TreeQueST: A Treemap-based Query Sandbox for Microdocument Retrieval

Dennis Thom and Thomas Ertl
Institute for Visualization and Interactive Systems
University of Stuttgart
Email: firstname.lastname@vis.uni-stuttgart.de

Abstract—Scatter/Gather-browsing has been proposed as a technique for information retrieval that fosters understanding of textual data and identification of key documents by means of exploration and drill-down. It has been found that such approaches are more expensive but not more effective than less interactive search solutions for traditional retrieval tasks. In this paper, however, we show that the rise of online microdocument platforms, such as Twitter, has brought new relevance to the technique for finding and understanding information about recent events. Our novel approach builds on hierarchical topic clustering combined with a treemap-based visualization to provide a highly interactive information management and query sandboxing space. Large volumes of data, only accessible through rate- and throughput-limited channels, can thus effectively be filtered and retrieved using iteratively optimized queries. We conducted a user study that demonstrates the performance of our approach compared to plain text search based on the Twitter engine.

Keywords—Visual Analytics; Twitter; Hierarchical Topics;

I. INTRODUCTION

The rise of microblogging services, such as Twitter, Tumblr or Weibo, established a new kind of information sharing behavior within the web. While traditional web documents, such as websites, news articles or blogposts, are often self-contained and provide comprehensive and broad information on a certain topic, the information shared in a single Twitter message is usually limited, very specific, and highly context dependent. It can nevertheless be of high relevance, as the communities discuss recent events and topics, and they often share inside information and firsthand accounts long before the public news media reacts. Recent research has already demonstrated that such data can have huge impacts in domains ranging from crisis response [1] to stock market analysis [2].

However, comprehensive utilization of these services for situation awareness and media intelligence poses new challenges in information retrieval. Textual search engines like Google or Bing are powerful tools for retrieval of traditional web documents, and the relevant information for a given query is often contained within the 10 or 20 top ranked results. The primary topic of websites can easily be identified based on word frequencies, and hyperlink-networks can then be used to assess their importance and centrality for a topic. By contrast, search engines for microblog entries (also more generally called microdocuments), such as Twitter Search\(^1\), often fail to deliver the information that was actually requested, which has several causes:

- Complete information about certain topics is often not contained in a small set of comprehensive documents, but distributed over hundreds of messages.
- The decision to reference (e.g., retweet) other messages is most often based on the popularity of the author and not on the relevance of the message. Ranked results based on link-centrality (e.g., retweet count) can thus fail to retrieve relevant content from unknown users.
- The low effort to share information leads to highly redundant message contents for popular topics. Top-ranked results can thus be dominated by sub-topics that are not relevant to the analyst.
- The shortness of the texts results in a lower chance that given query-words are contained in a relevant message. The analyst thus has to provide a more comprehensive list of keywords to cover all possibly relevant entries.

In addition to these issues, research on web analytics also suffers from the more general problem of rate-limited service APIs, which hinder comprehensive monitoring of events and topics with high message volumes. Although elevated access rights can be bought from the services, even many companies and government institutions lack resources to make arrangements with all possibly needed platforms.

In this work we propose a novel technique to tackle these challenges based on aggregated message collections and explorative query optimization. Our primary contribution is the design of an analysis loop for microdocument retrieval that employs hierarchical clustering and a highly interactive treemap representation of summarized clusters. In an iterative process analysts are helped to continuously improve their understanding of the inherent topic structure of a retrieval set, find relevant discussions and events, and build a textual query that perfectly fits their specific information needs. Our method bears strong similarities to Scatter/Gather [3], a document browsing technique that organizes unknown corpora based on cluster analysis and shows descriptive textual summaries to the user. Based on these summaries, groups of interesting topics are selected, which then serve as new seed set for the clustering. In past

\(^1\)https://twitter.com/search-home
research it has been found that the method is slightly more efficient but not more effective compared to keyword-based search engines. Although it was demonstrated that a user’s overall understanding of an unknown document collection is increased, the return of investment for traditional web search has been too low to popularize the method outside of research.

We think, however, that the basic principle of Scatter/Gather holds new potential if it is applied to the realm of microdocument retrieval based on our technique. To evaluate our method, it has been implemented in a prototypical system called TreeQueST (A Treemap-based Query Sandboxing Tool), which we will briefly introduce in the following section. In Section III we review related research in the areas of microdocument and social media analytics, Scatter/Gather-browsing, and visual information spaces. The technical details of our clustering scheme, which is based on the notion of tweet similarity, will be given in Section IV. There we will also describe how the resulting topic hierarchy is visualized for interactive exploration. The basic Scatter/Gather scheme, which is already implemented by that visualization, is then extended by a method for automated query creation based on user selected topics. The algorithm for this method will be introduced in Section V. We conclude with an evaluation of TreeQueST based on a user study in Section VI and final remarks in Section VII.

