Information Cartography: Creating Zoomable, Large-Scale Maps of Information

Dafna Shahaf, Jaewon Yang, Caroline Suen, Jeff Jacobs, Heidi Wang, Jure Leskovec
Stanford University
{dshahaf, crucis, cysuen, jjacobs3, hjw, jure}@cs.stanford.edu

ABSTRACT

In an era of information overload, many people struggle to make sense of complex stories, such as presidential elections or economic reforms. We propose a methodology for creating structured summaries of information, which we call *zoomable metro maps*. Just as cartographic maps have been relied upon for centuries to help us understand our surroundings, metro maps can help us understand the information landscape.

Given large collection of news documents our proposed algorithm generates a map of connections that explicitly captures story development. As different users might be interested in different levels of granularity, the maps are zoomable, with each level of zoom showing finer details and interactions. In this paper, we formalize characteristics of good zoomable maps and formulate their construction as an optimization problem. We provide efficient, scalable methods with theoretical guarantees for generating maps. Pilot user studies over real-world datasets demonstrate that our method helps users comprehend complex stories better than prior work.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; H.5 [**Information Interfaces and Presentation**]

Keywords

Zoomable Metro maps, Information, Summarization

1. INTRODUCTION

Faced with today's ever-increasing amounts of data, it is very easy to get lost in details and lose sight of the big picture. However, understanding the big picture is vitally important in order to make key decisions for our own lives, from financial decisions to political ones. We need intelligent and easily accessible tools to help make sense of all this information – and need it possibly more than ever.

Search engines have been traditionally relied upon for finding information in large corpora. Search engines are indeed effective in retrieving nuggets of knowledge, but their output – a list of search results – exhibits practically no structure. Thus, the task of fitting small nuggets of information into a single coherent picture remains difficult.

A possible solution to the above problems is to build *interactive* tools that *explicitly* show the relations among retrieved pieces of information. Multiple tools for summarizing and visualizing news stories already exist. However, we believe that the output of these systems is often not suitable for a news reader:

- Some systems' level of *resolution* is too coarse or too fine to be useful: Capturing relations between individual named entities [8] may be too fine grained, while relations between large complex and dispersed topics [5] may be too vague.
- In order to uncover the structure of a story, numerous tools have gone beyond search engines' list-output. Many of these approaches [19, 20, 4] boil down to *timeline* generation. However, this style of summarization only works for simple stories, which are linear by nature. In contrast, complex stories display a very non-linear structure: stories split into branches, side stories, dead ends, and intertwining narratives.
- Another popular representational choice is a graph [13, 6]. While a graph is surely more expressive than a timeline, these methods offer no notion of *path coherence*: the edges in the graph are selected because they pass some threshold, or belong to a spanning tree. We believe that the notion of coherent paths facilitates the process of knowledge acquisition for the users.

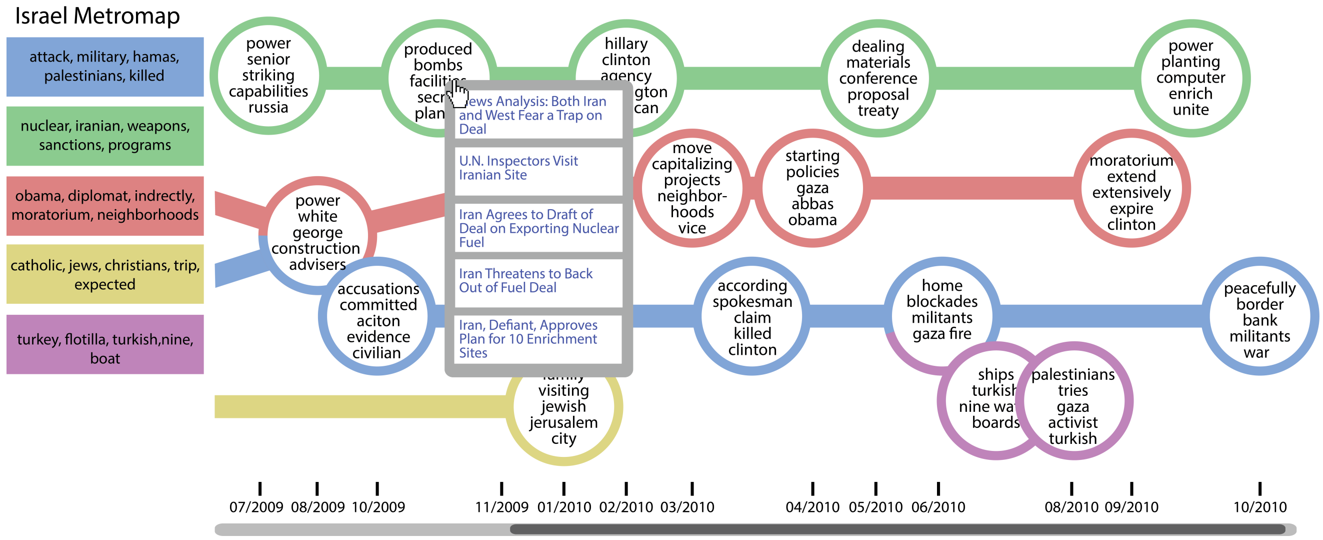
Recently, we took a step in that direction and proposed the notion of *metro maps of information* [18]. Metro maps are concise structured sets of documents maximizing coverage of salient pieces of information; in addition, the maps make explicit the various ways each piece relates to the others.

However, the current approach to metro maps has several shortcomings. For example, picking the right resolution of a metro map can be difficult. Some users are interested in a high-level overview, while others would like to go deeper into details. Current metro maps support only a single level of resolution, where each metro stop is a single news article. In addition, current metro maps are unable to extract sufficient structure and relationships between storylines. Finally, current maps are computationally inefficient, and can only be computed for a few hundreds of documents.

In this paper we extend and scale up *metro maps*. Let us start by revisiting the metro map metaphor: A metro map consists of a set of *metro lines* . Each metro line is a sequence of metro stops; each line follows a coherent narrative thread, and different lines focus on different aspects of the story. The metro map structure easily lends itself to many visualization techniques, allowing users to easily digest information at a holistic level.

Example 1. *Figure 1 shows an example output map of our system for the query "Israel". The map contains five metro lines. The leg-*

KDD'13, August 11–14, 2013, Chicago, Illinois, USA.
Copyright 2013 ACM 978-1-4503-2174-7/13/08 ...$15.00.


