

Implicit clustering of Web surfers with clickstreams

Scott Hendrickson, PhD
scott@drskippy.net
DRAFT

August 21, 2011

1. Introduction

This primer describes how to combine a Web surfer's succession of URLs (clickstream) with other users' clickstreams to identify similarities in user interest and to identify emerging topics of interest in real time with an Implicit Cluster Engine (ICE).

A hypothetical system is described here, along with some references to a prototype system implementing some of the ideas presented here. I give an overview of how users interact with ICE, what ICE does during user interactions and some explanation of how ICE creates a shared world for users to interact with people and content. This allows ICE to make real-time recommendations of relevant content.

Finally, I describe how ICE can be used as a search ranking tool based on emerging active URLs and the topics of the content found at those URLs.

Much of this work was conducted at and sponsored by Me.dium, Inc. which later became OneRiot, Inc. The intellectual property created while I was an employee at Me.dium, Inc. and that is described here is covered in patents applied for by that company. Some extensions and refinements suggested here are my own.

1.1. What is ICE trying to do?

ICE will organize information and people around their current Web browsing task in real time.

ICE can enable users to interact with one another in a task-specific context to find information, friends (social graph) and people with topic affinities (subject experts) and to explore the Web together. By combining real-time discovery and interaction among relevant people and "things," ICE allows users to interact with each other and with information in a manner that has many of the benefits of the way they interact in the real world, but with the advantages of speed, flexibility and lower cost of interaction in the online world.

The vision of ICE is accomplished by:

- Combining the attention streams of Web surfers.

- Determining the importance and relevance by evaluating a user's current activity and comparing it to other users' activities.
- Providing shared-world "physics" of interaction.
- Delivering browsing recommendations and identifying other Web surfers users who are interacting with similar intent.
- Ranking search results.

1.2. ICE System

A diagram of the system is shown in Figure 1. The key components of the system are user interactions (applications and sensors); "Match", the home of the Implicit Cluster Engine; and repositories (depicted as cylinders).

1.3. ICE provides three kinds of user value

ICE makes use of information available on the Web by:

- Recommending various sources of timely, relevant content in the form of *directed-surfer* implicit cluster exploration and explicit search results, with relevance ordered by the ICE recommendation algorithms.
- Providing discovery of very recent information (seconds) based on the real-time movement of other users—emerging hot topics.
- Socializing the browsing experience. (One product experiment created by Me.dium, Inc., was chat-plus-location for the use case of surf with friends—but users weren't buying.) In the future, ICE may provide evolution to a richer socialized browsing experience that will allow finding new friends based on affiliations or recommendations, online events, 1:1 online sales opportunities (OneRiot's current direction), etc. and search ranking.

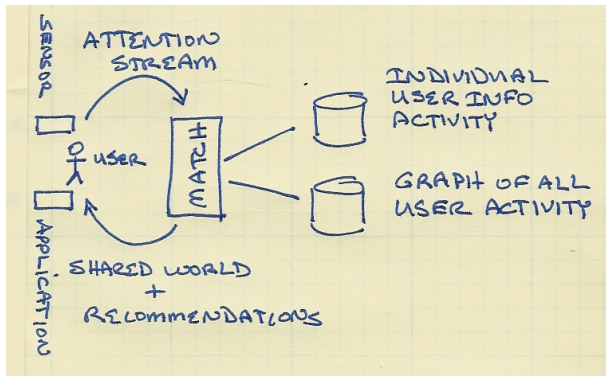


Figure 1: ICE provides user value by creating a shared world for user interactions and by making recommendations of relevant content in real time. Sensors provide attention stream data to the ICE servers. An individual user’s social graph, click stream, etc. are stored in a variety of repositories along with aggregate user activity in the form of resources (graph nodes) and correlations (graph edges). Aggregated data is used to create recommendations of relevant content. Additional applications may provide a wide variety of views into the shared world and presentation of recommendations.

1.4. ICE: Shared World and Recommendations

The core technology is the “Implicit Clustering Engine” (ICE). Users interacting with ICE realize value through two aspects: the shared world and recommendations.

Why make a distinction between the shared world and the recommendations? The shared world is the context and substrate for interactions. The shared world is what users interact with. The shared world gives the sense of consistency, the sense of place. It is the essence of the Web surfers’s location and context.

Recommendations, in contrast, provide pointers to content based on tasks, movements and interactions within the shared world. This situation is analogous to the function of a topographical map—it provides contextual clues, shows the general lay of the land and gives the user multiple points of alignment. But a topographical map does not recommend that you climb this mountain or avoid that dangerous canyon. We look to common sense, past experience, guide-books, friends and previous travelers for recommendations.

First, the shared world: In the topographical map, our interactions with the map are consistent and predictable. The map “physics” don’t change from moment-to-moment or person-to-person. The physics of the system is the consistent behavior that provides a concrete sense of shared world for the user and grounds the user for rich higher-level interactions with content, other users, and with ICE recommen-

dations.

The shared world embodies physics of:

- Location—where a person is located on the Web. Most users use the language of “location” for a Web site or a specific page, for example, people say “Where are you now? I want to see what you are seeing.” When we are browsing the Web, we respond with a URL, saying “This is where I am.” Additional properties of location that build on this basic unit are discussed below
- Proximity—What is near to what? Topographical maps provide context and scale because we know what the symbols mean, we can use the scale of the map to measure distance and we understand the concept of contour lines. More generally, proximity of interesting things in the map with a low enough acquisition cost lets me get there quickly and cheaply.
- Interaction—What artifacts can I leave behind in the shared world? Who can find them? And how can someone finding them interact with them? ICE uses clickstreams (breadcrumbs showing a path through the forest) and simple messaging between users. In the future, we can image tagging, graffiti, vanity markers (e.g. t-shirts indicating fan status, or affiliation), etc.
- Synchrony and asynchrony—Experience is real time. In order for users to have a rich experience, ICE must provide a real time experience of the shared world. However, asynchrony is also important of the shared world. Leaving a message (whether a URL you think they might enjoy, an IM, graffiti or an Easter egg) today for someone to receive in 5 min or tomorrow or next week enriches user experience.

