ABSTRACT
We present an intelligent visual analytic system called HARVEST (High-bandwidth Analytics via Responsive Visual Exploration, Summarization, and Tracking). HARVEST is designed to empower everyday business users to derive insight from large amounts of data. It combines three key technologies to support a complex, exploratory visual analytic process without requiring users to be visualization or computer experts. First, HARVEST employs a set of smart visual analytic widgets that can be easily reused across applications and support incremental visual updates required by a continuous visual analytic process. Second, it has a visualization recommendation engine that dynamically suggests suitable visual analytic widgets to users in context. Third, it supports the semantics-based capture of user visual analytic activity for the reuse and sharing of insight provenance. We have utilized HARVEST to help users perform realistic business tasks and our preliminary study shows its promise in comparison with a well-known visualization system.

Author Keywords
Visual analytics, Web-based visualization for the masses, Automated visualization design, Insight provenance, Collaboration

ACM Classification Keywords
H.4.3 [Communications Applications]: Information Browsers; H.5.0 [Information Interfaces and Presentation]: General

1. INTRODUCTION
In recent years, a large number of visualization systems have been developed to help users view, explore, and analyze information. The capabilities supported by these visualization systems vary broadly, ranging from supporting casual visual collaborations (e.g., ManyEyes [17] and Swivel [2]) to commercial-grade visual analytics (e.g., SpotFire [1] and MayaViz [5,15]).

At the same time, businesses have been creating and storing more data than ever before. Recognizing that valuable insights are buried within these mountains of information, companies have begun to push the use of visualization to drive their business decision-making process. Moreover, companies want to empower their employees to take part in such a process. However, most of today's visualization tools still target two niche audiences: (1) dedicated information analysts and (2) dashboard consumers.

Dedicated information analysts are those who have already acquired a high degree of visualization and computer skills and often use sophisticated visualization software to derive business insights. These tools, including Spotfire and Tableau [3], offer great visualization and analytical power, but are typically too complex for average business users, especially those with limited computer skills, let alone visualization expertise.

In contrast, there are dashboard consumers who are typically casual users of visualization. They often use common visualizations to view information (e.g., a business dashboard) or engage in collaboration (e.g., ManyEyes). By design, these tools require far less skill and are accessible to a much wider range of users. However, they lack several key capabilities, such as continuous exploration of large data sets, which are often required for supporting real-world business tasks. For this reason, these tools are typically used with data known a priori, such as static datasets or predefined key performance indicators (KPIs).

While both of these audiences benefit greatly from existing visualization tools, there is a third and perhaps largest class of users for whom existing tools are of limited value. This class of users, known as everyday business users, often have extensive domain knowledge but are not visualization or computer experts. Yet as part of their daily responsibilities, they perform situational analysis tasks over massive amounts of data for which visualization can be of great benefit.

For example, in our own company, employees often examine a large wiki site containing data about numerous projects underway within our organization. While the wiki effectively provides information on individual projects, it is very difficult for users to examine project patterns or trends. Neither can most existing visualization tools make this sort of task any easier for the average person. With most tools, users must go through several steps just to reach the point where s/he can actually visualize and explore the project information. First, the user must sift through the wiki site to reach the point where s/he can actually visualize and explore the project information. Second, s/he must navigate through a large wiki site containing data about numerous projects under consideration. With most tools, users must go through several steps just to reach the point where s/he can actually visualize and explore the project information. Third, s/he must transform the extracted data into a form that the target visualization tool can take. Finally, the user must repeat all of these steps to continue the analysis if s/he decides to examine project patterns or trends. Neither can most existing visualization tools make this sort of task any easier for the average user.

For example, in our own company, employees often examine a large wiki site containing data about numerous projects underway within our organization. While the wiki effectively provides information on individual projects, it is very difficult for users to examine project patterns or trends. Neither can most existing visualization tools make this sort of task any easier for the average person. With most tools, users must go through several steps just to reach the point where s/he can actually visualize and explore the project information. First, the user must sift through the wiki site to reach the point where s/he can actually visualize and explore the project information. Second, s/he must navigate through a large wiki site containing data about numerous projects under consideration. With most tools, users must go through several steps just to reach the point where s/he can actually visualize and explore the project information. Third, s/he must transform the extracted data into a form that the target visualization tool can take. Finally, the user must repeat all of these steps to continue the analysis if s/he decides to examine project patterns or trends. Neither can most existing visualization tools make this sort of task any easier for the average user.

As this example illustrates, average users require tools that empower them to perform continuous, situational visual exploration. Yet few existing systems are designed to support this type of analysis task, especially for the skill set of the average user. We attribute the absence of visualization tools suitable for this critical audience to the following four challenges.

