-wise but less similar by their raw content. To quantify the performance of the indirect nature of querying by document, the method is compared to simple cosign similarity between two documents.

The method for comparison involves sampling documents from the three different datasets over a two week period and pulling back their most relevant neighbors. Document similarity is restricted to pairs higher than .45 to provide comparable noise reduction to the thresholds in the storytelling graph. The returned documents are evaluated as a cluster by the within cluster sum of squares criteria on the vector space representation.

Impact of Topic Similarity Top ten terms of similar distributions of decreasing similarity					
`start`	`.9`	`.8`	`.6`	`.4`	`.3`
isaac	isaac	isaac	storm	haiti	fire
louisiana	louisiana	lousiana	isaac	isaac	storm
hurricane	hurricane	hurricane	hurricane	crisis	friday
orleans	storm	flooded	national	hits	officials
storm	orleans	baithwaite	gulf	portauprince	wednesday
home	residents	home	orleans	storm	winds
braithwaite	home	street	coast	earthquake	center
residents	parish	residents	tropical	people	rain
plaquemines	flooded	pontchartrain	rain	topical	house
flooded	mississippi	floodwaters	winds	city	damaged

Table 2: Examination of topic similarity thresholds related to Hurricane Isaac. At the lower end of similarity (.3 and .4), the topics appear to be more about general weather than Hurricane Isaac.

Example of evolving story from topic perspective				
09-12	**09-13**	**09-14**	**09-15**	**09-16**
teachers	school	teachers	teachers	teachers
chicago	teachers	school	union	chicago
strike	**strike**	union	chicago	union
unions	chicago	chicago	school	school
schools	union	**strike**	strike	strike
public	children	children	schools	schools
school	students	board	**deal**	**contract**
union	district	kids	students	**emanuel**
students	schools	day	rally	**delegates**
teacher	kids	students	teacher	**lewis**

Table 3: Example from topic perspective of evolving story between September 12-16th about Chicago teachers striking. The story moves towards the conclusion with the union representative Karen Lewis negotiating a deal with Mayor Rahm Emanual.

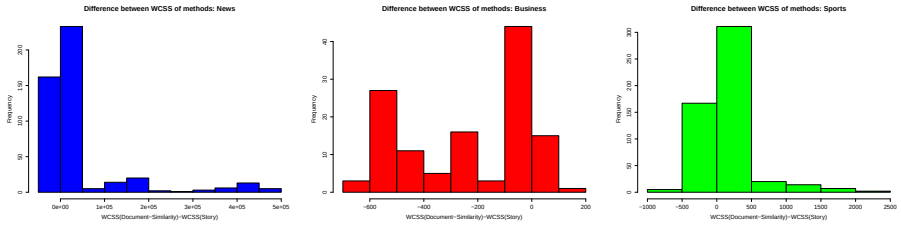

Figure 6: Histograms of the difference in WCSS between document queries using simple document similarity and the storytelling graph. Positive values indicate the storytelling graph produced a smaller WCSS and more coherent neighbors (good results).

$$WCSS = \sum_{i \in \mathbf{D_S}} \sum_{j}^{M} \|w_{i,j} - \mu_j\|^2 \tag{12}$$

The formula above shows the calculation for WCSS where μ is the mean term vector computed by taking the average word count across each document. Though what defines a good or bad value for WCSS is subjective, a bad result from this test would be neighboring documents returned by the storytelling graph that are wildly divergent from each other in comparison to document similarity, a relatively naive method. Highly divergent document peers occur when low document-topic thresholds are used. For thresholds of .6 and .15 for topic similarity and document-topic weighting, the WCSS value is frequently lower depending on the dataset. It follows from the topic modeling that the storytelling graph has the potential to perform better because neighbors are chosen from a learned similar group, but this is not guaranteed.

5.4 Graph Metrics

The final item presented for evaluation are the metrics on a storytelling graph. These metrics provide a quantitive way to compare subgraphs for different topics. The lifetime and mass of a story are somewhat trivial, so they are evaluated indirectly through their use in momentum. However, it should be noted that using the estimated Dirichlet priors to represent topic mass can be advantageous over the thresholded document count as the document count can be zero when other, more prominent topics squeeze the topic weight in question below the threshold. The issue arises due to the restriction that probability distributions must sum to 1.

Velocity of the topic is intentionally separated from its mass to distinguish between how often a topic arrises from how prominent a topic is. The calculation of momentum assumes a time window and for this project a window of 7 days is used. The table below shows example topics for a single day sorted by momentum. The metrics correctly increase the importance temporarily of a story

Example querying the storytelling graph by document starting from: "U.S. ambassador killed in Libya"
09-12
How the Benghazi attack unfolded Envoy to Libya dies in rocket blast U.S. vows to hunt down ambassador's killers U.S. ambassador killed in Libya
09-13
Inside the attack in Benghazi How the deadly attack in Benghazi unfolded U.S. warships move toward Libya Benghazi casualties
09-14
Opinion: Some 'don't get dissent' President Obama, Secretary Clinton Receive Bodies Of Americans Killed In Libya 4 held in Libya attack

Table 4: Example of similar documents returned using storytelling graph starting from the document "U.S. ambassador killed in Libya". The documents are clearly related.

with low velocity but a high mass such as the topics related to a kidnapping and a party bus full of teens overturning.