Israel Metromap
attack, military, hamas, palestinians, killed
nuclear, iranian, weapons, sanctions, programs
obama, diplomat, indrectly, moratorium, neighborhoods
catholic, jews, christians, trip, expected
turkey, flotilla, turkish,nine, boat
power senior striking capabilities russia
produced bombs faciliti secr plan
hillary clinton agency gton can
dealing materials conference proposal treaty
power planting computer enrich unite
power white george construction advisers
move capitalizing projects neighbor-hoods vice
starting policies gaza abbas obama
moratorium extend extensively expire clinton
accusations committed aciton evidence civilian
according spokesman claim killed clinton
home blockades militants gaza fire
peacefully border bank militants war
visiting jewish jerusalem city
ships turkish nine wa boards
palestinians tries gaza activist turkish
07/2009
08/2009
10/2009
11/2009
01/2010
02/2010
03/2010
04/2010
05/2010
06/2010
08/2010
09/2010
10/2010

ews Analysis: Both Iran and West Fear a Trap on Deal

U.N. Inspectors Visit Iranian Site

Iran Agrees to Draft of Deal on Exporting Nuclear Fuel

Iran Threatens to Back Out of Fuel Deal

Iran, Defiant, Approves Plan for 10 Enrichment Sites

Figure 1: An example metro map for the query "Israel" generated by our system. Each color represents a storyline (going from left to right). Both storylines and metro stops are labeled, and the user can hover over a stop to read associated articles.

end on the left shows the important words for each line. The lines revolve around Gaza, Iran, US diplomacy efforts, religion, and the flotilla incident. Each line consists of nodes (metro stops), also labeled with important words. Hovering over a metro stop brings up a list of articles associated with that stop. One can observe that the Gaza and US diplomacy lines intersected during the visit of the US envoy, and how the flotilla line splits from the blockade node.

Overview of Contributions. In this work we advance the idea of metro maps in four major directions:

- **Metro stops:** In our system metro stops are represented by *sets of words* instead of single documents. This allows for greater flexibility and diversity in the structure and size of a metro stop. Clusters can be thought of as temporal topics, grouping together words that co-occur extensively during a specific time window.
- **Zooming:** Regarding metro stops as sets of words naturally enables zooming, as stops can vary in scope, from focused to broad. This allows users to zoom in on a topic they are interested in, or zoom out to get a broader overview.

This fundamental change requires us to completely re-think the mathematical formulation of metro maps. The most important change is the development of an innovative algorithm for the analysis of the underlying structure of a story:

- **Map structure:** Different stories have different structure: some stories are almost linear, while others are a lot more complex. The maps of [18] had the number of lines and their length given as *input*, and thus could not distinguish between linear and complex stories. We develop a new notion of map structure that alleviates this issue.
- **Scalability**: Our algorithm scales far beyond what was previously possible and allows for generating maps from large corpora of documents. As the algorithm depends on the size of the document set only linearly, we can summarize hundreds of thousands of documents.

From the technical point of view, we develop novel formalizations of metro map quality scores. In addition, we devise a new algorithm for uncovering the underlying structure of news stories. We also devise a novel scalable bipartite graph clustering method that identifies densely connected overlapping clusters, which we use to identify potentially good metro lines.

Our Approach. Our system operates as follows. We start with a large corpora of documents and a user-defined query (e.g., "Middle East"). We extract all relevant documents and apply our graph-based word clustering method to find clusters to be used as building blocks for our maps. We then identify the underlying structure of the topic. We optimize an objective function that prefers longer coherent storylines whenever possible; this way, linear stories become linear maps, while complex stories maintain their interweaving threads. The objective, while hard to optimize exactly, is submodular, and can be efficiently approximated within guarantees.

A topic may be very complex, but the user's attention span is still limited. To keep things manageable, our final step is to restrict the size of a map. We select lines to satisfy two competing goals: ensuring that the map covers aspects which are important to the user, while also encouraging diversity. We rely on submodular optimization again to optimize the map within a map-size budget.

We evaluate our zoomable metro maps, comparing them to a number of competing approaches: web search, topic detection and tracking, and our previous metro maps. Our user study shows that zoomable metro maps help people understand a topic better: Users preferred us to competitors $58\% - 71.8\%$ of the time with statistical significance. Comparing ourselves to the previous maps allows us to isolate the benefits of our new framework.

2. CONSTRUCTING AN OBJECTIVE

The problem of finding a good metro map is hard, especially because it is difficult to understand what we are looking for. Recognizing whether a map is good or not is easy for humans, but it is a very intuitive property.

We first review the desired properties of a metro map, extending and elaborating on the criteria outlined in [18]. We shall briefly present these criteria, motivate and formalize them. Later, we present a principled approach to constructing maps that optimizes tradeoffs among these criteria.

2.1 General Considerations

Our first concern is **Scalability**. We would like our maps to scale to large corpora, potentially including millions of documents. Our previous approach did not support this order of magnitude, both computationally and visualization-wise. One of the challenges of handling large corpora is picking the right granularity of a map: some users are interested in a high-level overview of a topic, while others would like to go deeper into details.

For this reason, we would like our maps to support **Multi-resolution**: the maps should be zoomable, with each level of zoom showing finer details and interactions. Following our map metaphor, the higher levels of zoom should show the highways (major storylines). As the users zoom in, they discover smaller streets, service roads, and eventually even walking trails.

Let us now consider the building blocks of our maps. In order to support multi-resolution, our metro stops can no longer be single articles (as in [18]). Similar to topic representation in topic modeling, we choose to focus on *clusters of words* as our metro stops. Clusters should group together words from a specific time step.

Focusing on clusters naturally raises the question of **Cluster Quality**. Most importantly, the clusters should be *cohesive*: the words belonging to the same cluster should be closely related during the corresponding time period.

For example, consider the story of LeBron James. LeBron James is a professional basketball player. He played for the Cleveland Cavaliers between 2003 and 2010. In 2010, he left the Cavaliers for Miami Heat in a highly publicized (and highly criticized) free agency period.

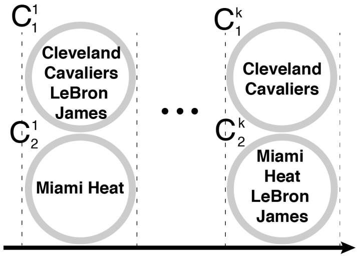


Figure 2: Simplified word clusters for the story of LeBron James. Clusters C_j^i belong to time step i.

Figure 2 shows (simplified) clusters for the LeBron James story. Throughout 2003-2009, the words "LeBron James" and "Cleveland" are clustered together, while "Miami" is a different cluster. In 2011, the word "LeBron" is more often mentioned with "Miami", so it has moved to the Miami cluster.

Now that we know what our metro stops are, we focus on the metro lines. A key requirement is that each line is **Coherent**: following the papers along a line should give the user a clear understanding of the evolution of a story. Most importantly, coherence is a *global* property of the line. It is not enough for every pair of consecutive clusters to be related: there should be a global theme running through the line.

A map is more than merely a set of lines. There is a lot of information in its **Structure** as well. Storylines can intersect, split and merge. Intuitively, different stories have different structure: some stories are almost linear, while others are a lot more complex.

Furthermore, we believe that some stories have a natural structure. For example, when asked to draw a map of the LeBron James story, people's maps often resemble Figure 3: The blue storyline follows Cleveland Cavaliers and the red follows Miami Heat. The third line, about LeBron, splits from Cleveland and joins Miami. Our previous maps have not been able to uncover such natural structures; in this work, we will explicitly search for it.