ICE is the technology from which unique content and people recommendation capabilities, as well as, a search ranking emerges. There are many sources of attention stream data (real time Sensors) and many ways to enable user interaction through web applications (Applications). The shared world completes the loop and enables the to generate user value. In the following, ICE’s content and people recommendation and search capabilities will be referred to as “ICE Recommendations.”

1.5. What is in the rest of the document?

In the following, I describe the algorithmic technology of the ICE Engine, both the software and infrastructure technology of a hypothetical implementation of ICE. This description will point to potential future technology directions.

1.5.1. Algorithms

ICE relies on statistical analysis of user activity, weighting more frequent associations of activities as more significant than less frequent associations. This strategy has two important attributes:

1. It is capable of providing real-time recommendations based on the most recent data in the system; and
2. It favorably leverages the idea of Directed Surfers (DS), i.e. actual people, to explore relevant and timely content. (Contrast the DS with Web crawling Robots that are “obsessed” with completeness and have comparatively rudimentary heuristics for determining the importance of one link over another.)

ICE can create real-time recommendations because it does not perform global cluster analysis. Instead, it allows the DS to efficiently explore implicit clusters, i.e., clusters of related resources for which explicit cluster membership has not been explicitly identified by ICE.

Additionally, every exploration by the DS enhances the correlations of URLs within an implicit cluster. The result is a real-time clustering solution that quickly delivers the rich and relevant content while sidestepping the huge effort of global cluster analysis.

A further advantage is found in how the fuzzy nature of clusters is handled (in both time and proximity). With explicit clustering algorithms it is very difficult to deal with members falling in two clusters. With Implicit Cluster Exploration (ICX), DSs create connections with ambiguous URLs and very naturally choose the relations most highly valued based on the task they are currently doing. More on this later on.

1.5.2. Technology Implementation

Implementation ideas here are mainly drawn from the prototype system build by Me.dium, Inc. Not all of the algorithms and ideas outlined in this paper were implemented in the prototype.

The primary drivers of technology innovation are:

1. Response time (must be $< \frac{1}{2}$ second in order to be perceived as real time by a user). This is an interaction requirement, not an algorithm requirement. It does not mean, for example that everything ICE does must be done in less than half a second. This means that basic physics of the system must be embodied nearly instantly. Imagine a system that provides recommendations etc. on content up to, for example, 2 weeks past peak relevance. In this case, algorithms requiring computation times from $\frac{1}{2}$ s to 2 weeks are appropriate.
2. Attention Data. A reasonable target is for ICE address about 2 weeks worth of attention data for the entire

user base. This Content Time Horizon (T_c) is a consequence of from a steady state flow of incoming attention data (e.g. tens of thousands of users might share about 100s URLs per second as well as other attention data), the decay rate of less relevant data, and the Cache Size (S_c).

2. Pieces of ICE

2.1. The Web Surfer’s System

Sensors—any input of user attention data. ICE accepts attention data only from live sensors, i.e. sensors providing real time input of attention data through Web surfers’s proprietary API. Alternately, one could provide an open API, as well as the ability to leverage batch attention stream data.

Shared World—Provided by ICE. This is also the context for users to get recommendations.

Applications—Applications provide access to the experience of the Shared World and recommendations. Web surfers may use a Sidebar (both an Application and Sensor), have limited interactions through a Tool Bar (primarily a sensor) and get limited access to social graph information through the website. It is not hard to imagine a proliferation of applications to interact with the shared world. Here are some examples:

- Search from a toolbar
- Corporate intranet emerging topics application
- Applications for showing what sites are hot now
- Applications for following in real time events such as elections or breaking news stories
- Desintation search site
- Targeted advertising
- Emerging topics from Twitter for Facebook streams
- ...

2.2. ICE Core

The ICE core is comprised of 7 of logical components:

- A. Resource Correlation Scoring Algorithms (RCSA) do things like accumulate information and calculate correlations from user actions and enables Implicit Cluster Exploration (ICX). Other ICE Recommendation Algorithms could use other graphs in addition to URL traversal—Friendships, Alexa results etc. to supplement RCSA recommendations. Many signals useful for the applications above will leverage accumulated data on each graph.

- B. RCSA Cache. Responsible for retention, orderly decay, and real-time access to aggregate attention stream data for recommendations.
- C. Shared World Model. Including messaging, friendships, shared-world artifacts like place markers or activity flags, current user locations, etc.
- D. User Attention Record. User clickstream history, preferences, affiliations, etc. This is the mechanism a user uses to filter, block or emphasis content, friendships, etc.
- E. Tuning Console allows various elements of the RCSA to be tuned.
- F. Content Indexer. URL page content indexer and index storage. For example, Lucene.
- G. ICE Mixer. Allows weighting and mixing of results from ICE Recommendation Algorithms into a final recommendation result set for presentation to user via Applications (sidebar, widget, website, etc.). (This functionality may be implemented as Publishers.)

2.3. ICE Lingo

This section provides definitions for many ideas that can be modelled in the ICE Core.

2.3.1. ICE Interactions

Term	Description
Actor	User, sensor presence, node of social graph
Community	Logical groups of actors
Neighborhood	Logical groups of URLs
Task	Sequence of events
URL	Web location
Topic	Chat associated with a URL
Conversation	Chat between actors

2.3.2. Shared World and Recommendations

Term	Description
User Tail	Reverse time ordered list of most recently visited Locations (URLs)
Graph	Model structure of Nodes and Edges
Resources	ICE has four types of Graph Nodes: Users, Locations, Topics, Domains
Correlations	Numerical weights representing the edge strength between Resources. Typically, edge weights increase based on users traversing the nodes connected by the edges and decreases with time.
Resource Cluster	Groups of correlated resources, connected by edges of non-zero weight.
Friendship	Explicit connections between users.
Social Graph	Structure of Friendship relations. Users are nodes, edges are Friendship connections.
ICE Set	Result of the Resource Correlation Scoring Algorithm.
Result Set	Shared world and recommendation data sent to Applications.
Publisher	Transforms and filters ICE Set to create a Result Set.
Map (E.g. Application)	Display of User at a Location, may display Locations of Friends, shows movement between Locations.
ICE Knobs	Set the controllable parameters of the ICE Mixer.
Share Location (E.g. Event)	A specific (one of many) event processed by the ICE Server. Indicates a user navigated to a new URL and the Reason. (E.g. user clicked on recommended site in the Map, user clicked on URL recommended in chat by a friend, user typed URL in to the browser navigation bar, etc.).

