Relating Documents via User Activity: The Missing Link

Elin Rønby Pedersen[1]
Google, Inc.
Mountain View, CA 94043
elinp@google.com

David W. McDonald
University of Washington
Seattle, WA 98195
dwmc@u.washington.edu

ABSTRACT
In this paper we describe a system for creating and exposing relationships between documents: a user's interaction with digital objects (like documents) is interpreted as links – to be discovered and maintained by the system. Such relationships are created automatically, requiring no priming by the user. Using a very simple set of heuristics, we demonstrate the uniquely useful relationships that can be established between documents that have been touched by the user. Furthermore, this mechanism for relationship building is media agnostic, thus discovering relationships that would not be found by conventional content based approaches. We describe a proof-of-concept implementation of this basic idea and discuss a couple of natural expansions of the scope of user activity monitoring.

INTRODUCTION
Information overload is a severe challenge to both overall productivity and one's sense of personal accomplishment. A true and tested way to mitigate the problem is to organize and cluster the information. However, few people are good at keeping their files organized, and even for those there are sometimes a conflict between the organization principle and the actual needs; for instance, the same document belongs to several non-overlapping categories, or the categories erode and change over time. And for the rest of us, there is always a problem of finding the proper place to store the document when we are in a rush, and re-finding what we prematurely categorized.

In recent years we have seen different approaches to help people get the benefits of organized document storage without requiring them to do all the work themselves. Techniques might rely on some inherent or emergent structure in the documents, automatically discovered by parsing the documents. Relation building based on content have severe limitations: most of our content analysis tools are limited to text, but many documents today are not textual, and relations that matters to users may not be only those of categorical similarity. Further, the notion of document is growing to include things like data-driven web pages that can change on each visit and generally cause problems for content analysis approaches.

Figure 1: Side panel showing three open documents, one of which is in focus, and four unopened documents that (in this case) all relate to the focal document

Usage tracking offers a different approach: we look at the user's behavior – at how the user is handling the digital material – and we build the relations from there.

THE IVAN APPROACH
We designed and implemented the *Ivan* system to discover and visualize relations among documents. The point of departure is that many of those relationships are reflected in the ways she works with information. That is, the activity of doing a task directly reflects the relationship patterns in information. In this way the user interaction itself becomes a link to be discovered and maintained by the system.

Thus, Ivan can improve life for the information worker in two ways, (1) by helping her get back to clusters of documents that are used repeatedly, and (2) by offloading parts of the mental work that goes into to (re-) establishing and maintaining key task/document relationships.

In brief, Ivan monitors the user's activity, taking particular notice when documents are up on the screen together, when the user switches back and forth between some of them, or when the user cuts, copies and pastes from one document to another.

Document relationships and usage are derived from raw interaction data. Algorithms were developed to calculate and adjust the strengths of such relationships between documents serving a user interface that suggests related documents.

Our approach is best characterized as a blend of recommendation systems like Amazon's book recommendations: "when you previously used this document you also looked at these documents,"

IUI'08, January 13-16, 2008, Maspalomas, Gran Canaria, Spain.

1 The work reported here was performed while the first author worked at Microsoft.

and Google's page ranking [2]: "this document is one that you have used so much or so little with other open documents."

It is important to note that the captured usage data are generic in several senses of the word. They are *application agnostic*: we do not need privileged access to the individual applications as long as we can monitor the underlying system events. And they are also *media agnostic*, as opposed to most strategies for determining relevance and relatedness that establish the relations as a derivative of some sort of similarity in content or meta-data, e.g., Haystack [1] or Stuff I Have Seen [4].

Also, we do not attempt to make extensive assumptions about what the tasks a user might be engaged in that again might trigger the use of related documents. In this aspect we differ from past work on behavioral modeling, like [7], and usage-tracking systems like Lumiere [5], TaskTracer [3] and ActivityExplorer [6]. Instead we are adopting a *radical behaviorist* approach in the sense that it is of no consequence why the user might use two documents in timely proximity; we just note the fact that she does – assuming that she might later find it useful to be served (information about) documents that were used together regardless of whether they "belong" to a single task or several.

It is important to note that while this prototype was tightly scoped to demonstrate the feasibility of a pure activity based approach, it can be made to work alongside content-based approaches as we see them applied in desktop search and like.