II. THE TREEQUEST ANALYSIS APPROACH

The analysis workflow of traditional Scatter/Gather approaches is quite straightforward. First, the available documents are grouped into disjoint subsets using some cluster analysis method like K-Means or DBSCAN. The users are then presented with an aggregated representation, also called a ‘digest’, for each of these groups. They select one or more of these digests that seem relevant (gather), and the process is repeated with the union of the selected document groups as new initial set (scatter). This is done until the users reach a good understanding of the available data and/or find the documents they are interested in. TreeQueST basically follows that same Scatter/Gather-loop, but adds another step, namely the query, between the gather and the scatter phase. In the following subsections we will introduce the TreeQueST UI and explain how the basic Scatter/Gather scheme is reflected in the analysis. We will then describe how the query step is integrated in that process. However, in the following subsection we will first give a brief introduction to Twitter, which is used as exemplary data source throughout this work.

A. Twitter Data

Since its foundation in 2006 Twitter has gathered over 241 Million users that produce about 500 Million tweets per day, making it a perfect source to gather public opinions, thoughts and observations. Twitter offers two web application programer interfaces (API), called the ‘REST-API’ and the ‘Streaming-API’, which are, similar to most other services, both subject to significant access-rate-limitations. The REST-API allows requests based on keyword queries as well as metadata filters, and provides results up to a maximum age of seven days. Once reaching that age, tweets are removed from the search index. During important events tweets can be generated at a rate of more than 140,000 messages per second\(^2\). However, due to the rate-limitation, the REST-API only allows the collection of approximately 17,000 tweets within each 15 minute time-slot. The Streaming-API accepts similar request parameters and continuously streams data for the requests at a higher rate. However, it can not be used to retrieve past data, and each Twitter account may only create one standing

---

\(^2\)https://blog.twitter.com/2013/new-tweets-per-second-record-and-how
connection based on his fixed request parameters. Therefore, analysts will often be working with a subset of the actual document corpus, and they have to create smart queries in order to drill-down on this corpus without actually having it locally available.

B. User Interface and Basic Analysis Phase

The basic UI of TreeQueST can be seen in Figure 1. The analysis process begins by entering a query into the search box (1.A). This query can contain boolean operators like ‘OR’, ‘AND’ and ‘NOT’, and since we are using Twitter as the data source, one can also use ‘#’ and ‘@’ to specifically indicate hashtags and usermentions. The seed query used in Figure 1 was ‘ukraine’ and it was performed on June 14, 2014. By clicking ‘Send Query’ (1.B) a sample of up to 1000 recent tweets ranked by relevance is collected from the Twitter REST-API. The incoming tweets are preprocessed and hierarchically organized in a binary tree using our method for agglomerative clustering and a newly defined tweet similarity measure, which will be further described in Section IV-B. Altogether this process takes about six seconds. In contrast to the traditional scatter gather approach, we now present users with a visual representation of this hierarchy (1.C), which serves as an information management space and the primary playground for all further interactions. Section IV-C will provide further details on how this visualization is created and structured.

Individual tweets within the hierarchy are shown based on a spatialization that places similar tweets more closely to each other. In addition, the contents of each area, also called a topic, are represented as a weighted tag cloud of terms and/or hashtags that are most specifically relevant to the tweets in this area. The hierarchy can now freely be explored by changing the granularity of topics with the mouse wheel, selecting nodes to show individual tweets in a table (1.D), and zooming into the hierarchy to drill-down into topics that are considered interesting (1.E). During this process, users are supposed to get familiar with the data and gather relevant topic areas, which is done by right-clicking on them. The navigation is supported by a minimap of the complete tree as shown in Figure 1.F. It illustrates which node has been selected (blue area), what viewport is currently zoomed-in (green frame), and which topics have been gathered (green areas). Once a relevant subset of topics is gathered, users can go on with the query step, which will be explained in the next subsection.