2.3.3. User Attention Stream

Term	Description
Clickstream	For a user, the list of URL and timestamp of arrival at that URL for a session of navigations, may contain additional descriptors (see text).
Page Dwell	The time a user spends on a URL. (Technically, this is the time between URL ShareLocation events.)

2.3.4. Search

Term	Description
Index	Database of terms and pointer to where the terms are located
Indexer	Builds the Index from content found URLs
Index Results	List of URLs produced from the index, ranking bases only on Index content (frequency of terms, relative frequency of key terms, etc)
ICE Ranked Index Results	Re-ordering of Index Results based on the RCSA
Query Processor	Prepares user-input terms for a search of the index. May include spell checking, adding synonyms to the term list, interpretation based on context (e.g. Apple Pie vs. Apple iPad) etc.

2.4. User Activity, Resources and the ICE Algorithm

ICE collects and aggregates User Activity (in general, Attention Streams) as Resources, then uses the resources to make Result Sets available. The steps to create a Result Set are as follows:

- I. User takes an action recorded by a live sensor. Sensor transmits activities to the ICE Server.
- II. Filter unwanted actions. This step may involve ignoring URLs for editorial reasons (e.g. pornography) or ignoring URL for algorithmic reasons (e.g. dynamic URLs not reachable by a second user) or statistical reasons (this URL appears for the first time, so delay processing).
- III. Create or Update Resources. In this step, ICE activities include calculating resource correlations,

presence, tail, etc information. The elements of resources and some of the update concepts are discussed in more detail below. This step updates the state of the shared world and creates the data structures in the RCSA Cache.

If the Sensor is also an Application:

- IV. Process the Resource Correlation Scoring Algorithm to create a ICE Set. This is done by traversing the graph of correlated Resources. The RCSA algorithm is described in some detail below. This step involves possibly many reads from the RCSA Cache (to retrieve node and edge information).
- V. Supplement ICE Set, when required (E.g. Alexa).
- VI. Process the ICE Set according to the Mixer Settings for the Application. Transmit to the Application.

2.5. User Activity and Resources

In step III. above, Resources are created and updated. These resources are designed to support the process of make a match using the RCSA.

Activity	Resource Record Includes
User	Location of User Topics Created by the User User's Tail Presence
Location Data	Current visitors Common or frequent visitors (arrival, page dwell) Correlations to other Resources
Topic	Visitors Posters Correlations to other Resources
Domain	Visitors Common or frequent visitors Correlations to Resources Editorial Data?

2.6. Clickstreams

Clickstream data describes a user session. Clickstream data is also available from ICE for offline processing. A clickstream data entry includes [userid, sessionid, contextid, timestamp, URL, Domain, browser type, sensor type]. This data is useful in real time for statistical and pattern analysis of the data set.

Possible valuable offline analyses:

- Drive improvements to the RCSA
- Task identification
- User behavior patterns
- Explicit clustering

This data can also be repurposed for sale to 3rd party data consumers.

2.7. Tuning ICE

ICE “Knobs” control the mix of ICE Sets and supplemental Data

ICE Knob	Determines relative weighting
	Tuning RCSA:
Task	Resource correlations (primarily driven by user activities and how they are correlated with the activities of other users.
Domain	Domain Resource correlations
	Tuning shared world:
Hot Popular	URLs with the most users now URLs the most users recently visited. I.e. the most visited sites in the RCSA Cache.
Moving	URLs with the greatest rate of increasing number of visitors
Pinned	Seeded Locations, pre-determined Locations
Friends	URLS where Friends are Located Tuning Supplemental:
Third Party	E.g. Alexa, other sources of related URLs

3. Implicit cluster exploration

How does ICE work?

The objective of this section is to motivate and explain a design of RCSA. On one hand, there is the requirement of real time interaction. On the other, is the desire to find related Resources based on User activity.

The literature on clustering describes many possible clustering algorithms for determining the relative importance of edges in the graph (the global structure of clusters of nodes) and possibly establishing cluster membership (useful for making recommendations of related sites). These are generally impractical for real time interaction.

In contrast the idea behind ICE, consider a famous example, Google’s Page Rank algorithm. Rather than directly determine explicit members of clusters, Page Rank finds potential important cluster hubs.

Page Rank is a calculation of a relative score of every URL (global!) based on the link structure of the Web. Considering the scale of this global calculation, it is a remarkable technological achievement. The calculation can be partitioned to millions of machines and completed in a relatively short time. This is because Page Rank algorithm is iterative, proved to adequately converge in 10s of iterations and requires very infrequent inter-machine communication.

To determine potentially important cluster hubs, Google analyses the link structure of the Web by crawling it, recording the link structure, and indexing the content.

Sometime later, users explore the structure of the Web by entering a query to define a topic, and viewing a ranked list of sites. Google actually uses many sets of rules to rank sites. For simplicity, assume Google uses only term frequency and Page Rank. In this extremely simplified toy case, the ranked list is determined by a combination of the term frequency score and the Page Rank of all the sites containing the search term.

In this simplified Google toy case, all the URLs displayed are members in this explicit cluster with the most important hub nodes at the top.

Other explicit clustering techniques would use some description of edge and node structure to find related sites and establish their cluster membership. While these techniques may provide interesting recommendations of related sites, explicit clustering and other global ranking schemes are not viable for making real time recommendations on data from real time sensors because the global calculations on millions of related Resources cannot be performed in 1 s.

RCSA implements a new strategy for clustering related URLs. (RCSA actually clusters related Resources, but to simplify the explanation, we will only deal with URL resources for the remainder of the section.)

