Trains of Thought: Generating Information Maps

Dafna Shahaf
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA
dshahaf@cs.cmu.edu

Carlos Guestrin
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA
guestrin@cs.cmu.edu

Eric Horvitz
Microsoft Research
One Microsoft Way
Redmond, WA
horvitz@microsoft.com

ABSTRACT

When information is abundant, it becomes increasingly difficult to fit nuggets of knowledge into a sigle coherent picture. Complex stories spaghetti into branches, side stories, and intertwining narratives. In order to explore these stories, one needs a map to navigate unfamiliar territory. We propose a methodology for creating structured summaries of information, which we call *metro maps*. Our proposed algorithm generates a concise structured set of documents which maximizes coverage of salient pieces of information. Most importantly, metro maps explicitly show the relations among retrieved pieces in a way that captures story development. We first formalize characteristics of good maps and formulate their construction as an optimization problem. Then we provide efficient methods with theoretical guarantees for generating maps. Finally, we integrate user interaction into our framework, allowing users to alter the maps to better reflect their interests. Pilot user studies with a real-world dataset demonstrate that the method is able to produce maps which help users acquire knowledge efficiently.

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

Metro maps, Information, Summarization

1. INTRODUCTION

As data becomes increasingly ubiquitous, users are often overwhelmed by the flood of information available to them. Although search engines are effective in retrieving nuggets of knowledge, the task of fitting those nuggets into a single coherent picture remains difficult.

We are interested in methods for building more comprehensive views that *explicitly* show the relations among retrieved nuggets. We believe that such methods can enable people to navigate new, complex topics and discover previously unknown links. We shall focus on the news domain; for example, the system described in this paper can be used by a person who wishes to understand the debt crisis in Europe and its implications.

WWW 2012, April 16–20, 2012, Lyon, France.
ACM 978-1-4503-1229-5/12/04.

Previous news summarization systems with structured output [17, 18, 2] have focused mostly on *timeline* generation. However, this style of summarization only works for simple stories, which are linear in nature. In contrast, complex stories display a very non-linear structure: stories spaghetti into branches, side stories, dead ends, and intertwining narratives. To explore these stories, one needs a *map* to guide them through unfamiliar territory.

In this paper, we investigate methods for automatically creating *metro maps* of information. 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. Due to the sparsity of the output, it naturally lends itself to many visualization techniques. We chose to follow the metro-map metaphor: a metro map consists of a set of lines which have intersections or overlaps. Each line follows a coherent narrative thread; different lines focus on different aspects of the story. This visualization allows users to easily digest information at a holistic level, and also to interact with the model and make modifications.

Figure 1 shows a simplified metro map representing the debt crisis in Greece. The middle (blue) line details the chain of events leading from Greece's debt 'junk' status to the Greek bailout. The L-shaped (red) line is about strikes and riots in Greece. Both lines intersect at an article about the austerity plan, since it plays an important role in both storylines: it was a key precondition for Greece to get bailout money, but it also triggered many of the strikes.

To the best of our knowledge, the problem of constructing metro maps is novel. We believe that metro maps can serve as effective tools to help users cope with information overload in many fields. For example, maps can be a great vehicle for scientists exploring the research landscape. Our main contributions are as follows:

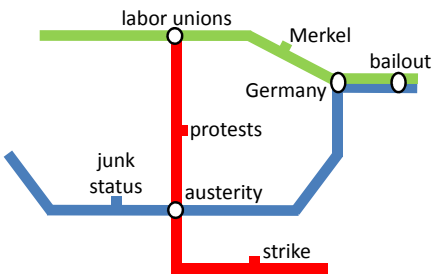


Figure 1: Greek debt crisis: a simplified metro map

- We introduce the concept of a metro map and formalize criteria characterizing good metro maps.
- We provide efficient methods with theoretical guarantees to construct good metro maps.
- We provide a randomized heuristic to generate a candidate set of good lines efficiently.
- We integrate user preferences into our framework by providing an interaction model.
- We conduct promising pilot user studies, comparing metro maps to Google News and to a state-of-the art topic detection and tracking system. The results indicate that our method helps users acquire knowledge more effectively on real-world datasets.

2. CRAFTING AN OBJECTIVE FUNCTION

What are desired properties of a metro map? In the following, we motivate and formalize several (sometimes conflicting) criteria. In Section 3, we present a principled approach to constructing maps that optimize tradeoffs among these criteria. First, we need to formally define metro maps.

DEFINITION 2.1 (METRO MAP). *A* metro map $\mathcal{M}$ *is a pair* (G, Π)*, where* $G = (V, E)$ *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.*

As an example, the map in Figure 1 includes three metro lines. Vertices V correspond to news articles, and are denoted by $docs(\mathcal{M})$. The lines of Π correspond to aspects of the story. A key requirement is that each line tells a coherent story: Following the articles along a line should give the user a clear understanding of evolution of a story.

Coherence is crucial for good maps, but is it sufficient as well? In order put this matter to a test, we found maximally-coherent lines for the query 'Bill Clinton' (using methods of Section 2.1). The results were discouraging. While the lines we found were indeed coherent, they were not *important*. Many of the lines revolved around narrow topics, such as Clinton's visit to Belfast, or his relationship with his religious leader. Furthermore, as there was no notion of diversity, the lines were very repetitive.

The example suggests that selecting the most coherent lines does not guarantee a good map. Instead, the key challenge is balancing coherence and **coverage**: in addition to being coherent, lines should also cover topics which are important to the user.

Finally, a map is more than just a set of lines; there is information in its *structure* as well. Therefore, our last property is **connectivity**. The map's **connectivity** should convey the underlying structure of the story, and how different aspects of the story interact with each other.

In Sections 2.1-2.3, we formalize **coherence**, **coverage** and **connectivity**. In Section 2.4, we explore their tradeoffs and combine them into one objective function.

2.1 Coherence

How should we measure coherence of a chain of articles? We rely on the notion of coherence developed in Connect-the-Dots (CTD) [16]. In the following, we briefly review this notion.

In order to define coherence, a natural first step is to measure similarity between each pair of consecutive articles along the chain. Since a single poor transition can destroy the coherence of the entire chain, we measure the strength of the chain by the strength of its *weakest link*.

Chain A	Chain B
• Europe weights possibility of debt default in Greece	• Europe weights possibility of debt default in Greece
• Why Republicans don't fear a debt default	• Europe commits to action on Greek debt
• Italy; The Pope's leaning toward Republican ideas	• Europe union moves towards a bailout of Greece
• Italian-American groups protest 'Sopranos'	• Greece set to release austerity plan
• Greek workers protest austerity plan	• Greek workers protest austerity plan

However, such a simple approach can produce poor chains. Consider, for example, Chain A. The transitions of Chain A are all reasonable when examined out of context. For example, the first two articles are about debt default; the second and third mention Republicans. However, the overall effect is associative and incoherent. Now, consider Chain B. This chain has exactly the same endpoints, but it is much more coherent.

