Incremental and Scalable Computation of Dynamic Topography Information Landscapes

Kamran Ali Ahmad Syed[1], Mark Kröll[2], Vedran Sabol[2], Stefan Gindl[3], Arno Scharl[3]
[1]Vienna University of Economics and Business
Research Institute for Computational Methods
Augasse 2-6, 1090 Vienna, Austria
[2]Know-Center GmbH Graz
Division for Knowledge Relationship Discovery
Inffeldgasse 13, 8010 Graz, Austria
[3]MODUL University Vienna
Department of New Media Technology
Am Kahlenberg 1, 1190 Vienna, Austria
kamran.syed@wu.ac.at, {mkroell, vsabol}@know-center.at, {stefan.gindl, arno.scharl}@modul.ac.at

ABSTRACT: *Dynamic topography information landscapes are capable of visualizing longitudinal changes in large document repositories. Resembling tectonic processes in the natural world, dynamic rendering reflects both long-term trends and short-term fluctuations in such repositories. To visualize the rise and decay of topics, the mapping algorithm elevates and lowers related sets of concentric contour lines. Acknowledging the growing number of documents to be processed by state-of-the-art Web intelligence applications, we present a scalable, incremental approach for generating such landscapes. The processing pipeline includes a number of sequential tasks, from crawling, filtering and pre-processing Web content to projecting, labeling and rendering the aggregated information. Processing steps central to incremental processing are found in the projection stage which consists of document clustering, cluster force-directed placement, and fast document positioning. We introduce two different positioning methods and compare them in an incremental setting using two different quality measures. The evaluation is performed on a set of approximately 5000 documents taken from the environmental blog sample of the Media Watch on Climate Change (www.ecoresearch.net/climate), a Web content aggregator about climate change and related environmental issues that serves static versions of the information landscapes presented in this paper as part of a multiple coordinated view representation.*

Keywords: Information Visualization, Incremental Projection Algorithm, Dynamic Information Landscape, Knowledge Evolution

Received: 20 December 2011, Revised 14 February 2012, Accepted 15 February 2012

1. Introduction

We are confronted not only with large, but also with continuously changing "*big data*" repositories. The concept of *information landscapes* is a powerful visual representation based on a geographic map metaphor, which conveys topical relatedness in large document repositories through spatial proximity in the visualization [28]. Hills represent clusters of topically similar documents, which are separated by sparsely populated regions represented as valleys or oceans. The height of a hill indicates the size of the corresponding topical cluster, while its compactness corresponds to the cluster's topical cohesion. Hills (clusters) are labeled with dominant terms and phrases from the underlying documents to provide orientation to the users. Due to their static nature, however, information landscapes cannot handle changes in the underlying datasets.

Dynamic topography information landscapes [40] are visual representations capable of conveying both topical relatedness and dynamic change. When a document repository evolves, new documents are added and old documents are removed, resulting in changes in the overall topical structure. Similarity relations and the position of documents will change – irrelevant topics may disappear altogether, while new topics will appear and increase in importance. The topography of dynamic information landscapes represents changes and topical shifts in the dataset as tectonic processes – rising and shrinking hills indicate the emergence and fading of topics, respectively. Hills moving towards or apart from each other indicate topical convergence or divergence of the corresponding topical clusters. When computing the layout of a dynamic information landscape, it is crucial that dataset changes are incorporated in the topography incrementally, so that regions not affected by changes will preserve their shapes and relative positions. Regions may undergo tectonic changes only if the corresponding parts of the dataset were subject to modification. In this way users will be able to quickly identify changing regions, while retaining orientation through recognition of stable parts of the topography.

Based on large and quickly evolving text repositories collected from public Web sources, this paper presents a scalable method to integrate dataset changes incrementally into existing landscape topographies. The proposed method provides three advantages, which are not simultaneously present in common dimensionality reduction approaches: (1) scalability with dataset size and high-dimensionality, (2) support for incremental computation, and (3) the capability to provide aesthetically appealing layouts, an important consideration when developing visual user interfaces for dynamic Web applications. Our approach relies on a processing pipeline composed of text preprocessing, projection, labeling and rendering stages. The central component of the pipeline is the projection stage, which combines incremental document clustering, incremental force-directed placement of clusters and, most notably, two alternative approaches to fast document positioning. Document positioning is given particular attention for two reasons: It is crucial for the overall performance due to the large number of documents to be processed, and it plays a decisive role for the quality and visual appearance of the layout. The document positioning methods are compared against each other using two different quality measures: a performance-optimized version of the standard stress-measure, and a cluster-based "*Hit Ratio*" method. It should be noted that the performance of the incremental algorithm is not the focus of this paper, as it was evaluated and compared to a non-incremental algorithm version in a previous publication [44].

2. Related Work

Information landscapes are commonly used to visualize topical relatedness in large document repositories, for example in [32], in [28] and in [2]. Static landscape visualizations, however, cannot convey changes. ThemeRiver [22] is a visual representation designed to represent changes in topical clusters, but it cannot express relatedness between documents or topical clusters. Visualization of topical changes through information landscapes with dynamic topologies were proposed in [37], albeit only for small datasets, and later extended in [39]. An approach suitable for larger datasets was demonstrated in [38]. It relies on 3D acceleration for animated morphing of landscape geometry, which makes it unsuitable for Web applications.

Visualization techniques in general have to cope with today's ever-growing data production and data consumption. Incremental algorithms provide the required functionality to process big data. Incremental algorithms do not recalculate their internal model from scratch for newly arriving data items and are thus capable of efficiently handling and seamlessly integrating continuously changing or growing data. In the context of generating dynamic information landscapes we review work on incremental dimensionality reduction and incremental clustering techniques:

2.1 Incremental Dimensionality Reduction

Dimensionality reduction techniques transform high-dimensional data into low-dimensional data seeking to lose as little information as possible. This transformation has turned out to be particularly useful in the field of visualization for projecting the high-dimension data into the lowdimensional visualization space. To face the growing amount of data, incremental variants have been developed usually on top of batch methods. Incremental unsupervised techniques include multi-dimensional scaling (cf. [5]), singular value decomposition (cf. [41], [7] or [20]), principal component analysis (cf. [3], [30] or [47]), random indexing (cf. [26]), locally linear embedding (cf. [27]) or isometric feature mapping (cf. [29]). Unsupervised methods are effective in finding compact representations, but ignore valuable class label information of the training data. Incremental supervised techniques are thus better suited for pattern classification tasks. Representatives of incremental supervised dimensionality reduction techniques include linear discriminant analysis (cf. 33], [50] or [24]), subspace learning (cf. [48]) or maximizing the margin criterion (cf. [49]).

2.2 Incremental Clustering

Incremental clustering algorithms can be traced back to the 1970s, cf. Hartigan's leader algorithm which requires only one pass

through the data [21], Slagle's shortest spanning path algorithm [43] or Fisher's COBWEB system, an incremental conceptual clustering algorithm [16]. The COBWEB system, for example, has been successfully applied to support fault diagnosis or bridge design [17]. Inspired by COBWEB, Gennari et al. proposed the CLASSIT [19] system which is capable of handling numerical datasets. In [8], the authors introduced an incremental clustering algorithm for dynamic information processing. In dynamic databases there is a constant adding or removing of data items over time. The idea is that these changes should be recognized in the generated partition without affecting current clusters. In the late nineties, several incremental clustering algorithms have been presented including BIRCH [51], incremental DBSCAN [15] to support data warehousing or Ribert et al.'s clustering algorithm to generate a hierarchy of clusters [36].

Incremental clustering algorithms are often closely linked to research domains and designed to meet domain-specific requirements, e.g. keeping the cluster diameter small [9] in case of information retrieval. Another application area is the analysis of gene expression data [11] to account for the amount of microarray experiments. Incremental clustering of text documents has been conducted as a part of the Topic Detection and Tracking initiative (cf. [1]) to detect a new event from a stream of news articles.