RCSA leverages the Directed Surfer (DS). ICE is based on a fundamentally different Web topology from link structure. In contrast with traditional search engine’s use of robots to determine the link structure of the web, RCSA relies on the browsing activity of a human, a Directed Surfer, to establish the edge structure of related nodes. Therefore, ICE is based on a fundamentally different Web topology from link structure.

While Google’s algorithm depends on the concept of a random surfer exploring hyperlinks on Web pages, ICE relies on task-directed surfers—very different than mathematically random surfers—to choose related links and explore unlinked, but related sites. Directed Surfers find unlinked-but-related sites because they are motivated to find specific answers, solutions to specific problems or perform specific tasks such as shopping, bill paying, reading the news, etc.

The Directed Surfer provides two benefits.

First, the directed surfer uses intelligence to crawl the Web. The DS follows links because they can be judged by a

human to be related or potentially interesting. Contrast this with a robot that is plodding away for completeness or based on very simplistic heuristics. The DS creates a node-edge structure based on goals, tasks and human intelligence.

Secondly, the requirement to display a complete list of cluster nodes as a result or recommendation set is relieved because a DS can rapidly explore an explicit cluster with the right tools. Presenting only part of an implicit cluster to a DS allows ICE to work in real time!

Regarding a user’s desire to find clusters of relevant URLs, ICE (specifically, the RCSA) can be thought of as an implicit cluster exploration (ICX) tool.

3.1. ICX Toy Example

A URL correlation between site A and site B is created when a user navigates (either by clicking a link, typing in a URL, copying a URL from a chat client, etc.) from A to B.

If a user then navigates to URL C, URLs B and C become correlated. In the RCSA, A and C become correlated as well, although slightly weaker, because of the additional space between them. If another user navigates from URL B to URL C, the correlation B–C is strengthened.

For a given URL, the Resource record in the RCSA Cache contains correlations to a pre-determined number of other URLs. The correlation scores are a combination of the proximity and number of clickstreams that contain the URL combination.

In this simplified model, the RCSA consists of using a matrix of URL correlations to lookup related sites. Consider a system in which 3 users (Anna, Bob, Cathy) visited the 15 URLs {A, B, C, D, E, F, G, H, I, J, K, L, M, N, O}.

Here are the toy clickstreams for each user:

User	Click Stream
Anna	A–B–C–D–L–M–N (marked red in Figure 2)
Bob	E–F–G–M–O (marked blue in Figure 2)
Cathy	H–I–J–K–L–M (marked green in Figure 2)

The correlation matrix is shown in Figure 2. It is a 15x15 matrix showing a correlation score between every combination of URLs appearing in our three user clickstreams. For example, the correlation between URL A and URL B is 1 because the A–B transition occurs in only Anna’s clickstream. Because the L–M transition appears in both Anna’s and Cathy’s click streams, it has correlation score of 2. The diagonal elements are self-correlations and have no meaning in the RCSA.

Making one more simplifying assumption, consider that for many URLs, the order in which they are visited is by chance, not because there is a progression inherent in the

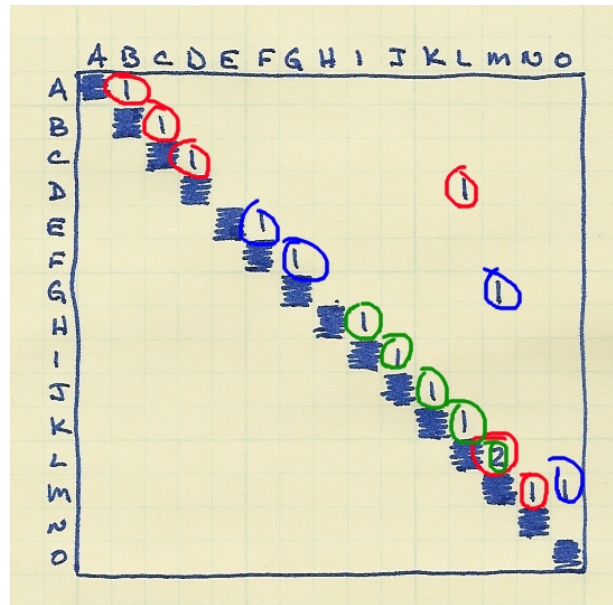


Figure 2: Three user clickstreams create the URL correlation matrix. The matrix shows correlation scores for only the direction the pages were navigated by the users.

content. For this example, let us assume that a forward correlation implies an equally important backward correlation. This has the effect of making the correlation matrix symmetric about the diagonal as shown in Figure 4.

Figure 5 shows the URL correlation matrix for the simple toy problem.

3.2. Recommending URLs

How does ICE use RCSA to recommend URLs?

Imagine a 4th user, Doug, starts by visiting URL L. L becomes the most recent URL in Doug’s Tail. The RCSA is run by finding L on the left side of the matrix and looking up all the URLs across the top that have a non-zero correlation with L.

Looking at Figure 6, the RCSA will recommend {M, D, K} with M at the top of the list because its correlation score, 2, is the highest.

Next, imagine Doug chooses to follow one of ICE’s recommendations and navigate to URL D. Doug’s Tail becomes [L, D] and Doug’s newly recommended set is based on both URLs in his tail: {M, C, K}. C has been added to the result set because transition D–C has correlation strength of 1. D and L are not in the recommended sites because Doug is at D and just came from L). M is still at the top of the list because of the higher correlation score.

The RCSA decreases the weights the contributions from URLs down the Tail so that sites Doug visits in the future

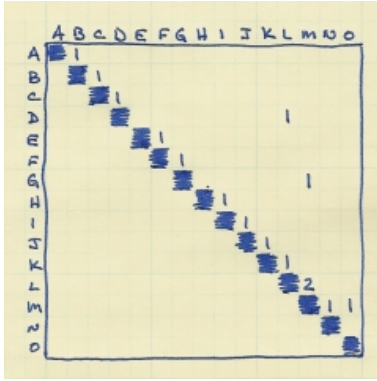


Figure 3: Three user clickstreams create a URL correlation matrix. The matrix shows correlation scores only in the direction the pages were navigated by the users.