Let us take a closer look at these chains: Figure 2 shows word patterns along both chains. Bars correspond to appearance of words in the articles depicted above them. For example, 'Greece' appeared throughout Chain B. It is easy to spot the associative flow of Chain A in Figure 2. Words appear for short stretches, often only in two neighbouring articles. Contrast this with Chain B, where stretches are longer, and transitions between documents are smoother. This observation motivates our definition of coherence.

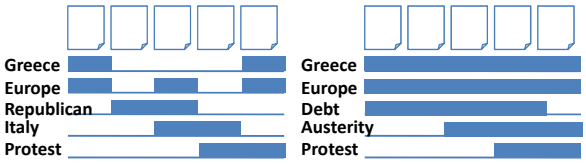


Figure 2: Word patterns in Chain A (left) and B (right). Bars correspond to the appearance of a word in the articles depicted above them.

We represent documents as feature vectors (for the sake of presentation, assume features $\mathcal{W}$ are words). Given a chain of articles $(d_1, ..., d_n)$, we first score each transition $d_i \rightarrow d_{i+1}$ by the number of words that both articles share:

$$Coherence(d_1, ..., d_n) = \min_{i=1...n-1} \sum_{w \in \mathcal{W}} \mathbb{1}(w \in d_i \cap d_{i+1})$$

However, word appearance alone is too noisy. Articles must use the exact same words; synonyms (or related words) are treated as unrelated. Also, all words are treated equally: the word 'Greece' is as important as the word 'today'.

Therefore, one can replace the indicator function $\mathbb{1}(\cdot)$ with a notion of *importance* of feature w in a transition. This notion takes word co-occurrence into account, reducing the noise considerably. It also considers the word's importance on a corpus level and on a document level (tf-idf). See [16] for details.

$$Coherence(d_1, ..., d_n) = \min_{i=1...n-1} \sum_{w} Importance(w \mid d_i, d_{i+1})$$

This objective guarantees good transitions, but associative chains like Chain A can still score well. However, these chains need to use a lot more words in order to achieve this high score, since many of their transitions use a unique set of words. On the other hand, coherent chains (like Chain

B) can often be characterized by a *small* set of words, which are important throughout many of the transitions.

Therefore, instead of summing $Importance(w \mid d_i, d_{i+1})$ over all features, the problem is transformed into an optimization problem, where the goal is to choose a small set of words (called 'active') and score the chain based on them alone. Constraints on possible choices (see [16]) enforce a small number of words and smooth transitions, imitating the behaviour of Figure 2 (right).

$$Coherence(d_1, ..., d_n) = \max_{activations} \min_{i=1...n-1} \sum_w Importance(w \mid d_i, d_{i+1}) \mathbb{1}(w \text{ active in } d_i, d_{i+1})$$

Finally, the coherence of a map is defined as the minimal coherence across its lines Π.

2.2 Coverage

In addition to coherence, we need to ensure that the map has high coverage. The goal of coverage is twofold: we want to both cover **important** aspects of the story, but also encourage **diversity**.

Our definition is inspired by [6]. Before we measure the coverage of an entire map, we consider the coverage of a *single* document. As in the previous section, documents are feature vectors. Let function $cover_{d_i}(w) : \mathcal{W} \to [0, 1]$ quantify the amount that document d_i covers feature w. For example, if $\mathcal{W}$ is a set of words, we can define $cover_{\cdot}(\cdot)$ as tf-idf values.

Next, we extend $cover_{\cdot}(\cdot)$ to maps. Since in our model coverage does not depend on map structure, it is enough to extend $cover_{\cdot}(\cdot)$ to a function over *sets* of documents.

A natural candidate for $cover_{\mathcal{M}}(w)$ is to view set-coverage as an additive process:

$$cover_{\mathcal{M}}(w) = \sum_{d_i \in docs(\mathcal{M})} cover_{d_i}(w)$$

Additive coverage is natural and easily computable. However, it suffers from one major drawback: since the coverage each document provides is independent of the rest of the map, additive coverage does not encourage **diversity**. In order to encourage diversity, we view set-coverage as a sampling procedure: each document in the map tries to cover feature w with probability $cover_{d_i}(w)$. The coverage of w is the probability at least one of the documents succeeded[1]:

$$cover_{\mathcal{M}}(w) = 1 - \prod_{d_i \in docs(\mathcal{M})} (1 - cover_{d_i}(w))$$

Thus, if the map already includes documents which cover w well, $cover_{\mathcal{M}}(w)$ is close to 1, and adding another document which covers w well provides very little extra coverage of w. This encourages us to pick articles which cover other features, promoting **diversity**.

We now have a way to measure how much a map covers a feature. Finally, we want to measure how much a map covers the entire *corpus*. Remember, our goal is to ensure that the map touches upon **important** aspects of the corpus. Therefore, we first assign weights λ_w to each feature w, signifying the importance of the feature. For example, if features are words (and stopwords have been removed), the weights can correspond to word frequency in the dataset.

[1]If $cover_{d_i}(w)$ are very small, we may want to sample more than once from each document.

We model the amount $\mathcal{M}$ covers the corpus as the weighted sum of the amount it covers each feature:

$$Cover(\mathcal{M}) = \sum_w \lambda_w cover_{\mathcal{M}}(w)$$

The weights cause *Cover* to prefer maps which cover important features of the corpus. In Section 6 we discuss learning a personalized notion of coverage.

2.3 Connectivity

Our final property is connectivity. There are many ways to measure connectivity of a map: one can count the number of connected components, or perhaps the number of vertices that belong to more than one line.

We conducted preliminary experiments exploring different notions of connectivity. These results suggest that the most glaring usability issue arises when maps do not show connections that the user knows about. For example, in a map about Israel, a line about legislative elections was not connected to a line about the chosen government's actions.

We came to the conclusion that the type of connection (one article, multiple articles, position along the line) was not as important as its mere existence. Therefore, we simply define connectivity as the number of lines of Π that intersect:

$$Conn(M) = \sum_{i<j} \mathbb{1}(p_i \cap p_j \neq \emptyset)$$

2.4 Objective function: Tying it all together

Now that we have formally defined our three properties, we can combine them into one objective function. We need to consider tradeoffs among these properties: for example, maximizing coherence often results in repetitive, low-coverage chains. Maximizing connectivity encourages choosing similar chains, resulting in low coverage as well. Maximizing coverage leads to low connectivity, as there is no reason to re-use an article for more than one line.

Let us start with coherence. As mentioned in Section 2, we are not interested in *maximizing* coherence. Instead, we treat coherence as a *constraint*: only consider lines above a certain coherence threshold τ, whether absolute or relative (see Section 5 for parameter tuning). In the following, we assume that τ is fixed, and denote a chain coherent if its coherence is above τ.

