System RDC: Relevant Data Condenser
A Knowledge System for a Cloud-immersed Culture

Eric Ciminelli and Jennifer Seitzer

Department of Mathematics and Computer Science
Rollins College, Winter Park, FL 32789

eciminelli@rollins.edu, jennifer.h.seitzer@gmail.com

http://myweb.rollins.edu/jseitzer

Abstract

Knowledge representation is an area of artificial intelligence that manages knowledge by most efficiently making the knowledge internally and externally representable, accessible, and usable. Among many other things, this management entails the acquisition, translation, longevity, and structural representation of the knowledge being represented. Our world has become one immersed in its knowledge by the plethora of physical repositories, sources, and subjects. This research addresses the ramifications of this immersion by asking the question of "how do these new characteristics of knowledge affect our computer system specifications and implementations?"

In this work, we delineate the characteristics of knowledge in the 2010s and present our response to them. System "Relevant Data Condenser (RDC)" is a *cloud-savvy app* for a smartphone that symbolizes the new computer system that is globally informed, locally understandable, lean, and portable. This endeavor is part of an overall project that involves knowledge representation in the Cloud. In this paper, we present the development framework and system architecture of RDC, including the incremental steps of its development process.

Introduction

Frameworks for knowledge representation methodologies have been proposed since the late 1960s. Logic, frames, semantic networks, graphs, as well as many others, have all been effective tools to accomplish the four requisites of any knowledge representation technique: representational and inferential adequacy, and representational and inferential efficiency [Russell 2010]. If we fast-forward 50 years, we find ourselves in a cyber world where we are literally immersed in our knowledge. The Von Neumann cycle of *fetch-decode-execute* driving our computational devices has seemingly been expanded to drive *us* at the macro level. We now *fetch* somewhere in the Cloud, *decode* into tweet-like morsels condensing unmanageable amounts of data into ingestible portions, and along with "*executing*" the knowledge in its problem-specific domain, we also log meta-knowledge of the information's time, date, source, and user-base. Our work presented here embraces this new paradigm by presenting a software system that is portable, cloud-driven, and accessible to all.

One of the newest genres of software systems is the app for the smartphone. Our research reported here integrates various techniques of artificial intelligence into a mobile app that is accessible to anybody who has a smartphone. Artificial intelligence (AI) is the area of computer science that attempts to put human thinking into computer programs. Our application eliminates the need for users to constantly "check the net" for new and relevant information on any topic they choose. It generates "*Alerts*" containing useful information that will come in a succinct text-message-like fashion (*phone notification*). AI techniques employed include smart web crawling (scanning the Web for valuable patterns) and automatic language generation related to natural language understanding (such as what Siri does). We illustrate the efficacy of RDC by using the topic of *cryptocurrency trading* (digital money).

The main producible from this work is a computer system comprised of the following components: 1) a web crawler, 2) an AI inference engine that performs web page ranking, parsing, and language generation, 3) a server that pushes relevant information to 4) a smartphone client app that displays the information. The topology of these architectural components can be seen below in Figure 1.

Characteristics of Knowledge in 2015

Today's knowledge is distributed, expansive, ephemeral, elusive (search engines may or may not capture it) and plentiful beyond belief. The great positive is the increased richness of our world-perception; the negative, however, is that contained in this plentitude, is the likelihood that inconsistencies and contradictions along with defunct data, is greatly increased.

Like the "brittle" knowledge of the 1980s' expert system that inspired us to endow our intelligent systems with machine learning capabilities, we are finding that the immense, distributed, ephemeral knowledge of today is also nudging us to augment our computer systems with new capabilities such as custom client-server relationships, GPS services, dedicated web-crawlers, and search engine access. Responding to this nudge, we created RDC.

System *Relevant Data Condenser* (RDC)

The RDC application is grounded in an internal engine that constantly scans through a repository of pre-specified web sites for keywords. Once those keywords are found, the page that contains the keyword is stored locally. Then, using a number of AI methods including a context-based ranking algorithm, the usefulness of each page based on the keyword is determined. Then, a brief message summarizing the information is generated and sent to the smartphone user. This is called "pushing" information to the user. In Figure 1 below, the overall system architecture of RDC is presented. Using this architecture, there are five basic steps that stay in a repetitive loop in the behavioral framework as shown in Figure 2.