Our last consideration is **Budget**. A topic may be very complex, but the user's attention span is still limited. To keep things manageable, we restrict the size of a map. These restrictions raise a question of selection: which lines should be displayed to the user?

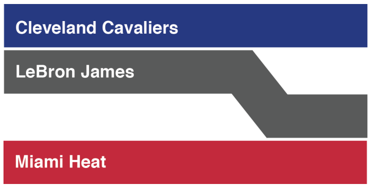


Figure 3: A hypothetical map that provides a high-level overview of LeBron James' move from Cleveland to Miami

Our goal in selecting the lines is twofold: we want to ensure that the map covers aspects which are important to the user, while also encouraging diversity. In other words, the lines should not be redundant. We call this property *coverage* under budget.

2.2 Formalizing the Objective

After discussing the general properties an objective function should satisfy, we now go ahead and formalize them. Before we begin, let us formally define our input and output.

We assume that we are given a set of documents $\mathcal{D}$, each with a corresponding time stamp. $\mathcal{D}$ are the documents we want to summarize and visualize; one can think of $\mathcal{D}$ as the result of a query. Our output is in the form of a metro map:

Definition 2.1 (Metro Map). A *metro map* $\mathcal{M}$ is a pair (G, Π), where $G = (\mathcal{C}, E)$ is a directed graph and Π is a set of paths in G. We refer to paths as metro lines. Each $e \in E$ must belong to at least one metro line.

We note our vocabulary by W. Each document is a multiset of W. Vertices $\mathcal{C}$ correspond to word clusters (subsets of W), and are denoted by *stops*$(\mathcal{M})$. The lines of Π correspond to aspects of the story. As an example, the map in Figure 1 includes five metro lines.

In the following sections, we formulate the problem of finding a good metro map given $\mathcal{D}$. We need to consider tradeoffs among the properties of Section 2.1: **Cluster quality**, line **coherence**, map **structure**, and coverage under **budget**. For example, maximizing coherence often results in repetitive, narrow-scope chains. Maximizing coverage leads to a disconnected map, since there is no reason to re-use a cluster for more than one line.

As discussed in [18], it is better to treat coherence as a constraint: a chain is either coherent enough to be included in the map, or it is not. Cluster quality and structure costs, on the other hand, should both be optimized. Combining this, we get:

Problem 2.2 (Metro Maps: Informal). $\mathcal{M}$ should satisfy:

- High cluster quality *cQual*$(\mathcal{C})$
- High structure quality *sQual*$(\mathcal{M})$
- Minimal line coherence, *Coherence*$(\mathcal{M}) \geq \tau$
- High coverage *Cover*$(\mathcal{M})$
- Maximal map size, $|\mathcal{M}| \leq K$

In Sections 2.2.1-2.2.4 we formalize all of the above notions. In Section 2.2.5 we formalize Problem 2.2 above.

2.2.1 Cluster Quality

We start by formalizing cluster quality *cQual*$(\mathcal{C})$. As mentioned in Section 2.1, our primary objective for clusters is cohesion: words in the same cluster during a period of time should be closely related to each other. In this section we formalize this notion.

In order to find cohesive clusters, we divide the query set $\mathcal{D}$ into time steps, and construct a word co-occurrence graph for each time step. See Section 3.1 for complete details. Figure 4 gives the intuition behind the process: On the left we see a co-occurrence graph. Each node in the graph is a word, and the edge strength represents

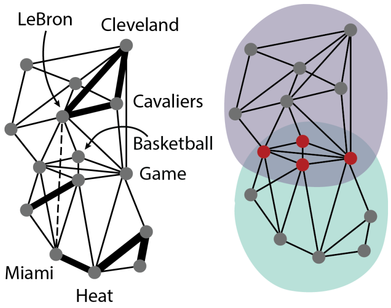


Figure 4: Left: word co-occurrence graph. Right: detecting communities.

how often the words co-occur with each other during that time step. For example, the word “LeBron” co-occurs often with “Cavaliers”.

Next, we prune the co-occurrence graph so that only the strongest edges remain. For example, perhaps LeBron was mentioned with Miami, but not often. The dashed edge in Figure 4(left) will subsequently be pruned.

Having built the word co-occurrence graph, we aim to find clusters of words that tend to co-occur (Figure 4 right). Formally, given a word co-occurrence graph $G(W, E)$ where W is a set of words, we want to detect densely connected clusters C_i of words.

Moreover, we expect that the clusters may overlap. For example, the words “basketball” and “game” are likely to appear in both clusters. The task of finding overlapping clusters, called “overlapping community detection”, has been widely studied [10, 15] and many methods are available. However, the current overlapping community detection methods assume *sparse* community overlaps: the more communities the nodes share, the less likely the nodes would be connected. However, in our task, communities are expected to produce *dense* overlaps: we observe that words belonging in multiple clusters together are often more likely to co-occur.

To come up with a notion of $cQual(\mathcal{C})$ that encourages densely overlapping clusters, we apply BigClam [21]. In the following we explain how to derive $cQual(\mathcal{C})$ from BigClam.

BigClam builds on a generative model for a network G which naturally captures the formation of dense community overlaps. In BigClam, each node w has a latent nonnegative membership weight F_{wc} to each cluster c. Given F_{wc}, F_{uc}, an edge between w, u is created with probability:

$$P(w, u) = 1 - \exp(-\sum_c F_{wc} F_{uc})$$

Note that BigClam assumes dense overlaps because nodes w, c would have higher edge probabilities if they belong to multiple communities c together ($F_{wc}, F_{uc} > 0$ for multiple c).

BigClam detects community memberships F_{wc} using maximum likelihood estimation. It optimizes the likelihood $l(F) = P(G|F)$ of a community membership matrix F:

$$\underset{F \geq 0}{\operatorname{argmax}}\, l(F) \sum_{(w,u)\in E} \log(1 - p(w, u)) - \sum_{(w,u)\notin E} p(w, u)$$

After learning F_{wc}, we regard w as a member of a cluster c if $F_{wc} > \delta$ for a given threshold. Most importantly, the likelihood of the matrix serves as a natural cluster quality for our purpose:

$$cQual(\mathcal{C}) = l(F)$$

2.2.2 *Coherence*

After formalizing cluster properties, we turn our attention to lines. The first properties of lines is *coherence*. We rely on the notion of coherence developed in Connect-the-Dots (CTD) [17]. In the following, we briefly review this approach.

In order to define coherence, a natural first step is to measure similarity between each pair of consecutive clusters along the line. The main insight of [17], however, was that coherence is a *global* property of the line, and cannot be deduced from local interactions.

Figure 5 demonstrates this idea: Suppose a line consists of three clusters, each containing four words. The first and the second clusters are similar, and so are the first and the third. However, the first and the third do not have any word in common. Local connections may give rise to associative, incoherent lines. On the other hand, coherent chains can often be characterized by a *small* set of words, which are important throughout many of the transitions.

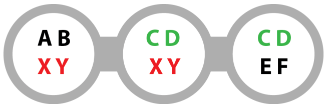


Figure 5: A line of length 3. The line is not coherent, despite the similarity of each pair of consecutive clusters.