We are left with coverage and connectivity for our objective. Suppose we pick connectivity as our primary objective. Our biggest obstacle is that coherent lines tend to come in groups: a coherent line is often accompanied by multiple similar lines. Those lines all intersect with each other, so choosing them maximizes connectivity. However, the resulting map will be highly redundant.

For this reason, we choose coverage as our primary objective. Let κ be the maximal coverage across maps with coherence $\geq \tau$. We can now formulate our problem:

PROBLEM 2.2. *Given a set of candidate documents $\mathcal{D}$, find a map $\mathcal{M} = (G, \Pi)$ over $\mathcal{D}$ which maximizes $Conn(\mathcal{M})$ s.t. $Coherence(\mathcal{M}) \geq \tau$ and $Cover(\mathcal{M}) = \kappa$.*

In other words, we first maximize coverage; then we maximize connectivity over maps that exhibit maximal coverage.

There is one problem left with our objective: consider two metro lines that intersect at article d. Our coverage function is a set function, therefore d is accounted for only once. In other words, replacing d in one of the lines can only increase coverage. Since there usually exists a similar

article d' which can replace d, the max-coverage map is often *disconnected*. Worse yet, it is often *unique*. In order to mitigate this problem, we introduce slack 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 maximizes Conn($\mathcal{M}$) s.t. Coherence($\mathcal{M}$) $\geq \tau$ and Cover($\mathcal{M}$) $\geq (1-\epsilon)\kappa$.*

for a given, small ϵ.

Finally, we need to restrict the size of $\mathcal{M}$; we chose to restrict $\mathcal{M}$ to K lines of length at most l. Alternatively, since some stories are more complex than others, one may prefer to add lines until coverage gains fall below a threshold.

3. ALGORITHM

In this section, we outline our approach for solving Problem 2.3. In Section 3.1 we represent all coherent chains as a graph. In Section 3.2 we use this graph to find a set of K chains that maximize coverage; in Section 3.3, we increase connectivity without sacrificing coverage.

3.1 Representing all coherent chains

In order to pick good chains, we first wish to list all possible candidates. However, representing all chains whose coherence is at least τ is a non-trivial task. The number of possible chains may be exponential, and therefore it is infeasible to enumerate them all, let alone evaluate them.

Instead we propose a divide-and-conquer approach, constructing long chains from shorter ones. This approach allows us to compactly encode many candidate chains as a graph. See Figure 3 for an illustration: each vertex of the graph corresponds to a short chain. Edges indicate chains which can be concatenated and still maintain coherence. A path in the graph corresponds to the concatenated chain.

It is tempting to concatenate any two chains that share an endpoint. That is, concatenate $(d_1, ..., d_k)$ and $(d_k, ..., d_{2k-1})$ to form $(d_1, ..., d_{2k-1})$. However, caution is needed, as combining two strong chains may result in a much weaker chain. For example, Chain B ended with an article about protests in Greece. If we concatenate it with a (coherent) chain about protests across the globe, the concatenated chain will change its focus mid-way, weakening coherence.

The problem appears to lie with the *point of discontinuity*: when we concatenate $(d_1, ..., d_k)$ with $(d_k, ..., d_{2k-1})$, there is no evidence that both chains belong to the same storyline, despite having a shared article d_k. From the user's point of view, the first k articles are coherent, but $(d_2, ..., d_{k+1})$ may not be. This observation motivates our next definition:

DEFINITION 3.1 (m-COHERENCE). *A chain $(d_1, ..., d_k)$ has m-coherence τ if each sub-chain of length m $(d_i, ..., d_{i+m-1})$, $i = 1, ..., k-m+1$ has coherence at least τ.*

The idea behind m-coherence is to control the discontinuity points. Choosing m is a tradeoff: Increasing m results in more-coherent chains, as the user's 'history window' is wider. However, it is also more computationally expensive. In particular, if $m = l$ the user remembers the entire chain, thus l-coherence is equivalent to the regular notion of coherence. If $m = 2$, the user only remembers the last edge; therefore, 2-coherent chains optimize transitions without context, and can result in associative chains like Chain A.

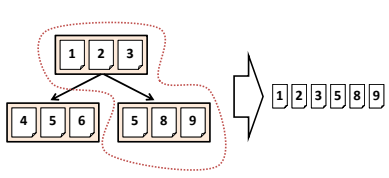


Figure 3: Encoding chains as a graph: each vertex of the graph corresponds to a short chain. A path in the graph corresponds to the concatenated chain.

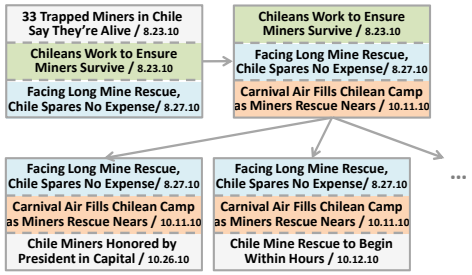


Figure 4: A fragment of the coherence graph $\mathbb{G}$ for $m = 3$. Note overlap between vertices.

In practice, we chose the highest m we could afford computationally (our website should handle queries in real time). After choosing an appropriate m, we rephrase Problem 2.3:

PROBLEM 3.2. *Given a set of candidate documents $\mathcal{D}$, find a map $\mathcal{M} = (G, \Pi)$ over $\mathcal{D}$ which maximizes Conn($\mathcal{M}$) s.t. m-Coherence($\mathcal{M}$) $\geq \tau$ and Cover($\mathcal{M}$) $\geq (1-\epsilon)\kappa$.*

Representing all chains whose *m-coherence* is at least τ is a less daunting task. Observe that m-coherent chains can be *combined* to form other m-coherent chains if their overlap is large enough. Specifically, we require overlap of at least $(m-1)$ articles:

OBSERVATION 3.3. *If chains $c = (d_1, ..., d_k)$ and $c' = (d_{k-(m-2)}, ..., d_k, ..., d_r)$ are both m-coherent for $k \geq m > 1$, then the conjoined chain $(d_1, ..., d_k, ..., d_r)$ is also m-coherent.*

The proof follows directly from Definition 3.1.

We can now construct a graph $\mathbb{G}$ encoding all m-coherent chains. We call $\mathbb{G}$ a *coherence graph*. Vertices of $\mathbb{G}$ correspond to coherent chains of length m. There is a directed edge between each pair of vertices which can be conjoined ($m-1$ overlap). It follows from observation 3.3 that all paths of $\mathbb{G}$ correspond to m-coherent chains.