C. Extension by Query Creation and Evaluation

As the users are usually just working on a sample of the complete Twitter data, accessible only through the rate-limited service APIs, the query creation is an integral part of TreeQueST. The system supports users in building an optimized query (1.G) in order to perform the scatter step on this ‘hidden’ document corpus. Section V will describe in more detail how a query is algorithmically built based on the gathered topics. Once a query is generated or manually entered into the search bar, users can immediately evaluate its effects by hitting the enter key before actually executing the query via the API. Tweets in the sample set that are covered by that query will then be highlighted in blue, and all other tweets will be shown in gray. This can be seen for the test query ‘putin’ that has been entered in Figure 1. Furthermore, topic tags that are highly cooccurrent to the current query are also highlighted (orange tags in Figure 1). Analysts can use the highlights to get a feeling for the portion and content of tweets that would be retrieved from the Twitter corpus, and investigate how well the individual topics in the hierarchy would be covered. They can then immediately modify the query by deleting or adding keywords and explore the results. After executing the query, the results of the API-request are combined with the tweets from the gathered topics in a new sample set, which will then again be clustered and visualized. The complete analysis loop of TreeQueST is illustrated in Figure 2. The query step can also be bypassed by the user if ‘Gather Query’ is clicked with an empty search bar. Similar to traditional Scatter/Gather, only the gathered topic-tweets will then be re-clustered without incorporating new retrieval results.

D. Scatterblogs Integration

Ultimately, the outcome of the analysis process is three-fold: First of all, by working with the document hierarchy, users will get a better understanding of the discussed topics, their interconnections, and individual document contents. Second, based on their research, they have evolved an optimized query that better fits their interest areas in terms of precision and recall. Third, during the process, they have retrieved a range of relevant documents and topics that establish an information space organizing their domain of interest.

The optimized query can now be used to continuously collect relevant documents in real-time using the REST- and/or Streaming-APIs of the service. The incoming microdocuments can then automatically be represented in the information space by recursively integrating them in the hierarchy using the similarity function. To this end, we have integrated TreeQueST with the Scatterblogs VA workbench for social media analysis [4], which can be seen in Figure 3. Tweets that have been retrieved based on the TreeQueST query are represented in the different views of Scatterblogs, including the map view and timeline. The temporal and spatial filter options can be used to restrict the sample collected by TreeQueST, and the system offers means to locate tweets even without provided geo-location. This also helps to assess geographical distribution of topics and queries.
Figure 3. The UI of TreeQueST has been integrated into the Scatterblogs framework, which enhances the analysis by showing the geographical and temporal distribution of messages in a map and timeline and allows the collection of streaming data.

III. RELATED WORK

The approach presented in this work can be located at the crossroads of Scatter/Gather-browsing, the use of visual information spaces, and recent research on social media and microdocument analysis in natural language processing (NLP), information retrieval, and visual analytics. In this section we will first address microdocument analysis, and examine how our approach compares to existing ideas. In the second subsection we review the Scatter/Gather-idea of Cutting et al. and related research. We conclude with an overview of hierarchical information structures based on space-filling visualizations.

A. Microdocument Analysis

Research on microdocument and microblogging analysis has seen exponential growth in the last couple of years. Besides visualization research, which will be addressed further below, the topic has received much attention in NLP and machine learning, where statistical methods for document retrieval, classification, and topic modeling for short and highly context dependent snippets have been developed. Sakaki et al. [5] apply Kalman and particle filters to detect the location of natural disasters based on Twitter messages, Weng et al. [6] use LDA-based topic modeling to find important information related to ongoing events and Schulz et al. [7] use a range of supervised classifiers and a highly sophisticated feature extraction pipeline to detect and rank incident related tweets. Statistical models, however, can only be applied to data that is locally accessible to the analyst. All of these approaches thus rely on a pre-extracted corpus or use pre-defined queries to collect streaming data. If relevant data is not covered by the corpus or the query in the first place, it will also bypass the model. In our work, we try to overcome the issue of rate-limited channels and provide a combination of model-based content aggregation and exploration together with query optimization in order to maximize recall.

The works in the information visualization and visual analytics domain can broadly be categorized in four application areas: Information diffusion analysis, sentiment and opinion mining, debate and news media intelligence, and situational awareness. Recent approaches primarily use topic modeling, classification, and entity recognition to find, filter, categorize and aggregate relevant microdocuments in interactive visualizations. For example, Abel et al. [8] present a web-based system that builds on faceted search and semantic entity extraction to retrieve relevant messages connected to emergency communication. Twitinfo from Marcus et al. [9] applies a custom peak detection scheme to identify and visually label unusual developments in Twitter debates. Similar to our system, both of the latter approaches use rate-limited requests to retrieve data for a user-defined seed query. However, they provide no means to manipulate or optimize that query based on aggregated data exploration.

