Interactive Storytelling - Bringing User Interaction to Path Discovery in Large Document Networks

Dipayan Maiti

November 8, 2011

Contents

Abstract

1 Introduction

The last couple of years have seen a veritable explosion of articles in popular media related to *big data* or the *data deluge* phenomenon. Without resorting to hard restrictions in terms of size or scope of data and with a view to keeping the definition flexible with fast evolving technologies, *big data* has typically been referred to in literature as data which is not amenable to existing methods of storage or analysis. Special reports from *The McKinsey Quarterly* [**cite report**] and *The Economist* [**cite report**] have dealt with this phenomenon in detail in quick succession. It is projected that global data will grow at an annualized rate of almost 40%. A major component of this data is 'multi-dimensional, multi-source, time-varying' unstructured textual data - this includes data from digital libraries, news sources, claim files, corporate emails and customer opinions from company-wide intranets, biomedical and health information systems, security and defense databases etc. As witnesses to conception of the *social-network* generation, where more than 30 billion pieces of social media content is added to Facebook alone in a month, it has hardly been a leap of faith for most companies to recognize text data as an important asset in their functioning. The focus on productivity based on large textual databases can be explained on one hand by the cheap storage costs and on the other hand by the sharp increase in computing power that supports more sophisticated algorithms. This shift in the focus on productivity based on harnessing text data has however been gradual - owing to the convergence of the aforementioned factors, it is only in recent times that mining of information from textual databases has been recognized as an active area of interdisciplinary research from the fields of computing, neuroscience, psychology, mathematics, statistics and economics.

In typical text mining applications, an abstraction for a large corpus of text documents is a similarity network with the notion of similarity being induced by terms or semantic associations between terms, that are shared between document pairs. Various similarity or distance metrics have been proposed and we explain our choice of distance in section 3.1. Two documents are *connected* as in a graph if their similarity is beyond a certain threshold or directly based on the fact if they share any terms or not. Algorithms for document clustering or graph clustering can be then be utilized to answer two broad sets of questions in unsupervised learning within document networks:

1) *Which documents form natural groupings or clusters based on their similarities?* - The goal is to group a collection of documents into coherent clusters such that the documents within a cluster share common characteristics while documents between clusters do not share any common characteristics. Discriminative clustering algorithms treat the problem of clustering as assigning documents to groups (based on pairwise similarities) such that a particular global clustering criterion is optimized (Zhao02). Generative clustering algorithms assume a data generating process as the model and estimate the parameters of the process by maximizing the fit of the data with the proposed model (Liu02). Cluster memberships for each document can then be assigned. In generalized clustering algorithms any document can *belong* to multiple clusters with differing values of evidence supporting the document's membership to any of the identified clusters. Once a certain clustering of the documents has been achieved, individuals groups might provide important insights about the underlying topics in a corpus. Another clustering strategy involves partitioning the weighted adjacency matrix for defining clusters; cuts are identified in the graph of documents to create partitions such that some cut-based criterion is optimized (Ding01). An overview of these algorithms is provided in (And07).

2) *Which documents and terms can be grouped together as co-clusters?* - The documents in the corpus are modeled as a bipartite graph between documents and terms. Since a document typically has a small subset of the terms of the corpus, the term-document occurrence matrix based on the vector space model is very sparse - a group of terms might characterize a subset of the documents but they are almost irrelevant (i.e. absent or almost absent) in the remaining documents. The goal then becomes defining a minimum cut vertex partition in the bipartite graph of the documents and terms to simultaneously define clusters in the term space and the document space (Dhil01).

Often however, our need goes beyond clustering of documents in the corpus or identifying pairs of documents which are *similar* to each other; we want to augment our idea of two *connected* (but not adjacent)

documents with information about the possible path that connects the two documents. This defines a third class of problems:

3) *How are any two non-adjacent documents related in a document network?* - The *Storytelling* algorithm (Kumar06; Shah10) constructs a path, henceforth called a *story*, by starting with a given document and then discovering and connecting seemingly unrelated documents along the way before finally ending in another specified document; throughout this process documents are brought into the path based on the criteria of generating the shortest path between the starting and ending document. The original Storytelling algorithm was proposed as a methodology to navigate through a graph of redescriptors defined over a set of objects and a collection of subsets over these objects. The similarity between two redescriptions was defined as the Jaccard distance between the two sets defined by the redescriptions. An $A^*Search$ algorithm was used for the path search.

The Storytelling algorithm has been applied to intelligence data (Hoss11) to aid the analyst in *foraging* information from a corpus containing news clips. In this case, Storytelling helps us understand how one document is related to a non-adjacent document via intermediate inter-document relationships – it does so by connecting two documents based on their term-vectors and creating a sequence of documents along the shortest path, from a pre-specified starting document to an ending document . An edge in the graph naturally signifies such a relationship via the overlapping and non-overlapping terms in the documents that define the edge. From a term-vector representation of document space, two documents are *unrelated* if the inter-document distance is greater than a certain threshold or due to a total absence of any common terms between them. The resulting story is supposed to span over key characters, places and events that point to one or more viable terrorist threats. The analyst then uses his subject matter expertise and semantic associations to accept, modify or reject the hypotheses contained in the story. The algorithm also has parameters at the analyst's disposal to control the adjacency matrix of the documents via constraints on clique size and the maximum nearest neighbor edge length - visually this translates to impacting the length of the story for the analyst. Clique size guides the path through regions of desired levels of connectivity and the thresholding distance controls the radius of the local neighborhood of a document in the graph.

1.1 An Example of Interactive Storytelling

We now proceed with an example that simulates a simplified intelligence scenario with fifty documents within which are embedded clues for a possible terror plot. These could possibly be a corpus of newspaper clips related to an imminent terror threat that has been disclosed by a pertinent law enforcement authority. We ignore parameters like clique size and threshold distance. As in a typical storytelling paradigm, an analyst identifies two documents that are unrelated in as much as their general theme is concerned but which are perhaps connected as a coherent *story* via a sequence of intermediate documents. Each document in our simulation corresponds to a bag of terms and their simulated counts within the document. However, to recreate a realistic scenario we also provide possible texts for the documents based on their respective bag of terms (but not their simulated counts). This lends a certain level of legitimacy in how we formulate our simplified example and will help us elaborate the strength of our algorithm later. The first document that has caught the intelligence analyst's attention reads as follows:

```
Document (D25) - A bank robbery was reported in the town of Blacksburg.
The robbers fled the scene of robbery in a red truck.
Blacksburg is a popular ski destination for tourists during the winter months.
```

; and the second document explicitly mentions a possible terror threat:

```
Document (D47) - The terror alert has been raised to orange level nationwide
due to a possible chemical attack. To avert a catastrophe, citizens are
requested to inform local authorities about any abandoned material
emanating a sweet odor as these might be potentially hazardous.
```