To compute dynamic topography information landscapes in an incremental and thus timely efficient manner, we integrate and combine incremental aspects into the generation process. (i) For clustering, we apply a simple, spherical k-means [14] and use previously computed partitions of the document set as initial state for incremental computations. (ii) We introduce a novel approach for document positioning which is essentially based on a simple spring forces-based model (cf. [18]) since we observed that landscapes generated with standard positioning method displayed geometrical edges. (iii) We use a force-directed placement (FDP) algorithm for projecting these high-dimensional cluster centroids into a 2D visualization space. The most attractive feature of the FDP algorithm is that it is intrinsically incremental when applied on a previously computed stable layout. Re-applying FDP on previous layout of centroids with modified similarities will produce a new layout closely resembling the previous one.

3. Algorithmic Approach

The overall process is a cascade of the three main components of the computation pipeline shown in Figure 1, which are outlined in the following and described in more detail in subsections 3.1 to 3.3. A separate description of the architecture's incremental aspects is presented in Section 3.4.

(1) The first component prepares a document-term matrix by fusioning the information on each document from the keyword relevance table and the word frequency table.

(2) The second component produces document positions in 3D space. In the first step, the kmeans clustering algorithm partitions the documents into topically related clusters. Afterwards, a force-directed placement algorithm (FDP) projects the centroids' positions of these topical clusters into the 3D space. For an incremental landscape generation the result of previous computations, 1C and 1D in Figure 1, provide the initial state for the clustering algorithm as well as for the centroid layout algorithm in 3D. Finally, the document positions are computed in 3D space.

(3) The third component computes properly labeled landscape images. We begin with transforming document positions from a 3D infinite space onto 2D finite surface. The 2D finite surface could be the surface of a $[1, 1]^2$-plane, the surface of a unit sphere or a surface of a unit cylinder (non-circular parts). The documents' layout position on the 2D finite surface is then used to model a topical landscape, which is essentially an elevation matrix on a 2D grid. A peek detection algorithm identifies important peeks on the topical landscape model along with underlying documents and inquires text descriptors for labeling against the reference corpora, 1E in Figure 1. A predefined coloring scheme shapes the landscape surface and peaks receive labels giving information about the cluster topics.

3.1 Generation of Document-Term Matrix

Based on previous research [25, 42], a crawler-based architecture gathers textual data and stores it in content repositories after analysis, conversion and annotation. After further pre-processing, the *webLyzard* media monitoring and Web intelligence platform (www.weblyzard.com) provides a flexible representation for each document in the form of a keyword relevance table and the document word frequency table (see Figure 1; 1A and 1B).

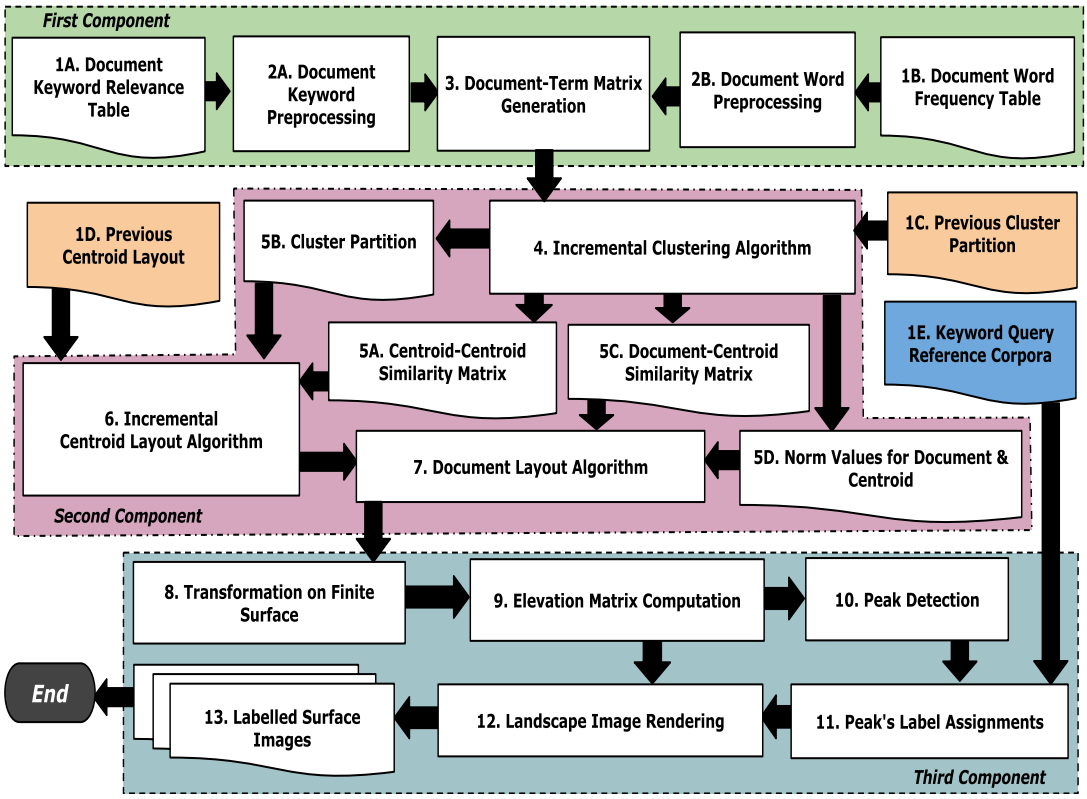

Figure 1. The processing pipeline of the incremental landscape computation framework

Using the information from 1A and 1B for each selected document, the preprocessing for document keywords is done at (2A), as well as, the preprocessing for document words is done at (2B). The preprocessing results are then linearly combined, using a tunable weighting ratio, into one augmented document-term matrix (3) with unique term IDs. We generally used 60% weighting for (2A) and 40% weighting for (2B). Note that both a keyword and a document word are assigned unique term IDs, hence they are different dimensions.

3.2 Clustering and Projection

The incremental clustering algorithm (4) receives the document term matrix (3) as its input and produces four outputs: a centroid-to-centroid similarity matrix (5A), a document-centroid relationship graph (5B), a documents-to-centroids similarity matrix (5C) and norm values for documents and centroids (5D). The previously computed clustering result (1C) may optionally be served to initialize the k-means algorithm module.

The algorithm for centroid positioning (6) gets the main input from (5A) and (5B). Nonincremental clustering initializes the centroid positions randomly, incremental clustering uses the positions from previous calculations for initialization (1D). The centroid positions, the centroid similarity matrix (5C) and the norm of centroids and documents (5D) serve as input for the computation of the document positions (7).

The following sections describe the document clustering, the cluster positioning as well as the document positioning.

3.2.1 Document Clustering

A spherical k-means algorithm partitions the documents into topical clusters [14]. The k-means++ method delivers a thoughtful choice and number of cluster seeds to address the algorithm's sensitivity regarding this aspect [4]. Cluster splitting and merging strategies [31] guess the number of clusters within specified minimum and maximum bounds. The Bayesian Information Criterion [34] improves these guesses for a better cluster cohesion. As human cognition puts limits to understanding complex visual representations, we limited the number of clusters to account for usability. In our experience, setting the limits for minimum and

maximum number of cluster to 30 and 40 resulted in meaningful and aesthetically pleasing information landscapes. For incremental clustering, previously computed partitions of the document set serve as the initial state. The system removes old documents from their respective clusters and adds new documents to the most similar cluster centroid. Subsequently, the k-means clusterer, including the split and merge procedure, further refines the partitions.

The algorithm's runtime complexity is $O(mnd)$ where as the number of clusters is m, the number of documents is n and the dimensionality of the term space is d. Since in our case $m \ll n$, and according to Heaps' law [23] d scales logarithmically with n, the clustering part of our algorithm is considered to scale with $O(nlog(n))$.

3.2.2 Cluster Positioning

The clusterer aggregates the set of documents into topical clusters, which are represented by their high-dimensional centroids. An FDP algorithm [18] performs a distance- preserving projection of the high-dimensional cluster centroids into the low-dimensional visualization space. In FDP attractive forces pull together topically similar centroids while dissimilar centroids are pushed apart from each other. Therefore, spatial closeness between centroids indicates topical closeness.