HierarchicalTopics [10] from Dou et al. is most similar to our approach in terms of hierarchical topic exploration. It uses an algorithm, called Topic Rose Tree, to create a hierarchy of LDA-extracted topics. An interactive visualization represents the topics and their temporal evolution as part of this hierarchy, and the user can manipulate it based on her mental model. In contrast to their approach, means to gradually explore the topic hierarchy based on their granularity to overcome shortcomings of automated cluster analysis is the primary focus of our visualization design. Furthermore, as an extension to the Scatter/Gather-idea, the visual topic hierarchy serves as a basis to select topics, generate optimized queries and filter the document set.

B. Scatter/Gather-browsing

Scatter/Gather, as a method for information retrieval, has first been popularized by Cutting et al. in their work 'Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections' [3]. Their method relies on partitional clustering, and it builds cluster representations based on their most frequent words and representative documents. As computing power was limited at that time, dynamic hierarchical clustering solutions, which are used in our method, were not an option in their research. However, in 1993 [11] they presented a method to pre-compute the hierarchy and use this as a basis to achieve constant interaction-time. The visualizations of these early approaches were usually plain-text lists of words and document titles or a windowed document browser as shown by Pirolli et al. [12]. In that work Pirolli et al. also present results of a user study, which demonstrated that Scatter/Gather helps participants to build a mental image of a given text collection, formulate richer keyword-queries, and get a feeling for the portion of relevant documents in the corpus. However, their study also showed that the method is not superior in effectiveness compared to plain keyword-search if the goal is just to locate specific documents. In our approach we revisit that discussion, and formulate the hypothesis that the latter is only true for
traditional text-documents, like articles, books, or websites. We suggest that, in the realm of microdocument retrieval, advanced visual aggregations and techniques for query-based exploration and drill-down can be of great help, both to better understand the available information and to identify key documents and topics.

More recent results from Ke et al. [13] proposed a novel partitional clustering algorithm as basis for Scatter/Gather, called LAIR2. They gave the users more control to interactively change clustering parameters, and they performed a user study with 24 participants. However, in contrast to our approach, neither short texts nor a visual representation of the topic hierarchy or query optimization were a focus of their work.

C. Visual Information Spaces

Space-filling tree-visualizations have been popular since Johnson and Shneiderman introduced the concept of treemaps in [14]. Our approach also uses the simple 'slice and dice' layout that was proposed in that work. Since then, there have been several new ideas and advances over that original concept, particularly to optimize aspect ratio, node containment and ordering (e.g. [15], [16]). However, similar to Blanch et al. [17], we think that interactive exploration plays the most important role in making large and complex data-sets accessible through this kind of visualization. In their work they thoroughly examine the notion of zoomable treemaps based on three different interactions: 'In Depth Navigation', 'In Breadth Navigation' and 'Direct Node Selection'. They allow the user to drill-down along a branch, move from a node to its neighbors on the same level, and select arbitrary nodes using gestures. Our system provides similar means for zooming. In addition, the user can manually change the display level of nodes using the mouse wheel to select, compare, and navigate between them. Furthermore, our system generates summarizations from the topic hierarchy and provides a minimap for orientation. Another approach has been proposed by Graham Wills [18], who also examined the use of interactive treemaps to view hierarchical clustering results. In this work he also introduces a means to represent individual data items in a spatialization generated directly from the hierarchical layout. However, exploration by zooming is not possible in his system and the complete hierarchy is always shown, which makes it difficult to navigate in deeper levels.

To the best of our knowledge, there are few approaches that address the use of hierarchical clustering and space-filling visualizations for information retrieval. In this regards the FISPA-system from Turetken et al. [19] can be considered most similar to our approach. They use a treemap visualization based on hierarchical clustering to organize web retrieval results. In contrast to our approach, they do not address microdocument retrieval and provide no means for sophisticated topic representation, change of granularity or query exploration. Traditional web searches, as already indicated, often deliver all relevant results within the top-ranked documents. Therefore, although a slight improvement in efficiency was achieved, the participants of their user study were not more effective in finding relevant results.

IV. HIERARCHICAL INFORMATION SPACE

In this section we describe in more detail how tweets are organized and represented in the hierarchical information space of TreeQueST. We use agglomerative clustering, a well-known cluster analysis technique, to build groups of related elements. As the technique heavily relies on a good similarity metric, we describe how the closeness of two tweets is characterized based on their textual content and extracted annotations in Subsection IV-B. We furthermore explain how the resulting cluster hierarchy is used to visually represent the dataset as hierarchical topics, and how the process is integrated with the interactive workflow in Subsections IV-C and IV-D.