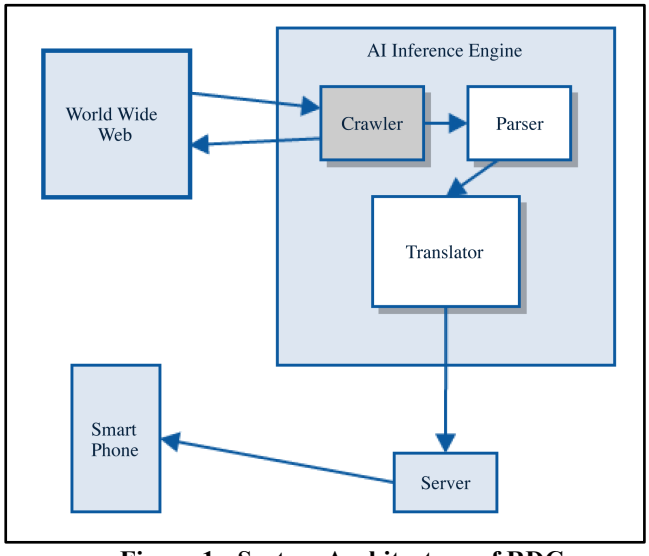

Figure 1: System Architecture of RDC

Begin RDF-Transaction
  App User subscribes to RDC,
    downloading the custom-client,
    uploading choice of topic words

  RDC establishes a persistent record
    for the user establishing parameters
    for client-server interactions

  For each client, RDC does the following:
    1. Every x minutes, keyword smart-crawl the repository (frontier of URL's) on the web
    2. Collect pertinent information using context-based ranking algorithm
    3. Parse pages and extract useful information
    4. Translate possibly verbose info into succinct messages
    5. Deliver messages to subscribers on their smartphones

  End For
End RDC

Figure 2: Top-Level Algorithm of System RDC

Software components of RDC

There are five main software components to the RDC that are cyclically invoked in the general operation of the RDC as shown in Figure 3 below. These activities do not include the formidable development work that was done to establish the underlying infrastructure of this operational cycle. This infrastructure includes the definition and maintenance of a user data base, the networking conundrums experienced when establishing any client-server relationship (three of which are to be discussed), and last, the considerable development required for GUI presentation on the iPhone (which were required to be coded in Objective-C).

One of the fascinating aspects of software development for the Cloud, is that the typical application assumes many roles including the seemingly opposite roles of client and server. RDC assumes the roles of client and server many times in its cycle of operation. The cycle begins with RDC crawling the web for information requested by its subscribers.

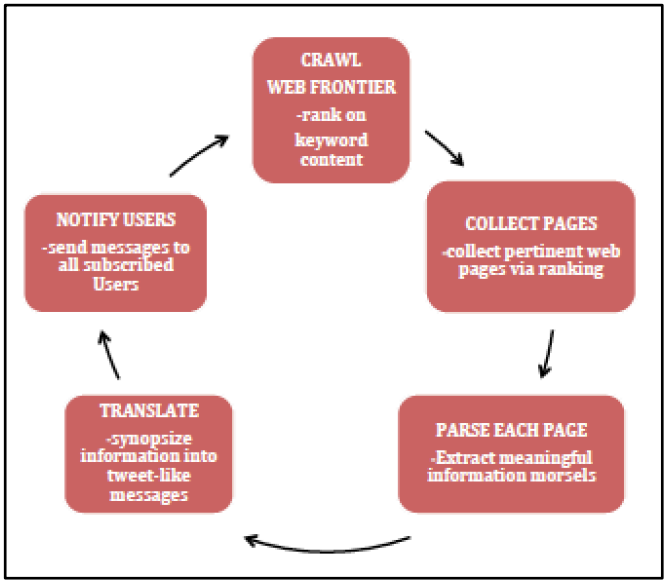

Figure 3: Software Components of RDC

(1) **The Web Crawler** requires that the AI inference engine become a client to many prospective web servers. A client is a program that, over a network, invokes a server for some file, message, or action. As a crawler, the client invokes many web servers requesting pertinent web pages. The web crawler of RDC was written with Java using well-known crawling packages available in the JDC (Java Development package).

(2) **The Filter** determines which of the procured web pages are pertinent to the topic currently being pursued and uses many of the techniques delineated in [Brin 1998] and [Shestakov 2013]. This is where future links of pages are either kept and enqueued on the crawler frontier, or discarded.