First, it is difficult for a system to pre-determine which visualization tools should be offered to users, since the targeted user tasks are often situational and unpredictable. Second, tools must support dynamic and incremental visual updates due to the exploratory and context-sensitive nature of many user tasks. Third, it is difficult for a system to accurately and meaningfully capture a user's insight provenance. Here, we use the term insight provenance to refer to the entire history and rationale of how insights are derived, including the relevant user's visual activity, information being explored, and the insight being extracted. Without insight provenance, users would be unable to easily re-use their analytical processes or share their techniques and conclusions with others. Finally, all of the above technical challenges must be addressed without placing any added burden on an average user.

To address these challenges, we are building HARVEST (High-bandwidth Analysis via Responsive Visual Exploration, Summarization, and Tracking), a visual analytic system that is designed for everyday business users. HARVEST combines three key technologies to support an exploratory visual analytic process without requiring users to be visualization or computer experts:

- Smart visual analytic widgets. A set of visualization
widgets with standard interfaces that can be easily reused across applications. They support semantics-based user interaction to help identify and capture user intention, and incrementally handle dynamic data sets retrieved during a continuous visual analytic task.

- Dynamic visualization recommendation. A context-driven, example-based approach that assists users in finding the proper visualizations for use in their context.
- Semantics-based capture of insight provenance. A semantics-based approach to modeling and capturing a user’s analytic process. It supports automatic detection of user action patterns for better visualization recommendation and flexible adaptation of a user’s analytic process.

The remainder of the paper begins with a discussion of related work, followed by an overview of HARVEST. We then highlight HARVEST’s three key technologies. Finally, we present a preliminary study that evaluates how well HARVEST assists average users in their analysis tasks in comparison with ManyEyes [17].

2. RELATED WORK

Our work on HARVEST is related to a large number of visualization systems developed in recent years. While each of these previous systems addresses a unique set of challenges, for the purpose of our work we characterize them along two dimensions: (1) the skill required to use the system (e.g., novices vs. trained experts) and (2) the level of support for continuous visual analysis.

Along the skill dimension, there are systems that focus on making visualization more accessible to average users. Swivel [2] and Many Eyes [17] are two such examples. Both sites allow users to upload static datasets and manually configure a visualization to display the uploaded data. Although both systems provide easy ways for users to map data to a visualization, the users must decide themselves which visualization to use. Moreover, neither tool supports a continuous analysis process in which users can incrementally request new data and/or new visualizations for representing the data. In contrast, HARVEST supports continuous visual analysis, allowing average users to easily view, explore, and analyze data without requiring extensive visualization or data manipulation skills.

At the other end of the skill spectrum, there are sophisticated visualization systems designed for highly trained analysts. These tools, including Visage [5,15], Tableau [3], and Spotfire [1], allow users to analyze dynamic data sets and smoothly switch from one visualization to another. While being powerful, these systems usually require users to have a certain level of visualization and computer expertise, since they often need to work directly with raw data tables and manually configure visualizations.

To lower the skill barrier, researchers have introduced intelligent techniques. For example, automatic visualization techniques help users select suitable visualization for their tasks (e.g., [13, 15]). However, these techniques are usually driven by a set of static rules without considering dynamic user behavior during an analysis process as HARVEST does. More recently, a template-based approach is used to support systematic visual discovery [14]. However, it relies on pre-defined procedures and does not suit the situational analysis tasks that HARVEST supports.

Falling between the extremes of easy-to-use and very sophisticated visualization tools, there are a large number of visualization systems that make various compromises. For example, there are domain-specific systems that provide users with pre-configured visualizations to support exploratory analysis of data already in memory. Examples include visual analysis tools for software design [9] and text documents [7]. Compared to these systems, HARVEST is designed to support data exploration and analysis in general, although domain-specific information (e.g., domain-specific visualization examples) can be easily incorporated to tailor visualizations to users working in a particular domain/application.

Our work is also related to research efforts that address specific challenges that HARVEST faces. For example, there are systems that automatically capture histories of user visual operations for their reuse [4, 10, 11]. However, these systems do not extract semantics of the histories as HARVEST does. More recently, the Aruvi system captures the tree-based structure of a user’s analytic activity using a set of domain heuristics [16]. In contrast, HARVEST focuses on extracting semantics of a user’s insight provenance independent of domains.

Despite the large body of work, few systems have focused on addressing all of the challenges that HARVEST faces. Consequently, HARVEST uniquely offers average users the usability and power to perform situational visual analysis tasks.

3. HARVEST OVERVIEW

HARVEST is designed to support a wide variety of visual analysis applications. It has a set of domain-independent, core components with separate data repositories that store domain-specific information (e.g., application data). We first provide an overview of the HARVEST architecture. We then describe a reference application that motivates our development effort. Finally, we use the reference application as an example to describe the typical HARVEST system flow.