THE IVAN SYSTEM

The Ivan system can present itself to the user as a simple side panel on the desktop with items representing open and related documents, as shown in Figure 1. While relation-focused graph visualization might be superior for displaying this kind of relationships, we chose to focus on proving the concept of relationship building. Also, the side panel requires little to no training and learning, we use it for our examples.

Whenever the user opens a document (through the file explorer or from an application menu), a corresponding item will appear in the side panel. Along with it will appear items for other documents that were previously used concurrently with the one just opened. At any one time, the panel will show open items and items related to open items. The items are ordered based on their strength relative to any open document.

Looking at the items in the sample side panel shown in Figure 1, the darker background color of an item signifies an open document, the one in focus being the darkest; and white background color signifies a related unopened document. Clicking on an item will cause the system to open the corresponding document (if it was not already open) and bring it into focus; the list of items in the side panel will be recalculated as a result.

TECHNICAL DESCRIPTION

The Ivan implementation was tightly scoped to highlight the unique aspects of an activity-based approach, i.e., establishing meaningful relationships by interpreting user activity as links.

The goal was to serve related documents to the user where the relationships are functions of prior user activity. Relatedness in this context is defined through the usage history, for instance,

- Documents are open or active at the same time or in close timely proximity
- Documents are interchangeably brought to focus ("clicking back and forth")
- Content is exchanged between documents, with copy and paste operations.

Realizing that a production system would need to reflect a much more carefully researched and designed concept of "a document", we nevertheless decided for simplicity in this initial implementation. We apply a pragmatic criterion: a document is something that can be shown in a window and usually lives in a named file in the file system. That works well in most cases, like spreadsheets, photos, CAD drawings, but breaks when what is seen on screen does not correspond to a named file: e-mail messages, server based web pages, database views. We also decided to support only static pages, leaving handling of dynamic pages (both the pages that are essentially database views and those that are on-the-fly calculations) for more thorough design considerations. Although for these types of special pages the activity based approach would have less trouble establishing a relationship because the critical relation information would be available – the user's activity.

System Architecture and Implementation

The system architecture of Ivan is an interchangeable UI with two major processing mechanisms underneath: (1) Activity capture, and (2) Relationship building. The side panel shown in Figure 1 can easily be replaced with other UIs, or the mechanism can be embedded in other applications, like desktop search.

The technical interface between the two processing components is a stream of events, composed of the following event types: Create document; Open document; Close document; Save document; Save document as; Copy/Cut material; Paste material; Mouse Click/Double-Click.

Event capture and processing

The event capture mechanism is designed to be "application agnostic", i.e., it is strictly generic with respect to the applications in the sense that it relies only on user actions in the window system interface. A less generic approach might have utilized adapters for major applications, possibly enabling more complex and task based activity capture; but it would have left us with a solution that would be much more vulnerable to change outside our domain of control.

Activity Data from Message Spying

Since we are interested in user actions over time, we need to understand how stored documents might relate to the windows on the screen. We capture events by "spying" on the messages sent between the applications and the two essential OS components, the window manager and file system. Message spying tools are built into most operating systems, like Microsoft's Active Accessibility and Filespy for file system instrumentation.

Divide between File System and Window System

There is a sharp divide between the stored and the displayed data in today's computing environment, and this divide is entirely managed by the application. Window manager components may provide information about what is happening within and between windows on the screen, but it is basically oblivious to the content that is being displayed and manipulated. This information is known by the file system. Thus, the major work consists of synchronizing and reconciling between the content in a window and which file in the file system is most related to that content.

Listening to window messages, we first have to recognize and discard events from a multitude of windows that only exist to provide meta-information (like alerts and dialogs). Afterwards we are left with more or less one window per document, with a couple major exceptions. First, some applications provide the user with the option of multiple views on the same document, each view carried in its own window. Second, some recent UI styles with dock-able or tabbed windowpanes can make the detailed user activity opaque to this type of monitoring. Correlation with the file system allows us to resolve most cases, but as the concept of "a document" is blurred this will be a problem that grows.

Analyzing the event streams

Even with careful message spying, there are events we do not see when monitoring at the window manager level, for instance all the events that are invoked from within an application through the menus. Since action like these can be essential to understanding what the user is doing, we need to take special measures to derive them. We are able to derive most of the detailed interaction directly or from correlation of events from the three monitored components (window manager, clipboard and the file system). When that fails we revert to doing screen content comparisons through the Accessibility API. For instance, in the current implementation we derive the Save and Save As… episodes from a combination of file system events and changes in the windows. Monitoring the clipboard captures Cut and Copy episodes, and Paste episodes are determined through analysis of changes in window contents.