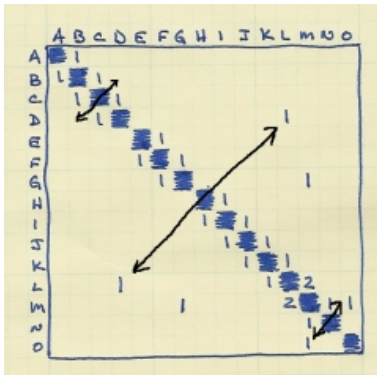


Figure 4: The URL correlation matrix is symmetric about the diagonal under the assumption that backward correlations have the same value as forward correlations.

are eventually not dominated by the high correlation of transition L–M.

Note too, that in implementation of RCSA, in contrast to this simplified example, many other resources are being correlated and processed to provide a results set. The final results are the mixture controlled by the ICE Knobs.

3.3. Exploring an implicit cluster

Who does Doug explore an implicit cluster?

Refer again to Figure 5. In the toy example it is possible to navigate to any of the URLs from any of other URLs by visiting recommended links. In Doug’s case, he merely has to visit every recommended URL and this will lead him to all sites in the system because they are all correlated.

In RCSA, this is not always the case. In fact, there can be many islands of URLs because the common sites that everyone visits (the sites that would bridge between islands of

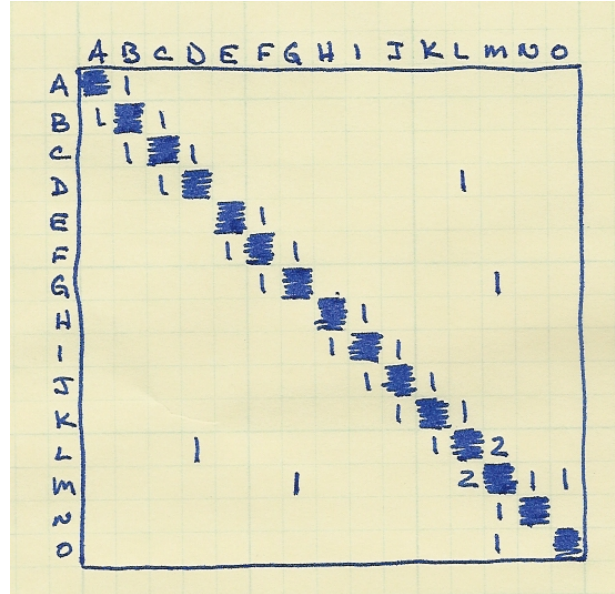


Figure 5: Symmetric matrix created using the process illustrated in 4.

connected URLs) appear too frequently to give meaningful correlations. That is, they are correlated with everything. Because of this, we ignore these (E.g. www.google.com).

A second reason Doug many not be able to navigate to all URLs based on recommendations is that the final Recommendation Set is always a subset (typically 5-20 recommended URLs) of the ICE Set (possibly hundreds of URLs) due to correlations score ranking, Friends, Pinning, Hot, etc. results occupying the available positions for display in the Application.

However, Doug can still navigate highly correlated sites and visit many of the central hubs of the clusters by navigating the first few recommendations. Additionally, once Doug performs his navigations, his clickstream has updated the matrix, further strengthening the connections between the sites he chooses to explore.

4. Search

From the toy example above, it is easy to see how to explore correlated URLs given that you arrive at a URL within the implicit cluster. However, users also find it useful (and are accustomed) to starting a search by declaring the topic of a cluster rather than trying to start at a related URLs.

To support this behavior, implementing a search capability so that a user can enter some terms and narrow the result set to URLs containing or very closely related to the terms entered. Adding this functionality is a natural extension of ICE and provides a direct means of experiencing content

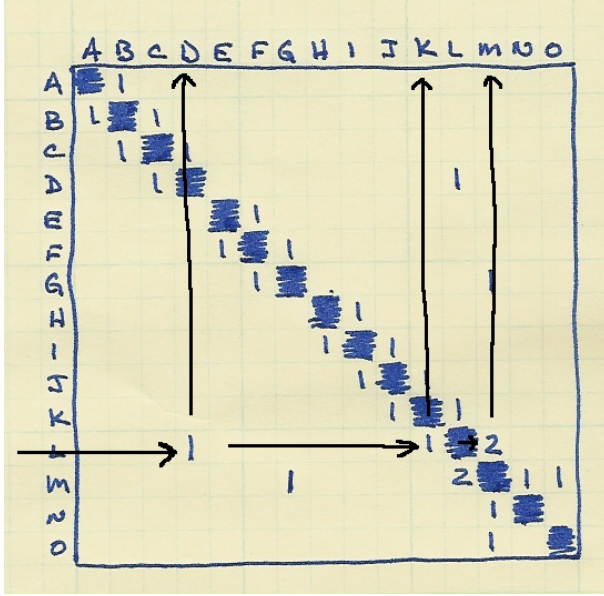


Figure 6: ICE Set based on user Doug visiting URL L. ICE recommends M, K and D as relevantly related URLs. URL M is at the top of the list because of its greater correlation score.

based on intent. Search is a familiar mode of interaction for users, so may be an effective way to introduce users to the shared world and recommendations created by ICE.

4.1. ICE-Ranked Index Results

A user interacting with the shared world creates a set of signals that can be used to rank-order search results. An indexer creates the index of URL content as they appear in user clickstreams. Contrast this with robot crawling: a DS “crawls” only the URLs relevant to their current task.

Since the ICE Core is modeling URL resources, a useful extension to the model is a content index. With this index, a user wanting to start exploring implicit clusters of sites in the system enter search terms in a toolbar, sidebar or web application.

Ranked search results are created by this process:

- A. Index-ranked results are returned from the index.
- B. The ICE Engine refines and enhances the result ordering by,
 1. Re-weighting URLs using the RCSA. Note that this can both re- order URLs and add additional URLs that are related by the RCSA, but that don’t contain the original search term.
 2. Adding URLs based on Resource correlations in addition to URLs such as friends, groups, editorial, etc.

By simple extension of the ICE Core scoring algorithms, we can create a search engine especially suited to emerging hot topics.

4.2. An ICE-Inspired URL Ranking Signal

There are some natural signals to add to the search ranking algorithms closely related to the ICE ideas outlined above.