3.1 Architecture

HARVEST is a web-based, client-server system built on top of standard web technologies (Figure 1). This design makes HARVEST easily deployable via standard browsers like Internet Explorer without requiring installation of special software.

Starting from the client side, HARVEST provides a user with three main interaction areas: a query panel for issuing data queries (Figure 1a), a visualization canvas for displaying user-requested

![Figure 1. An Overview of the HARVEST Architecture and user interface, including (a) a query panel, (b) a visualization canvas, and (c) a history panel.](image-url)
information (Figure 1b), and a history panel where a user can view and modify his/her ongoing exploration path (Figure 1c). Given a user’s input in any of the three areas, a request is first routed to the client-side coordinator. Depending on the type of user interaction, the coordinator triggers one of the two client-server communication paths in HARVEST: the action loop and the event loop.

The action loop is the primary client-server communication path in HARVEST. It involves all server-side components (indicated by the solid black lines in Figure 1). Here, an action represents an atomic, semantic step taken by a user in his/her visual analytic process. Each action has a type (e.g., Query or Filter) that represents a user’s specific analytic intention and a set of parameters (e.g., data concepts and constraints in a Query action). Once an action reaches the server-side of HARVEST, it is processed by three modules: the query manager, the visualization recommender, and the action tracker. The query manager is responsible for interpreting and executing data queries (e.g., SQL queries to databases). Once query results are obtained, the visualization recommender selects a proper visualization to encode the retrieved data. Depending on the quality of the data, it may also decide to transform the data (e.g., normalization) for better visualization [18]. Once a visual response is created, it is then sent back to the client-side coordinator to update the visual canvas. The action tracker, meanwhile, logs every user action and the corresponding HARVEST response. It attempts to dynamically infer a user’s higher-level semantic constructs (e.g., action patterns) from the recorded user actions to capture a user’s insight provenance.

In contrast, events are triggered by lower-level, intermediate user interactions. For example, HARVEST recognizes the Query action as a whole but not the intermediate query building steps, like adding a new constraint in the query panel. These intermediate steps are considered events. When an event requires the attention of a server-side module, the event loop provides a shortcut between the client and individual server-side components (the dotted lines in Figure 1). For example, a query-building event may request a list of context-appropriate query prompts from the server. In this case, the event loop involves only the query manager on the server side. The event loop allows HARVEST to quickly satisfy intermediate needs without involving all of the server components as done in the action loop.

HARVEST maintains a library of visual analytic widgets containing the visualizations that it supports. It is also connected to several external databases holding various information (e.g., application data).

3.2 Reference Application

Our work on HARVEST is motivated by the common information needs of employees within our own company. Our organization maintains a large wiki site describing all ongoing research projects. Each project page is a semi-structured text document, containing a project description, the people working on the project, and several other important pieces of information. New projects are added to the wiki regularly, and updates are constantly contributed by people related to a project, including project members and managers.

While it is relatively easy to look up information about individual projects in the wiki, there is no easy way to obtain a quick overview of a collection of projects. Yet in many cases, higher-level summaries of information may be most valuable.

Consider a researcher named Alice who is putting together a new proposal for a computer vision research project. To help prepare the proposal, she would like to analyze all of the existing projects first. To scope her project properly, for example, Alice must decide how many “person-years” (PYs) could be realistically funded. To help answer this question, Alice would like to view the distribution of PYs in funded projects, especially in the area of computer vision. Similarly, Alice could better position her proposal if she could discover which funding programs were historically most likely to accept computer vision proposals. In addition, she would like to identify potential collaboration partners by examining related projects and their team information.

3.3 Typical HARVEST Flow

The information required to answer each of Alice’s questions is contained within the project wiki. However, there is no easy way for Alice to extract the needed insights. To help people like Alice, HARVEST provides an intuitive set of tools for them to perform visual analysis tasks and obtain insights. In this section, we trace the steps of Alice to illustrate the typical flow of a HARVEST application. Figure 2 shows HARVEST screenshots corresponding to various points in the flow.

After logging into HARVEST, Alice is shown a screen with links to her past analyses and a button to start a new task. Assume that Alice starts a new task and arrives at the main visual analysis interface that provides full access to query, visualization, and history management tools. She starts by using the query panel to build a query “summarize the number of projects by discipline.”

Once the query is submitted, the client forwards a new Query action to the server. On the server side, the action is processed by three HARVEST core components: (1) the query manager interprets the GUI input to formulate a SQL query and then executes it, (2) the visualization recommender automatically composes a bar chart encoding the retrieved data, and (3) the action tracker records the Query action as part of Alice’s insight provenance. Accordingly, the client is updated to reflect the newly created visualization in the canvas and the newly recorded Query action in the history panel (Figure 2a).