Our storytelling algorithm under the absence of any interaction provides the analyst with the following *story*.

```
Document (D25) - A bank robbery was reported in the town of Blacksburg.
The robbers fled the scene of robbery in a red truck. Blacksburg is a popular
ski destination for tourists during the winter months.
```

```
Document (D29) - The Red Trucking Co. has filed for bankruptcy due to defaulting
farmers. Loans to farmers have become increasingly risky due to the prolonged winter.

Document (D45) - Terror has gripped farming communities nationwide due to the
spate of defaults on farmers' loans. The government is trying to avert the
possible drying up of loans to the farmers as the lucrative orange season approaches.

Document (D47) - The terror alert has been raised to orange level nationwide
due to a possible chemical attack. Citizens are requested to inform local
authorities about any abandoned material emanating a sweet odor as these might
be hazardous.
```

The story and a visualization of the document space based on multidimensional scaling (MDS) is provided in figure 1. Our choice of MDS is arbitrary and any efficient visualization algorithm could have been used. The progression of the story based on the resulting path might not be based on the analyst's expected hypotheses and he often wants to provide guidance in terms of favoring or ignoring certain paths. Such guidance from the user will henceforth be termed as *feedback*. Feedback will be strictly in the domain in which the analyst is comfortable with i.e. in the visual domain where the documents and stories are visualized on a two-dimensional display.

In our example, the analyst believes that the bank robbery might be related to financing the procurement or manufacturing of any hazardous chemical. Under these circumstances the analyst provides a feedback - the specific feedback is in terms of two documents that he believes should either be in the story or perhaps should have information about how **D25** and **D47** are related. To that effect, he identifies a document (**D43**) with listings of chemical factories in the state. Such a document in the corpus happens to be a catalogue for recent hirings in the state chemical industries. The second document (**D20**) identified by the analyst merely lists the hazardous chemicals that have been found abandoned recently in the state along with their odors to help the law enforcement officials identify them. We also assume that the feedback explicitly restricts that terms corresponding to document **D43** and **D20** must appear in the sequence the feedback is injected i.e. we should expect the story to *evolve* from a robbery to possible connection to local chemical plants and then maybe to recent cases of spills or abandonment of hazardous chemicals. Our final goal is to provide a new *story* that takes into account this feedback and reweighs the importance of the terms of the corpus. Such a reweighing induces a different proximity structure between the documents, which in turn provides a story which is hopefully consistent with the analyst's view of how the threat might have progressed. After incorporating the feedback in the algorithm, the new story is as follows:

```
Document (D25) - A bank robbery was reported in the town of Blacksburg.
The robbers fled the scene of robbery in a red truck.
Blacksburg is a popular ski destination for tourists during the winter months.

Document (D49) - Local chemical factory SweetDemolition recently closed
down and retracted its open hiring positions due to a change in ownership.
Located in the outskirts of Blacksburg, it provided employment to
about ten employees.

Document (D30) - Locals reported a sweet odor from an abandoned chemical factory.
Authorities investigated the report but found no hazardous substance in the
premises. The factory recently retracted its open hiring positions and
has since been abandoned.

Document (D47) - The terror alert has been raised to orange level nationwide
due to a possible chemical attack. Citizens are requested to inform local
authorities about any abandoned material emanating a sweet odor as these might
be hazardous.
```

The new story and the visualization of documents based on a reweighing of the terms in the similarity metric is provided in figure 2. Firstly we notice that the new story now incorporates documents that have

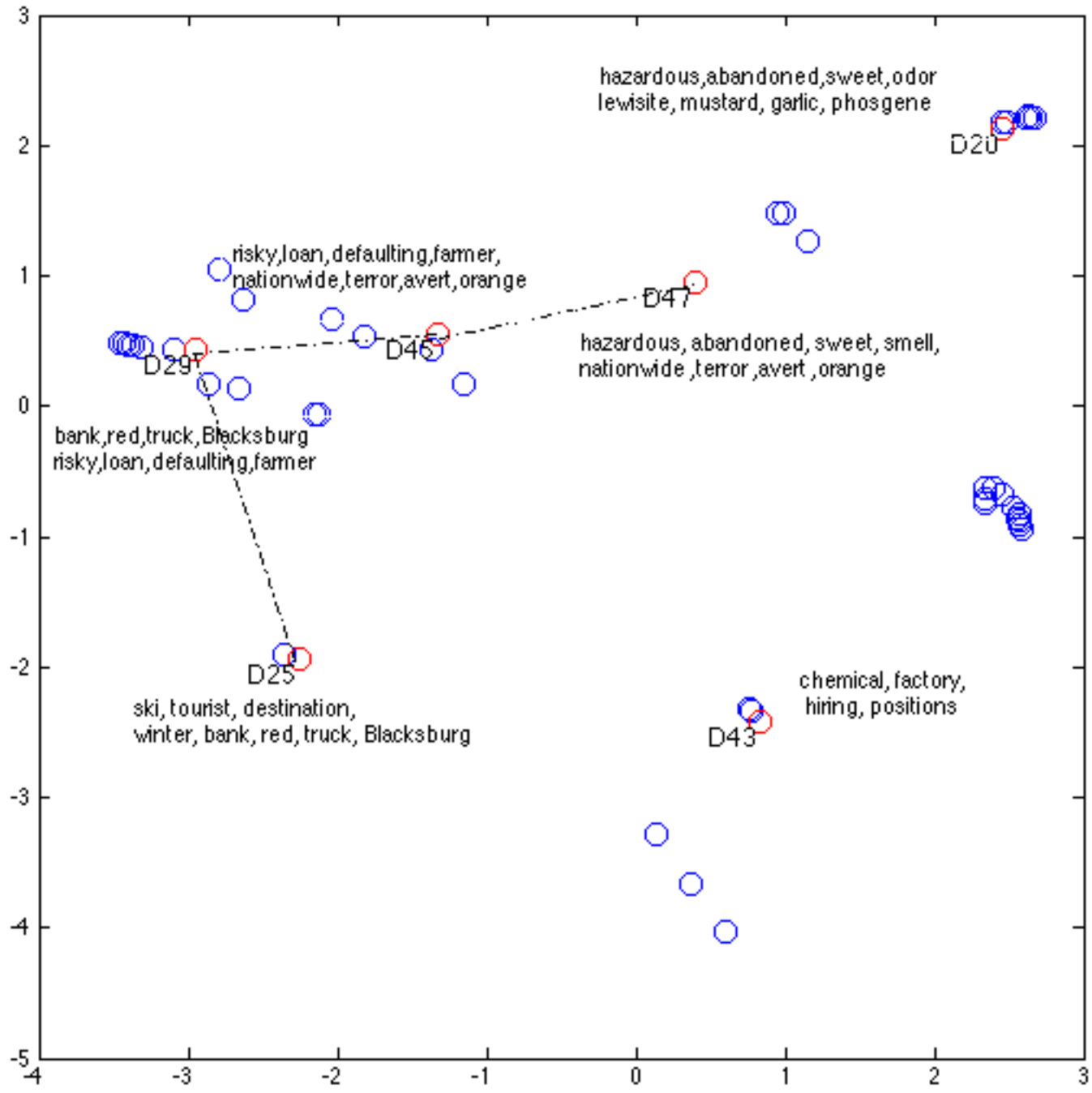

Figure 1: Visualization of the document space based on multidimensional scaling (document='O') with important documents being annotated by their respective bag of terms. The story is denoted a dashed line.