The main advantage of the FDP algorithm is its capability to produce accurate and aesthetically pleasing layouts. However, a brute-force implementation of FDP scales poorly having a cubic (in this case $O(m^3)$) time complexity. Yet, as in our approach $m \ll n$ and the number of clusters has a constant upper limit, the running time of cluster positioning can be considered constant. Also, as the number of clusters is small, the FDP method terminates very quickly.

The most attractive feature of the FDP algorithm is that it is intrinsically incremental when applied on a previously computed stable layout. As long as the changes in the dataset are comparatively small, the impact of incremental clustering is reflected in small changes of similarities between cluster centroids. When FDP is re-applied on a previously computed cluster layout using the modified similarities, the resulting layout will closely resemble the previous one.

3.2.3 Document Positioning

Our earlier scheme of document positioning was based on non-overlapping space partitioning [40]. The deBerg's Delaunay triangulation algorithm (cf. [12]), with a time requirement of $O(m \log m)$ time where m is the number of centroids, partitions the 2D space into non overlapping triangular regions with centroid positions at respective vertices. The similarity of the document to the centroids at the vertices of the triangle determines its position. We used a similarity-based ranking scheme to find the most similar triangle for a document. As a preprocessing step to facilitate successive searches for the document's most similar triangle, the similarity-based ranking scheme takes $O(m^2)$ time to link each centroid positions and their centroid neighbors' positions in 2D layout. By using Barycentric coordinates the document position is then assigned to a triangular region, after brute force search in $O(m)$ time. Thus the overall time for positioning n documents is $O(nm + m \log m + m^2)$. With $m \ll n$ and m having an upper bound, we can assume that the algorithm runs in $O(n)$ time. This algorithm is very fast, but we observed visual shortcomings in the resulting landscape. Mainly, the topography of the generated landscapes follows geometrical edges, because documents tend to be positioned on straight lines connecting cluster centroids. This negatively affects aesthetics and compromises the viewing experience.

To preserve a linear running time along with pleasing viewing experiences of a realistic landscape without troubling artifacts, we introduce two novel approaches for document positioning: (i) the first approach is based on a simple spring forces-based model (cf. [18]) and (ii) the second approach is based on geometric concepts. In the following, we will describe these two approaches in detail.

3.2.3.1 Spring Force Model Approach for Document Positioning

Assume that the document is attached to each centroid $\boldsymbol{c}$ by a spring having a spring constant k_c. Intuitively, we assume that this spring constant k_c is proportional to the similarity between the document and the centroid $\boldsymbol{c}$ in the n-dimensional space.

According to Hooke's law, the spring force $\vec{\boldsymbol{F}}_c$ exerted on the document due to the centroid $\boldsymbol{c}$ at position $c(x_c, y_c, z_c)$ is proportional to the displacement between the document position $P(x, y, z)$ and the centroid position.

$$\vec{\boldsymbol{F}}_c = -k_c \vec{\boldsymbol{r}}_{\boldsymbol{c}} \qquad (1)$$

The negative sign shows the force exerted on the document is in the opposite direction of the displacement $\vec{\boldsymbol{r}}_{\boldsymbol{c}}$ which is

essentially the difference between the position vector $x\,\boldsymbol{i} + y\,\boldsymbol{j} + z\,\boldsymbol{k}$ for the document and the position vector $x_c\,\boldsymbol{i} + y_c\,\boldsymbol{j} + z_c\,\boldsymbol{k}$ for the centroid $\boldsymbol{c}$.

$$\vec{\boldsymbol{r}}_c = (x - x_c)\,\boldsymbol{i} + (y - y_c)\,\boldsymbol{j} + (z - z_c)\,\boldsymbol{k} \tag{2}$$

Here $\boldsymbol{i}$, $\boldsymbol{j}$ and $\boldsymbol{k}$ are unit vectors in the direction of the x-axis, y-axis and z-axis, respectively.

For the document to be positioned in an equilibrium state, the sum of all forces exerted on the document ought to be zero.

$$\vec{\boldsymbol{F}}_{doc} = \sum_{\forall c} \vec{\boldsymbol{F}}_c = -\sum_{\forall c} k_c \vec{\boldsymbol{r}}_c = \vec{\boldsymbol{0}} \tag{3}$$

Substituting $\vec{\boldsymbol{r}}_c$ from equation (2) in equation (3),

$$-\sum_{\forall c} k_c (x - x_c)\,\boldsymbol{i} - \sum_{\forall c} k_c (y - y_c)\,\boldsymbol{j} - \sum_{\forall c} k_c (z - z_c)\,\boldsymbol{k} = \vec{\boldsymbol{0}} \tag{4}$$

This implies that

$$\sum_{\forall c} k_c (x - x_c) = 0\,, \quad \sum_{\forall c} k_c (y - y_c) = 0\,, \quad \sum_{\forall c} k_c (z - z_c) = 0 \tag{5}$$

In other words, if we resolve all the forces exerted on the document along the Cartesian coordinate axes, then the x-component, y-component and z-component of the resultant force $\vec{\boldsymbol{F}}_{doc}$ is also zero. Hence, using summation properties in equation (5) we have

$$x = \frac{\sum_{\forall c} k_c x_c}{\sum_{\forall c} k_c}\,, \qquad y = \frac{\sum_{\forall c} k_c y_c}{\sum_{\forall c} k_c}\,, \qquad z = \frac{\sum_{\forall c} k_c z_c}{\sum_{\forall c} k_c} \tag{6}$$

The right hand sides of each formula in equation (6) represent the weighted averages over xcomponents, y-components and z-components of the centroids. Consequently, the document position $P(x, y, z)$ simply reads as

$$x = \sum_{\forall c} s_c x_c\,, \qquad y = \sum_{\forall c} s_c y_c\,, \qquad z = \sum_{\forall c} s_c z_c\,, \tag{7}$$

$$\textit{where} \quad s_c = \frac{k_c}{\sum_{\forall Q} k_Q} \Rightarrow \sum_{\forall c} s_c = 1$$

As the spring force constant k_c is proportional to similarity between the document and the centroid $\boldsymbol{c}$ in the n-dimensional space, the weights s_c in equation (7) are, therefore, essentially the normalized similarity of documents with the centroid c in n-dimensional space.

Thus, the positions of n documents partitioned into m clusters, with given clusters' layout positions and normalized similarity with the cluster centroids, can be computed by using equation (7) in $O(mn)$. As $m \ll n$ and m has a fixed upper bound, the computation time is $O(n)$, i.e. linear with the number of documents.

3.2.3.2 Geometric Approach for Document Positioning

Consider the document has the highest cosine similarity with a centroid c from the given set S of total m centroids. The scalar product of a document vector $\vec{\boldsymbol{r}}$ and the centroid vector $\vec{\boldsymbol{r}}_c$ in three dimensional spaces will be

$$|\vec{\boldsymbol{r}}|\,|\vec{\boldsymbol{r}}_c|\cos\vartheta_c = x_c x + y_c y + z_c z \tag{8}$$

We assume that for all centroids $c \in S$, the cosine similarity $\cos\vartheta_c$ between the document vector $\vec{\boldsymbol{r}}$ and the centroid vector $\vec{\boldsymbol{r}}_c$ is an invariant with respect to dimension space transformation, and the magnitude of all position vectors is linear with respect to dimension space transformation. In other words, if $|\vec{\mathrm{R}}|$ is the norm of document vector in n-dimensional space, $\cos\theta_c$ is the cosine similarity in n-dimensional space and the scaling factor k is a constant of proportionality for transforming vector magnitude from n-dimensional space into three dimensional space, then we could formulate our assumptions as

$$\begin{cases} \cos \vartheta_c = \cos \theta_c \ , \forall c \in S \\ |\vec{r}| = k\,|\vec{R}| \end{cases} \tag{9}$$

Here, the quantities on left hand sides of the equations are in three dimensional space and the quantities on right hand sides of the equations are in n-dimensional space.