A. Agglomerative Clustering

Agglomerative clustering is a popular method of distributing elements into similarity groups. Starting with the complete set of elements - in our case the result set from a Twitter-API request - the algorithm initially considers each element as an individual cluster. Based on a predefined distance function between elements, the method then repeatedly finds the two clusters with minimum distance and merges them into a larger cluster. For clusters containing more than one element, this distance is derived from the pairwise element distances using one of several existing methods. We use the average of pairwise distances in our implementation. The clustering terminates when all elements have been merged into a single, large cluster. During the process, the algorithm builds a binary tree of all merges that have been performed, which is basically the final output of the method. This tree then reflects the similarity structure of the set, such that the sub-trees of nodes near the root usually have a high pairwise distance that becomes smaller when moving towards the leaves. In our system we use a slightly modified Weka [20] instance to perform the clustering, as it provides a very fast implementation based on priority queues. Although agglomerative clustering is a quite simple method, the actual challenge is to define and implement a metric that well estimates the similarity distance of two tweets.

B. Tweet Similarity

The most popular measures for document similarity are several forms of edit distance, which work well on shorter strings with few variations, and cosine similarity, which is well suited for larger documents that are considered similar if they address the same subject matter. Although tweets are very short documents, they are often considered to be
of similar content if they share a certain amount of low-frequent words. We thus adhere to cosine similarity, and build several feature vectors to form our distance measure. In addition to plain-text elements, tweets and other sorts of microdocuments provide a range of meta-features that can be used to assess their similarity. In our data processing phase we thus extract the following feature vectors from a tweet:

- **Hashtags - F₁** - The Twitter community has established '#' as a prefix for unique names that indicate a common topic. This has been widely accepted and recognized throughout the service. Some time ago, Twitter began to automatically detect and link such annotations and to make them accessible through the API. As a hashtag is usually a clear statement of the author, telling us that the tweet is related to some well-known topic, it is a perfect indicator of its subject matter.

- **URLs - F₂** - A large portion of tweets contains a URL as an invitation to its reader to visit the corresponding website or to make a comment about it’s content. Mentioned URLs are thus another powerful feature to determine the topic of a tweet.

- **Usernames - F₃** - Similar to hashtags, the '@' sign is used as a prefix to highlight usernames in tweets, either to indicate the recipient of the message, when used at the beginning, or as a reference to that user, when used in the middle. As users can often be assigned to the discussion around a specific topic, we use the usernames as well as the username of a tweet’s author as additional features.

- **Terms - F₄** - After removing hashtags, usernames, URLs and punctuation, the rest of the textual content of the tweet is tokenized, lower-cased, and lemmatization and stemming are performed. The resulting tokens are then also used as features.

Low-frequent words, such as 'ukraine' or 'putin', are often of higher specific relevance to a document’s subject matter than high-frequent words, such as 'the', 'and', or 'word'. It is therefore common to use the inverse-document frequency, defined as

\[ \text{idf}(w) = \log \frac{|T|}{|T : w \in t|} \]

with \( T \) being the set of documents, to assess how 'unusual' a word \( w \) is. The idf is usually applied in conjunction with the term-frequency (tf) of a word within the given document. However, because of the shortness of Twitter messages, words rarely appear more than once in a given message. For sake of simplicity we therefore always assume that the tf for a given word and message is either 0 or 1. Using a comprehensive set of more than 3 billion tweets, which were collected from the Streaming-API in the course of the last three years, we were able to compute a sophisticated dictionary of idf values for all hashtags, URLs, usernames and terms that have been used at least 10 times during that time. As Twitter is constantly evolving, this dictionary is continuously updated on a daily basis. On this basis we build our feature vectors \( F_1(t), \ldots, F_4(t) \) of a given tweet \( t \), such that an element will be either 0 if \( t \) lacks a certain hashtag/URL/username/term, and the idf value of that hashtag/URL/username/term otherwise. Based on these feature vectors, we define our distance as follows:

\[ d(t_1, t_2) = 1 - \sum_{i=1}^{4} w_i \cdot \frac{F_i(t_1) \cdot F_i(t_2)}{\|F_i(t_1)\| \cdot \|F_i(t_2)\|} \]

with \( w_i \) being a weight assigned to each feature vector according to its power to indicate the subject matter. Based on our experiments we use \( w_1 = 0.3, w_2 = 0.3, w_3 = 0.2, \) and \( w_4 = 0.2 \).

### C. Visual Tree Representation

Traditionally, the output of agglomerative clustering is often used to produce a partitioning of the element-set into disjoint groups, comparable to the result of K-Means or similar partitional clustering approaches. To this end, a fixed distance threshold is chosen, and, starting from the root, the tree is traversed depth-first. Every time a node with siblings of smaller distance than the threshold is encountered, all elements of the sub-tree are collected into one of the final clusters. This, however, can lead to clusters that heavily vary in size and numbers and poorly reflect actual similarity groups. By giving users interactive means to explore the resulting hierarchy, starting from clusters of high diversity, and moving to clusters of high similarity, we want to turn that drawback into a helpful tool of identifying actual topics within the structure.