We are still left with the task of finding short coherent chains to serve as vertices of $\mathbb{G}$. These chains can be generated by a *general best-first* search strategy. In a nutshell, we keep a priority queue of sub-chains. At each iteration, we expand the chain which features the highest coherence, generating all of its extensions. When we reach a chain of length m, we make it into a new vertex and remove it from the queue. We continue until we reach our threshold. Since the evaluation function used to sort the queue is admissible (as a subchain is always at least as coherent a chain which extends it), optimality is guaranteed. In Section 4, we present a faster method to find good short chains.

Example Coherence Graphs Figure 4 shows a fragment of a coherence graph for $m = 3$. The figure depicts multiple ways to extend the story about the trapped Chilean miners: one can either focus on the rescue, or skip directly to the post-rescue celebrations.

3.2 Finding a high-coverage map

In the previous section, we constructed a coherence graph $\mathbb{G}$ representing all coherent chains. Next, we seek to use

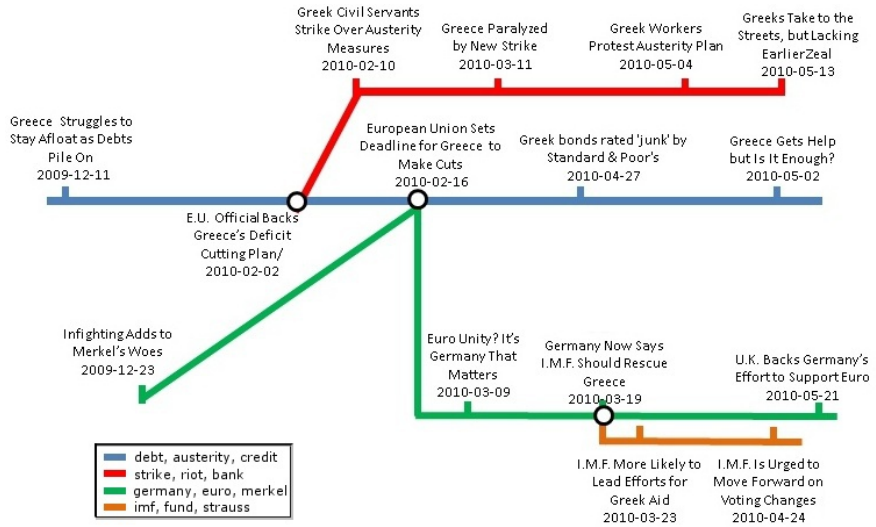


Figure 5: An example of our results (condensed to fit space). This map was computed for the query 'Gree* debt'. The main storylines discuss the austerity plans, the riots, and the role of Germany and the IMF in the crisis.

this graph to find a set of chains which maximize coverage, subject to map size constraints.

PROBLEM 3.4. *Given $\mathbb{G}$ coherence graph, find paths $p_1 \ldots p_K$ s.t. $Cover(docs(\bigcup_i p_i))$ is maximized, and $|docs(p_i)| \leq l$.*

This problem is NP-hard, which necessitates resorting to approximation methods. First, let us pretend that we can enumerate all paths of $\mathbb{G}$ that contain up to l documents. Then, we can take advantage of the *submodularity* of $Cover(\cdot)$:

DEFINITION 3.5 (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, $Cover(\cdot)$ is submodular since reading some article v after already reading articles A provides more coverage than reading v after reading a superset of A [6].

Although maximizing submodular functions is still NP-hard, we can exploit the classic result of [13], which shows that the greedy algorithm achieves a $(1 - \frac{1}{e})$ approximation. In other words, we run K iterations of the greedy algorithm. In each iteration, we evaluate the *incremental* coverage of each candidate path p, given the paths which have been chosen in previous iterations:

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

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

Let us revisit our assumption: unfortunately, enumerating all candidate paths is generally infeasible. Instead, we propose a different approach: suppose we knew the max-coverage path for each pair of fixed endpoints, documents d_i and d_j. Then, we could modify the greedy algorithm to greedily pick a path amongst these paths only. Since there are only $O(|\mathcal{D}|^2)$ such pairs, greedy is feasible.

Computing the max-coverage path between two endpoints is still a hard problem. In order to solve it, we formulate our problem in terms of *orienteering*. Orienteering problems are motivated by maximizing some function of the nodes visited during a tour, subject to a budget on the tour length.

PROBLEM 3.6 (ORIENTEERING). *Given an edge-weighted directed graph $G = (V, E, len)$ and a pair of nodes s, t, find an s-t walk of length at most B that maximizes a given function $f : 2^V \rightarrow \mathbb{R}^+$ of the set of nodes visited by the walk.*

We set all edge lengths to 1. We want a path containing at most l articles; since each vertex of $\mathbb{G}$ corresponds to m articles, and the overlap is $m - 1$, we set the budget B to be $l - m$. In addition, we want f to reflect the incremental coverage of path p given the current map, so we define

$$f(p) = IncCover(p|\mathcal{M})$$

We adapt the submodular orienteering algorithms of [4] to our problem. This is a quasipolynomial time recursive greedy algorithm. Most importantly, it yields an $\alpha = O(\log OPT)$ approximation. We combine the greedy algorithm with submodular orienteering. At each round, we compute approximate best-paths between every two documents (given the chains which have been selected in previous iterations) using submodular orienteering. We then greedily pick the best one amongst them for the map. The algorithm achieves a $1 - \frac{1}{e^\alpha}$ approximation.

The main bottleneck in our algorithm 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 chain 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.

3.3 Increasing connectivity

We now know how to find a high-coverage, coherent map $\mathcal{M}_0$. Our final step is to increase connectivity without sacrificing (more than an ϵ-fraction of) coverage.

In order to increase connectivity, we apply a local-search technique. At iteration i, we consider each path $p \in \Pi_{i-1}$. We hold the rest of the map fixed, and try to replace p by p' that increases connectivity and does not decrease coverage. At the end of the iteration, we pick the best move and apply it, resulting in $\mathcal{M}_i$.

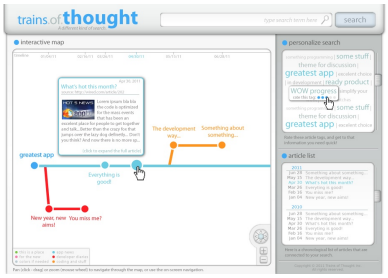
In order to find good candidates to replace a path p, we consider the map without p, $\mathcal{M}_{i-1} \setminus p$. We re-use the tech-

nique of submodular orienteering and compute approximate max-coverage paths between every two documents. In order to guide the process, we can bias the orienteering algorithm into preferring vertices that already appear in $\mathcal{M}_{i-1} \setminus p$. We consider all chains which do not decrease map coverage, and pick the one which maximizes connectivity. We stop when the solution value has not changed for T iterations.

EXAMPLE 1 (MAP). *Figure 5 displays a sample map generated by the methodology. This map was computed for the query 'Gree* debt'. The main storylines discuss the austerity plans, the riots, and the role of Germany and the IMF in the crisis. In order to facilitate navigation in the map, we have added a* legend *feature. We assign a few characteristic words to each line. The words chosen to describe each line carry the highest incremental coverage for that line.*

4. IMPLEMENTATION

We have created a website that allows interactive visualization of metro maps, which we hope to launch soon.