Therefore, the problem can be transformed into an optimization problem, where the goal is to choose a small set of words (called ‘active’ words), and score the chain based only on these words. In order to ensure that each transition is strong, the score of a chain (given a set of active words) is the score of the weakest link.

$$Coherence(C_1, ..., C_n) = \max_{W \in activations} Coherence(C_1, .., C_n|W) \quad (2.1)$$

$$Coherence(C_1, .., C_n|W) = \min_{i=1...n-1} score(C_i \to C_{i+1}|W) \quad (2.2)$$

Constraints on possible activations encourage a small number of words and smooth transitions, imitating the behavior observed for coherent chains in [17]. Finally, the coherence of a map is defined as the minimal coherence across its lines Π.

2.2.3 *Structure*

Even restricting ourselves to coherent lines, the number of lines remains formidable. Our next goal is to find a set of coherent lines that captures the true structure of a story. “Story structure” is a rather intuitive property: For example, natural disasters stories usually tend to be relatively linear: a disaster strikes, relief efforts arrive, recovery begins. On the other hand, stories about financial crisis tend to involve multiple interacting actors and span many geographic locations.

In order to recover the story structure, we aim to minimize the number of coherent lines, such that all clusters belong to some line. Since some clusters cannot join any other cluster to form a coherent line, we treat them as singleton lines:

$$sCost(\Pi \mid \mathcal{C}) = |\Pi| + |\{c \notin \Pi\}| \quad (2.3)$$

In other words, the cost of a structure Π is the number of lines, plus the number of clusters not covered by Π. For example, in Figure 6 two storylines (Miami and Cleveland) cover all of the clusters. The cost of these lines is therefore 2. The quality of a structure $sQual(\Pi \mid \mathcal{C})$ is defined to be $-sCost(\Pi \mid \mathcal{C})$.

If a story is relatively linear, most of it can be captured by one (long) coherent line. Therefore, a map optimizing this objective would be mostly linear. Maps of complex topic, on the other hand, would still be forced include multiple shorter lines.

Minimizing Equation 2.3 produces longer storylines whenever possible. However, this objective suffers from one major drawback. Consider again the story of LeBron James in Figure 6. The two storylines (Miami and Cleveland) indeed cover all of the clusters,

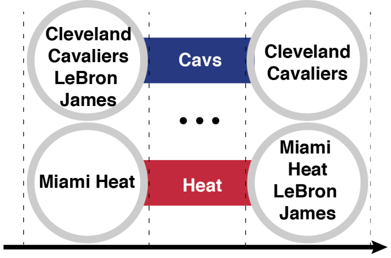


Figure 6: Minimizing number of story lines (Equation 2.3): Note that information about LeBron James' switch is completely lost.

but this structure is missing an important storyline: the one about LeBron himself.

The problem seems to stem from the idea that a line going through a cluster c covers the entire content of c. More realistically, the line may only cover c *partially*. Therefore, we modify the objective.

First, we associate each line with a set of words that it covers, $\mathcal{W}(\pi)$. In our framework, the words associated with a line are the words that make it coherent (see Section 2.2.2). For example, the blue line is associated with "Cleveland" and "Cavaliers".

When a line goes through a cluster, $\mathcal{W}(\pi)$ are the only parts of the cluster that the π covers. Refer to Figure 7 for an example: since the blue line covered only the Cleveland part of the top-left cluster, another line is still needed to cover "LeBron".

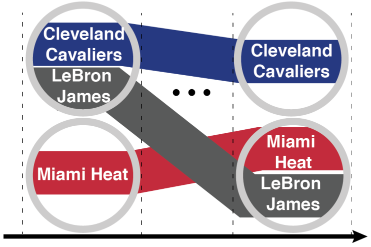


Figure 7: Optimizing Equation 2.4: Although there are more story lines, the story of LeBron James' move is captured.

Formally, the cost of a set of lines Π is:

$$\textit{sCost}(\Pi \mid \mathcal{C}) = \sum_{c \in \mathcal{C}} |c \setminus \bigcup_{\pi . c \in \pi} \mathcal{W}(\pi)| + \sum_{\pi \in \Pi} |\mathcal{W}(\pi)| \qquad (2.4)$$

The cost is 1 for each cluster-word that is not covered by Π. This way, if a line covers a word in multiple clusters, we only pay for this word once: when we pick that line.

For example, in Figure 6, the cost of having no lines at all would be the number of uncovered words: $4 + 4 + 2 + 2 = 12$. The two lines in Figure 6 cost 2 each, and leave four words uncovered ("LeBron", "James" in two clusters), so their total cost is $2 \cdot 2 + 2 + 2 = 8$. The cost of each of the three lines of Figure 6 is still 2, but they cover all of the words. Therefore, the total cost is $3 \cdot 2 = 6$, and this solution will be preferred.

2.2.4 Budget: Coverage

We now formalize a notion of coverage. The goal of the coverage measure is twofold: we would like high coverage maps to revolve around **important** topics, while also being **diverse**. To achieve this, we use the coverage notion of [7]. In the following, we briefly review this notion.

In [7], we are given a function $\textit{cover}_c(w)$, measuring how well a cluster c covers a given word w. We then extend $\textit{cover}_c(w)$ to functions measuring the coverage of sets of clusters, $\textit{cover}_C(w)$. In order to encourage diversity, this function should be submodular. In [7], we chose

$$\textit{cover}_C(w) = 1 - \prod_{c \in C} (1 - \textit{cover}_c(w)) \qquad (2.5)$$

Thus, if the map already includes documents which cover w well, $\textit{cover}_{\mathcal{M}}(w) = \textit{cover}_{\textit{stops}(\mathcal{M})}(w)$ is close to 1, and adding another cluster which covers w well provides very little extra coverage of w. This encourages us to pick documents that cover new topics, promoting **diversity**.

Now that we have defined coverage in terms of individual words, we can focus on coverage for an entire corpus. To ensure that we bias towards **important** words, we compute an importance score λ_w for each word w. Finally, we define corpus coverage as:

$$\textit{cover}_{\mathcal{M}}(\mathcal{C}) = \sum_{w} \lambda_w \textit{cover}_{\mathcal{M}}(w) \qquad (2.6)$$

The weights cause *Cover* to prefer maps that cover important documents. They also offer a natural mechanism for personalization: With no prior knowledge about the user's preferences, we set all of the weights to 1. This is equivalent to finding a map that covers as much of the corpus as possible. In [18] we discuss learning weights from user feedback, resulting in a personalized notion of coverage.

2.2.5 Tying it all Together

We have now defined all the pieces of Problem 2.2, and can now define the objective formally. First, we formulate the problem of finding a map without budget constraints.