Given that we have a transformation from n-dimensional space to three dimensional space for a given set of centroids *S*, hence their position vector magnitudes in n-dimensional space and in three dimensional space are known, we could roughly estimate the scaling factor *k* by summing all centroid position vector magnitudes. Here, we could use any suitable measure of vector norm. The assumptions in equation (9) lead us to a scaling factor *k*:

$$k_{estimate} \stackrel{def}{=} \frac{\sum_{\forall c \in S} |\vec{r_c}|}{\sum_{\forall c \in S} |\vec{R_c}|} \tag{10}$$

Thus, with known document vector magnitudes, known cosine similarity and unknown document position (*x*, *y*, *z*) in the 3D space, the equation (8) represents an equation of a plane at a perpendicular distance $d_c = |\vec{r}||\vec{r_c}| \cos \vartheta_c$ from the origin of the three-dimensional Cartesian coordinate space. We could use a measured value of the perpendicular distance d_c as

$$d_c \stackrel{def}{=} k_{estimate} |\vec{R}||\vec{r_c}| \cos \theta_c \tag{11}$$

Thus equation (8) will become

$$x_c x + y_c y + z_c z = d_c \tag{12}$$

Choosing another two centroids (*b and a*) from the given set *S* of *m* centroids, the equation generates a system of linear equations for the intersection point (*x*, *y*, *z*) of three non-parallel planes.

$$\begin{bmatrix} x_a & y_a & z_a \\ x_b & y_b & z_b \\ x_c & y_c & z_c \end{bmatrix} \begin{bmatrix} x \\ x \\ x \end{bmatrix} = \begin{bmatrix} d_a \\ d_b \\ d_c \end{bmatrix} \tag{13}$$

The solution for the above system of linear equations may or may not exist (for example, the solution does not exist if a line joining any two centroids' positions also passes through the origin of three dimension space). The solution, if it exists, is a candidate for documents position (*x*, *y*, *z*) and can be computed by using the simple Cramer's rule. As the number of choices for centroid *b* and centroid *a* is $O(m^2)$, we can compute all the candidates for the document position in $O(m^2)$ time. As *m* has a fixed upper bound, the solution is computed in constant time. Finally, the most desirable candidate will be selected by comparing its distance from the position of centroid *n*. Thus, the positions of *n* documents can be computed in linear time $O(n)$.

3.2.3.3 Estimation of Scaling Factor k

Equation (10) gives us an unsophisticated estimate of scaling factor *k*, although, the given transformed positions of all centroids are results of Force Directed Placement (FDP) algorithms. The FDP algorithms generally use scalar products instead of position vectors. Furthermore, in contrast to cosine similarity between document and centroid as assumed in equation (9), the given cosine similarity between centroids are not necessarily reflecting invariance.

Therefore, a better estimation for the scaling factor *k* will, arguably, be the one computed by using scalar product in the three dimension space and in the n-dimension space.

$$k_{estimate} \stackrel{def}{=} \sqrt{\frac{\sum_{\forall a, c \in S} \vec{r_a} \cdot \vec{r_c}}{\sum_{\forall a, c \in S} \vec{R_a} \cdot \vec{R_c}}} \stackrel{def}{=} \sqrt{\frac{\sum_{\forall a, c \in S} |\vec{r_a}||\vec{r_c}| \cos \vartheta_{ac}}{\sum_{\forall a, c \in S} |\vec{R_a}||\vec{R_c}| \cos \vartheta_{ac}}} \tag{14}$$

Nevertheless, the computation time $O(m^2)$ for $k_{estimate}$ using equation (14) is higher as compared to computation time $O(m)$ using equation (10).

3.2.4 Space Transformation, Landscape Creation and Peak Labeling

Using document layout positions we computed the corresponding location on a finite two dimensional surface illustrated in (Figure 1; 8), which is then used to compute an elevation matrix (9) that represents an information landscape model. This matrix is then utilized in (10) to identify peak locations, heights and list of documents related to the peak.

The peak detection algorithm is based on a kernel window convolution over the landscape model. The list of documents under the peak is then used in the peak label assignment module (11) for subsequent queries and comparisons with the semantically tagged reference corpora (1E), which is continuously refined by the *webLyzard* platform.

The assigned labels to peaks are placed on the information landscape surface images (13) by using coloring scheme for elevation values (9) and by using labeling heuristic algorithms of the landscape image rendering module (12).

3.2.4.1 Space Transformation on 2D Surface

In earlier versions of our processing pipeline, we employed a 2D version of FDP for computing document positions in 2D. In contrast, in our new geometric approach for computing document positions, the number of axes is equal to the number of most similar centroids which are being used in computing the position. As the computation for document position needs at least three centroids for realistic landscape, we used a 3D version of FDP modules for computing document positions in 3D space. This also gives us more flexibility in visualization metaphors. For example, instead of simple projection on a fixed 2D plane we also used projections on curved surfaces of a cylinder and a sphere. Projection on a sphere is achieved by using (longitude, latitude)pairs, and the projection (u, v) on cylinder surface is computed by

$$u = \frac{1}{2}\, tan^{-1}\left(\frac{y}{x}\right),\ v = z, \qquad where\ tan^{-1}\frac{y}{x} \equiv atan2(y, x) \in [-\pi, \pi] \tag{15}$$

In contrast to a classical landscape in a 2D plane, projection on the curved surface of a cylinder yields a landscape with no boundaries on its left and right sides. Projection on a sphere yields a landscape with no boundaries at all resulting in a 3D "*topical planet* " visualisation. The reason we first project into the 3D space using FDP and then apply an additional transformation for projection onto a 2D surface (such as a 2D plane, the surface of a cylinder, or a sphere) is that it significantly simplifies the realization of the projection method. This allows us to always use the same FDP algorithm (using Euclidean distance metrics) for projecting in 3D, saving us the effort and the complexity of developing specialized FDP-versions (and the corresponding distance metrics) for projecting onto surfaces of different geometrical bodies. Instead, we apply well-known, simple mathematical transformations to transform from the 3D space onto the geometrical body surface.

3.2.4.2 Landscape Modeling

An information landscape is modeled by an elevation matrix with a specific resolution (for example 4096 x 4096). In this model, a document is represented by a small peak of fixed height having a shape of a Gaussian curve. The peak is placed at the corresponding position over the underlying matrix cells. The document influences the landscape by its height and the radius of its peak. The height value on a matrix cell is influenced by the document's peak height at that location, and the radius reflects how far the document's influence spreads. The elevation values of the underlying matrix cells are superimposed values of the peak heights, reflecting the documents' density at that particular location.

3.2.4.3 Peak Detection

We used a kernel window-based peak detection algorithm for identifying the significant peaks of the landscape (cf. [45] [35]).The center value of the matrix cell is compared with the average of the convolution of the window with the elevation matrix. If the center value is higher than the average convolving value, then the location is identified as a peak. After detecting all significant peaks, documents are assigned to their nearest peak by comparing their Euclidean distance in the landscape.

3.2.4.4 Label Computation

The set of documents under a peak is evaluated against the semantically tagged reference corpora. The evaluation process comprises of chi-square test of significance with Yates' correction for identifying over-represented terms. Furthermore, for discovering the frequently appearing text fragments within the same sentences and within the documents, it also uses the *term cooccurrence* analysis, based on pattern matching algorithm, along with *trigger phrases*, based on regular expressions [25, 42]. Finally in the resultant list, redundant nouns and synonymous labels are filtered out by using regular expressions and lookups in the WordNet library, and/or lookups in the customized dictionary of synonyms.

3.2.4.5 Map Generation and Label Placement

The last step in our landscape generation process encompasses (i) the proper coloring of each pixel, depending on the values of the corresponding density matrix cells, and (ii) the proper assignment of textual labels over the surface image.

We used the color blue to express lowest height values, then green, brown and finally light gray, for the highest height values. The slopes of the height are used for picking different shades from our color scheme. Consequently, a resulting landscape surface image resembles a geographic map having the hills at large document density areas, and valleys or oceans at the low document density areas. A screen shot of a landscape surface area is presented in Figure 3

Finally, a heuristic point feature label placement algorithm [10] is used for laying out labels on the basis of labeling quality evaluation criteria reflected in the following basic rules.

1. No label should overlap with another label

2. No label should overlap with the boundary of the image.

3. No label should overlap another peak location.

4. A peak's label has only four possible placements near the peak locations. These placements are rectangular spaces at top-right, bottom-right, top-left and bottom-left of the peak locations. At most one of these placements could be occupied by the peak's label.