We therefore display the cluster tree based on a binary space partitioning. Starting with the root node, the visual space is recursively shared between the siblings by splitting it alternating in a vertical and horizontal fashion. A recursion path is terminated when it either reaches a leaf-node, or when the similarity distance of a node’s siblings is below an interactively adjustable threshold. We then call such nodes 'display-nodes'. The next subsection will explain in more detail how they are rendered. The distance threshold is initially determined by finding the first steep slope of distances using a breadth-first search of the tree and the z-score of encountered distances. It can subsequently be modified by users using the mouse wheel.

As the tree is usually re-rendered in a few milliseconds, we can achieve fluid interactions allowing users to intuitively change the granularity of the represented information space. Furthermore, they can select represented nodes with the mouse and zoom into that area. In this case the selected node is used as the new root node for the visualization.

### D. Tag Clouds, Spatial Layout and Exploration Lens

Once the above described splitting has terminated at some leaf- or display-node, the tweets in its sub-tree are visualized...
in two layers within the space that has been assigned. For the top layer we generate a weighted tag cloud based on either the terms, hashtags, or usermentions that are most prominently used within the tweets of the sub-tree. By this means, users can get a quick indication of potential topics represented in the display-node. The tags are weighted by our global idf-dictionary and a smaller idf-dictionary that is specifically generated for the cluster hierarchy. That way, we can show the tags that are most specifically relevant to the display-node. The tag clouds are rendered based on our own Wordle-inspired [21] implementation. We chose to use a global scale to determine the size of a tag based on the square root of its weight. The tag relevance is then comparable between topics. In addition, the occurrence of larger tags gives users a good indication whether a display-node already constitutes a coherent topic, or whether they should further explore its siblings.

In the second layer we generate a spatialization of the tweets that groups more similar elements within each others neighborhood. This allows users to assess the portion and density of tweets within a display-node and also helps to understand the distribution of tweets in the query exploration, as will be explained in the following section. The existing clustering hierarchy provides a simple means to spatialize the tweets of a sub-tree according to their similarity. Starting from its leaf-nodes, we generate their spatial layout by placing their single tweet somewhere within a $[0,1] \times [0,1]$ coordinate space. We then move upwards in the tree and merge the layouts of siblings of a node by the ratio of the number of tweets they contain. It can be determined based on a nodes tree depth whether the merge has to be done in a horizontal or vertical fashion. The process is illustrated in Figure 4. By this means, the similarity information is preserved on any display-level, and closely related tweets will also be spatially close to each other.

We furthermore utilize this layout to provide an additional content exploration tool, called a content lens; an idea that has already been featured in our previous works [4]. A circular or square-shaped ‘lens’ can be moved over the data using the mouse cursor. The most prominent words and/or hashtags within the lens are then displayed as a tag cloud around it. Users can change the size of the lens and choose whether the term relevance should be ranked based on idf-weighted or absolute frequency. Applying the lens, users can quickly asses the range of contents within a node before they decide to perform a drill-down. Content exploration is furthermore supported by an optional sentiment highlighting based on the SentiStrength [22] library, which can be activated for the spatialized tweets. Users can thus find and explore areas of positive or negative affection towards or related to a topic to inform their reasoning. We chose to color the individual tweets according to the respective sentiment instead of the whole cell based on the aggregate sentiment. Outliers are thus more visibly highlighted from the dominant sentiment in a cell, and, because of the similarity spatialization, users can easily detect coherent sentiment structures in deeper levels of the hierarchy while they still investigate upper levels.

V. QUERY EXPLORATION

Within the TreeQueST analysis process, the system generates queries that try to cover a large number of tweets contained in the gathered topics and as few tweets as possible from all non-gathered topics. In this section we will describe how this query is algorithmically created. It is then inserted into the search bar to be executed through the Twitter-API and allow users to retrieve more tweets for the topics they consider interesting. As the resulting query will not always achieve the desired results, we also provide users with means to manually manipulate it and explore the effects on the current sample set.

A. Algorithmic Query Construction

Suppose we have a set of tweets $T_g$, called the wanted results, in our case the tweets from the user-gathered topics, and another set $T_b$, called the unwanted results. Furthermore, we have a set of all regular words, i.e. terms and hashtags, that are contained in tweets of the union $T_g \cup T_b$. In a naive approach we could just try to build a boolean query, e.g. as a disjunction of conjunctive clauses of negated and non-negated words, that covers exactly all tweets from $T_g$ and none of the tweets from $T_b$. We could then use a method for boolean function minimization, like the Quine-McCluskey algorithm, in order to find a shortest query that imposes a less restrictive behavior when applied to a larger corpus and does not exceed limits for request length of the service API.