In Problem 2.2, we stated that both cluster quality and structure quality should be optimized. The structure of a map depends immensely on the clusters: given bad clusters, no structure can save the map. Therefore, cluster quality is our primary objective. However, during our experimentation we learned that top-quality clusters are sometimes missing words (mostly due to noisy data). Furthermore, these missing words often hinder lines from reaching the coherence threshold. In order to mitigate this problem, we introduce slack ϵ into our objective:

Problem 2.3. Given a set of candidate documents $\mathcal{D}$, find a map $\mathcal{M}^* = (G^*, \Pi^*)$ over $\mathcal{D}$ which minimizes $\textit{sCost}(\Pi^* \mid \mathcal{C}^*)$ s.t. $\textit{Coherence}(\mathcal{M}) \geq \tau, \textit{cQual}(\mathcal{C}^*) \geq (1-\epsilon)\textit{optQual}$. *optQual* is the maximal $\textit{cQual}(\cdot)$ achievable over $\mathcal{D}$; τ and ϵ are given parameters.

$\mathcal{M}^*$ is an optimal map with no budget constraints. Next, we find a subset of $\mathcal{M}^*$ subject to our budget:

Problem 2.4. Given a set of candidate documents $\mathcal{D}$, find a map $\mathcal{M}^*_K = (G, \Pi)$ over $\mathcal{D}$ which maximizes $\textit{Cover}(\mathcal{M}^*_K)$ s.t. $\mathcal{M}^*_K \subseteq \mathcal{M}^*, |\mathcal{M}^*_K| \leq K$.

There are multiple ways to define the size of the map. We will show how to restrict the number of lines and the number of clusters in Section 3.4.

3. ALGORITHM

In this section, we devise an algorithm for solving Problem 2.4. The approach is outlined below:

- Find optimal structure, $\mathcal{M}^*$:
  - Find an initial set of clusters $\mathcal{C}$ optimizing *cQual*$(\mathcal{C})$ (Section 3.1)
  - Find an initial set of lines Π optimizing *sQual*$(\Pi \mid \mathcal{C})$ (Section 3.2)
  - Perform local search to improve lines without sacrificing cluster quality (Section 3.3)
- Find a subset map $|\mathcal{M}^*_K| \leq K$ maximizing coverage (Section 3.4)

The approach follows directly from our problem definition. In order to solve Problem 2.4, we first need to find the optimal structure, $\mathcal{M}^*$ (Problem 2.3). The optimal structure consists of a set of clusters of quality close to optimum, and a (low-cost) set of lines going through them.

In Section 3.1, we find an initial set of good clusters. In Section 3.2, we find a low-cost set of lines going through these clusters. We then run local search in order to improve the map (Section 3.3). Finally, in Section 3.4 we find a high-coverage subset of $\mathcal{M}^*$ satisfying budget constraints.

3.1 Finding Good Clusters

We start by finding an initial set of good clusters for each time step. As mentioned in Section 2.2.1, we create a co-occurrence graph for each time step; we then use BigClam, a fast overlapping community detection method for undirected networks [21].

We consider two strategies when dividing $\mathcal{D}$ into time steps: steps encompass either a fixed amount of time or a fixed number of documents. Oftentimes, the popularity of a query over time varies; there are bursts of intense popularity, followed by long stretches of relative unpopularity. As we wish for more detail during popular times, we use the latter.

To find closely related words, we construct a word co-occurrence graph for each time step. Nodes of the graph are individual words, and the edge weight indicates the strength of the connection between two words. We experimented with several weighting strategies; the strategy that performed the best listed the top 50 tf-idf words for each document, and set to weight of edge (u, v) to be the number of documents that had both u and v in that list. This weighting mechanism disregards both rare and overly popular (and therefore uninformative) words using tf-idf, while still favoring popular words that co-occur together frequently.

To ensure strong co-occurrence between words in resulting clusters, we next prune the co-occurrence graph, removing edges whose weights are less than 5% of the maximum achievable edge weight. Note that this weight depends on the number of documents in each time window.

In order to find communities in each co-occurrence graph, we use BigClam (see Section 2.2.1). BigClam uses a block coordinate ascent approach for maximum likelihood estimation, which is very scalable as each iteration takes a near constant time [21].

3.2 Finding Good Lines

Now that we have a good set of clusters $\mathcal{C}$, we need to find a set of coherent lines going through them that minimizes the cost *sCost*$(\Pi \mid \mathcal{C})$ (see Equation 2.4). As the number of lines is exponentially large, we start by heuristically generating a pool of candidate lines; in Section 3.2.2 we find a subset of these lines that minimizes structure cost.

3.2.1 Generating Candidate Lines

After finding word clusters $\mathcal{C}$ for each time step, we aim to group the clusters that may form a coherent line together. Note that computing similarity between each pair of clusters is not enough. In Section 2.2.2, we noted that coherence is a global property of the line; in other words, there should be a small set of words that captures the entire line.

We want to simultaneously find groups of words that belong to the same clusters, and clusters that all use similar words. *i.e.*, we co-cluster the clusters and the words. To achieve this, we extend [21] (see Section 2.2.1) and develop a group detection method for a word-cluster bipartite graph.

Formally, we are given a bipartite network $B(C, W, E)$ where C is one set of nodes (the word clusters), W is the other set of nodes (the words), and E denotes the set of the edges between them. We assume that the nodes have latent group membership factors, and the edges in the bipartite network B are generated from these factors.

The algorithm is an extension of the algorithm described in Section 2.2.1. Word $w \in W$ has membership weight F_{wg} for group g, and cluster $c \in C$ has membership weight H_{cg}.

The edge probability between w and c is:

$$p(w, c) = 1 - \exp(-F_w H_c^T), \qquad (3.1)$$

where F_w, H_c are weight vectors for node w and c respectively ($F_w = F_{w\cdot}$, $H_c = H_{c\cdot}$). As before, this function has a nice probabilistic interpretation, and allows us to compute the gradient for the likelihood of a single node in near constant time [21].

In order to detect the groups, we fit the model, finding the most likely affiliation factors F, H to the input bipartite network B by maximizing the likelihood $l(F, H) = \log P(B|F, H)$ of B:

$$l(F, H) = \sum_{(w,c)\in E} \log(1 - \exp(-F_w H_c^T)) - \sum_{(w,c)\notin E} F_w H_c^T.$$

To maximize $l(F, H)$, we employ a block coordinate gradient ascent approach [12], where we update F_w for each word w with other variables fixed, and then update H_c for each word cluster c with other variables fixed. Finally, we add words and clusters to group g if their membership weight is above some threshold. Please refer to Section 4 for a sample output group.

3.2.2 Finding a Good Structure

Even after we found a pool of candidate lines, minimizing *sCost*$(\Pi \mid \mathcal{C})$ is a hard problem; luckily, *sQual*$(\Pi \mid \mathcal{C}) = -$*sCost*$(\Pi \mid \mathcal{C})$ is a submodular function:

Definition 3.1 (Submodularity). Function f is submodular if for all $A, B \subset V$ and $v \in V$ we have $f(A \cup \{v\}) - f(A) \geq f(B \cup \{v\}) - f(B)$ whenever $A \subseteq B$.