HARVEST-generated visualizations not only present users with the requested information, they also serve as an input mechanism for users to further their data exploration. For example, Alice selects a subset of five bars that correspond to five disciplines in

![Figure 2. Screenshots illustrating the typical user workflow in our reference HARVEST application.](image-url)
which she is interested. Once the disciplines are selected, she issues a Filter action using the bar chart’s context-sensitive menu. In response to this action, HARVEST updates the visualization to reflect Alice’s new data interests. Both the query and history panels are also updated to include the new data constraints and the Filter action, respectively (Figure 2b).

For all projects in the five selected disciplines, Alice now wants to examine the correlations among four variables: the discipline, funding partner, project PY, and related industry. To do so, Alice modifies the current query and submits it. Given this new Query action, HARVEST creates a parallel coordinates visualization to encode the updated data (Figure 2c). As shown here, Alice’s analysis goals and data interests evolve over the course of her task, making it impossible to know ahead of time which data sets Alice would like to analyze or the proper visualizations to use. For this reason, HARVEST supports context-sensitive queries and dynamically recommends appropriate visualizations in context.

In addition to automatically composing a top-recommended visualization, HARVEST provides users with a set of alternative views. The alternatives are displayed as thumbnails next to the visualization canvas (Figure 2a–c). A user can click on any thumbnail to switch to the alternative visualization. For example, Alice can change to a FanLens [12] view (Figure 3) of the same data shown previously using a parallel coordinates plot (Figure 2c).

Whenever a user action (e.g., Query and Filter) is performed, HARVEST updates its internal semantic representation of Alice’s insight provenance. Externally, the performed action is displayed in the history panel so that Alice can manipulate or reuse her past actions as visual analysis macros. For example, Alice could easily adapt her analysis to a new set of disciplines by editing the parameters of her previous Filter action (Figure 3).

To return to her work at a later time, Alice can bookmark her work at any point of the analysis. Each bookmark records not only the visualization state, but also the associated exploration path that led to the saved point in time. We call this path a user’s analytic trail. Besides restoring her saved trails (Figure 4), Alice can share them with co-workers or re-purpose them for new tasks.

4. Key HARVEST Technologies

To empower everyday business users like Alice to perform complex visual analysis tasks, HARVEST combines three key technologies: (1) smart visual analytic widgets, (2) dynamic visualization recommendation, and (3) semantics-based capture of insight provenance. In this section, we review each of these technologies, highlighting their unique features in addressing the challenges that HARVEST faces.

4.1 Smart Visual Analytic Widgets

The goal of empowering average users to perform continuous analysis tasks places several requirements on HARVEST’s underlying visualization tools. First, individual visualizations must accommodate users’ evolving data interests due to the exploratory nature of their tasks. Second, each tool must support HARVEST’s effort to capture the semantics of a user’s insight provenance. Third, HARVEST should easily reuse existing visualization tools or adopt new ones to support a wide variety of user tasks and applications. To fulfill these requirements, HARVEST employs a set of smart visual analytic widgets supporting: (1) incremental visual updates, (2) semantics-based user actions, and (3) standard APIs.

4.1.1 Incremental visual updates

Few existing visualization tools support incremental updates. Instead, most of these tools work with a closed data set for the lifetime of the visualization. In these cases, a new visualization must be created if the underlying data changes. However, abruptly-switching to a new visualization disrupts the visual continuity of a display and reduces a user’s visual momentum. As a result, it prevents users from comprehending information across successive displays [20].

To maintain the desired visual momentum while a user is shifting data foci, a subset of our visual widgets is designed to support incremental visual updates. They implement a visual context management module, which uses an optimization-based approach to dynamically decide how to best update the existing visualization to incorporate the new data [19].

4.1.2 Semantics-based user actions

One of HARVEST’s key goals is to capture the semantics of insight provenance, which can be used to help share and re-purpose a user’s visual analytic processes. Since a large part of a user’s activity is interacting with visual widgets, ideally these widgets should recognize the semantics of user activities as they occur. This is in contrast to most existing visualization tools that support
To achieve our goal, we implement visual analytic widgets to support a set of actions, which are semantics-based interaction primitives. Each action is defined by a type that represents a user’s specific intention (e.g., Filter and Sort) and a set of parameters (e.g., a Sort action has sorting dimension and order). To support user actions consistently across different visualization tools, we have surveyed a variety of tools and developed a standard catalog of user actions, including Filter, Sort, Pan, and Bookmark. Each visual analytic widget supports a subset of the actions in our catalog that is most suitable for the specific visualization metaphor. For example, a map supports a Pan action but a parallel coordinates plot does not. However, both support a Filter action that lets a user narrow down a data set.