In this section, we discuss some of the practical implementation issues. In particular, the algorithm of Section 3 takes several minutes for most queries, which is not acceptable for a website. Following an analysis of our algorithm, we propose a method to speed up our bottlenecks.

4.1 Analysis

Let us analyze the complexity of the algorithm presented in Section 3. Suppose our input consists of a set of documents $\mathcal{D}$. The algorithm is composed of the following steps:

1. Constructing coherence graph vertices generated by general best-first heuristic (Section 3.1) may require solving $O(|\mathcal{D}|^m)$ linear programs in the worst case. The number of edges may be $O(|\mathcal{D}|^{2m})$: however, using an implicit representation (hashing prefixes and suffixes of each vertex) we only need an expected time of $O(|\mathcal{D}|^m)$ to store them.

2. The coverage step (Section 3.2) requires K iterations of the greedy algorithm. Each iteration requires solving $O(|\mathcal{D}|^2)$ orienteering problems using the quasipolynomial algorithm of [4].

3. The connectivity step (Section 3.3) performs local search. Each iteration requires in the worst case solving another $O(K|\mathcal{D}|^2)$ orienteering problems.

4. The visualization step, which was not described earlier, is based on a force-directed layout. The algorithm assigns forces among edges as if they were springs, and simulates the graph like a physical system. We have modified the algorithm so that the graph respects chronological order among vertices (by using x as a temporal axis); we have also tried to minimize the number of kinks and turns in a line.

Steps 1-3 are computationally intensive. As noted earlier, lazy evaluations lead to dramatic speedups in steps 2 and 3 without losing approximation bounds. Furthermore, steps 2 and 3 are very easy to parallelize. Step 1, on the other hand, is both harder to parallelize and requires an external LP solver. In the next section, we propose a practical method to speed this step up.

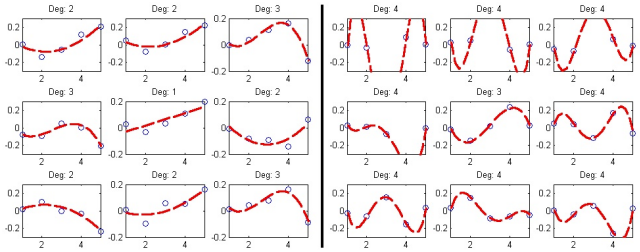


Figure 6: Behaviour of the top-9 features for Chain B (left), and B' (see in text, right). x axis represents document position in the chain, y axis represents PCA feature value. Circles correspond to the actual feature values in the chain, and dashed lines are the lowest-degree polynomial which fits these points within a specified error. The degree of each polynomial appears above it.

4.2 Speeding up the coherence graph

We are interested in a fast way to build a coherence graph $\mathbb{G}$. The CTD coherence notion was created with long chains in mind; we can take advantage of the fact that chains we are interested in are *short* (length m). Shorter chains are simpler: articles are often more related to each other, and there is less topic drift. In terms of Section 2.1, selected words are active throughout the entire chain, which makes them easier to identify; for example, we can look at the words with the highest median (or average) value.

Next, we look at the behaviour of these words throughout the chain. We observe that in many good chains, words exhibit a smooth behaviour: in Chain B, the word 'austerity' becomes more important as the chain progresses, while 'EU' decreases and 'debt' is stable. In poor chains, words tend to fluctuate more.

In order to formalize this, we look at the series of coefficients of each word throughout the chain, and try to express it as a low-degree polynomial. Figure 6 demonstrates this approach. We tested two chains: Chain B and Chain B', which was obtained from Chain B by replacing an intermediate article by an article about Hungary and the Greek debt. Intuitively, this change made Chain B' less coherent.

Figure 6 shows the behaviour of the top 9 features for Chain B (left), and B' (right). As words were too noisy, we used PCA components instead. x axis represents document position in the chain, y axis represents (PCA) feature value. Circles correspond to the actual values, and dashed lines are the lowest-degree polynomial which fits them within a specified error. The degree of each polynomial appears above it (note that one can always can fit a polynomial of degree 4 through 5 points). As expected, Chain B needs lower-degree polynomials than Chain B'.

We have experimented with low-degree polynomials as a measure of chain quality. Chains displaying low-degree behaviour were usually coherent, but some of our hand-picked coherent chains required high-degree. That is, the process seems biased towards false negatives. However, according to our observations, this bias does not pose a problem when $\mathcal{D}$ is large enough. If a coherent chain did not score well, there was usually a similar chain which scored better.

Algorithm 1: `FindShortChains`$(\mathcal{D}, m)$

input : $\mathcal{D}$ a set of documents, m desired length
output: A set of chains of length m.

1 **for** *K iterations* **do**
2 Randomly select $(d, d') \in \mathcal{D}^2$;
// Create a model of a chain between d and d'
3 **foreach** $(w) \in$ `importantFeatures`$(\{d, d'\})$ **do**
4 $Model_{d,d'}(w) =$ `polyfit`$((1, d(w)), (m, d'(w)))$;
// Evaluate other documents
5 Initialize array *FitDocs* to $\emptyset$;
6 **foreach** $d'' \in \mathcal{D} \setminus \{d, d'\}$ **do**
// Find best position for d'' (null if far)
7 $i'' =$ `bestPos`$(d'', Model_{d,d'})$;
8 **if** $i'' \neq$ *null* **then**
9 $FitDocs[i''] = FitDocs[i''] \cup \{d''\}$;
10 $score =$ `getScore`$(FitDocs)$;
11 Record best model as $Model^*$, and its *FitDocs* array as $FitDocs^*$;
12 **if** $Model^*$ has a low score **then return** ;
// Good model found. Reestimate from fitting docs
13 **foreach** $(w) \in$ `importantFeatures`$(FitDocs^*)$ **do**
14 $Model^*_{new}(w) =$ `polyfit`$(FitDocs^*(w), degree)$;
15 **return** *extractChains*$(FitDocs, Model^*_{new})$;

We can now describe our algorithm for finding good short chains (see Algorithm 1). Our approach is inspired by RANSAC [8]. RANSAC is a random sampling method. In each iteration, we randomly select a set of candidate pairs of articles, $\{(d, d')\}$, to be used as chain endpoints (Line 2). We then hypothesize a model for the rest of the chain (Line 4). A model is a sequence of predicted values for each feature. For example, if d and d' both display high levels of feature w, we expect the rest of the documents in any coherent chain to display similar high levels. If d displays higher levels of w, we expect to observe this trend in the rest of the chain. Since we only have two points to estimate the model from, we simply fit a line between them.