As soon as a window/file binding has been established we monitor the event stream for each document and ascertain that only a well-formed stream of events are being passed on to the relationship builder. The check is done through a simple state machine designed with some tolerance and recovery capability.

Relationship Building

The relationships are between documents. Our concept of a document includes any document-like entity, i.e., traditional documents, spreadsheets, pictures, messages, and web pages.

Relations

We focus on symmetrical relationships for pairs of documents. There is no technical obstacle to including asymmetrical relations and relations between more than two documents, however, they are not required by the use scenarios considered here.

Relationships between documents are established when a document is opened and there are already other documents open. Initially a relation is tentative; this is reflected in a low setting of the relation strength. A relation between two documents is strengthened when the user performs actions that involve both documents, like cut, copy, and paste, and clicking back and forth between them, and when they are subsequently open together.

Heuristics for Strength of Relationships

The most important determinant of strength is time proximity in use. A relation is created as soon as two documents are open within the same time frame (a user setting, default setting is that there is a moment when both documents are open). The relation is initially assigned the value 1, and over its lifetime it may move in the interval of 1 to 10.

The strength of established relations changes over time as a direct result of user actions (usage triggered contributions), as well as relation management operations such as "fading" (automatic decreases) and "confirmation" (usage triggered adjustments). Any user action on a document will cause a recalculation of strength between this document and all other documents it is related to. The calculation takes place in several steps: 1) calculate fading since last use; 2) calculate contribution from current event; 3) calculate any confirmation effect, 4) adjust strength to fit into value range; 5) adjust strength to equalize across relations.

Fading happens when a relation remains unused (lack of confirmation) for a while. Confirmation happens when the user activates a proposed relation, e.g., clicks on a suggested related document, or when two documents are repeatedly brought up together. As events of the same type occur repeatedly on a particular relation, we adjust the effect of the event on the strength. In general, we define two formulas for each event type: a "first occurrence" formula and a "subsequent occurrence" formula.

We impose limitation on how much the strength can grow and shrink: we limit the strength range to 1-10, and we apply a simple logarithmic mapping to keep values within.

EVALUATION

The primary purpose of the Ivan implementation was to gauge if simple and raw user activity data could indeed become useful links between documents. In this section we provide an evaluation of technical feasibility and usefulness of the approach.

A handful of people used the system over a couple of months, providing log data, ongoing design feedback and final assessment of the potential usefulness of relationships from user activity tracking. There was a general agreement that activity linking among documents is helpful.

False Positives or Rich Associations

At the outset we were worried about the pervasiveness of multitasking and not being able to distinguish between tasks that the documents were used in, thus creating false positives (from a task based point of view) in the set of related documents. During initial use of the system, it turned out to be much less of a problem, perhaps even a feature: we are showing documents that are related in time and interaction, not necessarily task organized.

What we see here is an example of how we mentally organize complex information: associations such as time, place and other contextual dimensions can play as important a role as logical structure and categories.

Document Types Based on Usage

During initial use of the system we noticed that certain documents did not get registered as the users would prefer, for instance, documents used for reference were never or very seldom changed. Thus they got a lower rating on relatedness as time went by (fading). It seemed like there should be different kinds of usage.

That led us to define document types based on usage patterns and incorporate that into our algorithms. We can identify reference documents by monitoring the usage pattern over time: long time on display, frequent visibility or focusing of the window, and minimal pasting or typing into the document are indicators of a reference document. Thus, in case of reference documents (and thereby the relations they are part of), we suspend the calculation of fading. Resulting from the first couple weeks of use of the system, we identified four different usage based categories:

- Reference documents
- Frequently used documents
- Transient documents
- Other documents.

The metric used for categorization is based on the amount of user activity in a document, i.e. accumulated and average time a document has been open and visible to the user. This very simplistic usage metric can easily be improved and also expanded to include calculations that take rhythmic usage patterns into consideration. It should be noted that this metric, just like the usage-tracked relations, is independent on the specific content of the documents.

FUTURE DIRECTIONS FOR USAGE TRACKING

While we were able to establish and adjust document relationships in a way that resonated with the users concepts of relatedness, we also realize that there are severe challenges to pulling the right information from the windows and file system.