terms which the analyst wanted to include in the story in a specific order. Secondly, the two specified documents which were identified by the analyst as feedback do not show up in the new story. While the algorithm has picked up on clues from the analyst with respect to which terms might be more important in the story, it has not simply *stitched* together a new story by merely forcing it to pass through the feedback

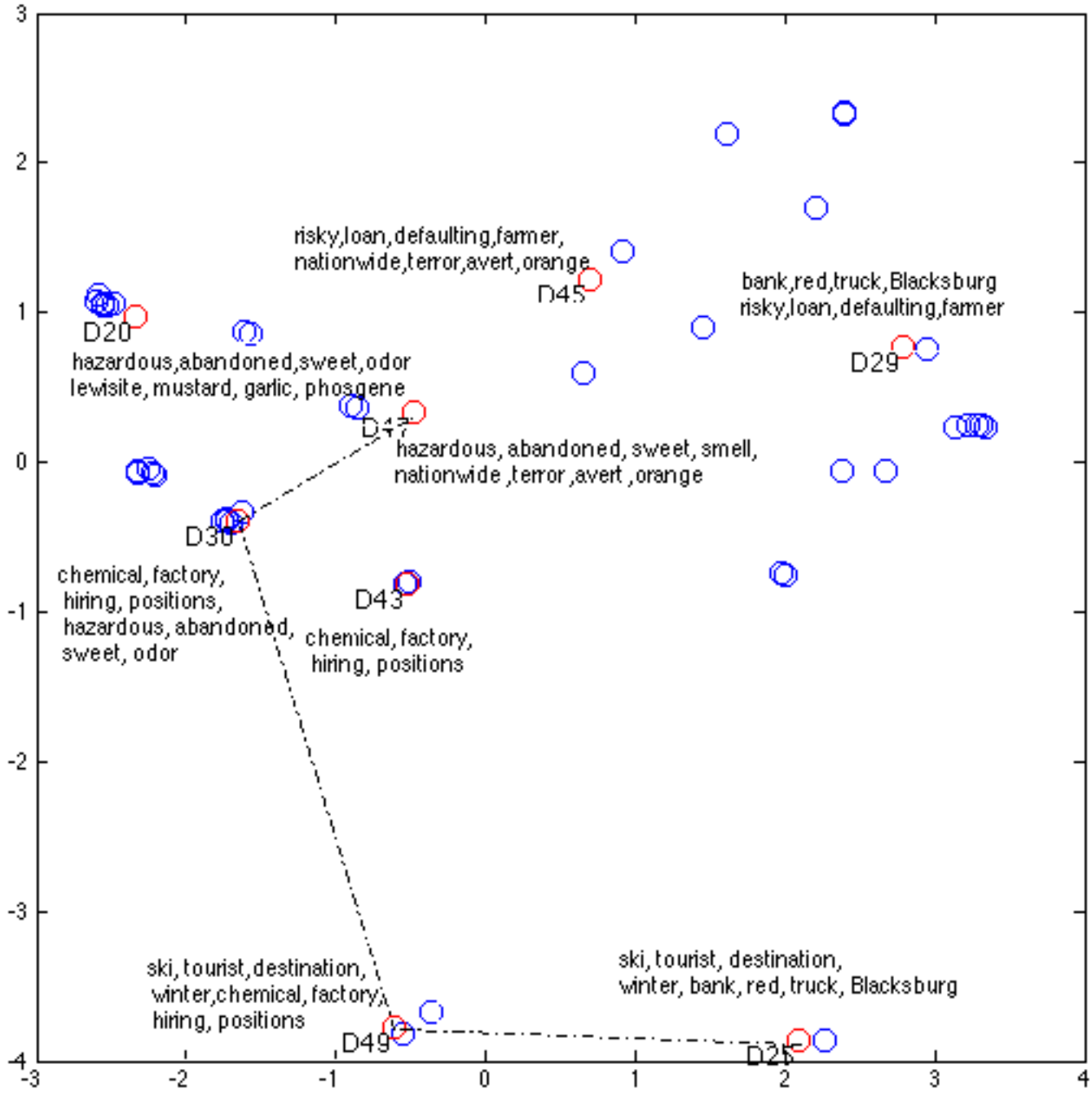

Figure 2: Visualization of the reweighed document space based on multidimensional scaling (document='O') with important documents being annotated by their respective bag of terms. The new story is denoted a dashed line.

documents. This is clearly a strength for the algorithm as it *learns* from the feedback and provides a story that it considers more appropriate based on some criterion (the shortest path criterion in this case).

This framework in which the analyst does not directly interact with the parameters of the model (clique size and maximum edge length threshold in the existing formulation of Storytelling) but instead interacts

in the visual domain to inject his feedback into the underlying model is known as *Visual To Parametric Interaction* or *V2PI* (Leman10). Section 2 explains the V2PI framework. Section 3 introduces the Interactive Storytelling algorithm as a formulation of the *inverse shortest path problem* in the V2PI framework, discusses the scope of the feedback we incorporate in our algorithm, and follows up by some implementation details of the algorithm. We also discuss 1) interpretation of re-weighting of the term space, specific to our distance measure and feedback, 2) uniqueness properties of the term weights after feedback, which is a solution of an optimization problem, 3) probable measure of divergence between the analyst's expectation and the algorithm's output, 4) a framework to incorporate multiple but sequential feedback along with spatially varying term weights. In section 4, we apply our interactive Storytelling on publicly available datasets, followed by conclusion and future work.