We then evaluate our model by finding other articles that closely fit the model's prediction. For each article d'', we find the best position in the chain: i.e., the position that minimizes d'''s distance from the model (Line 7, function `bestPos`). If d'' is close enough to the model, this position is recorded in the *FitDocs* array.

After all documents were tested, Function `getScore` computes the number of chains that can be generated from *FitDocs*. If there are no chronological constraints, this is simply $\prod_i |FitDocs[i]|$: there are $|FitDocs[i]|$ options for the ith position. Otherwise, we construct a directed acyclic graph corresponding to constraints, and apply a linear-time algorithm to count the number of possible paths.

We repeat this process for multiple candidate pairs, and then pick the best model. Since the model was estimated only from the initial two articles, we re-estimate it from all of its close articles *FitDocs* (Line 14). Finally, we extract short chains that are a close fit to the re-estimated model.

Our algorithm is not guaranteed to succeed, since it may not draw documents that capture a good model. However, since many document pairs do encode a good model, the algorithm works well in practice. It is also fast and easy to parallelize. In addition, the algorithm provides an interesting interpretation of m-coherence: One can think of a chain as a ride through feature-space. Each sub-chain has a smooth trajectory, when projected on important-feature axis. Because of the large overlap between sub-chains, we are only allowed gentle adjustments to the steering wheel as we progress throughout the chain.

As expected, the algorithm tends to recover topics that are heavily represented in the dataset; topics that are poorly-represented are less likely to be sampled. Nevertheless, the chains recovered are of comparable quality to the chains recovered by methods of Section 3.1.

5. USER STUDY

In our user study, we evaluate the effectiveness of metro maps in aiding users navigate, consume, and integrate difference aspects of a multi-faceted information need. Our experiments were designed to answer the following questions:

- Accuracy: How well do the documents selected for the map summarize the topic of the task?
- Micro-Knowledge: Can the maps help users retrieve information faster than other methods?
- Macro-Knowledge: Can the maps help users understand the big picture better than other methods?
- Structure: What is the effect of the map structure?

We assembled a corpus of 18,641 articles from the International section of the *New York Times*, ranging from 2008 to 2010. This corpus was selected because of the material's relative accessibility, as news articles are written with a broad reader population in mind. Stopword removal and stemming have been performed as a preprocessing step.

We created three news exploration tasks, representing use cases where the reader is interested in learning about the trapped Chilean miners, the earthquake in Haiti, and the debt crisis in Greece. We refer to these tasks as *Chile*, *Haiti*, and *Greece*. Our tasks were chosen in order to cover different scenarios: The *Chile* task is very focused, concentrating on a single geographic location and a short time period. The *Haiti* task is broader, and the *Greece* task was the most complicated, as it spans multiple countries for a long period of time. In the following, we outline our evaluations.

5.1 Accuracy

In this study, we evaluate the map's content. Before we let users interact with our system and look for information, we want to know whether the information is there at all.

For each task, three domain experts composed a list of the top ten events related to the task. The experts composed their lists separately, and the top ten events mentioned most often were chosen. For example, *Chile* events included the accident, miners' discovery, miners' video, drill beginning and completion, first and last miner outside, release from the hospital, media coverage of the saga, and the presidential ceremony held in the miners' honor.

We then asked the experts to identify those events in metro maps of different sizes (3-6 lines of length at most 6). Below we measure *subtopic recall* (fraction of the important events that are successfully retrieved) of our method. In general, results are high: many of the important events are captured.

Lines	3	4	5	6
Chile	80%	100%	100%	100%
Haiti	50%	70%	80%	80%
Greece	30%	60%	60%	70%

Note that high subtopic precision (fraction of retrieved documents which are relevant to the top ten events) is not a desired property of metro maps: high precision means that

the maps is very focused on a small set of events, implying repetitiveness. If the top ten events are already covered, submodular coverage will try to cover side stories as well.

5.2 Micro-Knowledge and Structure

In this study, our goal is to examine maps as retrieval tools; we wish to see how maps help users answer specific questions. We compare the level of knowledge (per time step) attained by people using our prototype vs. two other systems: Google News and TDT. **Google News** is a computer-generated site that aggregates headlines from news sources worldwide. News-viewing tools are dominated by portal and search approaches, and Google News is a typical representative of those tools. **TDT** [12] is a successful system which captures the rich structure of events and their dependencies in a news topic.

We computed maps by methods of Section 4. We set m=3 for quick computation. After experimenting with several other queries, we set the coherence threshold to top 15%. Instead of fixing the number of chains, we continued to add chains until additional coverage was less than 20% of the total coverage (since we use greedy coverage, there will be at most 5 chains).

We implemented TDT based on [12] (cos+TD+Simple-Thresholding). We used the same articles $\mathcal{D}$ for maps and for TDT. We picked $\mathcal{D}$ using broad queries: 'chile miners', 'haiti earthquake' and 'gree* debt'. We queried Google News for 'chile miners', 'haiti earthquake' and 'greece debt' (plus appropriate date ranges). We did not restrict Google News to NYTimes articles, as not all of them are included. We ensured that all systems display the same number of articles: for Google News, we picked the top articles. For TDT, we picked a representative article from each cluster. The purpose of the study was to test a single query. We defer the evaluation of the interactive component to Section 6.

We note that comparing the different systems is problematic, as the output of Google News and TDT is different both in content and in presentation (and in particular, cannot be double-blind), so it is hard to know what to attribute observed differences to. In order to isolate the effects of document selection vs. map organization, we introduce a hybrid system into the study: the system, **Structureless metro maps** displays the same articles as metro maps but with none of the structure. Instead, articles are sorted chronologically and displayed in a fashion similar to Google News.

We recruited participants in the study via Amazon Mechanical Turk. Each user chose the number of tasks they were interested in doing out of the three tasks available. For each selected task, one of the four methods was assigned randomly. To make the users more comfortable with the system (and unfamiliar map interface), we asked them to do a warm-up task: copy the first sentence of the tenth article. After the warm-up, users were asked to answer a short questionnaire (ten questions), composed by domain experts. Users were asked to answer as many questions as possible in 10 minutes. In order to counter bias introduced by prior knowledge, the users had to specify the article where the answer was found. A *honey pot* question (an especially easy question, that we expect all workers to be able to answer) was used to identify spammers. After removing users who got this question wrong, we were left with 338 unique users performing 451 tasks.

A snapshot of the users' progress (number of correct answers) was taken every minute. Our interest is twofold: we wish to measure the user's *total knowledge* after ten minutes, and also the *rate* of capturing new knowledge. Figure 7 shows the results. x axis corresponds to time, and y axis corresponds to the average number of correct answers.

The results indicate that metro maps are especially useful for complex tasks, such as Greece. In this case, maps achieve higher scores than Google and TDT at the end of the test, as the advantage of structure outweighs the cost of ingesting the additional structure and grappling with an unfamiliar interface. Perhaps more importantly, the rate of capturing new knowledge is higher for maps.