In other words, f is submodular if it exhibits the property of *diminishing returns*. Intuitively, *sQual*$(\cdot)$ is submodular since adding a line π to a set of lines always has the same cost, $|\mathcal{W}(\pi)|$, but the number of words that π covers for the first time can only be smaller when added to B s.t. $A \subseteq B$.

Although maximizing submodular functions is still NP-hard, we can exploit some recent results. However, most results apply to non-negative, monotone submodular functions, and our function is neither: in fact, it monotonically increases to some point and then starts decreasing. First, we come up with an equivalent objective that is non-negative. Let w_c be the total number of words in clusters, $\sum_{c\in\mathcal{C}} |c|$. Our goal is to maximize

$$2 \cdot w_c - \sum_{c\in\mathcal{C}} |c \setminus \bigcup_{\pi.c\in\pi} \mathcal{W}(\pi)| - \min(\sum_{\pi\in\Pi} |\mathcal{W}(\pi)|, w_c) \qquad (3.2)$$

The objective is still submodular, and has the same argmax_{Π}. It is non-negative, although still non-monotone. We use [9] , that gives a deterministic local-search $\frac{1}{3}$-approximation and a randomized $\frac{2}{5}$-approximation algorithm for this case.

3.3 Improving the Structure

In previous sections we found a good set of clusters and lines going through them. In this section, we apply local search to improve the results. As defined in Problem 2.3, we are willing to sacrifice some cluster quality in order to obtain better structure score.

At each iteration, we consider adding cluster c (which is not completely covered) to line π. We consider the words associated with π, and add them to c by increasing their membership weights F_{wc} (see Section 2.2.1). We then compute the likelihood of this new membership matrix, and the new structure score. We repeat this computation for all clusters and lines. At the end of the iteration, we pick the best move and apply it; the search stops when no improvements can be made.

3.4 Finding a Map Subject to Budget Constraints

We have now solved Problem 2.3 and found $\mathcal{M}^*$. Next, we need to find a subset of it subject to budget constraints (Problem 2.4).

Luckily, the coverage function of Equation 2.6 is submodular as well. This time, the function is both monotone and non-negative, so we can exploit the classic result of [14], which shows that the greedy algorithm achieves a $(1-\frac{1}{e})$ approximation. In other words, if our budget is K lines, we run K iterations of the greedy algorithm. In each iteration, we evaluate the *incremental* coverage of each candidate line π, given the lines which have been chosen in previous iterations:

$$\mathit{IncCover}(\pi|\mathcal{M}) = \mathit{Cover}(\pi \cup \mathcal{M}) - \mathit{Cover}(\mathcal{M})$$

That is, the additional cover gained from π if we already have clusters of $\mathcal{M}$. We pick the best line and add it to $\mathcal{M}$.

If instead of restricting the number of lines we want to restrict the number of clusters, we use a classical "bang-for-the-buck" greedy algorithm instead.

3.5 Zooming

Some users are interested in a quick, high-level overview of a topic, while others wish to delve into the details. For this reason, we want our maps to be *zoomable*. As maps are very expressive, there are multiple ways to interpret zoom interactions. In this section, we explain how to implement three such interpretations.

Time Resolution: Zoom level may affect the resolution of a time window, allowing users to choose between daily, weekly or even yearly advances in the story. This is easy to implement: all we need to do is split $\mathcal{D}$ into a different number of time windows (Section 3.1). Note that the pruning threshold is relative to the number of documents in each time window; therefore, strong co-occurrences will still be captured.

Cluster Resolution: The zoom level could be interpreted as a measure of cluster cohesiveness: when zooming out, related clusters should merge together. When zooming in, clusters should break into their most-dense parts.

Luckily, our map formulation supports zooming naturally: Refer to BigClam again (Section 2.2.1). In some cases, an edge may happen between two nodes who do not share any community affiliations. To account for such case, BigClam assumes a base edge probability ε between *any* pairs of nodes, which a user can specify a priori. By varying the value of ε, the user can tune the edge density of the detected communities, because BigClam would not detect communities whose edge density is lower than ε. (This mechanism is similar to setting clique size k in a clique-percolation algorithm.)

Story Resolution: Finally, the user may choose to zoom into a particular metro line, or perhaps a metro stop. In this case, we construct a subset of $\mathcal{D}$ corresponding to the user interest, and rerun the map algorithm on these articles alone.

Of course, one may wish to combine several zooming methods.

3.6 Complexity and Running Time

We now examine the time complexity of our algorithm. The important thing to note is that we first compile our query set $\mathcal{D}$ to a sequence of co-occurrence graphs. The process of generating these graphs is linear in $\mathcal{D}$, but the size of the graph does not depend on $\mathcal{D}$ at all; rather, it is always limited by the size of our vocabulary W. Thus, our dependency of the size of $\mathcal{D}$ is linear, and our algorithm scales well. Note that unlike many clustering methods, we do not need to compute distances between each pair of documents.

The BigClam algorithm for finding communities takes $O(|E|)$ time for a full iteration [21]. The generation of our word-cluster bipartite graph and the group detection within this graph are both linear processes as well.

Finally, when satisfying the budget constraints (Section 3.4), we use a greedy algorithm, whose main bottleneck is the need to re-evaluate a large number of candidates. However, many of those re-evaluations are unnecessary, since the incremental coverage of a line can only decrease as our map grows larger. Therefore, we use CELF [11], which provides the *same* approximation guarantees, but uses lazy evaluations, often leading to dramatic speedups.

Our system is able to process datasets that [18] could not handle. For example, we generated a map concerning Mitt Romney during the 2012 election season. The query set contained 70390 articles (extracted from a Spinn3r [3] dataset), and the map took about 11 minutes to compute. Considering many of the bottlenecks are easy to parallelize, we believe that we can achieve even more significant speedups with a parallel implementation.

Note that while our system could in principle support even larger query sets, the use case we have in mind for maps rarely necessitates it. In particular, we imagine that very broad queries ("U.S.") would be less common than narrower ones ("health care reform").

4. EVALUATION

In this section we evaluate our results. We start by closely inspecting two example maps. Later, we conduct a user study in order to evaluate the utility of the maps.

4.1 Qualitative Evaluation

Our first qualitative evaluation case is the query "Israel". We now follow the creation process of a map. We start by constructing clusters from the co-occurrence graph.

Clusters. The clusters discovered by our method successfully capture major issues related to Israel in the dataset: **Iran** (words in the cluster: *iran, nuclear, iranian, enrich, fuel, agency, uranium, programs, weapons, energy, atomic...*), **negotiating settlements** (*settlements, palestinians, netanyahu, negotiator, west, east, bank, jerusalem, abbas, freezing, building...*), **assassination in Dubai** (*killed, dubai, passport, mabhouh, suspect, hotel, assassinated...*) and the **flotilla incident** (*blockades, gaza, flotilla, ships, raided, activist, commandos...*).