2 The Visual To Parametric Interaction (V2PI) Framework

The preceding example shows that the knowledge discovery process in document networks is complicated – the conclusions about the data are as important as the path to reach those conclusions. It is further complicated in large document networks due to inherent limits to human cognition when it comes to making sense of large amounts of data – short term memory can process only as many as seven distinct pieces of information (Miller56). A viable recourse has been to devise visualization methodologies that glean important pieces of information from a large dataset and then allow the user to delve in to parts of data in more detail that he deems interesting, by interacting with the visualization. Thus, incorporating visualization provides a visual metaphor to the knowledge discovery model – e.g. proximity in spatial layout of data points automatically refer to similarity, edges in networks possibly refer to hidden relationships and scatterplots make trends visible. Add to that the plethora of human interactions that are possible on a visual display, and we have a rich framework for analyzing massive datasets. (Keim08) also point out, that a visual framework simplifies our understanding of *analyzing our analyses* for future reference – hence replicating an atypical analysis or collaborating between multiple users and across different levels of abstraction become easier. Analyst's Workspace (Hoss11), Interactive Principal Component Analysis (Jeong09), Interactive Multi-Dimensional Scaling (Buja08) are all examples of such visual systems of knowledge discovery. The interactions here can be termed as *parameter level interaction* – the analyst directly modifies the parameters of the model and updates the visualization. The new visualization provides the analyst with a template to update his existing set of hypotheses. The challenge in implementing this visual analytic framework is that the responsibility of understanding the effect of parameters on the final visualization falls on the analyst – the user or the analyst should be well versed in the mathematical formulation of the problem to modify the underlying parameters.

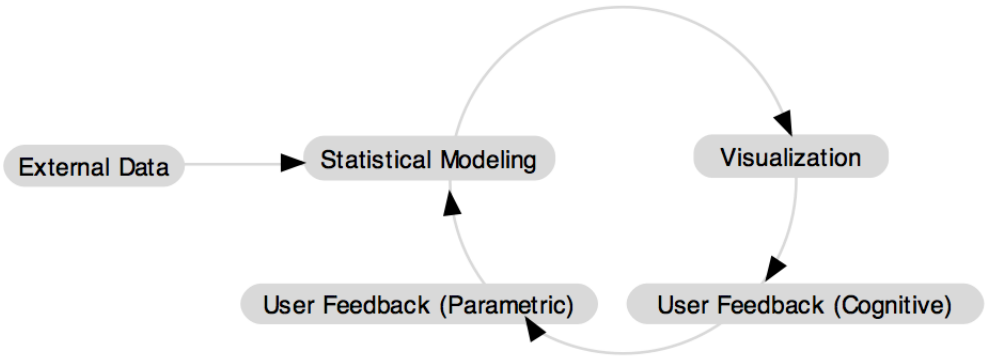

Figure 3: The Visual To Parametric Interaction (V2PI) Framework

In our framework we deviate from the current methodology of parameter level interaction and propose that the user solely focus on interacting with the data – this type of interaction is termed *object level interaction* (Endert11). Since a user is more comfortable with just interacting with the data visualization

and validating hypotheses, rather than having to understand the technicalities involved in the model, we believe this has broader appeal and acceptability for the analysts. In addition, our framework allows us to employ more sophisticated algorithms to model the data and at the same time enhance its usefulness or predictive ability based on the analyst's inputs. The typical steps in an interactive Storytelling algorithm based on such a framework will be as follows:

I) Algorithm provides a visualization based on initial values of model parameters. In *Interactive Storytelling*, this visualization is based on the shortest path in the document network between the the starting document (s) and an ending document (t), as defined by the analyst.

II) Analyst interacts with observations (i.e. documents in the two-dimensional visualization) to inject feedback based on his semantic reasoning of the data. This feedback is in terms of documents that the analyst prefers or disagrees with in the *story*. The analyst is completely shielded from the path searching and visualization algorithms.

III) Under a certain predefined mapping of the analyst's observation-level interaction (the intended feedback is termed as *cognitive feedback*) to a mental model of reasoning (e.g. bringing documents closer implies that they are similar) , the parameters of the model are re-weighted to reflect the his understanding of the data (*parametric feedback*). In our algorithm, this is based on satisfying as many user constraints as possible to ensure that the path specified by the analyst is favored over other possible local choices that have been evaluated.

IV) System regenerates an updated visualization based on the new parameter values of the model. The process continues iteratively, as does sensemaking, for the duration of the analytic process.

3 Interactive Storytelling

Following the steps outlined in the V2PI framework, we elaborate each step of the Interactive Storytelling algorithm. In the process, we also introduce appropriate notation as and when required.

I) **Intial visualization :–** The initial path is based on the original Storytelling algorithm – using term vector representation for document space, find the shortest path between s and t . Since the document network is usually extremely large, the $A^*Search$ algorithm is used to provide a path from s to t . The two important facets of the algorithm in this step are *term weighting* and defining the *distance metric*.

Confronted with hundreds of terms, it becomes important that a filtering method be applied up front to minimize computational overhead and reduce model complexity by filtering out non-relevant terms. The same criteria can also be used for term-weighting as a preprocessing step. A variety of distance measures have been proposed for path finding algorithms in document networks. A brief overview of such term-weighting schemes and distance measures is discussed in (Noreault80). We compare Inverse Document Frequency and Gini Index based weights in the preprocessing step and discuss Manhattan distance as our choice of the distance metric.

3.1 Term-Weighting and Distance Metric

For text data, it has been been observed that for the distribution of a term of rank r in the corpus (the most frequently observed term is of rank one) follows an inverse power law i.e. for the term ξ indexed by its rank r (and hence denoted by $\xi_{(r)}$), $P(\xi_{(r)}) \propto r^{-\alpha}$, where α is a positive exponent which is usually very close to 1 (Zipf's law) (Li92). (Jones72) suggests a weighting scheme which essentially assigns a weight proportional to $-log\frac{n_{\xi_{(r)}}}{N}$ to term $\xi_{(r)}$ where $||D|| = N$ is the number of documents in the corpus and $||\{d \in D : \xi_{(r)} \in d\}|| = n_{\xi_{(r)}}$ is the number of documents in which term $\xi_{(r)}$ occurs. Hence for α close to one, the weight is proportional to $-log(\frac{1}{r})$; it is the *Inverse Document Frequency* (IDF) weight. IDF is not scaled and ranges from infinity to zero. IDF measure for a term that occurs in all documents of the corpus is zero while the theoretical limit for IDF for a term occurring in only one document in a corpus of infinite size is infinity. We use $\alpha = 1$. We also compare our performance

by using *Gini Index*(GI). GI is based on the distribution of the term within the documents and measures the inequality of term occurrence in the documents. Consider an term ξ_i that occurs x_{ij} times in document d_j. The GI for ξ_i is given by,

$$GI_i = \frac{\sum_{j=1}^{N}\sum_{k=1}^{N}|x_{ij} - x_{ik}|}{2N^2\mu_i}$$

where μ_i is the average frequency for term ξ_i and given by,

$$\mu_i = \frac{\sum_{j=1}^{N} x_{ij}}{N}$$

The *Gini Index* ranges from zero, when ξ_i occurs equally frequently in all the documents $d_1, \ldots, d_N$, to a theoretical maximum of one when all but one of the documents in a corpus of infinite size ($N \to \infty$) has term ξ_i (with any non-zero frequency). It is noteworthy that the frequency of term ξ_i in the lone document does not affect GI_i. The GI can be used to rank (and hence filter) terms. After the filtering step, GI can be used in weighting x_{ij}. The weighted x_{ij} is defined as

$$GI_i \times \frac{x_{ij}}{||d_j||} \times \frac{||\xi_i||}{F}$$

where,

$$||d_j|| = \sum_{i=1}^{K} x_{ij} \text{ : sum of all term frequencies in document } d_j$$

$$||\xi_i|| = \sum_{j=1}^{N} x_{ij} \text{ : sum of term frequencies for term } \xi_i \text{ across all documents in corpus}$$

$$F = \sum_{i=1}^{K}\sum_{j=1}^{N} x_{ij} \text{ : sum of frequencies across all terms and documents}$$