5. A peak location could be tagged with at most five labels.

3.3 Incremental Computation

An initial landscape computation is required for subsequent computation of incremental landscapes. The algorithm is applied on the initial dataset and the result of the clustering partitions and layout position of the document is saved for future use. Every time we have some fresh data available, the equal number of data has to be retired for keeping the data chunk size in fresh computations. Therefore, after initialization from the previously computed stable partition, the incremental k-means algorithm removes the retired documents and adding new documents to the most similar clusters which leads to a number of k-means iterations for subsequent stable partition. FDP algorithm minimizes the average stress value hence the successive iteration will finally stop at the first local minima for the average stress value.

An example showing information landscapes for a dynamically changing sample documents from environmental blogs is shown in Figure 2. The sequence of incrementally computed information landscapes, visualizing 5,000 documents each, reflects weekly changes from 25 March 2012 to 6 May 2012. Images in right column of Figure 2 are results of document computation using geometric planes intersection methods while the images in right column of Figure 2 are results of spring force based computations for document positioning. Approximately 10% of the dataset changed between each individual steps – resulting in transformations of portions of the topography, while the overall structure remains recognizable.

Considering that all parts of the algorithm except clustering scale linearly (or better) with the number of documents, the time complexity of the entire landscape generation process will be dominated by clustering, yielding a running time of $O(n\log(n))$.

4. Evaluation

4.1 Metrics Used for Evaluation

In order to evaluate how good our document placements could retain its respective positions in higher dimension we will use following two measurements.

4.1.1 Stress Function

In Multidimensional scaling (MDS) literature, the loss function also known as raw stress function or measure of badness-of-fit is fairly formulated as [13, 6]

$$Stress = \sum_{i=1}^{n-1} \sum_{j=i}^{n} w_{ij} | f(p_{ij}) - d_{ij}(X) |^2 \qquad (16)$$

The MDS-literature refer similarity or dissimilarity values p_{ij} as the proximity values, and the corresponding distance $d_{ij}(X)$ as

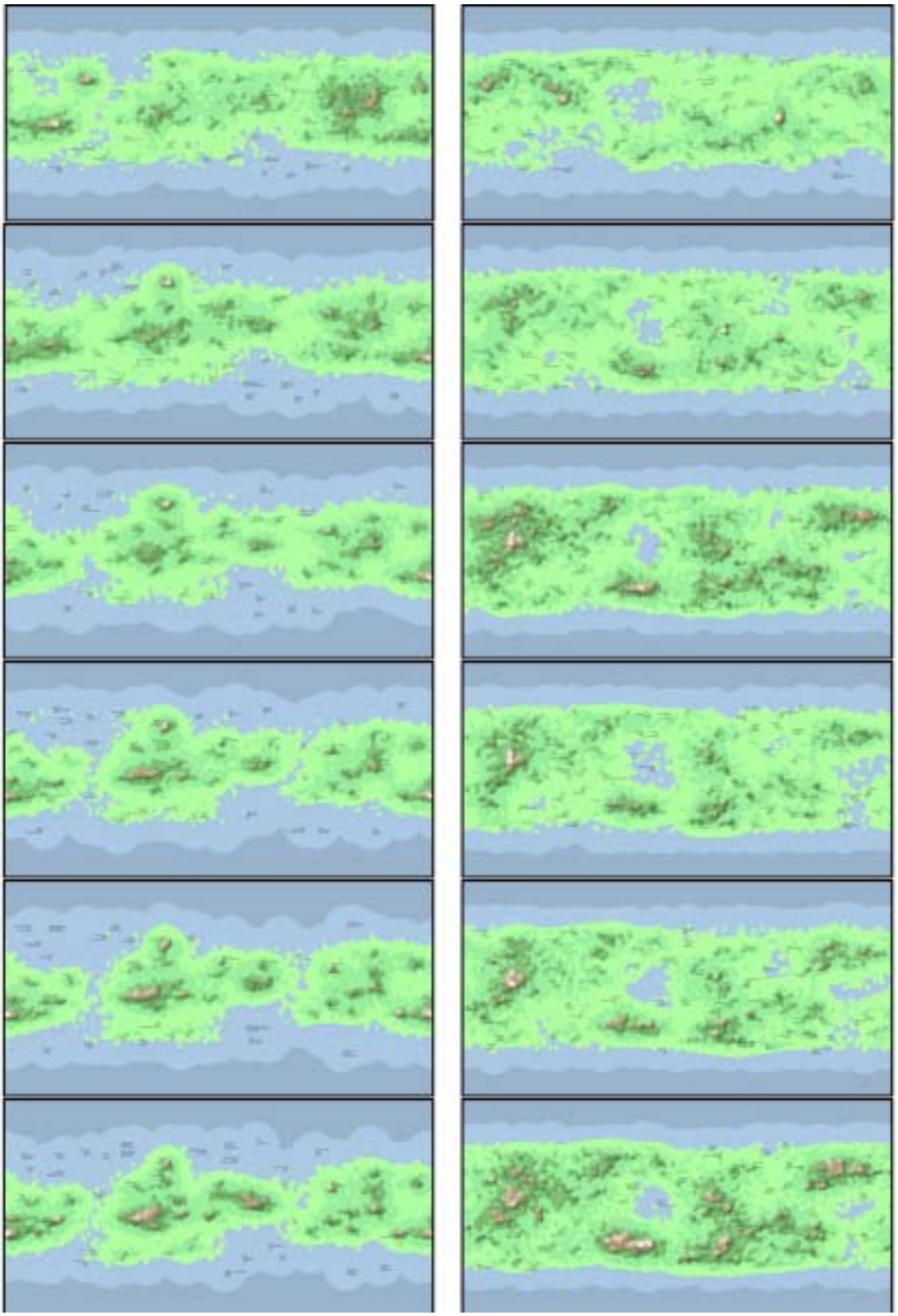

Figure 2. Temporal sequence of six incrementally computed information landscapes by using two different document position methods (spring force on left and geometric on right), reflecting weekly changes (top to bottom) in the underlying document set