Lines. Next, our bipartite community detection discovers lines. For example, our system correctly identifies the **Iran** line which contains the words *american, obama, iranian, iran, agency, united, council, efforts, diplomats, weapons, nuclear, negotiator, syria, russia, enrich, uranium, sanctions, atomic, planting, energy, fuel*. The line nicely passes seven clusters, among them (*iran, diplomats, programs, nuclear, syria*) and (*supporters, american, obama, president, iran, security, united, council, weapons, nuclear, sanctions*).

These words and clusters are highly related to the topic of Iran, and allow for the creation of very coherent lines.

The Map. Finally, the resulting map is shown in Figure 1. The map contains five metro lines. The legend on the left shows the important words for each line: The top-coverage lines that were found revolve around Gaza, Iran, US diplomacy efforts, religion, and the flotilla incident.

Reading the map from left to right (obeying the chronological order of events), our system successfully identifies the structure of the story: Israel and Gaza were in conflict in 2009. Americans have been trying to negotiate between the sides, including a visit of the U.S. envoy (intersection of the diplomacy and Gaza lines). After the war, the blockade persisted (blue line); in May 2010, a flotilla intended to break the blockade (purple line splitting from blue). The Israel-Turkey relations reached a low point after the incident (end of the purple line). The map also shows a line about the atomic bomb threat of Iran (green). Following the green line from left to right we can learn about the escalation of the story, including harsher and harsher sanctions on Iran.

Finally, the yellow line is about religion. It originally contained only two clusters. Suppose we wish to further explore it; when zooming into the yellow line, our system identifies nine new clusters, some of which are pictured in Figure 8(a). The new level reveals stories that were invisible before zooming in, such as the Pope's visit to Jerusalem (blue), clashes over national heritage sites (red) and an attack on a Rabbi (green).

The qualitative evaluation demonstrates the potential of our method. Chronological and topical relationships are captured by the rich structure of the map. These relationships are efficiently conveyed to the user by our automatically generated visualization.

O.J. Simpson Map. We now consider another map. Figure 8(b) shows a detail of the map corresponding to the notorious O.J. Simpson trial. The map reveals a rich and intricate structure: The blue line (trial) and the yellow line (evidence) are very closely intertwined. The blue line focuses on the judge, jury and details of the murder itself. The yellow line focuses on physical evidence, blood samples, and the tapes with racial epithets by an Los Angeles Police Department detective. Both lines merge again for the verdict.

The green line focuses on the media's fascination with the trial, and how cable ratings soar. Interestingly, the map even captures the aftermath of the Simpson trial (red line), including the effect on Gen. Colin Powell's campaign and the renewed focus on domestic issues (e.g., the number of black men in their 20's who are imprisoned). All in all, we conclude that our system correctly identified the intricate relationships and individual story lines of the O.J. Simpson trial case.

4.2 Quantitative Evaluation via a User Study

Evaluating metro maps quantitatively is a difficult task. There is no established golden standard for this task, and even ground truth is hard to define. Since the goal of the maps is to help people navigate through information, we decided to conduct a user study. The study aims to test whether the maps that we generate are useful for humans seeking information about a topic.

In previous work [18], we explored the tasks that maps are most suitable for. That work indicates that metro maps are best used as a high-level overview of events, rather than as a tool to answer narrow questions. To this end, we evaluate our maps by comparing them with three established alternatives:

Google News: a news aggregator that endeavors to cluster search results by event, in roughly chronological order [2]. Google News was chosen because of its popularity, as Google is the first choice for many users when seeking information on the web. To avoid bias, we stripped the web page of the logo and typical formatting.

KeyGraph [16]: a graph-based topic detection and tracking (TDT) solution. Topic Detection and Tracking is a well-established area of research dealing with capturing the rich structure of events and their dependencies in a news topic. KeyGraph was selected because of its structured output.

Previous Maps [18]: our previous work on maps. We wish to compare our work to our previous work. However, our previous work would not scale to our current query set size. For the sake of the comparison, we decided to first run our clustering step (Section 3.1), and treat each cluster as a single document for the algorithm of [18]. Note that this comparison favors our previous map design as in its original form it simply would not be able to scale to the dataset size. In addition, as both systems operate over the same set of metro stops, we can attribute differences in performance entirely to our new formalizations of structure and coverage metrics.

Datasets. For the user study, we extracted two sets of articles from the New York Times: one about Obama's presidency in 2009, and another about the O.J. Simpson trial in 1995. These two stories have been selected to gauge the effect to story recency on our results. Our New York Times corpus contained tens of thousands of articles; the Obama query set contained $\approx$ 4000 articles, and the O.J. dataset contained $\approx$ 2000.

Note that Google News maintains its own database of news information, which is a superset of our dataset.

Preprocessing. The dataset we use contains undesired HTML, and other relics of web scraping. During a pre-processing step, we clean the dataset by checking for duplicates, matching URLs against blacklists, and running the data through an HTML parser. We remove stop words and compile word counts based on stemmed representations.

4.2.1 Study Procedure

We conduct the study with 27 in-person users. Users were affiliated with Stanford, and were compensated for their time. In the first phase of the study, each user is randomly assigned two new summarization systems out of the four competitors: one to learn about Obama's actions in 2009, and one to learn about the O.J. Simpson trial. Our goal was to test which system helped the users to better understand the big picture.

We believe that the true test of one's own understanding of a topic is their ability to explain it to others. Therefore, we asked our participants to write a paragraph for each story, summarizing the story to someone not familiar with it.

In the second phase, we used Amazon Mechanical Turk [1] to evaluate the paragraphs. At each round, workers were presented two paragraphs (map user vs. competitor-system user). The workers were asked which paragraph provided a more complete and coherent picture of the story; in addition, they justified their choice in a few words. We collected 1000 online evaluations this way.

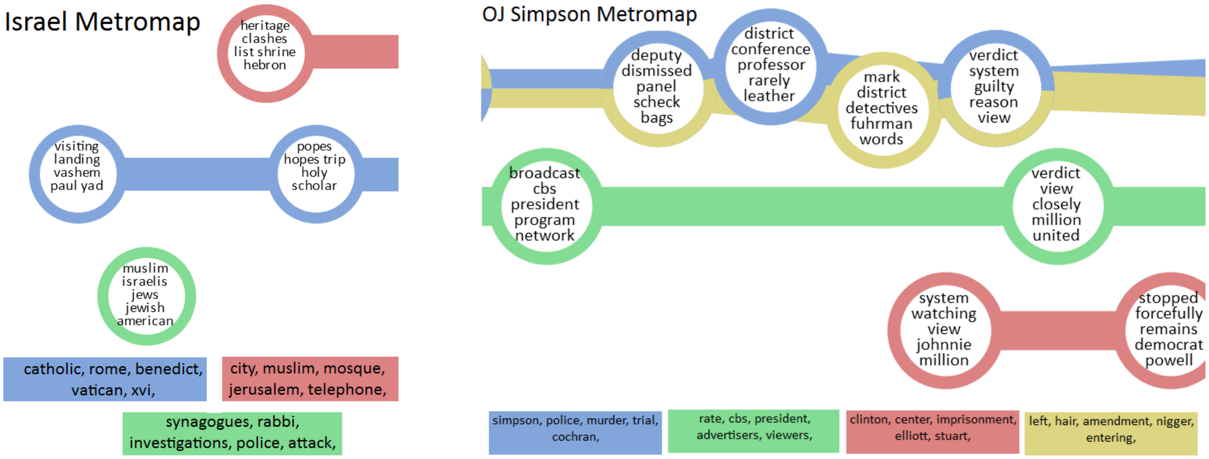
The in-person user-study guarantees high-quality paragraphs, while the Mechanical Turk service allows us to obtain more accurate statistical information with a larger sample size.