Here, I propose signals that may enhance the determination of very active URLs or groups of related URLs. This section contains work not included in the Me.dium, Inc. patents or included in their prototype. None of the signals in this section have been proven to enhance the ICE Core.

Some motivations:

- Simple input data requirements
- “Local” calculations—don’t have to solve global clustering problems
- Normalized signals
- Orthogonal signals (don’t mix too many measures in to any one signal)
- Focus on signals for each of two ideas: (1) signals for “Hottness” (both freshness and emerging interest); (2) signals for validating activity (relative interest)

4.2.1. Correlation Data and Signals

- List of correlation tuples

$$tuple = \begin{pmatrix} URL, \\ timestamp_{last\ traversal}, \\ C(float), \\ type\{TO, FROM, BOTH\} \end{pmatrix} \quad (1)$$

where C is the correlation coefficient.

- The number of tuples kept is currently limited to 16, but could be reset as needed.
- The signal used in the measure of heat is,

$$S = \frac{Count(TO, BOTH)}{20} \text{ with } 0 \leq S \leq 0.8 \quad (2)$$

- Also available in the ICE cache, the correlation coefficient is calculated by

$$C' = f(C, t) + \delta(t_{dwell}) \quad (3)$$

where $f(C, t)$ is a time decay function that scales like $exp(-t/t_0)$ for large t and t_0 is chosen to decay $C \rightarrow 0$ for unvisited URLs over approximately 2 weeks.

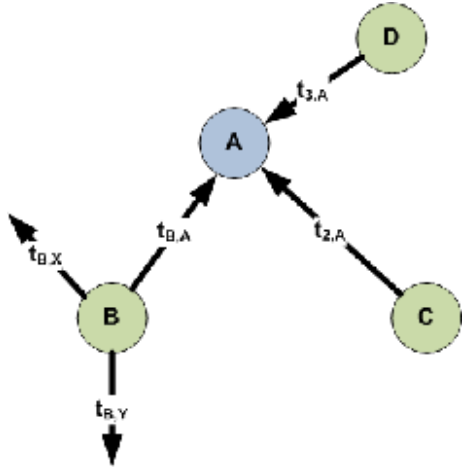
4.2.2. Additional Data Elements

These are additional in the sense that they are not in the ICE Cache as described above.

New data elements to add to the location object (LAD) are outlined below. Each data element has the quality that it that the source and destination URLs of a user traversal are currently managed by the *ICE* session and can be updated without changing the underlying interactions of ICE and the Cache.

No time decay of data is proposed. My thinking is that decay from the system should be controlled by (1) falling user activity and (2) desired overall decay profile based on staleness. These signals are intended to focus validity of emerging content.

If the current system depends very explicitly on the signal Eq. (2), we may have to use it in conjunction with these signals until a decay signal is developed.



- In-bound degree of node URL. This is a single (*int*). Signals based on this metrics are discussed below.
- List of tuples representing visitors traversing *out* of the node,

$$tuple_{trav} = (URL, t_{out}(int)) \quad (4)$$

where t_{out} is the number of surfers traversing outward to “destination URL”.

The maximum list size, $N_{out} = \max(n_{out})$, in the list is configurable.

The list of tuples will grow to where $n_{out} < d_{out}$: Because there is no time decay of the list, a random-removal queuing scheme will be required to allow new, hotter destinations to emerge when in the list.

- List of tuples representing the Relevance Contribution for the node

$$tuple_{cont} = (URL, \alpha(float)) \quad (5)$$

where α is the relative contribution weight or score for a path into the URL based on the choices of the surfer.

- Calculating α . For nodes (URLs) *A* and *B*, consider the case where users traverse the path $B \rightarrow A$. According to above, the node *B* has a list of all traversals *out*, $t_{B,i}$ of the node, including the value $t_{B,A}$. The list of traversals is the out-bound degree of the node, $d_{out,B}$.

The total visitors leaving node *B* is,

$$v_B \equiv v_{out,B} = \sum_i^{d_{out,B}} t_{B,i} \quad (6)$$

(Referring to the example depicted in the picture, the sum is over $i = A, X, Y$.)

α_A is the fraction of surfers leaving *B* via path $B \rightarrow A$ relative to the total number of traversals from *B* by any path, over the probability of choosing *B* via path $B \rightarrow A$ at random from the paths all of the surfers leaving *B*,

$$\begin{aligned} \alpha_A &= \frac{\frac{t_{B,A}}{v_B}}{\frac{1}{d_{out,B}}} \\ &= \frac{d_{out,B} t_{B,A}}{v_B} \end{aligned} \quad (7)$$

$$\text{with } 0 \leq \alpha < d_{out,B}$$

If the tuple list is truncated at N_{out} , substitute the $d_{out} \approx N_{out}$ in Eq. (7). In general,

$$\alpha_A \quad \begin{cases} < 1 \text{ less likely than random} \\ \approx 1 \text{ like random} \\ > 1 \text{ more likely than random} \end{cases} \quad (8)$$

These data support multiple signals. It would be useful to begin capturing this data while I determine which signals (some proposed below) can make the most valuable contribution to the measure of hot sites.

4.2.3. Signals

The basic idea is to use α 's and node degrees (d_{in}, d_{out}) to weight the relative validity of traffic on a URL.

Relative Contribution Weight. For node *A*, determine signals from list of $\{\alpha_{A,i}\}$,

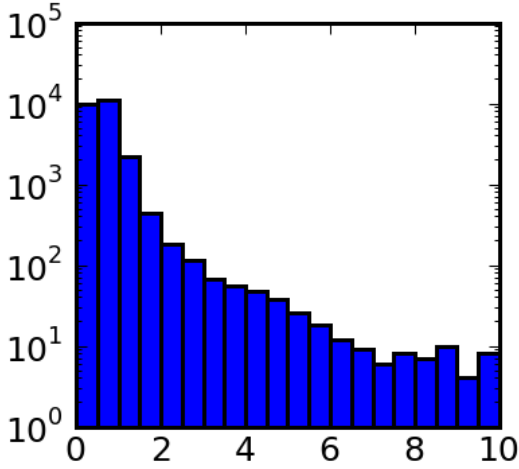


Figure 7: Log-scale histogram of α parameter.

$$\alpha_A = \sum_i^{d_{in}} x_i \begin{cases} x_i = 0 & \text{for } \alpha_{A,i} < 1 \\ x_i = \alpha_{A,i} - 1 & \text{for } \alpha_{A,i} > 1 \end{cases}$$

$$S_1 = \frac{2}{1 + \frac{1}{\alpha_A}} - 1 \quad (9)$$

with $-1 < S_1 < 1$
and $d_0 = \text{max degree (max length of list)}$

Alternatively, use other common aggregation methods such as $\langle \alpha_{A,i} \rangle_{avg}$, $\langle \alpha_{A,i} \rangle_{max}$, or $Count(\alpha_{A,i})$ where $\alpha_{A,i} > 1$, etc. for the heat signal, S_1 .