After the preprocessing, for M terms in the corpus, the term vector for document s is denoted by $\boldsymbol{x_s} = (x_{s1}, \ldots, x_{sM})$. We denote any path P with L edges by $< s\ , d_{P(1)}, d_{P(2)}, \ldots, d_L, t\ >$. Here $d_{P(i)}$ is ith document in the path after s ; the path is of length L. For an arbitrary edge $e_{ij} = (d_i, d_j)$ in any path and a weight vector $\mathbf{w} = (w_1, \ldots, w_K)$, the cost of the edge is given by the weighted Manhattan distance metric:

$$c(e) = ||e_{ij}||_w = \sum_{k=1}^{K} w_k \Delta_{(ij)k}, \text{where } \ \Delta_{(ij)k} = |x_{ik} - x_{jk}|.$$

The total cost of the shortest path P^* is calculated as, $c(P^*) = \sum_{k=1}^{K} w_k \Delta_k^*$, where $\Delta_k^* = \sum_{e_{ij} \in P^*} \Delta_{(ij)k}^*$. Here Δ_k^* is the contribution by the kth dimension towards the total cost of P^*. Similarly, $gScore(l)$ for a node l can be expressed as the sum of edge costs for edges in the shortest path from s to l i.e. $gScore = \sum_{k=1}^{K} w_k \Delta_k^{gScore(l)}$ where $\Delta_k^{gScore(l)}$ is defined equivalently as before. Being a cost function itself, the heuristic distance for a node m is given by the straight line distance to document t i.e. $h^*Score(m) = \sum_{k=1}^{K} w_k \Delta_{(mt)k}^*$ where $\Delta_{(mt)k}^* = |x_{mk} - x_{tk}|$. Manhattan distance follows the triangle inequality and hence $hScore$ is an admissible heuristic. $fScore(l)$ can be evaluated as the sum of $gScore(l)$ and $hScore(l)$. The reason for using the Manhattan distance will be understood in section 3.4.

II) **Cognitive Feedback :–** Once the initial path is provided in a visualization, depending on his mental model of the data, the analyst injects his feedback into the system.

3.2 Feedback in Storytelling

We focus our attention to three possible types of user feedback:

(a) The investigator selects a sequence of documents $\mathcal{C} =< C_1, \ldots, C_K >$ which should *perhaps* be a segment of the shortest path between documents s and t . In the most general formulation of this feedback, the path between any two consecutive documents C_i and C_{i+1} in $\mathcal{C}$, could be a direct edge or a shortest path via documents that the investigator does not yet have information about. The order of the documents is important since the investigator expects to see the documents in the order specified in $\mathcal{C}$.

(b) The investigator selects a single document $\mathcal{C}$ or a set of documents $\mathcal{C} =< C_1, \ldots, C_K >$ which should perhaps, *not* be in the shortest path between documents s and t . Since the feedback specifies documents that might be excluded from the shortest path search, the order of the documents in $\mathcal{C}$ is irrelevant.

(c) The investigator selects a sequence of documents $\mathcal{C} =< C_1, \ldots, C_K >$ which is already in the shortest path between documents s and t , but insists a new permutation $\pi(C) =< \pi(C)_1, \ldots, \pi(C)_K >$ of the K specified documents in the path between s and t . As with the feedback of the first type, the path between any two consecutive documents $\pi(C)_i$ and $\pi(C)_{i+1}$ in $\pi(C)$, could be a direct edge or a path via documents that the investigator does not yet have information about.

Although the user specifies a subset of documents as her feedback, the shortest path after incorporating the feedback might or might not include all the specified documents.

III) **Parametric Feedback :–** The goal of the parametric feedback is to provide a new set of term weights $\mathbf{w}^* = (w_1^*, \ldots, w_K^*)$ for the Manhattan distance metric, such that the new path is consistent with the analyst feedback. Hence our goal is to search for the appropriate $\mathbf{w}^*$ such that the path specified by the analyst is the shortest path over all possible complete or partial path options (note that the original weights are taken to be uniform i.e. $w_k = \frac{1}{K}, k = 1, \ldots, K$ in the initial visualization). A complete path connects s and t , and a partial path is a path that starts from s but has been abandoned in course of the $A^*Search$. We include paths from the *unconstrained* $A^*Search$ over the document network as well as the paths obtained from s to t under the constraints that it has to go via $\mathcal{C} =< C_1, \ldots, C_K >$.

In the next subsections, we elaborate on the constrained $A^*Search$ algorithm, the process of defining constraints as a set of linear inequalities based on the analyst's feedback and the solution to this system of inequalities.

3.3 Constrained $A^*Search$ Algorithm

Consider the case where the investigator insists that a document C (currently not in the path between s and t) be introduced in the path between s and t . Such a feedback by the user might be motivated after reading document C, or after searching for certain terms that seem contextual to the story connecting documents s and t (and which are present in C). This can be easily extended to the case where the user insists that documents $C_1, \ldots, C_K$ be in the path between s and t in the specified order. There could be various scenarios under which the investigator introduces a specific document or a sequence of documents as being pertinent to the context of the story; we however ignore such considerations that precede the investigator's feedback.

We now elaborate on three important artifacts of our algorithm: 1) a redefined heuristic distance $h^*(,)$ between a node and the ending document, 2) the concept of *Ancestry* and 3) a rule to decide ties amongst its predecessors when the open node with the smallest $fScore$ expands to an existing open or closed node in its neighborhood. Denoting neighborhood documents of an arbitrary document D by $\mathcal{N}(D)$, there are five neighbors for D in $\mathcal{N}(D) = (D_1, \ldots, D_5)$ in figure(4). The neighbors can be defined as k nearest neighbors for a predefined k or for a predefined neighborhood radius of r around $\boldsymbol{x_D}$. The heuristic distance between a neighbor D_2 and the ending document t in the original $A^*Search$ algorithm is given by $h(D_2, C)$ (figure(4a)). Our redefined heuristic distance is given by $h^*(D_2, C) = h(D_2, C) +$