While the underlying implementations of a particular type of action may vary across visual analytic widgets, actions with the same type are considered semantically equivalent by HARVEST. This allows HARVEST to respond directly to a user’s intent without knowing the specific inner workings of each visualization.

4.1.3 Standard APIs
To reuse visual analytic widgets across different applications and also to easily adopt other visualization tools in HARVEST, we have defined a standard set of APIs for programming visual analytic widgets. Table 1 lists the main APIs. Using this design, we have implemented a wide range of visual analytic widgets (Figure 5).

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Interface</td>
<td>Methods for supplying input data, including full data sets or incremental data updates.</td>
</tr>
<tr>
<td>Visual Interface</td>
<td>Methods for specifying visual preferences or visual operators for incremental visual updates.</td>
</tr>
<tr>
<td>Action Interface</td>
<td>Methods for reporting user activity to outside components in the form of actions (Section 4.1.2).</td>
</tr>
</tbody>
</table>

Table 1. Standard APIs of visual analytic widgets.

To assist average users in effectively using visualizations in their tasks, we develop a visualization recommendation engine. Given a user’s request, our engine automatically recommends the top-N suitable visualizations to the user. This engine is built on our previous effort in using example-based learning to automate visualization generation [21]. However, our previous work is limited to handling small data sets in a relatively static environment. To support rapid and continuous user interaction with large data sets in real-world HARVEST applications, we have extended our work in the following three aspects.

4.2 Dynamic Visualization Recommendation
To meet HARVEST’s practical needs, we first improve the speed and result quality of our example-based learning algorithm.

Given a user request, from a database of visualization examples, our example-based learning uses a similarity metric to retrieve the examples that are most similar to the request. The top-N matched examples are then directly reused for or adapted to the new situation (e.g., new data). Previously, we used a brute-force, graph-based similarity measuring [21]. This method is inadequate in providing the desired system response time that HARVEST demands in certain interaction situations, especially when creating complex visualizations or visualizing large data sets. We thus use a polynomial-time, dynamic programming approach to approximate graph-based example matching.

Furthermore, our recommendation engine ensures that the top-matched examples can be instantiated with the intended data to produce a quality visualization. It uses two strategies. First, it uses a set of example-specific rules to quickly weed out examples that can not be instantiated with the intended data. For example, a bar chart instantiation requires a data set that has a numeric dimension. If the data to be visualized does not meet the criterion, our recommendation engine would rule out the use of a bar chart. Second, our engine also dynamically determines which, if any, data transformations (e.g., data cleaning and normalization) should be used for producing an effective visualization [18].

4.2.1 Practical example-based recommendation.
To meet HARVEST’s practical needs, we first improve the speed and result quality of our example-based learning algorithm.

Given a user request, from a database of visualization examples, our example-based learning uses a similarity metric to retrieve the examples that are most similar to the request. The top-N matched examples are then directly reused for or adapted to the new situation (e.g., new data). Previously, we used a brute-force, graph-based similarity measuring [21]. This method is inadequate in providing the desired system response time that HARVEST demands in certain interaction situations, especially when creating complex visualizations or visualizing large data sets. We thus use a polynomial-time, dynamic programming approach to approximate graph-based example matching.

Furthermore, our recommendation engine ensures that the top-matched examples can be instantiated with the intended data to produce a quality visualization. It uses two strategies. First, it uses a set of example-specific rules to quickly weed out examples that can not be instantiated with the intended data. For example, a bar chart instantiation requires a data set that has a numeric dimension. If the data to be visualized does not meet the criterion, our recommendation engine would rule out the use of a bar chart. Second, our engine also dynamically determines which, if any, data transformations (e.g., data cleaning and normalization) should be used for producing an effective visualization [18].

4.2.2 Visual context-preserving recommendation
Besides improving the practicality of our work, we augment it to facilitate a continuous visual analysis in HARVEST.

Visual context switching impacts a user’s ability to extract and integrate information across successive displays [20]. To minimize visual disruption between user requests, our engine uses a set of heuristics to rank matched examples based on their abilities of pre-

Figure 5. Samples of visual analytic widgets currently used in HARVEST.
serving a user’s visual context. For example, one of the heuristics states that whenever possible, visualizations similar to the current display should be favored for maintaining visual continuity. Assume that Alice asks to summarize projects by the company’s organizational structure after exploring the correlations of various project attributes using a parallel coordinates plot (Figure 2c). By the above heuristic, our engine would recommend the use of parallel coordinates as its top choice, although the same data set can just as well be visualized by a FanLens.