Desktop or cloud

We may envisage other user activity monitoring techniques to come looking in two different directions. First, we can look for platform support at the desktop, hoping that operating system designers will recognize the usefulness of activity data and integrate support of secure, non-invasive monitoring. Second, leaving the desktop behind to consider the ubiquitous computing future, in "the cloud" of information access it will likely be much easier to harvest a rich activity event stream in that environment.

Search and Organization

We see two very different potential application areas of our usage based relationships: (1) ranking or narrowing results from other search approaches; and (2) widening the search basis by suggesting related documents for content based search approaches. As soon as we consider an integrated approach we are faced with the question of how to weigh different filter metrics. We are particularly eager to find a "peaceful" coexistence of our usage based approach and the prevalent Date/Time ranking most used today.

While the focus in Ivan was on the relationships between documents, we recognized that the documents themselves have interesting qualities based on the user activity "on them", for instance, the length of time a document is open; the amount of change that takes place on a document, and its role within a larger set of active documents. Tracking the activity on a document (e.g., the type of activity, the amount of activity) can, in its simplest form, be used to flag documents in existing folder views; we can add read/unread and edited/unedited flags. As we develop better usage classifications of documents we can provide the option to sort on usage type of document.

Other usage applications

By leveraging the relationships between documents, we can extend text search beyond just what is in the focused window in order to infer additional topics that might be of interest and to target ads accordingly. This combination of content and usage monitoring is similar to the search scenario mentioned above.

The area of knowledge management and leveraging the "tribal" knowledge around workflow currently relies on manual documentation of process and presents itself as another area that could benefit from automatically gathered usage information. The idea is that through tracking the actions a user takes, a basic workflow template can be created.

With a robust mechanism for tracking usage we can extend the scenarios beyond document management into processes. Perhaps the scenario is rather futuristic, but assume for a moment that our applications were componentized in a way that, when using certain features within an application, the user would establish paths or patterns of usage. These patterns can be used to recompose applications and populate them with functionality based on the usage patterns.

CONCLUSIONS

In this project we saw how a simple proof of concept prototyping revealed some technical challenges (the difficult of matching file system events with window events), while also allowing us to get a first sense of the usefulness of the approach (users' immediate grasp of the concept, confirmation of benefit assumptions).

The technical challenge to obtain a reliable and non-invasive stream of user interaction events needs be addressed. While waiting for the desktop operating systems to accommodate our needs, we also suggest turning to a promising alternative in the web application and services: with a rapidly increasing range of web-based application services (such as various Live services, Groove, Google apps, Salesforce.com) we may some day no longer need to bother about the desktops.

Looking ahead we see activity based linking as an interesting additional source of data being available for numerous improvements in both functionality and user experience.

ACKNOWLEDGMENT

This research was carried out as part of a project in the Office of the CTO, Microsoft. The authors also thank Jeanine Spence for unwavering support and contributions.

REFERENCES

1. Adar, E., Kargar, D. and Stein, L.A., Haystack: Per-User Information Environments. In Proc. CIKM'99, (1999), 413-422.

2. Brin, S., and Page, L., The Anatomy of a Large-Scale Hypertextual Web Search Engine. In Computer Networks 30(1-7): 107-117 (1998)

3. Dragunov, A.N., Dietterich, T.G., Johnsrude, K., McLaughlin, M., Li, L. and Herlocker, J., TaskTracer: A Desktop Environment to Support Multi-tasking Knowledge Workers. In Proc. IUI'05. ACM Press 2005, 75-82.

4. Dumais, S.T., Cutrell, E., Cadiz, J., Jancke, G., Sarin, R. and Robbins, D.C., Stuff I've Seen: A System for Personal Information Retrieval and Re-Use. In Proc. SIGIR'03, (2003), 72-79.

5. Horvitz, E., J. Breese, D. Heckerman, D. Hovel, K. Rommelse, The Lumière Project: Bayesian User Modeling for Inferring the Goals and Needs of Software Users. In Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence. Morgan Kaufmann, 1998.

6. Millen, D.R., M.J. Muller, Werner Geyer, Eric Wilcox, Beth Brownholtz. Understanding users and usage patterns: Patterns of media use in an activity-centric collaborative environment. In Proc. CHI 2005 ACM Press, 2005.

7. Oard, D. W., and Kim, J., Modeling Information Content Using Observable Behavior. In Proceedings of the 64 Annual Meeting of the American Society for Information Science and Technology, USA, 2001.