$h(C, B)$ as in figure(4b). If the investigator insists that documents $C_1, \ldots, C_K$ be in the path between s and t in the specified order, it is given by $h^*(D_2, C) = h(D_2, C_1) + h(C_1, C_2) + \cdots + h(C_K, B)$. While $h^*(,)$ ensures that the $fScore$ of a document depends on its path via the sequence of feedback nodes $\mathcal{C}$, it must consider the subset of $\mathcal{C}$ which already exists in the shortest path from s to D to estimate the heuristic distance, $h^*(D, B)$. To this end, we define on a property of a node called its *Ancestry*, that keeps track of the subset of the feedback nodes that already exists in the shortest path from the starting document to the said node. Recursively, the *Ancestry* $\mathcal{A}(D_i)$ of an arbitrary neighbor of D is defined as $\mathcal{A}(D_i) = \mathcal{A}(predecessor(D))$ if D is not a feedback node and $\mathcal{A}(D_i) =< \mathcal{A}(predecessor(D)), D >$ if D is the feedback node immediately after the subsequence $\mathcal{A}(predecessor(D))$ in $\mathcal{C}$. The starting document has an empty *Ancestry*. A node with a longer subsequence of $\mathcal{C}$ in its *Ancestry* compared to another node is said to have a *richer Ancestry*. Hence based on figure(4c), $h^*(D, B) = h(D_2, C_2) + h(C_2, B)$ since $\mathcal{A}(D) = C_1$. To break ties for an open or closed node, a predecessor with richer *Ancestry* is always favored over another. If the ancestries are comparable, for an open node the predecessor with the smaller $gScore$ is chosen, and for a closed node the predecessor with the smaller $fScore$ is chosen (refer to Appendix for details about the algorithmic implementation).

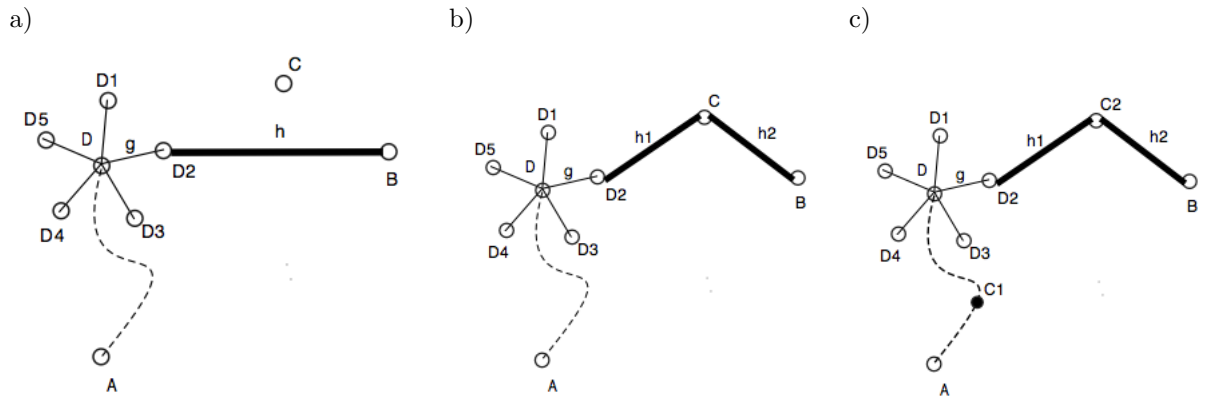

Figure 4: a) Heuristic distance $h(D_2, B)$ based on original $A^*Search$ b) Heuristic distance $h^*(D_2, B)$ based on $A^*Search$ via feedback node C c) Heuristic distance $h^*(D_2, B)$ based on $A^*Search$ via feedback nodes $\mathcal{C} = (C_1, C_2)$ where *Ancestry* of D is $\mathcal{A}(D) = C_1$. Shortest path from s to D is given by the dashed line.

3.4 Constraints as System of Inequalities

However, the path P^* from s to t via the feedback nodes $\mathcal{C}$ is not the shortest path under the current weighting of the terms (typically uniform weights over all terms) - it is the shortest path consistent with user feedback. Note that our algorithm induces an acyclic graph $G(V, E)$ on the nodes that have been visited as part of the above search process. Let e^* be an edge in P^* and e be an edge in $E - P^*$. Our goal is to find the term-weight vector $\mathbf{w}$ such that the costs of the edges are consistent with the user feedback. A degenerate solution for $\mathbf{w}$ will be based on the set of following conditions: for all $e^* \in P^*$, $c(e^*) = 0$ and for all $e \in E - P^*$, $c(e) = \infty$ (or a suitably large number). In our approach, we solve for $\mathbf{w}$ based on a set of inequalities motivated by the following observations: any edge e^* is bounded by a maximum cost given by β_{e^*} and any edge e is bounded by a minimum cost given by α_e (all other edge costs remaining fixed), such that P^* is indeed the shortest path from s to t . In other words, if the cost of an edge e^* in the shortest path crosses the upper threshold β_{e^*}, or if the cost of an edge e not currently in the shortest path falls below the lower threshold α_e, all other edge costs remaining fixed, P^* will no longer be the shortest path from s to t . The former is called the *upper shortest path tolerance* and the later the *lower shortest path tolerance*, of an edge in a shortest path algorithm. Hence our inequalities will be of the form,

$$c(e^*) \leq \beta_{e^*}, \text{for all } e^* \in P^*$$
$$c(e) \geq \alpha_e, \text{for all } e \in E - P^*.$$

These inequalities are less restrictive than the degenerate constraints discussed above and hence correspond to a $\mathbf{w}$ that is just discriminatory *enough* between the edges in the feedback consistent path and the edges outside of it. The upper and lower shortest path tolerances were studied by (Ramaswamy05) to address questions related to sensitivity analysis for shortest paths in an undirected graph. Notationally, $d^{e,k}(s\ ,t\ )$ denotes the cost of the shortest path from s to t , and for an arbitrary path P and $e \in P$, $c^{e,k}(P)$ denotes the cost of the path P with $c(e) = k$, with all other edge costs fixed. The tolerances provided in (Ramaswamy05) are:

$$\begin{aligned} \beta_{e^*} &= d^{e^*,\infty}(s\ ,t\ ) - c(P^*) + c(e^*) \\ \alpha_e &= c(P^*) - d^{e,0}(s\ ,t\ ) \end{aligned} \tag{1}$$

so that the inequalities for the respective edges in E become,

$$c(P^*) \leq d^{e^*,\infty}(s\ ,t\ ), \text{for all } e^* \in P^* \tag{2}$$
$$c(e) \geq c(P^*) - d^{e,0}(s\ ,t\ ), \text{for all } e \in E - P^*. \tag{3}$$

Since $c(e^*) > \beta_{e^*}$ implies $d^{e^*,\infty}(s\ ,t\ ) < c(P^*)$, which is exactly what we want to avoid, the first equality follows. To have an intuitive understanding of equations (1), we note that β_{e^*} is given by the difference of two path costs – the cost of the shortest path from s to t that *bypasses* e^* (and hence e^* with an infinite cost) i.e. $d^{e^*,\infty}(s\ ,t\ )$, and the minimum cost of P^* with e^* in the path (and hence e^* with a cost of zero), $c(P^*) - c(e^*) = c^{e^*,0}(P^*)$. Note that if $e = (l, m)$, then $d^{e,0}(s\ ,t\ ) = min(c(P^*), d(s\ ,l) + d(m,t\ ))$. For the second equality, if the shortest path from s to t is unchanged even with with $c(e) = 0$ i.e. $d^{e,0}(s\ ,t\ ) = c(P^*)$, then the lower tolerance for $c(e)$ is zero. If however, the constraint $c(e) = 0$ favors an alternate path through e (and hence not P^*), the lower tolerance for e is given by the drop in path cost that this alternate path allows for over P^*.