4.2.3 User behavior-driven recommendation
To better assist users in their tasks, we also extend our work to tailor visualization recommendation to dynamic user behavior. A user’s behavior during visual analysis often signals implicit analytic needs. Assume that Alice is interacting with a FanLens that hierarchically encodes the number of projects by discipline and by sponsor (Figure 3). To compare the number of projects by sponsor in each of the disciplines, she clicks on each discipline (a slice) to expand it. Alice must repeat her actions for all disciplines.

To better help Alice in the above situation, we use a rule-based approach to make pattern-based visualization recommendation. Each rule maps a user action pattern to an implied visual task. For example, the pattern demonstrated by Alice in our example above, is mapped to a visual comparison task. Based on the inferred visual task, our visualization recommendation engine suggests suitable visualizations. In HARVEST, the action tracker is responsible for dynamically identifying user action patterns (Section 4.2.2). Using the above example, the action tracker detects a Scan pattern based on Alice’s repetitive actions. Once the pattern is detected, our engine recommends the use of a bar chart to encode the desired information for comparison. As a result, the recommended visualization not only better meets a user’s needs, but it also reduces the number of required user actions.

4.3 Semantics-Based Capture of Insight Provenance
Visual analytic tasks are often complex and time consuming. To make the process easier for average business users, HARVEST’s action tracker component maintains a semantics-based model of a user’s visual analytic activity. This model is then used to enable more effective visualization recommendation, and to allow flexible adaptation of a user’s analytic process to new situations. We refer to the model of user activity as insight provenance because it contains the entire history and rationale of how insights are derived during a user’s analytic process.

4.3.1 Identification of user analytic trails
From our empirical studies [6], one of the distinct semantic constructs observed in a user’s analysis process is a logical sequence of user actions leading to an insight. We refer to this sequence of user actions as an analytic trail. Using our reference application scenario, Alice has performed three actions, Query⇒Filter⇒Query to reach the state shown in Figure 2(c). She now bookmarks (Bookmark) this state to save it as “Project distribution for 5 disciplines” This sequence of four actions, Query⇒Filter⇒Query⇒Bookmark, becomes one of Alice’s analytic trails.

As this example illustrates, trails define a user’s exploration path and its semantics (e.g., captured by the action types and parameters). Trails typically have a well-defined structure that can be automatically identified by our action tracker using a set of rules. For example, one rule checks for certain action types, like Bookmark or Annotate, which signal the end of a trail.

When users save their work (e.g., using the Bookmark action), HARVEST preserves both the final state of the visualization as well as the user’s entire analytic trail. When a bookmark is later restored, the trail is restored as well. This allows a user to review the exploration context in which an insight was discovered. This feature is especially useful during collaborative tasks, allowing users to see not only what has been found, but also how.

4.3.2 Automatic pattern detection in user actions
In addition to identifying a user’s analytic trails, the action tracker performs pattern detection over the user’s recently performed actions in search of meaningful activity. HARVEST currently detects three user action patterns: Scan, Flip, and DrillDown.

A Scan pattern occurs when a user successively inspects a series of visual objects that represent similar data sets. Using the example given in Section 4.2.3, a Scan pattern occurs when Alice repeatedly clicks on several slices encoding disciplines to expand them. A Flip pattern occurs when a user switches between data sets with varied values along a single dimension. For example, Alice might “flip” back and forth between two visualizations that encode the project distribution by PYS for year 2007 and 2008, respectively. The main difference between a Scan and a Flip pattern is that a Scan pattern indicates a user’s interaction with the same data set, while a Flip pattern suggests a user’s interaction with different data sets. A DrillDown pattern occurs when a user performs a series of filters along several different dimensions. For example, Alice might perform three straight filtering operations: first for “projects from 2008,” then “projects related to graphics and visualization,” then finally “projects for the Banking industry.”

A DrillDown pattern indicates that a user is narrowing down the focus of his/her analysis.

We use a rule-based approach to pattern detection. Each rule includes two parts: (1) a regular expression that encodes the required action sequence, and (2) a set of features describing the required action parameters. If a sequence of actions matches the regular expression and also has the required parameters, it is considered to be a pattern. For example, one of the rules for detecting a Scan pattern is defined by a regular expression [1,4] which matches sequences of four or more Inspects. In addition, the required parameters of the Inspect actions must reference the same type of data objects. Once a pattern is detected, both the pattern type (e.g., Scan) and its parameters are reported back to HARVEST. The detected pattern is then used by the visualization recommender to dynamically provide suitable visualizations that better help users in their tasks.

4.3.3 Flexible adaptation of analytic trails
One of the main benefits of capturing action trails is to allow users to adapt their previous analysis processes to new tasks. As a user interacts with the system, HARVEST externalizes a user’s exploration path in the history panel (Figure 1c). Each time a new action is performed, the panel is updated to reflect user’s ongoing analysis. Similarly, when a user re-visits a saved bookmark, the corresponding trail is restored and externalized through the history panel.