The structureless methods do better for the simple task of Chile. Upon closer examination of the users' browsing patterns, it seems that many of the answers could be found in the (chronologically) first and last articles; the first article provided the basic facts, and the last summarized the story. We believe that this is the reason for the map's performance.

Let us examine Structurelss Maps. As discussed earlier, the fact that Structurelss Maps outperforms Google News is due to article selection. Metro maps and structureless maps seem comparable, but Metro Map users acquire knowledge more quickly, especially for complex stories; e.g., consider the first few minutes of the Greece task in Figure 7.

As a side note, the small number of correct answers is worrisome. We found that the main cause of mistakes was date-related questions; many of the participants entered the article's date, rather than the event's. Since about 30% of our questions involved dates, this affected the results severely. In addition, the majority of Turk users are non U.S-based (and non-native English speakers)[2]. When we conducted a preliminary survey across CMU undergrads, the average number of correct answers was significantly higher.

Finally, we compare the *ease of navigation*. If a user has a question in mind, we estimate the difficulty of finding an article containing the answer by computing the number of articles that users clicked per correct answer:

Maps	SL Maps	Google	TDT
2.1	3.74	5.28	4.91

Metro maps require the least amounts of clicks to reach an answer. Most importantly, maps did better than structureless maps, demonstrating the utility of the structure.

5.3 Macro-Knowledge

The retrieval study in the previous section evaluated users' ability to answer specific questions. We are also interested in the use of metro maps as high-level overviews, allowing users to 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 recruited 15 undergraduate students and asked them to write two paragraphs: one summarizing the Haiti earthquake, and one summarizing the Greek debt crisis. For each of the stories, the students were randomly assigned either a metro map or the Google News result page (stripped of logo and typical formatting, to avoid bias).

We then used Mechanical Turk to evaluate the paragraphs. At each round, workers were presented a two paragraphs (map user vs. Google News 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 ('Paragraph A is more...').

After removing spam, we had 294 evaluations for Greece, and 290 for Haiti. 72% of the Greece comparisons preferred

[2]http://www.behind-the-enemy-lines.com/2010/03/new-demographics-of-mechanical-turk.html

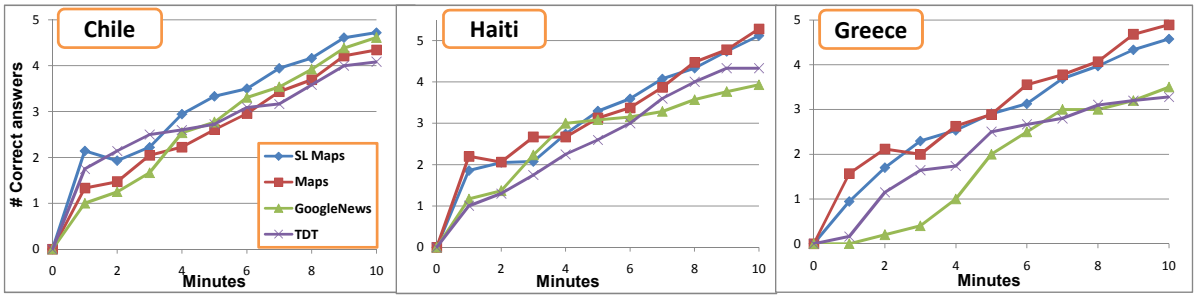


Figure 7: User study results: average number of correct answers amongst users vs. time. As the task gets more complex, metro maps become more useful.

Figure 8: Tag clouds representing descriptions of Google News (left) and Map (right) pararaphs. Note maps receive more positive adjectives.

map paragraphs, but only 59% of Haiti. After examining the Haiti paragraphs, we found that the all paragraphs were very similar; most followed the same pattern (earthquake, damages, distributing aid). None of the paragraphs mentioned the Haitian child smugglers, and only one mentioned the temporary laws for dislocated Haitians, despite the fact that both stories appeared in the map. A possible explanation was given by one of the participants: "I chose to not use the American politics. If this is a summary about the event I wanted to remain as objective as possible". In other words, map users avoided some storylines intentionally. As in the previous section, maps are more useful for stories without a single dominant storyline ('the event'), like Greece.

Finally, Figure 8 shows tag clouds of the words workers chose to describe the Greece paragraphs. Sample map paragraphs descriptions include 'gives a clear picture' and 'gives a better understanding of the debt crisis'. Google News paragraph descriptions included 'good but just explained about the facts' and 'more like a list of what happened'.

6. PERSONALIZATION AND INTERACTION

Models of interaction can be naturally integrated with metro maps. In order to be useful, the model must be capable of representing users' interests. We rely on user feedback in order to learn preferences and adjust the maps accordingly. In the following, we illustrate the potential of *learning a personalized coverage function* with an example.

Since our coverage objective is a set function, the most natural notion of feedback from a machine learning perspective would be for users to provide a single label for the map, indicating whether they like or dislike it. However, this approach is not practical. Since there are exponentially many such maps, we are likely to need an extensive amount of user feedback before we could learn the function.

Even more importantly, this approach makes it hard for users to form expressive queries. Ideally, we would like to support queries of the form 'I want to know more about the involvement of Germany in the debt crisis', or 'I do not want to know about Wyclef Jean's Haitian Presidential bid'. However, labeling entire maps – or even single documents – is just not rich enough to support this query model. Indeed, the user could indicate that they dislike Wyclef Jean articles shown to them, but there is no way for them to specify that they would like to see something which is not on the map.

We propose to let the user provide *feature-based feedback* instead. Feature-based feedback provides a very natural way for supporting the queries mentioned above. For example, the user could increase the importance of the word 'Germany' and decrease the importance of 'Wyclef Jean' to achieve the desired effect.

There has been growing recent interest in feature-based feedback. [5] proposed a discriminative semi-supervised learning method that incorporates into training affinities between features and classes. For example, in a baseball vs. hockey text classification problem, the presence of the word "puck" can be considered as a strong indicator of hockey. We refer to this type of input as a *labeled feature*.

Unlike previous approaches that use labeled features to create labeled pseudo-instances, [5] uses labeled features directly to constrain the model's predictions on unlabeled instances. They express these soft constraints using generalized expectation (GE) criteria – terms in a parameter estimation objective function that express preferences on values of a model expectation.

We apply the idea of labeled features to metro maps. We aim at creating two classes of documents, roughly meant to represent 'interesting' and 'non-interesting'. Initially, we have no labels; we compute a metro map (as discussed in previous sections) and display it to the user. In addition, we show the user a *tag cloud*. A tag cloud is a visual depiction of words, where size represents frequency (see top-right of the website figure in Section 4). The cloud describes the documents of the map. We let users adjust word importance. For example, importance of 0.9 implies that 90% of the documents in which the word appears are interesting to the user. The relative transparency of the model allows users to make sense of feature weights.