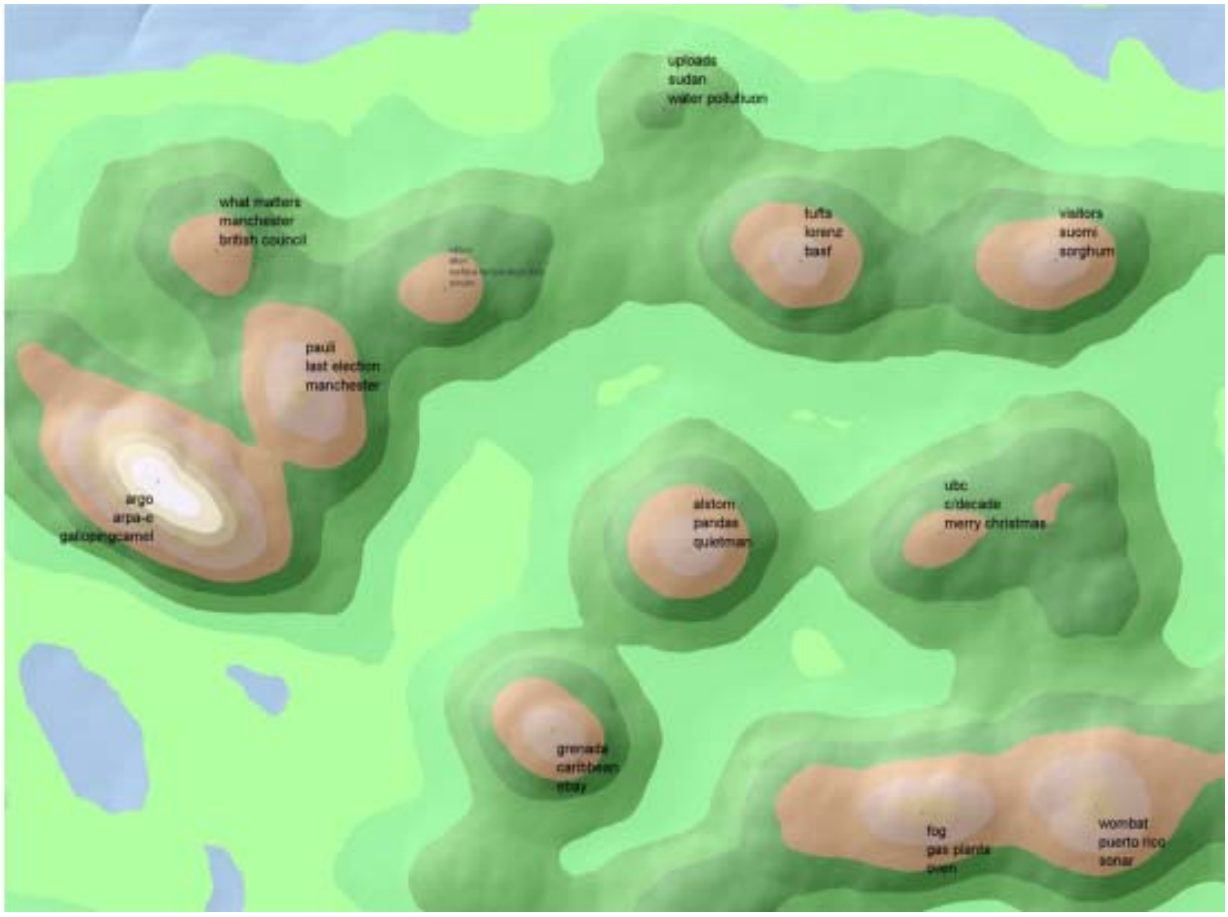

Figure 3. A screenshot of the information landscape computed for 5000 documents

the disparity values, in MDS space X which is referred as configuration space. The function $f: p_{ij} \rightarrow d_{ij}(X)$ specifies the MDS model. Here w_{ij} is the weight to be chosen appropriately. For handling missing data, we could choose $w_{ij} = 0$, if data is not available, otherwise $w_{ij} = 1$ [13].

It is required that the proximity values must be mapped to their corresponding distances, i.e. $f(p_{ij}) = d_{ij}(X)$. However, the proximity always contains noises due to measurement imprecision. Thus, one must consider our transformation as $f(p_{ij}) \approx d_{ij}(X)$, where ≈ means "*equal except for some small discrepancy*". [6].

4.1.2 Similarity Estimation

In our case, only $C_{ik} \forall K \in S$, the cosine similarity of document i with centroid K is available, where S is the set of all centroids. As the direct similarity measure between document i and document j is computationally expensive to obtain we therefore propose an estimation of similarity between two documents by employing known values of similarity between document and centroids. The estimation is based on the idea that in the absence of an observation an average value could be used. In other words, from the stand point of document i , the similarity of document i with the centroid of document j could be taken instead of the direct similarity between ments i and j. The same argument follows from the stand point of document j. Combining both perspectives by using an average formula is a reasonably accurate estimated value. We used geometric mean.

$$s_{ij} = \sqrt{c_{iJ} c_{jI}}, \qquad c_{iI} > c_{iK} \forall K \in S - I, \qquad c_{jJ} > c_{jK} \forall K \in S - J \tag{17}$$

Here, the centroid of document i is designated as I and the centroid of document j is designated as a J. Clearly, the similarity value c_{iI} of document i with centroid I is greater than the similarity value c_{iK} of document i with any other centroid, say the

centroid K, likewise $c_{jJ} > c_{jK}$.

Intuitively, $c_{max} = max\,(c_{iK})$ and $d_{max} = max\,(d_{iK})$ for $\forall\, K \in S$, $i \in [1, n]$ where d_{iK} is the Euclidian distance between the position of document i and the position of centroid K in MDS configuration space X, then

$$f(p_{ij}) = d_{max}\left(1 - \frac{\sqrt{c_{iJ}\,c_{jI}}}{c_{max}}\right) \tag{18}$$

In simple words, c_{max} is the maximum similarity value of a document and a centroid whereas d_{max} is the maximum distance between a document and a centroid in configuration space X, i.e., our final 2D layout space.

We end up with an average version of stress formula in equation (16)

$$Stress = \sqrt{\frac{2\sum_{i=1}^{n-1}\sum_{j=i}^{n}\left|d_{max}\left(1 - \frac{\sqrt{c_{iJ}\,c_{jI}}}{c_{max}}\right) - d_{ij}(X)\right|^2}{n\,(n-1)}} \tag{19}$$

The lower the stress value in equation (19), the better will be the positioning of the documents.

4.1.3 Hit Ratio

We propose a measure that tells us how many documents are still close to its top most similar centroid as compared to other centroids. In other words, a document which belongs to a cluster "*A*", in higher dimension, belongs to the same cluster "*A*", hence a hit, in the layout configuration space. We consider this as a better measure for the visual sense because it tells some degree of certainty about the document clustering. If the document is still belongs to same partition then it is more likely to be sitting around similar documents, even though, in side that partition the order of similarity between documents may have compromised. To formulate this measure we define a function for each document with position vector $\vec{r_i}$.

$$hit\,(r_i) \stackrel{def}{=} \begin{cases} 1, & iff\ \exists A \in S : |\vec{r}_{i,A}| < |\vec{r}_{i,K}| \wedge |\vec{R}_{i,A}| < |\vec{R}_{i,K}|\ \forall K \in S \\ 0, & otherwise \end{cases} \tag{20}$$

Here, $\vec{r}_{i,A}$ and $\vec{R}_{i,A}$ represents difference vectors between document and centroid A in low and high dimensional space respectively. Thus, we define our metric

$$hit\ ratio \stackrel{def}{=} \frac{1}{n}\sum_{i=0}^{n} hit\,(\vec{r_i}) \tag{21}$$

The higher the hit ratio in equation (21), we consider, the better will be the document positioning.

4.2 Experimental Setup

To facilitate the evaluation of the incremental computation framework we used seven consecutive incremental computations for 5000 documents from environmental blogs datasets of the Media Watch on Climate Change [25] during the period from 25 March 2012 to 6 May 2012 (weekly changes). In the *webLyzard* framework, each week new documents are gathered via a Web crawler and are available for landscape computations. Since, the new documents replaces the equal number of retiring documents from the fresh 5000 documents set, the landscapes for the fresh document set, need to be computed incrementally based on results of previous landscape computations of the last weeks 5000 documents set.

Furthermore, for the purpose of evaluation, we computed four of our document positioning scheme for the same set of clustered documents. The landscapes images (only first six weeks) of two document positioning schemes are shown in Figure 2. The results of using spring force based method utilizing all centroids (designated S1 in Figure 4 and Figure 5) is in the left column of the Figure 2, while the results of our geometric plain intersection method (designated G in Figure 4 and Figure 5) is presented in the right column of the Figure 2. One could feel the incremental viewing experience of landscapes in screening the images from top to bottom for each column of Figure 2.

Two additional versions of spring force based methods by choosing the similarity weights and the number of centroids in using

equation (7) are also investigated. In the second version of spring force based methods (designated S2 in Figure 4 and Figure 5) we doubled the similarity value corresponding to the most similar centroid of document and in the third version (designated S23 in Figure 4 and Figure 5) we further limited the number of centroids to the top most three. The documents positioning within the incremental landscapes are finally evaluated using the stress value computations, equation (19), and our proposed hit ratio measures, equation (21). The results of the stress value and hit ratios are presented in Figure 4 and Figure 5 respectively.

The stress value could arguably reflect the goodness of the document mapping from the n-dimensional space to the two dimensional space. However, the stress value computation requires the computation of distances (dissimilarity) for all pairs of documents in high-dimensional space, which is computationally expensive and is quadratic in time. To speed up the process and to be capable of handling large datasets, we introduce a faster variant which approximates the true stress values. Instead of computing dissimilarity between documents using their respective similarity, we substituted those similarity values with a geometric mean of the similarity between one document and centroid of the second document as an estimated value of similarity between both documents.