4.2.2 Results and Discussion

Baseline	Preferred us	p-value
Google	71.8%	$< 1.0 * 10^{-16}$
TDT	63.1%	$2.0 * 10^{-7}$
Previous Maps	58%	.0021

The table above displays the study results for all three baselines. "Preferred us" is the percentage of paragraphs generated by our system that users preferred over the paragraphs generated using the baseline. Our maps win all three comparisons, with percentage as high as 71.8% against Google News. The p-values resulting from all baselines are less than $0.01/3 = 0.0033$, showing that our results are significant with 99%, using the Bonferroni correction.

As noted earlier, both previous and new map representations operate over the same set of clusters. Therefore, the gap (58% : 42% in favour of the new maps) is due to our new structure and coverage algorithms.


Israel Metromap
heritage clashes list shrine hebron
visiting landing vashem paul yad
popes hopes trip holy scholar
muslim israelis jews jewish american
catholic, rome, benedict, vatican, xvi,
city, muslim, mosque, jerusalem, telephone,
synagogues, rabbi, investigations, police, attack,
OJ Simpson Metromap
deputy dismissed panel scheck bags
district conference professor rarely leather
mark district detectives fuhrman words
verdict system guilty reason view
broadcast cbs president program network
verdict view closely million united
system watching view johnnie million
stopped forcefully remains democrat powell
simpson, police, murder, trial, cochran,
rate, cbs, president, advertisers, viewers,
clinton, center, imprisonment, elliott, stuart,
left, hair, amendment, nigger, entering,

(a) Zooming into the yellow line of Figure 1

(b) OJ Simpson Map

Figure 8: Sample maps.

User Feedback:. We have asked users of the first phase for some feedback regarding the systems they have tested. 84.6% of the users who have used our new map representation have preferred it to the competitor. Sample comments included:

"It was so nice to see the different lines. I immediately could tell some of the important things I should write in the paragraph" / "The labels on the circles [metro stops] were of tremendous help." / "Getting a sense of time makes a big difference for me. I can also see which things are close to each other and which are unrelated."

5. CONCLUSIONS AND FUTURE WORK

In this paper, we have developed a *scalable* algorithm for creating metro maps. Given a query and a large collection of documents, our system extracts and visualizes interconnected storylines associated with the query. Our system pays particular attention to the *structure* of the maps, and accommodates different *granularities* of information through multi-resolution zooming. Our system reveals nuances and connections between events in a manner that is easily understandable by humans. The system is scalable since the runtime is only *linear* in the size of the query result set. Evaluation based on a user study, as well as manual inspection of extracted maps reveals superior performance of our system when compared to present state of the art. Overall, users tend to glean better understanding of events with metro maps than with competing systems.

Potential directions for future work include automatic query generation and refinement based on temporal dynamics of news. Incremental algorithms could allow maps to handle real-time streaming news. Novel visualizations and user interaction techniques would be another fruitful venue: for example, we consider "infinite" maps where users can seamlessly navigate the whole space of topics and events. We also hope to utilize summarization techniques to succinctly represent the content of a node. Most importantly, we hope this line of work will find practical uses and help people deal with the ever-increasing problem of information overload.

Acknowledgements. We thank Andreas Krause for his valuable comments. This research has been supported in part by NSF IIS-1016909, CNS-1010921, CAREER IIS-1149837, IIS-1159679, ARO MURI, DARPA SMISC, DARPA GRAPHS, Brown Institute for Media Innovation, Okawa Foundation, Docomo, Boeing, Allyes, Volkswagen, Intel, Alfred P. Sloan Fellowship and the Microsoft Faculty Fellowship.

6. REFERENCES

[1] Amazon mechanical turk, http://www.mturk.com/, 2013.

[2] Google news, http://news.google.com/, 2013.

[3] Spinn3r, http://spinn3r.com/, 2013.

[4] James Allan, Rahul Gupta, and Vikas Khandelwal. Temporal summaries of new topics. In *SIGIR '01*, 2001.

[5] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet allocation. *JMLR*, 2003.

[6] Rohan Choudhary, Sameep Mehta, and Amitabha Bagchi. On quantifying changes in temporally evolving dataset. In *CIKM '08*, pages 1459–1460, New York, NY, USA, 2008. ACM.

[7] Khalid El-Arini, Gaurav Veda, Dafna Shahaf, and Carlos Guestrin. Turning down the noise in the blogosphere. In *KDD '09*, pages 289–298, New York, NY, USA, 2009. ACM.

[8] Christos Faloutsos, Kevin S. McCurley, and Andrew Tomkins. Fast discovery of connection subgraphs. In *KDD '04*, 2004.

[9] Uriel Feige, Vahab S. Mirrokni, and Jan Vondrak. Maximizing non-monotone submodular functions. In *FOCS '07*, pages 461–471, Washington, DC, USA, 2007. IEEE Computer Society.

[10] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75 – 174, 2010.

[11] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. Cost-effective outbreak detection in networks. In *KDD' 07*, 2007.

[12] Chih-Jen Lin. Projected gradient methods for nonnegative matrix factorization. *Neural Computation*, 19(10), October 2007.

[13] Q Mei and C Zhai. Discovering evolutionary theme patterns from text: an exploration of temporal text mining. In *KDD '05*, 2005.

[14] G. Nemhauser, L. Wolsey, and M. Fisher. An analysis of the approximations for maximizing submodular set functions. *Mathematical Programming*, 14, 1978.

[15] G. Palla, I. Derényi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005.

[16] Hassan Sayyadi, Matthew Hurst, and Alexey Maykov. Event detection and tracking in social streams. In *ICWSM*, 2009.

[17] Dafna Shahaf and Carlos Guestrin. Connecting the dots between news articles. In *KDD '10*, 2010.

[18] Dafna Shahaf, Carlos Guestrin, and Eric Horvitz. Trains of thought: Generating information maps. In *WWW '12*, 2012.

[19] Russell Swan and David Jensen. TimeMines: Constructing Timelines with Statistical Models of Word Usage. In *KDD' 00*, 2000.

[20] Rui Yan, Xiaojun Wan, Jahna Otterbacher, Liang Kong, Xiaoming Li, and Yan Zhang. Evolutionary timeline summarization: a balanced optimization framework via iterative substitution. In *SIGIR*, 2011.

[21] J. Yang and J. Leskovec. Overlapping community detection at scale: A nonnegative matrix factorization approach. In *WSDM*, 2013.