Figure 7 is a log-scale histogram of α sampled from 1,000 randomly chosen URLs from the cache. Because $\alpha = 1$ represents probability that is the same as random choice, values below 1 are contributing a detracting score while $\alpha > 1$ represents that this path has a better than random chance of being chosen by users. The plot below shows values up to $10x$ as probable as random occurred approximately 10 times over 1,000 samples.

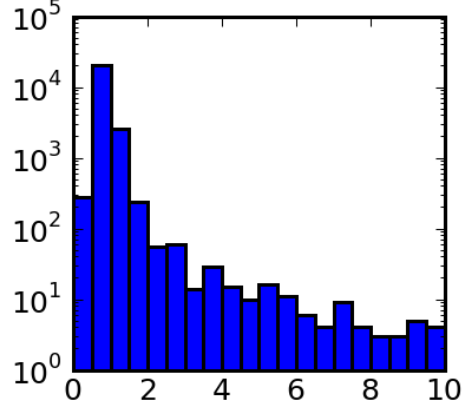
To simplify the calculation, the statistics presented here were calculated by choosing a URL (B) and calculating the $\alpha_{B \rightarrow j}$ s for all of the outgoing links from B . Therefore, the data above shows the special property

$$\text{Avg}_j \alpha_B = \sum_j^{d_{out}} \alpha_j = 1 \quad (10)$$

While this is true across the entire data set, a random subset of the URLs for which we (alternately) calculate the

(input) signal $\alpha_{A \leftarrow j}$ from all the j s that lead to URL A will not have this property.

A variation of this signal would be to count only unique-user path traversals. The figure below shows the distribution of α when unique-user path traversals instead of all traversals.



This is an improvement as there are a few sets of URLs with many traversals between them, all made by the same user. This gives the “fatness” around 4-5 in Fig. 7. It appears that maintaining a short list of users traversing a path will suffice to filter this unwanted behavior. Unfortunately, a list would have to be maintained for each path from a URL (the number of lists is equal to d_{out}). I am working to quantify the minimum list size required.

In-Degree. Use the number of incoming paths surfers traversed as a measure of overall connectedness. One challenge of this signal is that paths grow with visitors (more people explore more paths) and with time (given more time, visitors find more paths). “Hot” items may have rapidly changing visitation patterns and be the top choice from the source URL(s) while having only $d_{in} \approx 1$. This is a useful validity indicator for the medium timescale in the life of an emerging “Hot” URL.

Determined from existing data,

$$d_{in,A} = Count(\alpha_{A,i})$$

with $0 \leq d_{in,A} < \infty$

Proposed signal,

$$S_2 = \frac{2}{1 + \frac{1}{d_{in,A}}} - 1 \quad (11)$$

with $0 < S_2 < 1$
and $d_{in,A} > 0$

Fraction of paths In-Degree. Use the ratio of incoming to outgoing paths surfers traversed as a measure of desirable

connectedness. More in than out denotes a session or task end point; more out than in indicate a jumping-off point.

Determined from existing data,

$$r \equiv \frac{d_{in,A}}{d_{in,A} + d_{out,A}} = \frac{Count(\alpha_{A,i})}{Count(\alpha_{A,i}) + d_{out,A}} \quad (12)$$

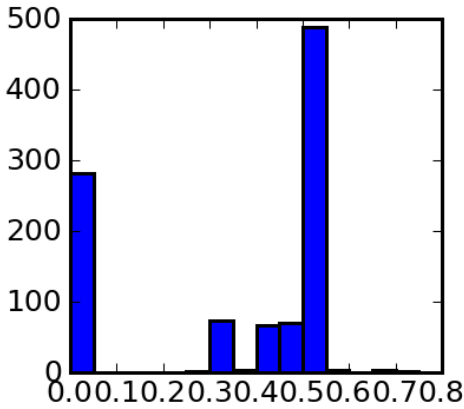
with $0 \leq d_{in,A} < 1$

Proposed signal,

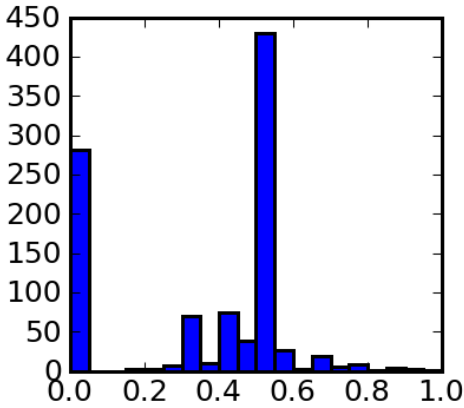
$$S_3 = 2r - 1 \quad (13)$$

$$-1 < S_3 < 1$$

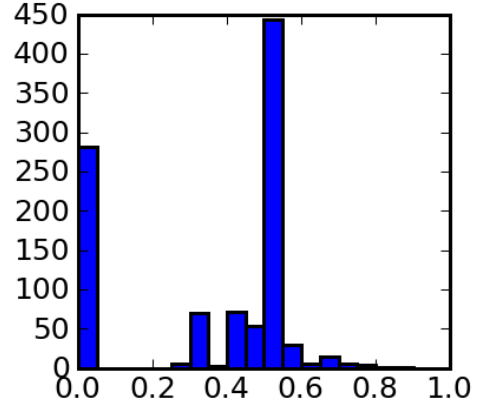
Below is a histogram of the fractional in-degree, Eq. 12 for 1,000 randomly chosen URLs from the cache. The majority of URLs (over 400 of 1,000). Interesting (and unexplained) are the large number 0



Fraction in-traversals



and fraction unique-person in-traversals



look somewhat similar and may provide alternative signals to fraction-in-degree.