When the user is done adjusting word importance, we train a MaxEnt classifier on $\mathcal{D}$ using those constraints. The classifier then assigns a score μ_i to each document $d_i \in \mathcal{D}$. μ_i represents the probability that d_i is interesting to the user. Given μ_i, we define a personalized notion of coverage:

$$\textit{per-cover}_i(j) = \mu_i \cdot \textit{cover}_i(j)$$

Weighting the coverage by μ_i causes non-interesting articles to contribute very little coverage. Note that non-interesting articles may still be picked for the map, e.g. if they are a coherent bridge between two areas of high interest.

In order to demonstrate the algorithm, we increase the importance of 'IMF' in the Greek debt map. The new map focused more on the IMF, including articles such as 'I.M.F.

May Require Greece to Cut Budget and Jobs', 'I.M.F. Is Playing the Role of Deal Maker in Europe', 'E.U. Leaders Turn to I.M.F. Amid Financial Crisis'. After *decreasing* the importance of 'Greece', a new line appeared, focusing on the *Spanish* economic struggle. Representative articles include 'Spain Seen as Moving Too Slowly on Financial Reforms' and 'I.M.F. Gives Backing to Spain's Austerity Measures'.

7. RELATED WORK

To the best of our knowledge, the problem of constructing metro maps automatically is novel. There has been extensive work done on related topics from topic detection and tracking to summarization and temporal text mining.

Our work differs from previous work in two important aspects. Our system has **structured output**: Not only does our system pick nuggets of information, it explicitly shows connections among them. Prior work, in contrast, has been limited largely to list-output models. In the summarization task [14, 2, 15], the goal is often to summarize a corpus of texts by extracting a list of sentences. Other methods [10, 20, 19] discover new events, but do not attempt to string them together.

Numerous prior efforts have moved beyond list-output, and proposed different notions of storylines [1, 17, 18, 2]. Graph representations are common across a variety of related problems [9, 7, 12] , from topic evolution to news analysis. However, in all of those methods, there is no notion of **path-coherence**. In other words, 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.

Finally, different notions of coherence and coverage have been proposed in the literature. For example, enhancing coverage has been explored in the context of ranking and summarization (see MMR [21]). We chose not to use MMR as it does not provide approximation guarantees, and could not be combined with our orienteering algorithm. Modeling *coherence* via lexical relations was studied in [3]. However, their notion is restricted to chains of related words (Machine, Microprocessor, Device). In contrast, we generate coherent chains of *articles* by taking multiple concepts into account.

8. CONCLUSIONS AND FUTURE WORK

We have presented a new task, creating structured summaries of information, which we call *metro maps*. Given a query, our algorithm generates a concise structured set of documents which maximizes coverage of salient pieces of information. Most importantly, metro maps explicitly show the relations between the retrieved pieces.

We formalized the characteristics of good metro maps and provided efficient methods with theoretical guarantees. Our approach finds concise maps, making it well-suited for complement existing visualization and user interaction approaches. In particular, we integrate user preferences into our framework by providing an appropriate user-interaction model based on feature-based feedback.

We conducted pilot user studies, testing our algorithm on a real-world dataset. The study showed the promised of the proposed approach as an effective and fast method for creating valuable metro maps.

In the future, we plan to pursue richer forms of input, output and interaction, and the incorporation of higher-level semantic relations into our framework. In addition, we would like to apply our methods to other datasets, such as scientific publications. We believe that metro maps will enable users to better cope with information overload.

Acknowledgments: This work was partially supported by ONR PECASE N000141010672, ARO MURI W911NF0810242, and NSF Career IIS-0644225. Dafna Shahaf was supported in part by Microsoft Research Graduate Fellowship.

9. REFERENCES

[1] A. Ahmed, Q. Ho, J. Eisenstein, E. Xing, A. J. Smola, and C. H. Teo. Unified analysis of streaming news. In *WWW'11*, 2011.

[2] J. Allan, R. Gupta, and V. Khandelwal. Temporal summaries of new topics. In *SIGIR '01*, 2001.

[3] R. Barzilay and M. Elhadad. Using lexical chains for text summarization. In *ACL Workshop on Intelligent Scalable Text Summarization*, 1997.

[4] C. Chekuri and M. Pal. A recursive greedy algorithm for walks in directed graphs. In *FOCS '05*, 2005.

[5] G. Druck, G. Mann, and A. McCallum. Learning from labeled features using generalized expectation criteria. In *SIGIR '08*, pages 595–602. ACM, 2008.

[6] K. El-Arini, G. Veda, D. Shahaf, and C. Guestrin. Turning down the noise in the blogosphere. In *KDD '09*, 2009.

[7] C. Faloutsos, K. S. McCurley, and A. Tomkins. Fast discovery of connection subgraphs. In *KDD '04*, 2004.

[8] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24:381–395, June 1981.

[9] Y. Jo, J. E. Hopcroft, and C. Lagoze. The web of topics: discovering the topology of topic evolution in a corpus. In *WWW '11*, 2011.

[10] J. Kleinberg. Bursty and hierarchical structure in streams, 2002.

[11] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *KDD*, 2007.

[12] R. Nallapati, A. Feng, F. Peng, and J. Allan. Event threading within news topics. In *CIKM '04*, 2004.

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

[14] A. Nenkova and K. McKeown. A survey of text summarization techniques. In C. C. Aggarwal and C. Zhai, editors, *Mining Text Data*. 2012.

[15] D. Radev, J. Otterbacher, A. Winkel, and S. Blair-Goldensohn. Newsinessence: summarizing online news topics. *Commun. ACM*, 48:95–98, October 2005.

[16] D. Shahaf and C. Guestrin. Connecting the dots between news articles. In *KDD '10*, pages 623–632, New York, NY, USA, 2010. ACM.

[17] R. Swan and D. Jensen. TimeMines: Constructing Timelines with Statistical Models of Word Usage. In *KDD' 00*, 2000.

[18] R. Yan, X. Wan, J. Otterbacher, L. Kong, X. Li, and Y. Zhang. Evolutionary timeline summarization: a balanced optimization framework via iterative substitution. In *SIGIR' 11*, 2011.

[19] Y. Yang, T. Ault, T. Pierce, and C. Lattimer. Improving text categorization methods for event tracking. In *SIGIR '00*, 2000.

[20] Y. Yang, J. Carbonell, R. Brown, T. Pierce, B. Archibald, and X. Liu. Learning approaches for detecting and tracking news events. *IEEE Intelligent Systems*, 14(4), 1999.

[21] C. X. Zhai, W. W. Cohen, and J. Lafferty. Beyond independent relevance: methods and evaluation metrics for subtopic retrieval. In *SIGIR '03*. ACM, 2003.