This solution, however, suffers from a range of problems. First, computing a minimized form of a boolean query is considered an intractable problem. Since we have a very large set of variables, even fast algorithms like the espresso logic minimizer would render the integration of query generation as part of a fluent interaction scheme almost impossible. Second, depending on the tweets, the optimal solution would often still be quite large, as there might be outliers that have to be covered with indispensable query clauses. Third, in some cases there might be no existing solution at all, e.g. if $T_g \cap T_b \neq \emptyset$.

We therefore implemented a heuristic solution, which might accept some unwanted results, and also miss some
wanted results, in order to allow fast computation and powerful queries. Furthermore, since our set of wanted results \( T_f \) is partitioned into subsets \( L_1 \cup \ldots \cup L_m = T_g \), called the *topics*, which might be of individual relevance to the user, we introduce the additional constraint that our query should also cover as many topics as possible.

We assume to have a global idf-dictionary \( idf_{global} \), as it has been introduced in Section IV-B. Let \( W(T) \) be the set of all terms and hashtags that are used in a given tweet set \( T \), and let \( \|T\|_w \) be the number of tweets in \( T \) that contain term or hashtag \( w \). Furthermore, let \( q \) be our initially empty query, i.e. the one that was used to initially load the sample data-set. Our algorithm then works as follows:

1) Find the word \( w \in W(\bigcup_{i=1}^m L_i) \) that maximizes the weight
\[
idf_{global}(w) \cdot \frac{\|T_g\|_w}{1 + \|T_g\|_w} + \sum_{i=1}^m \|L_i\|_w
\]

2) If \( idf_{global}(w) \geq idf_{global}(q_0) \), set \( q := q OR w \), else, set \( q := q OR (q_0 AND w) \).

3) Remove all tweets from \( L_1 \) \ldots \( L_m \) that contain \( w \).

4) If \( \bigcup_{i=1}^m L_i \) \( \neq 0 \), continue at 1., else, terminate.

The final result will be a query that reasonably generalizes to unknown tweets, prefers relevant topic tweets over non-relevant ones, and usually tends to reduce the set of tweets that are considered within the Twitter corpus. The latter condition mimics the general Scatter/Gather behavior of rather reducing the document set in every iteration. Furthermore, due to step 2, the algorithm tries to add topic-relevant query words to cover tweets missing the initial seed query. If the query is too long for the service API, we cut it between two conjunctive clauses right below the maximum allowed length.

**B. Query Testing and Manipulation**

Sometimes the result of automated query generation still requires refinement, as the algorithm chose keywords that users consider misleading, or the coverage of topics does not reflect their preferences. They can thus manually manipulate the query in the search bar, and the corresponding results in the local sample are immediately highlighted, as mentioned before. They can then further explore the topic hierarchy and try to understand the coverage. The content lens, as introduced in the previous section, can be a valuable tool in this process. Sometimes only certain areas of a display-node are covered by the query, and the lens can be used to investigate their contents before zooming into the area.

When used as part of the Scatterblogs framework, the system provides additional capabilities to explore and integrate query results. Once a query is generated or entered, the corresponding tweets are highlighted on the worldmap and in a timeline. Users can thus understand the geographic and temporal coverage of the query within the sample. Once users have created a well-optimized query, they can furthermore activate the stream collection for that query, and corresponding results will continuously be collected and represented within the framework. The results will also be recursively inserted into the final topic hierarchy based on the previously introduced tweet distance.

**VI. User Study**

In this section we present results of a user study that was conducted during a single day, on September 03, 2014, to evaluate the real-time capabilities and overall performance of TreeQueST. We selected two broad keywords, 'ukraine' and 'barack obama', which were both frequently mentioned media entities during these days. Six study participants, all graduate students in the computer science department, assumed the role of political journalists tasked with investigating these topics using live Twitter data. They were asked to find relevant events connected to the keywords, investigate the communities reactions and opinions, and identify news content that could dominate further media debates. For each participant, we randomly selected one of the two topics to be investigated with TreeQueST. The other topic had to be investigated with a plain text search tool based on the Twitter search engine in order to establish a baseline for comparison. For each tool, the participants were given five minutes to get familiar with it, and then they had ten minutes to investigate the topic. Finally, they were asked to rate both tools in an anonymous online survey.