(3) **The Parser** is a Java program that searches through HTML and related code to find morsels of information that can be used as small messages to be pushed to users.

(4) **The Translator** is responsible for transforming the verbose to the succinct. In particular, it is the message generator to be served to the clients of RDC.

(5) **The Notifier** is a message pusher that notifies users of important information that was just found then delivers the messages to the smartphone users.

Subscribers to RDC choose a topic(s) of their choice. This topic then forms one of the queries that RDC regularly "checks." We exemplify this notion by presenting how one might use RDC to keep up to date on the topic of *cryptocurrency*.

Example: Cryptocurrency (aka Digital Money)

Cryptocurrency is beginning to have a presence in the economic and social world [Alstyne 2014]. Traders are taking advantage of how volatile the cryptocurrency market is right now. Cryptocurrencies are just beginning to be used in the world. There are now automated teller machines that allow withdrawals of cryptocurrency for actual cash [Jervis 2014]. Until they gain a wide popularity and recognition, however, their exchange rates will continue to fluctuate greatly on a daily basis.

In this example domain, our system is helpful because of its ability to report these fluctuations by being the first to know about any news that goes on in the world of cryptocurrency. One must realize that the exchange rates for cryptocurrencies normally change after news gets out of a certain event that can affect the price. For instance, when a popular business begins to accept a cryptocurrency, it seems that as soon as the news spreads, the price rises. Similar to stock market trading, a societal custom tells us that those who are the first to know the news are the ones who are most likely to succeed in the trading world.

An important part of this work is that the topic of interest can be changed. We choose to exemplify our software system using the topic of cryptocurrency. However, as in many artificial intelligence applications, the underlying software infrastructure can be extracted and applied to virtually any other subject. For example, intelligent expert systems originally written for medical diagnosis were decomposed and then applied to computer system configuration [Russell 2010]. Other areas of application beyond trading include disaster alert systems, institution-specific local data such as 'deadline to drop classes' messages, and any other uses where immediate information is important.

The RDC User Experience

The users of the application interact with the RDC via their smartphones. Inside the application, the user selects the type of cryptocurrency they want to monitor from a list. Each currency selected will have its own tab where the current price along with a graph displaying the price over time can be seen. At the top of the interface there is a news feed where any relevant news that the web-crawler finds is displayed. If the user clicks on the news *scroller* it will

bring them to a page where a list of relevant articles found will be displayed. Selecting an item will open an iPhone browser and bring the user to the source web page where the data was found by RDC.

The most powerful feature of the app is the sending of notifications to its subscribers. Depending on the type of cryptocurrency the user selects to monitor, alerts discovered by the crawler will appear on the home screen in a form such as, "*PC World says that LiteCoin is expanding. Now is a good time to buy." or "LiteCoin hit a low for the week at $742."*

In future work, the app will also include notifications for significant price drops or gains. The user will be able to dictate the price at which they want to be alerted and the app will do so accordingly.

RDC User Client and Graphical User Interface

The graphical user interface (GUI) and network client communication software was written in objective-C. The GUI is organized so that the top shows the news scroller followed by the different types of currencies. Under this top portion is the current price in green along with other useful information including a price graph. Figure 4 shows a rendering of the interface for the topic of cryptocurrency.

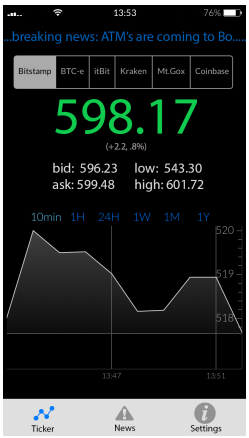

Fig. 4: **RDC Smartphone GUI.**
A screen mockup of the *ticker* page in the application. The top shows the news *scroller* followed by the different types of currencies. Under that is the *current price* in green along with other useful information including a *price graph*.

Clients and Servers in System RDC

The process of getting operational communication of RDC on the Internet was multi-faceted and one that entailed writing many preliminary clients and servers using socket calls and socket libraries for C, C++, Java, and Objective-C. There were many versions that had to be discarded in our path to a functional cloud-savvy app. The first author (who was the main developer) built several different versions of web-interfaces – experimenting with many programming languages including PHP. The main AI inference engine server is a customized server (that evolved out of an Apache server) running on a computer (the server machine) that the main developer built himself.