The formula in equation (7) for spring force based methods and the formula in equation (13) for geometric intersection of planes based methods, reflects the mapping of document from the n-dimensional space to the two dimensional space. However, a document belongs to a particular cluster may not get a position closer to its centroid position in two dimensional space. Instead, it may be closer to another centroid. This is the consequence of compromised and imperfect centroid layout positions in the two dimensional space, which results in the centroids losing receptiveness and similarities notions of their corresponding n-dimensional space.

As the documents needed to be placed closer to the respective centroids, it is a good idea to weight more for its own centroid or to cherry pick the top most similar centroids with some scheme of weighting and disregard or eliminate the less similar centroids. We observed that considering top three most similar centroids with doubling the similarity of the documents with its own centroid turns outs to place more than 50% of documents near its own centroid position in two dimension space. All measurements were performed on a machine equipped with a 2.66GHz Intel Xeon X5355 CPU, 8GB of memory, running 64 bit versions of Linux and Java v1.6.0_29.

4.3 Comparison of Document Positioning Methods on Incrementing Landscapes

The stress of G in Figure 4 is increasing monotonically while the stresses values of S1 and S2 decreases first and then start increasing for subsequent computation.

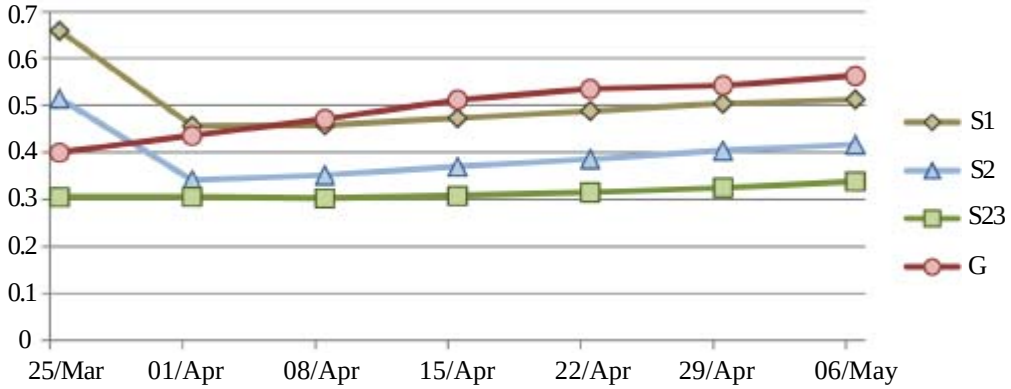

Figure 4. Comparison of average raw stress values, in incremental computation from 25 March 2012 to 6 May 2012 (weekly changes), computed by four different document positioning methods, (i) geometric plain intersection method, designated as G, (ii) spring force method with all centroids, designated as S1, (iii) spring force method with all centroids and weighting the most similar centroid twice, designated as S2, (iv) spring force method with only three most similar centroids and weighting the most similar centroids twice, designated as S23

Although, the stress values might be attributed to nature of unpredictable incremental changes in dataset, the choices of weights for the top most similar centroid and cherry picking of centroids in spring force based methods, is clearly, has a positive impact on stress values as exposed by S1, S2 and S23 in Figure 4. The impacts of these choices are more obvious in Figure 5

where the hit ratio of S2 and S23 has a jump start and are reaching considerably high hit ratio values, above 50% for S2 and above 70% for S23, as compared to S1. The hit ratio of P is not changing significantly for successive incremental landscape computation.

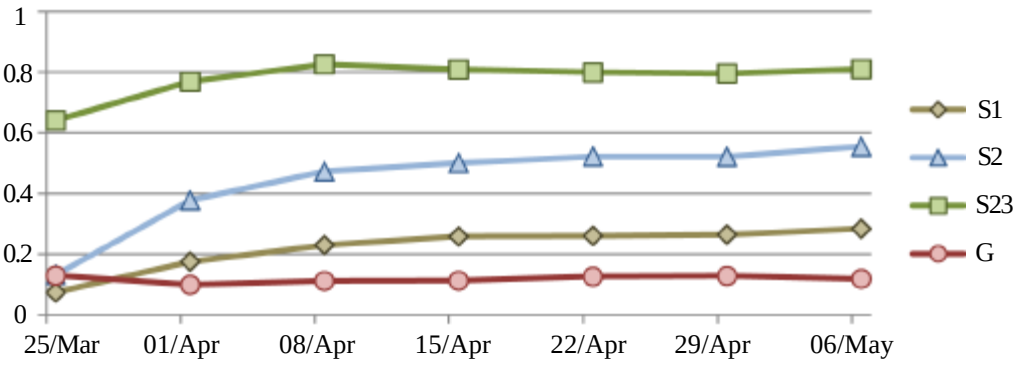

Figure 5. Trends of hit ratios, in incremental computation from 25 March 2012 to 6 May 2012 (weekly changes), computed by (i) geometric plain intersection method, designated as G, (ii) spring force method with all centroids, labeled as S1, (iii) spring force method with all centroids and weighting the most similar centroid twice, labeled as S2, (iv) spring force method with only three most similar centroids and weighting the most similar centroids twice, labeled as S23

5. Conclusion

This paper has introduced and evaluated a scalable incremental algorithm for generating dynamic topography information landscapes. The algorithm was applied on text data to visualize the evolution of document repositories where, in our opinion, it produces visually appealing layouts. Although methods based on force-directed placement are known to produce visually pleasant layouts, an empirical confirmation for our algorithm is still missing, but will be addressed in a future user evaluation.

Our method combines well-known algorithmic approaches, such as k-means clustering and force-directed placement, and introduces a novel method for fast document positioning which relies on previously computed cluster centroid positions. The central feature of the algorithm is it's incrementally. As shown by comparing the quality of incrementally computed layouts, incremental operation tends to improve initial stress values and hit ratios. Scalability of the algorithm in its current version is determined by the clustering step. As it is the only part of the algorithm with a time complexity which is (slightly) worse than linear, further optimizations will focus on increasing the scalability of this clustering step.

Future work will also focus on improving the layout quality by utilizing semantic information in the process of calculating similarities between documents, e.g. during clustering. These semantics will help us to better handle linguistic concepts such as synonymy and thus to capture more implicit, meaningful associations amongst textual resources.

Acknowledgement

The work presented in this paper was developed within the DIVINE project (www.weblyzard.com/divine), funded by the Austrian Ministry of Transport, Innovation & Technology (BMVIT) and the Austrian Research Promotion Agency (FFG) within the strategic objective FIT-IT (www.ffg.at/fit-it). The Know-Center is funded within the Austrian COMET Program (Competence Centers for Excellent Technologies) under the auspices of BMVIT, the Austrian Ministry of Eco-nomics and Labor, and by the State of Styria.

References

[1] Allan, J. et al. (1998). Topic Detection and Tracking Pilot Study: Final Report. *In*: *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*. Lansdowne, US.

[2] Andrews, K. et al. (2002). The InfoSky Visual Explorer: Exploiting Hierarchical Structure and Document Similarities. *Information Visualization*, p. 166–181.

[3] Artac, M., Jogan, M., Leonardis, A. (2002). Incremental PCA for on-line visual learning and recognition. *In*: *Proceedings of the 16th International Conference on Pattern Recognition*, p. 781-784.

[4] Arthur, D., Vassilvitskii, S. (2007). K-means ++: The advantages of careful seeding. *Proceedings of the 18th annual ACM-SIAM symposium on Discrete algorithms*, p.1027–1035.

[5] Basalaj, W. (1999). Incremental multidimensional scaling method for database visualization. *In*: *Proceedings of SPIE - The International Society for Optical Engineering*, p. 149-58.

[6] Borg, I., Groenen, P.J.F. (2005). *Modern Multidimensional Scaling: Theory And Applications*. 2nd ed. New York: Springer-Verlag. Springer Series In Statistics.

[7] Brand, M. (2002). Incremental singular value decomposition of uncertain data with missing values. *Lecture Notes in Computer Science*, p.707–720.

[8] Can, F., 1993. Incremental clustering for dynamic information processing. *ACM Transactions on Information Systems*, p. 143–164.