To simplify our formulation of the inequalities with output from our path search algorithm, we insist on using a stricter inequality for (3) by using $d(s\ ,l) + d(m,t\ ) \geq gScore(l) + h^*Score(m)$. Hence equation (3) can also be written as:

$$\left.\begin{aligned} c(e) \geq c(P^*) - gScore(l) - h^*Score(m), \\ c(e) \geq 0, \end{aligned}\right\} \quad \text{for all } e \in E - P^*. \tag{4}$$

Denote the cost of the shortest path from s to t via an edge $e \in E$ by $d(s\ ,e,t\ )$. The cost of the shortest path bypassing an edge $e^* \in P^*$ is given by $d^{e^*,\infty}(s\ ,t\ ) = min_{e \in E-P^*}(d(s\ ,e,t\ ) \mid e^* \notin d(s\ ,e,t\ ))$. In figure 5, the red edge is one such $e^* \in P^*$. Denote the subtree induced by the search process with e^* as the root node (the tree corresponding to dashed lines) by $\tau(e^*)$ and the remainder of the tree by $\tau^C(e^*)$ (the tree corresponding to solid lines). Based on the search process that induced the graph G, we would expect the shortest path from s to t via any edge in $\tau(e^*)$ to have e^* in it. Hence, $d^{e^*,\infty}(s\ ,t\ )$ should be based on paths via edges in $\tau^C(e^*)$. Similar to equation (4), all such path costs can be estimated heuristically by the $fScores$ for open nodes in $\tau^C(e^*)$ (green nodes in figure 5). Hence the inequality $c(P^*) \leq min_{e \in E-P^*}(d(s\ ,e,t\ ) \mid e^* \notin d(s\ ,e,t\ ))$, can be replaced by a set of inequalities,

$$c(P^*) \leq fScore(o), \forall\ o \text{ in the set of open nodes in } \tau^C(e^*). \tag{5}$$

Using weighted Manhattan distance metric, all the inequalities in (4) and (5) can be expressed in the form $A\mathbf{w}\ \leq \mathbf{0}$ for a real matrix A with m rows and K columns for m open nodes from the $A^*Search$.

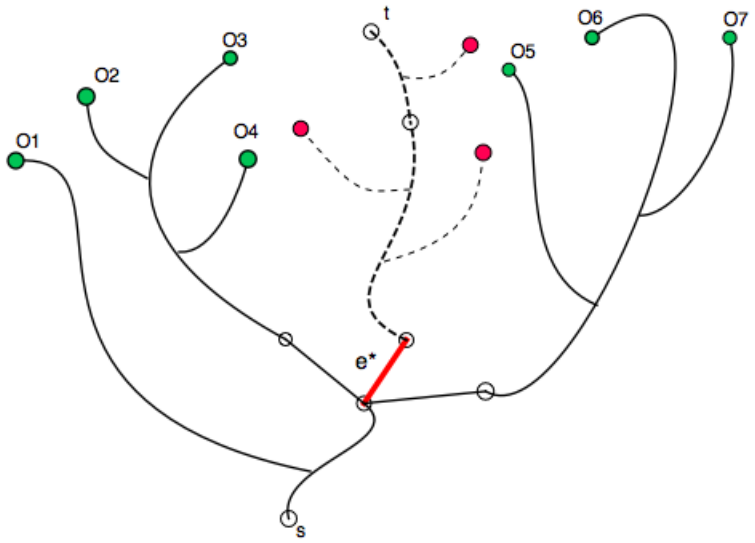

Figure 5: Dashed lines represent the tree $\tau(e^*)$ and solid lines represent the tree $\tau^C(e^*)$. Green nodes are candidate open nodes in $\tau^C(e^*)$ for inequality 2. Red nodes are open nodes in $\tau(e^*)$ and do not contribute to inequality 2.

3.5 Solution of Linear Inequalities in a Least Square Sense

Since there might not be a feasible set of weights that satisfy all the linear inequalities, we search for a solution which satisfies the system in a least square sense. Our solution is then defined as:

$$\mathbf{w}^* = min_{\mathbf{w}} \; ||(A\mathbf{w})_+||^2,$$

where $(A\mathbf{w})_+$ is an $m-$vector whose i-th element is $max((A\mathbf{w})_i, 0)$. We solve the system by Han's algorithm (Han80) and for a large system of inequalities the algorithm due to (Bramley94) can be used.

This is also an appropriate place to interpret the weights $\mathbf{w}$, in weighted Manhattan distance metric, within the context of our storytelling algorithm. Since the cost of any path $P^{(o)}$ based on an open node o in the $A^*Search$ is given by $c(P^{(o)}) = \sum_{k=1}^{K} w_k \Delta_k^{(o)}$, then any row in A can denoted by: $[\Delta_1^* - \Delta_1^{(o)}, \Delta_2^* - \Delta_2^{(o)}, \ldots, \Delta_K^* - \Delta_K^{(o)}]$. Hence the $k-th$ column in A denotes the difference in contributions towards the path cost, by the term ξ_k, between P^* and an arbitrary path $P^{(o)}$. If ξ_k is does not contribute to the path cost in P^* but does so in $P^{(o)}$, then it is more likely to be up-weighted – such a term would not be much important to the analyst since it probably does not show up in any of the documents in the analyst's preferred path, P^* but only in the local search space. On the other hand, if ξ_k is in P^* but probably not in $P^{(o)}$, it will be down-weighted so as to minimize the objective function $||(A\mathbf{w})_+||^2 = ||(A\mathbf{w})||^2$. Note that here the optimization is done on the difference of two path costs and not on any actual path cost.