HARVEST supports flexible manipulations of the actions in a trail. The manipulations include: Undo, Delete, and Modify. Undo allows a user to reverse his/her most recently performed actions one by one. A user can also modify or delete any action in the trail. This allows users to quickly adapt their previously performed trails to new contexts. When a trail is altered, the sequence of actions is essentially replayed by HARVEST as if it were an analytic macro whose parameters had been changed. Furthermore, a user can jump to any action and starts his/her new analysis process from there. This feature is especially powerful when combined with bookmarks. Rather than starting an analysis from scratch, a user can reload a saved trail by restoring a bookmark. S/he can then select any action in the trail to use as a starting point for his/her new analysis. Alternatively, she can re-use the entire trial and simply modify individual action parameters to meet her new needs.

5. Lessons Learned
In the process of developing HARVEST, we have learned several important lessons which we would like to share with the wider information visualization community. It is our hope that we can work together to address some of the remaining challenges.
UI consistency across visual analytic widgets. As mentioned in Section 4.1, HARVEST uses a library of visual analytic widgets. Although all these visual analytic widgets share a set of standard APIs, they may be adopted from different sources. For example, we have adopted third-party visualization components from ManyEyes and JFreeChart. Because of the variations in their original design and implementation, widgets from different sources may support different forms of user interaction even when supporting the same user intention. For example, one widget may support visual object selection using a left mouse button, while the other requires the use of right mouse button.

While our users appreciated the variety of visualization tools that HARVEST offers, they found the system as a whole lacks consistency in supporting user interaction with different widgets. The situation worsens when a user frequently switches from one visualization to another during the analysis process and the two tools happen to have totally different interaction models. To remedy this situation, we have used two strategies. First, we design new widgets to match the interaction model used in the majority of the existing visual widgets. Second, when recommending a visualization to a user, HARVEST attempts to suggest visualization tools that have similar interaction model as that of the visualization currently in use. Despite our strategies, we would hope that the information visualization community as a whole could work together and agree upon a standard set of interaction models to be supported across various visualization tools. Consequently, these tools can be better used together in systems like HARVEST.

Support for incremental data and visual updates. One of our major design goals is to support incremental and continuous data analysis. To achieve this goal, we have designed and implemented a set of smart visual analytic widgets that support incremental visual updates based on data changes. However, most existing visualization tools are designed for handling closed data sets. This traditional paradigm does not support the workflow of our target users. To support incremental data updates, these visualizations must be re-engineered, which often requires the change of the source code. We hope that in the future more visualizations are designed to meet this requirement, making them more useful for complex visual analysis tasks.

Transitions between visualizations. Most of the HARVEST’s visual analytic widgets provide animated transitions within their own context through the support of incremental updates. However, animating the transition between two different visual metaphors remains to be a challenge. This problem is more acute in HARVEST, as it supports a continuous visual analysis where users may frequently switch from one visualization to another. Some recent work has begun to address this challenge in part, e.g., supporting animated transitions in statistical data graphics of the same data set [8]. However, the difficulty is to support transitions between two arbitrary visualizations of partially overlapping data sets, e.g., transitioning from a parallel coordinates plot showing multi-dimensional company attributes to a network diagram displaying company relationships. Currently HARVEST does not directly address this issue, but it attempts to minimize visual disruptions by recommending subsequent visualizations similar to the visual metaphors on display (Section 4.2.2).

6. Evaluation
We have applied HARVEST to the reference application described in Section 3. This application allows average users in our company to visually analyze a collection of 1500 projects that was previously accessible only through a standard wiki interface.

6.1 Study Setup
To evaluate the effectiveness of HARVEST in supporting average users in their analytical tasks, we designed and conducted a comparison study. Since we are not aware of any existing systems that address the exact same set of challenges as HARVEST does, we decided to compare HARVEST with ManyEyes [17].

We chose ManyEyes for three main reasons. First, ManyEyes and HARVEST both target the similar audience. Second, we were able to use an internal version of ManyEyes with the confidential data set in our target application. Third, we had access to the ManyEyes source code so we could make the customizations required for our experiments. Specifically, we augmented ManyEyes with our own visualization widgets to ensure that both systems were equipped with the same set of visualization tools.

In this study, we focused on evaluating two key features of HARVEST: (1) automated visualization recommendation and (2) semantics-based capture of insight provenance. Since ManyEyes does not support these features, through comparison we would like to assess whether and how these two features better assist users in their analysis tasks. We designed two similar but not identical tasks to avoid potential learning effects. Each task required a user to take multiple steps to analyze a set of projects that meet certain criteria (e.g., belonging to particular project areas and managed by certain organizations). At the end of each task, the user must identify various characteristics of these projects (e.g., what percentage are cross-lab projects).