[9] Charikar, M., Chekuri, C., Feder, T., Motwani, R. (1997). Incremental clustering and dynamic information retrieval. *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, p. 626-34.

[10] Christensen, J., Marks, J., Shieber, S. (1995). An empirical study of algorithms for point feature label placement. *ACM Trans. on Graphics*, 14 (3) 203–232.

[11] Das, R., Bhattacharyya, D. K., Kalita, J. K. (2009). An incremental clustering of gene expression data. *Nature & Biologically Inspired Computing*, p.742-47.

[12] de Berg, M., Cheong, O., van Kreveld, M., Overmar, M. (2008). *Computational Geometry: Algorithms and Applications*. Springer-Verlag.

[13] de Leeuw, J. (1977). Applications of convex analysis to multidimensional scaling. *In*: Barra, J. R., Brodeau, F., Romier, G., van Cutsem, B. eds. *Recent developments in statistics*. Amsterdam,The Netherlands: North-Holland. p.133-45.

[14] Dhillon, I. S., Modha, D. S. (2001). Concept decompositions for large sparse text data usingclustering. *Machine Learning*, p.143–175.

[15] Ester, M. et al. (1998). Incremental clustering for mining in a data warehousing environment. *In*: *Proceedings of 24th International Conference on Very Large Data Bases (VLDB-98)*, p.323-33.

[16] Fisher, D. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, p.139–172.

[17] Fisher, D. et al. (1993). Applying AI clustering to engineering tasks. *IEEE Expert: Intelligent Systems and Their Applications*, p. 51–60.

[18] Fruchterman, T., Reingold, E. (1991). Graph Drawing by Force-directed Placement. *Software - Practice and Experience*, 12, p. 1129 -1164.

[19] Gennari, J., Langley, P. Fisher, D. (1989). Models of incremental concept formation. *Artificial Intelligence*, p. 11-61.

[20] Gorrell, G. (2006). Generalized Hebbian Algorithm for Incremental Singular Value Decomposition in Natural Language Processing. *In*: *Proceedings of Interspeech*, p. 97-104.

[21] Hartigan, J. A. (1975). *Clustering Algorithms*. New York, NY: John Wiley and Sons, Inc.

[22] Havre, S., Hetzler, E., Whitney, P., Nowell, L. (2002). ThemeRiver: Visualizing thematic changes in large document collections. *IEEE Transactions on Visualization & Computer Graphics*, p. 9-20.

[23] Heaps, H. (1978). Information Retrieval: Computational and Theoretical Aspects. p. 206–208.

[24] Hiraoka, K. et al. (2000). Successive learning of linear discriminant analysis: Sanger-Type Algorithm. *In*: *Proceedings of the International Conference on Pattern Recognition*, p. 2664-67.

[25] Hubmann-Haidvogel, A., Scharl, A., Weichselbraun, A. (2009). Multiple coordinated views for searching and navigating web content repositories. p.179 (12): 1813-1821.

[26] Kanerva, P., Kristofersson, J., Holst, A. (2000). Random indexing of text samples for latent semantic analysis. *In*: *Proceedings of the 22nd Conference of the Cognitive Science Society*, p. 103– 106.

[27] Kouropteva, O., Okun, O., Pietikäinen, M. (2005). Incremental locally linear embedding. *Pattern Recognition*, p. 1764-67.

[28] Krishnan, M. et al. (2007). Scalable visual analytics of massive textual datasets. 21*st IEEE Int'l Parallel and Distributed Processing Symposium. IEEE Computer Society*.

[29] Law, M., Jain, A. (2006). Incremental nonlinear dimensionality reduction by manifold learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, p. 377-91.

[30] Li, Y., Xu, L., Morphett, J., Jacobs, R. (2003). An integrated algorithm of incremental and robust PCA. *In*: *Proceedings of the International Conference on Image Processing*, p. 245-48.

[31] Muhr, M., Granitzer, M. (2009). Automatic cluster number selection using a split and merge kmeans approach. *Proceedings of the 20th International Workshop on Database and Expert Systems Application*, p. 363-67.

[32] Muhr, M., Sabol, V., Granitzer, M. (2010). Scalable Recursive Top-Down Hierarchical Clustering Approach with implicit Model Selection for Textual Data Sets. *Database and Expert Systems Applications, DEXA, International Workshops*, p.15-19.

[33] Pang, S., Ozawa, S., Kasabov, N. (2005). Incremental linear discriminate analysis for classification of data streams. *IEEE transactions on systems man and cybernetics*, p. 905-14.

[34] Pelleg, D., Moore, A. (2000). X-means: Extending k-means with efficient estimation of the number of clusters. *In*: *Proceedings of the 17th International Conference on Machine Learning*, p. 727–734.

[35] Razaz, M., Hagyard, D. M. P. (1999). Efficient convolution based algorithms for erosion and dilation. *In*: *Proceedings of the IEEE-EURASIP Workshop on Nonlinear Signal and Image Processing (NSIP'*99*)*, p. 360–363.

[36] Ribert, A., Ennaji, A., Lecourtier, Y. (1999). An incremental hierarchical clustering. *Proceedings of the Vision Interface Conference*, p. 586-91.

[37] Sabol, V., Granitzer, M., Kienreich, W. (2007). Fused Exploration of Temporal Developments and Topical Relation-ships in Heterogeneous Data Sets. *In*: *Proceedings of the 11th International Conference Information Visualization*, p. 369-75.

[38] Sabol, V., Kienreich, W. (2009). Visualizing Temporal Changes in Information Landscapes. *Poster and Demo at the EuroVis*.

[39] Sabol, V., Scharl, A. (2008). Visualizing Temporal-Semantic Relations in Dynamic Information Landscapes. *11th International Conference on Geographic Information Science, Semantic Web Meets Geospatial Applications Workshop*.

[40] Sabol, V. et al. (2010). Incremental Computation of Information Landscapes for Dynamic Web Interfaces. *Proceedings of the 10th Brazilian Symposium on Human Factors in Computer Systems*, p. 205-08.

[41] Sarwar, B., Karypis, G., Konstan, J., Riedl, J. (2002). Incremental singular value decomposition algorithms for highly scalable recommender systems. *In*: *Proceedings of the 5th International Conference on Computer and Information Science*, p. 399-404.

[42] Scharl, A., Weichselbraun, A., Liu, W. (2007). Tracking and modeling information diffusion across interactive online media. 2 (2) 135-45.

[43] Slagle, J., Chang, C., Heller, S. (1975). A clustering and data-reorganizing algorithm. *IEEE Trans. Syst. Man Cybern*, p. 125–128.

[44] Syed, K. A. A. et al. (2012). Dynamic Topography Information Landscapes - An Incremental Approach to Visual Knowledge Discovery. In Cuzzocrea, & Dayal, U., eds. *Data Warehousing and Knowledge Discovery - 14th International Conference, DaWak 2012, LNCS 7448*. Vienna, Austria, Springer.

[45] van Herk, M. (1992). A fast algorithm for local minimum and maximum filters on rectangular and octagonal kernels. *Pattern Recognition Letters*, 13 (7) 517-21.

[46] webLyzard, n.d. *www.weblyzard.com*. [Online].

[47] Weng, J., Zhang, Y., Hwang, W. (2003). Candid covariance-free incremental principal component analysis. *IEEE Trans. Pattern Analysis and Machine Intelligence*, p. 1034-40.

[48] Yan, J., Cheng, Q., Yang, Q., Zhang, B. (2005). An incremental subspace learning algorithm to categorize large scale text data. p. 52-63.

[49] Yan, J. et al. (2006). Effective and efficient dimensionality reduction for large-scale and streaming data preprocessing. *IEEE Trans. on Knowl. and Data Eng*, p. 320-33.

[50] Ye, J., Janardan, R., Kumar, V. (2005). IDR/QR: An incremental dimension reduction algorithm via QR decomposition. *IEEE Transactions on Knowledge and Data Engineering*, p. 1208-22.

[51] Zhang, T., Ramakrishnan, R., Livny, M. (1996). BIRCH: An efficient data clustering method for very large databases. *Proceedings of the International Conference on Management of Data*, p. 103-14.