IV) **Re-visualize Data :–** Once the new weights are calculated, they are applied to the Manhattan distance metric and the $A^*Search$ process is repeated. Intuitively, when the analyst insists that documents **D43** and **D20** should be in the story in the specified order, his feedback is interpreted as bringing documents with terms in **D43** and **D20** closer to the shortest path from s to t . As we explained above, this is accomplished by redistributing the unit weight on $\mathbf{w}$ in such a way that terms like ξ_k which are in the user specified documents get a lower weight and the excess weight is attributed

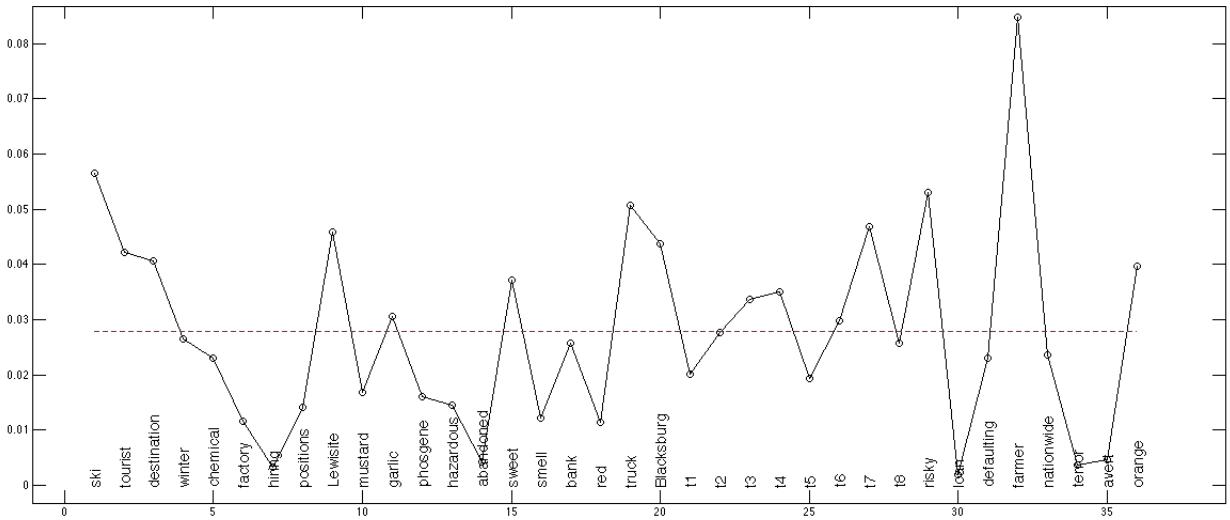

Figure 6: The red line indicates the uniform weights (for weighted Manhattan distance) for the initial visualization and path search in the Storytelling algorithm i.e. figure (1). The black line indicates the weights that were learned from the algorithm based on the analyst's feedback i.e. corresponding to the story in figure (2)

to the remaining terms, all the time ensuring that our objective function is minimized. Figure 6 shows the weights used for the terms in the weighted Manhattan distance metric in our earlier example – the red line shows the uniform weights in the initial path search. The black line represents the same term weights after we have learned from the analyst's feedback. Terms like *chemical, factory, hiring, positions, hazardous, abandoned* and *smell* in **D43** and **D20** have been down-weighted allowing documents **D49** and **D30** to come in to the new story.

4 Conclusion and Future Work

References

[And07] Nicholas O. Andrews and Edward A. Fox. Recent developments in document clustering. Technical report, 2007.

[Bramley94] R. Bramley and B. Winnicka. Solving linear inequalities in a least squares sense. *SIAM J. Sci. Comp*, 17:17–275, 1994.

[Buja08] Andreas Buja, Deborah F. Swayne, Michael L. Littman, Nathaniel Dean, Heike Hofmann, and Lisha Chen. Data visualization with multidimensional scaling. *Journal of Computational and Graphical Statistics*, 2008.

[Dhil01] Inderjit S. Dhillion. Co-clustering documents and words using bipartite spectral graphpartitioning. Technical report, Austin, TX, USA, 2001.

[Ding01] Chris Ding, Chris H. Q. Ding, Xiaofeng He, Hongyuan Zha, Ming Gu, and Horst Simon. A min-max cut algorithm for graph partitioning and data clustering, 2001.

[Endert11] Alex Endert, Chao Han, Leanna Maiti, Dipayan and House, Scotland Leman, and Chris North. Observation-level interaction with statistical models for visual analytics. Technical report, 2011.

[Han80] Han S. P. Least-squares solution of linear inequalities. In *Tech. Rep. TR-2141*, University of Wisconsin-Madison, 1980. Mathematics Research Center.

[Hoss11] *Helping intelligence analysts make connections*, 2011.

[Jeong09] D. H. Jeong, C. Ziemkiewicz, B. Fisher, W. Ribarsky, and R. Chang. iPCA: An Interactive System for PCA-based Visual Analytics. In *Computer Graphics Forum*, volume 28, pages 767–774. Blackwell Publishing Ltd, 2009.

[Jones72] K. Sprck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11–21, 1972.

[Keim08] Daniel A. Keim, Florian Mansmann, Daniela Oelke, and Hartmut Ziegler. Visual analytics: Combining automated discovery with interactive visualizations. In *Proceedings of the 11th International Conference on Discovery Science*, DS '08, pages 2–14, 2008.

[Kumar06] Deept Kumar, Naren Ramakrishnan, Richard F. Helm, and Malcolm Potts. Algorithms for storytelling. In *In Proc. KDD06*, pages 604–610, 2006.

[Leman10] Scotland C. Leman, Leanna House, Alex Maiti, Dipayan and Endert, and Chris North. A bidirectional visualization pipeline that enables visual to parametric interaction (v2pi). Technical report, 2010.

[Li92] W. Li. Random texts exhibit zipf's-law-like word frequency distribution. *IEEE Transactions on Information Theory*, pages 1842–1845, 1992.

[Liu02] Xin Liu, Yihong Gong, Wei Xu, and Shenghuo Zhu. Document clustering with cluster refinement and model selection capabilities. In *IN PROCEEDINGS OF THE 25TH ANNUAL INTERNATIONAL ACM SIGIR CONFERENCE ON RESEARCH AND DEVELOPMENT IN INFORMATION RETRIEVAL*, pages 191–198. ACM Press, 2002.

[Miller56] George A. Miller. The magical number seven, plus or minus two some limits on our capacity for processing information. *The Psychological Review*, 63:81–97, 1956.

[Noreault80] Terry Noreault, Michael McGill, and Matthew B. Koll. A performance evaluation of similarity measures, document term weighting schemes and representations in a boolean environment. In *Proceedings of the 3rd annual ACM conference on Research and development in information retrieval*, SIGIR '80, pages 57–76, 1981.

[Ramaswamy05] Ramkumar Ramaswamy, James B. Orlin, and Nilopal Chakravarti. Sensitivity analysis for shortest path problems and maximum capacity path problems in undirected graphs. *Math. Program., Ser. A*, 102:355–369, 2005.

[Shah10] M. Shahriar Hossain, Michael Narayan, and Naren Ramakrishnan. Efficiently discovering hammock paths from induced similarity networks. *CoRR*, abs/1002.3195, 2010.

[Zhao02] Ying Zhao and George Karypis. Criterion functions for document clustering: Experiments and analysis. Technical report, 2002.