Lastly, three prominent stories found during the middle of the corpus (2012-09-01) are queried with the resulting subgraphs' documents plotted by day. These stories happen to be related to President Obama's re-election campaign, turmoil in Gaza, and Hurricane Irene. The Obama and Gaza topics are fairly enduring in this dataset so they remain steady. The hurricane, however, was a more isolated event which began August 21, 2012 and ended September 1, 2012. In all three cases the momentum metric of the storytelling graph accurately captures lifetime.

6 Conclusion and Future Work

This project implements a web application to explore RSS feeds and display information in such a way that emphasizes time and the connectedness of topics. To accomplish this goal, a storytelling graph was introduced created from LDA topic modeling at daily intervals. The web applications allows exploring a data stream both from the perspective of an abstract topic and and a concrete document. Both modes of operation follow directly from the queries possible on the story graph. Metrics on the storytelling graph are introduced to further prioritize topics and supply quantitive methods to evaluate the prominence of a story. The resulting implementation is a fun and easy to use way to follow

Most prominent topics by Momentum on 2012-09-01				
Mass	Mass_α	Velocity	Momentum	Topic
244	0.634	9.0	2196.0	romney obama president republican mitt campaign
128	0.163	8.857	1133.714	gaza benghazi hamas dogs missiles obama dome walk
93	0.170	2.286	212.571429	isaac louisiana hurricane storm orleans home residents
9	0.029	0.571	5.143	state penn football coach campus sandusky paterno
11	0.200	0.286	3.143	syrian government syria damascus rebels violence
6	0.015	0.429	2.571	police officers car henderson kidnapped arrested duran
8	0.020	0.286	2.286	bus party fernandez 16 head hospital teens overpass

Table 5: View of prominent stories by momentum for a given day (2012-09-01). The momentum weighting provide an balanced view of import topics at a given point in time.

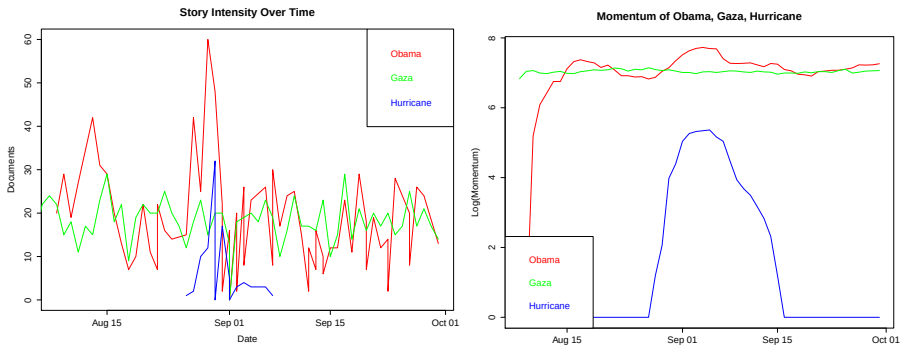

Figure 7: Document count by story over time (left) and Momentum by story over time (right). The log(momentum) view offers a smoother curve that gives a more visually intuitive understanding of the change in a story's prominence over time.

stories over time.

Future improvements might include enhancements to article extraction and data cleaning, integration with natural language processing (NLP) techniques to identify actors and agents present in topics, and further display improvements to the front end. Proper article extraction proved paramount, and yet artifact and ambiguous topics remain that need be suppressed. It is likely that further data cleaning methods can still improve upon this issue. NLP named entity extraction was considered, but time did not permit a full integration. Identifying entities in documents would further enhance the presentation of an evolving story. Lastly, further visualization efforts the storytelling graph and the metrics can improve the user experience.

References

[1] Blei, D. M., Ng, A. Y., Jordan, M. I. (2003). Latent Dirichlet Allocation. Journal of Machine Learning Research, 3, 993-1022.

[2] Griffiths, T. L., Steyvers, M. (2004). Finding scientific topics. Proceedings of the National Academy of Science, 101, 5228-5235.

[3] Khurdiya, A., Dey, L., Raj, N. and Haque, M. Multi-perspective linking of news articles within a repository. International Joint Conference on Artificial Intelligence, 22.

[4] McCallum, Andrew Kachites. "MALLET: A Machine Learning for Language Toolkit." http://mallet.cs.umass.edu. 2002.

[5] Wallach, H. Structured Topic Models for Language. B.A., University of Cambridge (2001); M.Sc., University of Edinburgh (2002)

[6] Kohlschtter, C. Boilerplate Detection using Shallow Text Features. WSDM 2010 – The Third ACM International Conference on Web Search and Data Mining New York City, NY USA.