Population-Normalized Degree Growth. As popular nodes become more popular, there is a pattern of the winner-take-all tendency of the node to accumulate more links inward as more visitors discover the URL. Use the growth degree of a node as an indication of advanced “Hotness”.

Determined from existing data,

$$g = \frac{d_{in,A}}{G(v_{in,A})}$$

with $0 \leq g < \infty$
and g undefined at $d_{in,A} = 0$

Use existing data to determine the average growth profile,

$$d_{avg} = G(v_{in,A})$$

Proposed signal,

$$S_4 = \frac{2}{1 + \frac{1}{g}} - 1 \quad (14)$$

with $-1 < S_4 < 1$

There is much work to do to verify that these signals have a useful dynamic range (i.e. all of the interesting behavior is not piled up in a small region of the domain of these signals), understand the cost of computing these signals across the entire cache, and finally, that the overall system can be tuned and trained to make use of any or all of these signals.

5. Implementation

Some of the ideas outlined above were implemented in a Medium, Inc. prototype. Some of what was learned is outlined here.

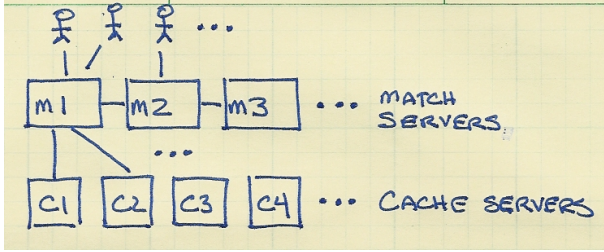


Figure 8: ICE “Match” Servers and ICE Cache Servers deployment architecture. Users interact with the ICE Servers through Sensors and Applications.

Referring back to section Logical Components of the ICE Engine above, it may make sense to start by implementing logical components A, G and parts of C (location, recent user attention stream) in the core ICE Server implementation. Messaging (chat) and social graph could be provided by a 3rd party chat server, e.g. XCP. The Web site applications touch this functionality where required and also maintain user identity and profile information. Widget (a portable, modular type of Application) functionality is interfaced through the Web applications and directly to the ICE Server.

ICE Servers manage the shared world and interface with the RCSA Cache. See Figure 8. For efficiency, each ICE Server has a local Cache corresponding to RCSA Resources for its shared world. User sessions are pinned to a ICE Server.

ICE Servers and RCSA Cache are configured as in Figure 8. The figure shows M1, M2, M3 up to N_m match servers and C1, C2, C3, C4 up to N_c cluster servers. The RCSA Cache is partitioned into hashed addressed caches of size $\frac{1}{N_c}$. Cache addressing as efficient as the hash address scheme and can easily be scaled from 1 to many servers.

The bottle neck in this architecture will occur when the communication overhead between the two layers (between M’s and C’s) dominates the total communication time. Estimates of when this will become a limiting constraint and potential solutions are discussed below.

5.1. Steady State Cache Dynamics

While the time to process RCSA depends on algorithm calculations (including the time required to access the RCSA Cache), the RCSA Cache size determines the scale of Resource correlations. This implies that the Cache size is the limiting factor for the number of implicit clusters or, more generally, the richness of content in the system.

As described above, the RCSA Cache contains four types of Resources: Location, User, Domain, and Topic.

Each of the four scales differently with the number of users. For example, User Resources scales with active users; the number of Topic Resources may be much smaller than the number of Active Users (these topics are emerging from the collective behavior of the users).

Regarding Location and Domain Resources, URLs enter the system continuously as Users browse the Web. Each unique URL requires a Location Resource in the Cache. When a URL is repeated, its Location Resource is updated.

For example (from the Me.ium, Inc. prototype), in 2 weeks in January, 2008, the system processed a volume of 45M URLs, 7.4M of these URLs were unique implying 7.4M Location Resource Cache objects created in the 2-week period. Many fewer Domain Resources are created for the same set of URLs: only 997K unique domains were associated with these URLs. Over the two week period, almost 9M Resource Objects were created in the RCSA Cache.

When the RCSA Cache has reached capacity, Resources are dropped from the Cache. This is actually done somewhat gracefully by decaying Resource correlations over time. The effect is that Resources fade away before they disappear. However, decaying resource correlations does not increase the Cache capacity.

Once the cache is full, equilibrium is maintained as the number of new URLs entering the system for a given time period equals the number of Location Resources dropped from the cache for that same time period.

The calculation of equilibrium is complicated by the fact that the distribution of URLs entering the system for a given population is a Long-Tail (scale-free, closely approximated by a Power Law) distribution. This means that the number of new URLs (never seen by the system before, or at least no longer in the cache) entering the system declines over time for a given population. For the data set cited above, the number of new URLs on day 15 was about 15% of the total URLs seen that day. This implies that for a Cache Size and URL rate resulting in $T_c = 2$ weeks of Resource object storage, the system has to drop Resource objects from the Cache at a rate of approximately 15% of the daily URL volume.

For a higher URL volume per day, the same Cache Size implies a shorter T_c . Because of the long tail, the number of new URLs is a larger percentage of the daily Volume as the number T_c decreases. Therefore, the churn of Resources is not only higher for shorter T_c , but it is accelerating with decreasing T_c .

6. Conclusion

I have provided an overview of a novel system that enables users to explore clusters of like sites, people, content and tasks in real time as they emerge from the activity of other

users. This approach is novel in that it tries to use recent activity on a number of graphs simultaneously to identify and make accessible to users clusters of like behavior without directly solving the global clustering problem.

This is only outline of an idea that can be matured with additional experimentation and work. The original business models explored that would have leveraged this work were not viable. At the time of writing (May 2011), some of this work was salvaged for another startup attempt at OneRiot, Inc.