We recruited eight users in our study, all of whom were familiar with the application domain. We asked each participant to perform both tasks, one using ManyEyes and the other using HARVEST. Before the task, we gave every user a 15-minute tutorial each on ManyEyes and HARVEST. We allotted 30 minutes for each task, during which we recorded a detailed log of the user’s analytic behavior. We also timed each user step from the moment when a visualization was shown to the moment when the user switched to another visualization or made his/her next query. At the end of each task, we asked the user to complete a short survey. First, we asked the user to rate the usability of both systems on a scale of 1 (least) to 5 (best). Then, we asked them to comment on the least and most liked features in the two systems.

To ensure a fair comparison, we had participants use the HARVEST query interface in both tasks, since ManyEyes does not include any query tools. Moreover, we ignored the time spent to both query and upload data before they were visualized.

6.2 Results and Analysis
We analyzed both objective and subjective data collected from our study. Figure 6 shows that HARVEST performed significantly better by our two objective metrics: task completion time and error rate. Users of HARVEST were able to complete their tasks significantly faster (p<0.0001), with about 40% time reduction at each step of the task on average (Figure 6a). Note that we ignored the time spent retrieving, formatting, and uploading data as required by ManyEyes. Thus, the time accounted for was spent by a user to select a visualization, interact with the visualization to analyze the information, and switch to a new visualization if needed.

Based on our observations, we attribute this significant reduction in time mainly to HARVEST’s visualization recommendation, which quickly led users to proper visualizations for their task. In our post-study survey, users rated visualization recommendation as
one of their most favorite features of HARVEST. In contrast, ManyEyes required users to choose a visualization on their own. Most users had difficulty in finding the right visualizations for the task at hand. One user commented on the difficulty of this process:

“Initially I used simple visualizations like bar chart. But they gave me so little information and I had to take many steps to answer one question. Later I chose more complex visualizations but they were so complex and I didn’t even understand them.”

Our results also indicated a significant difference in task error rate between the two systems (p<0.01) (Figure 6b). Task error rate measured the number of project characteristics that a user had successfully identified in a task. We normalized this number to be a percentage value, with 0% indicating that no project property is correctly identified and 100% representing that all project properties are correctly recognized. When we checked user results with facts from the original content, we found that there was a 75% reduction in error rate on average when a task was performed using HARVEST (5.6%) vs. using ManyEyes (22%).

Based on our analysis, we attribute the sharp drop in error rate to HARVEST’s ability to let users easily explore data from different angles. In ManyEyes, changing data or visualizations can be onerous, requiring a significant amount of work. From our observations and users’ comments, we believed that users often chose to settle for the visualization and data set they already had, rather than dig deeper for real answers. In contrast, users commented that HARVEST made it “easy to switch” to alternative visualizations and different data sets. Moreover, HARVEST’s automated analytic trail management facility made operations like “go back” or “undo” trivial. In essence, it was the seamless integration of HARVEST’s key technologies that led to more accurate results. As one user commented, “[there was] coordination among [the] query GUI, analytic trail, and visualization [in HARVEST]...,” where you could “modify/specify queries from any of the three.”

From users’ subjective feedback, the participants also overwhelmingly favored HARVEST (mean rating of 4 out of 5) over ManyEyes (mean rating of 2.6) for the tasks that they performed.

During this study, we did not collect enough objective data to statistically assess how the automatic capture of a user’s insight provenance might have aided the participants in their analysis. This is mainly due to the relatively short duration of the tasks. The participants did not have a compelling need to re-examine or reuse their activity since it occurred over a short period of time (30 minutes). However, even for these short tasks, we observed that some users made use of the history panel to review what they had done so far or to edit their prior actions. We are planning additional studies to more effectively measure the impact of these features.

7. Conclusions

In this paper, we have presented HARVEST, an intelligent visual analytic system designed to empower everyday business users to derive insight from large amounts of data. Motivated by the real-world requirements of average business users, we reviewed HARVEST’s overall architecture and highlighted three of its key technologies. First, we described our library of smart visual analytic widgets that can be easily reused across applications and can cope with large dynamic data sets as often encountered within a continuous visual analytic process. Second, we presented HARVEST’s visualization recommendation engine that dynamically recommends suitable visual analytic widgets to users in context. Third, we explained how HARVEST supports the semantics-based capture of user insight provenance for its reuse and sharing.

We applied HARVEST to a real-world analysis application within our company using realistic data and tasks. We designed and conducted a comparison study that evaluated our prototype against ManyEyes, a well-known visualization system targeting the same type of users. Our preliminary results show that HARVEST performed significantly better by two objective metrics: task completion time and error rate. The subjective feedback from our participants also positively confirmed the value of HARVEST.

REFERENCES