**A. Findings and Comments**

The participants were encouraged to document all relevant findings, observations, and problems with screenshots and think-aloud comments during their investigation. Major news items surrounding the given topics were usually immediately indicated by TreeQueST right after the seed query was entered, as can be seen in Figure 5. The participants thus easily discovered the execution of journalist Steven Sotloff by IS terrorists, Obama’s public reaction to that, the temporary cease fire between Russia and Ukraine, as well as a
speech given by Obama to reassure Estonia and other Baltic states that defenses in eastern Europe will be reinforced. Several participants commented positively on the indication of entities, such as names, places, and institutions, that could be easily connected to the topics using the tag clouds. Some participants had already learned about some of the major news items from other media before participating in the study. When using the plain text search, they performed equally well on these topics, since they knew in advance, which queries might be successful. However, participants commented on several occasions that they tend to randomly read messages in the top-ranked documents before they try a new query, and that they thus ignore possibly relevant information on the following pages.

With TreeQueST all participants made heavy use of the possibility to drill-down on major topics in order to find sub-topics, to highlight query words, and to crate new queries based on selected topics. By this means, participants discovered additional newsworthy information, e.g. that US stocks on wall street opened higher because of the cease fire, that the announcement of an ongoing truce from Ukraine were initially denied by Russia, which lead to ongoing confusion in the community, and that several Twitter users requested Obama to give back his Nobel peace prize. Participants commented that the plain text search did not support similar discoveries, as it gave no clue in which direction they should broaden their queries. Two participants made frequent use of the possibility to highlight the sentiment of tweets and tags, which, however, did not lead to significant additional findings in these cases.

B. Survey

The online survey was conducted on the same day. It comprised one questionnaire for each tool consisting of Likert scales and written feedback. First, the participants had to rate five subjective statements about the tools on a 5-point Likert scale, ranging from 1 (strongly disagree) to 5 (strongly agree). The statements corresponded to five defining elements of information retrieval performance:

- Overview - I could get a good overview of the topic.
- Precision - I could quickly find relevant news items.
- Recall - I did not feel to miss any relevant information.
- Optimization - I was enabled to optimize my queries.
- Complexity - The tool was easy to use.

The results in Figure 6 show that TreeQueST enabled participants to get a better overview of the topic and the feeling to achieve better retrieval results in terms of precision and recall. They also felt better supported in optimizing their queries based on initial retrieval results. Correspondingly, it was observed during the investigations that participants tried to find further query words in retrieval sets of the plain text search, but commented that additional support would be helpful. The results furthermore show that TreeQueST has a steep learning curve, which reflects our observations that not all features, such as the sentiment highlighting or the exploration lens, were used by all participants, and that some participants mentioned that they already forgot some features from the introduction.

The participants were furthermore asked to provide written feedback based on two questions. First, we wanted to know what they liked better about each tool compared to the other. They liked about TreeQueST that it felt more powerful, provided better overviews, gave them more control over the investigation, and was more enjoyable to use. They also mentioned that the structured overview of TreeQueST would motivate them to read tweets from various sub-topics, which they might have ignored in the plain text search, as they rarely looked beyond the top-ranked documents. About the plain text search they commented that it was more easy to use and to learn. One participant also mentioned that it was a more stateless approach, meaning that he would not lose relevant views and preliminary analysis results when re-iterating the query. In the second question we asked what they would like to change about the tools. Amongst others, they commented that TreeQueST should have a separate text field for query testing, that the query words should be highlighted in the tweets, and that the tag clouds should be improved in terms of scaling and colors.

To conclude the questionnaires, a German school grade, ranging from 1 (best) to 6 (worst), had to be given to rate the overall usefulness of each tool in supporting the task. On average, plain text Twitter search scored 2.8, and TreeQueST scored 1.8, both with a standard error of 1.6.

VII. Conclusion

In this work we have described how highly interactive microdocument exploration and iterative query optimization, based on the Scatter/Gather-idea, can successfully improve situation awareness and media intelligence. We introduced our newly designed notion of tweet similarity to extract the inherent topic structure of a microdocument collection with hierarchical clustering. We furthermore showed how the inevitable shortcomings of automated clustering can be tackled if analysts can explore the cluster hierarchy
through zooming and fluent granularity changes to quickly find meaningful and relevant topics. Using our automated query generation algorithm, they can then trigger an analysis loop that continuously advances their query vocabulary, the relevance of retrieved sample results, and their own understanding of the ongoing event or topic. Finally, we presented the results of our user study, which convincingly demonstrated the usefulness, applicability, and potential of the TreeQueST approach.

ACKNOWLEDGMENT

We thank all participants of our user study and the reviewers for their valuable suggestions and comments.

REFERENCES