Incremental Development of Cloud-savvy programs

In [Freeman 2013], the author discusses the notion of TSTMTCTMI: the simplest toy model that captures the main idea of the problem. This was a guiding principle for us in our pursuit. Because there are so many facets, functional units, internal and external links in any one system, we approached our endeavor incrementally. That is, we built a fully functional (yet an extremely primitive version) of RDC in a few months. Then, module by module, we improved and replaced each part to attain a more satisfying version of RDC.

Current Version and Status of RDC

The RDC is a fully functional system consisting of five components: (1) the Crawler, (2) the Filter (3) the Parser (4) the Translator, and (5) the Push-Server (Notifier).

The crawler has been revised many times and is now designed to cooperate with Bing. The software method implementing this cooperation is called *bingResults*. On invocation to the crawler, the method returns the top three news results (sorted by relevance from the past hour). Once the HTML from the page returned is loaded into a string, the program picks it apart to get the title, URL, and a short description from each news story. This information is stored in the News class. Moreover, this latest version of the crawler has implemented multi-threading where it is possible for multiple crawler threads simultaneously running for multiple search words at the request of the user.

The parser successfully performs both filtering and translation. This involves condensing possibly verbose information into succinct messages. The newest version of the RDC parser effectively cooperates with the commercial search engine, Bing although the material of [Meyerzon 2000] was used extensively in earlier versions. The current version returns the top three news results (sorted by relevance from the past hour). Once the HTML from the returned page is loaded into a string data structure, the program decomposes it to extract the title, URL, and a short description from each news story. This information is then inserted into the Java implementation "News Class" (internally represented as a linked-list).

Upon insertion to the News Class, the story is checked to verify non-duplication. If it is a fresh story, the title, URL, and description of the story are concatenated into a single string separated by a unique character that enables the string to be pushed to all of the clients in one message. When the client receives this string, it performs processing invisible to the iPhone user. In particular, it can extract the title, URL, and description from each of the new stories because it knows the unique separation character. Once the client receives the message, it then displays the message in the form of a notification and adds it to the news page of the App.

Conclusion

Today's knowledge is ubiquitous, ephemeral, and massive. It is becoming apparent that most applications will involve interaction involving many clients, servers, and web services. The Relevant Data Condenser is one such application that is able to access, represent, and deliver knowledge both adequately and efficiently. Additionally, we believe that the RDC system serves as a model for the software application that is cloud-savvy, portable, ubiquitous, connected, communicating, and intelligent. These characteristics are lofty and convenient. And, alas, they are difficult to implement. We have shown, by proof of construction, that it is possible to create these successful systems using an incremental and varied approach. In our endeavor, we had many modules that failed or that were not adequate before we found the language and/or technique that worked.

In future work, we intend to expand this system to the Android smartphone platform. Additionally, we hope to actively expand our iPhone user-base to test and augment the RDC to scale up to a much larger number of clients.

Acknowledgements
We are greatly indebted to the reviewers of this paper, and to the Rollins College Student Faculty Collaborative Scholarship program for funding this project.

References

[Alstyne 2014] "Why Bitcoin has Value" *Communications of the ACM* , 57 No. 5, pages 30-32, 2014.

[Brin 1998]. Brin, Sergey, and Lawrence Page. "The Anatomy of a Large-Scale Hypertextual Web Search Engine." *The Anatomy of a Search Engine*. Computer Networks & ISDN Systems, 30 (1998, pages 107-117). Web. 13 Feb. 2014.

[Freeman 2013]. Freeman, W. "How to do Research." *http://people.csail.mit.edu/billf/publications/How_To_Do_Research.pdf.*

[Jervis 2014]. Jervis, Rick. "Bitcoin ATMs Come to USA." *USA Today*. Gannett, 20 Feb. 2014. Web. 22 Feb. 2014.

[Meyerzon 2000]. Meyerzon, Dmitriy, and Sankrant Sanu. "Method of Web Crawling Utilizing Address Mapping." *Microsoft Corporation, Patent granted 11/7/2000.* Downloaded using Google Patents on 2/12/2014.

[Russell 2010] Russell, Stuart J., Peter Norvig, and Ernest Davis. "Intelligent Agents Artificial Intelligence: A Modern Approach" 3rd Edition. Prentice Hall, Upper Saddle River NJ, 2010.

[Shestakov 2013]. Shestakov, Denis. Intelligent Web Crawling. *IEEE. Intelligent Informatics Bulletin*, Dec. 2013. Web. 12 Feb. 2014.