Effective Nutrition Label Use on Smartphones

Vladimir Kulyukin
Department of Computer Science
Utah State University
Logan, UT, USA
vladimir.kulyukin@usu.edu

Tanwir Zaman
Department of Computer Science
Utah State University
Logan, UT, USA
tanwir.zaman@aggiemail.usu.edu

Sarat Kiran Andhavarapu
Department of Computer Science
Utah State University
Logan, UT, USA
sarat.kiran@aggiemail.usu.edu

***Abstract*—Proactive nutrition management is considered by many nutritionists and dieticians as a key factor in reducing and controlling cancer, diabetes, and other illnesses related to or caused by mismanaged diets. As more and more individuals manage their daily activities with smartphones, smartphones have the potential to become proactive diet management tools. Many grocery products sold worldwide have nutrition labels (NLs). Unfortunately, even highly motivated consumers sometimes find it difficult to locate or to comprehend them. The literature on NL use by consumers contains several recommendations to improve retention and comprehension of nutrition information: 1) central positions of NLs; 2) nutrients sorted by health relevance; 3) explanation of nutrients; 4) reduced visual clutter around NLs; 5) increased visual salience through contrast and orientation; 6) increased surface size of NLs. In this paper, a system is presented that satisfies recommendations 1, 3, 4, and 6. The system's front end is implemented as a smartphone application. The smartphone application runs on the Google Nexus 4 smartphone with Android 4.3 or 4.4. The system's back end is currently a four node Linux cluster used for image recognition and data storage. The presented system has broader implications for food policy. The position advocated in this paper argues that the current NL design on product packages does not necessarily have to change to make NL use more effective. Rather, consumers can use their smartphones to design and manipulate their own NL presentation schemes suitable to their specific nutrition needs without requiring product manufacturers to change physical product packages.**

Keywords—mobile computing; cloud computing; nutrition label use; nutrition management; electronic commerce

I. Introduction

U.S. Department of Agriculture estimates that U.S. residents have increased their caloric intake by 523 calories per day since 1970 [1]. Mismanaged diets are estimated to account for 30-35 percent of cancer cases. A leading cause of mortality in men is prostate cancer. A leading cause of mortality in women is breast cancer. Approximately 47,000,000 U.S. residents have metabolic syndrome and diabetes. Diabetes in adults and children appears to be closely related to increasing obesity levels. It is estimated that by 2030 the prevalence of diabetes in the world will reach 4.4%, which will equal to approximately 366 million people [2]. Due to the long-term complications of diabetes, many countries will likely see an increase in blindness, kidney failures, and amputations. Many nutritionists and dieticians consider proactive nutrition management to be a key factor in reducing and controlling cancer, diabetes, and other illnesses related to or caused by mismanaged diets.

Many products sold worldwide have nutrition labels (NLs). In the U.S., the display of nutrition information is mandated by the Nutrition Education and Labeling Act (NLEA) of 1990 [3]. Similar initiatives or legislative acts (e.g., EU FLABEL [4]) exist in other countries. Unfortunately, even highly motivated consumers, who look for NLs to make healthy food choices, sometimes find it difficult to locate and to comprehend nutrition information on many products [5]. Recent investigations of NL use by consumers have used digital cameras to track consumers' eye movements to better understand how consumers locate and understand NLs [6]. These studies have identified four key factors that appear to impede comprehension and retention of nutrition information: 1) label's location on the package; 2) presentation of information within the label; 3) label's surface size; and 4) surrounding visual clutter. Consumers report that they can better locate NLs positioned centrally on a side with a small amount of surrounding visual clutter. Consumers also report failures to comprehend nutrition terms and to read small font sizes in NLs [7].

Several recommendations are made in the NL use literature to improve retention and comprehension of nutrition information: 1) central positions of NLs; 2) nutrients sorted by health relevance; 3) explanation of nutrients; 4) reduced visual clutter around NLs; 5) increased visual salience through contrast and orientation; 6) increased surface size of NLs [5].

In this paper, a system is presented that satisfies recommendations 1, 3, 4, and 6. The system's front end is implemented as a smartphone application. The application runs on the Google Nexus 4 smartphone with Android 4.3 or 4.4. The system's back end is currently a four node Linux cluster used for image recognition and data storage.

The front end smartphone sends captured frames to the back end cluster across a wireless data channel (e.g., 3G/4G/Wi-Fi) where barcodes, both skewed and aligned, are recognized [10]. Corresponding NLs are retrieved from a cloud database, where they are stored as HTML documents, and sent across the wireless data channel back to the smartphone where the HTML documents are displayed on the touchscreen. Wikipedia links to important nutrition terms are embedded for better comprehension. Consumers can use standard touch gestures (e.g., zoom in/out, swipe) available on mainstream smartphone platforms to manipulate the label's surface size. The NL database currently includes 235,000 products compiled from public web sites by a custom crawler.

The remainder of this paper is organized as follows. In Section II, the system's overview is presented and vision-based nutrition extraction methods are discussed to give the reader a broader background about the front end of the system. In Section III, the node cluster is described in detail. Section IV presents several stress test experiments with the system and discusses the results. Section V presents our conclusions, outlines the strengths and limitations of our system, and discusses several implications for proactive nutrition management and food policy.

II. Related Work

A. Overview

Modern nutrition management systems assume that users understand how to collect nutritional data and can be triggered into data collection with digital prompts (e.g., email or SMS) [7]. Such systems often underperform, because many users find it difficult to integrate nutrition data collection into their daily activities due to lack of time, motivation, or training. Consequently, they eventually turn off or ignore numerous digital stimuli [8].

To overcome these challenges, we have begun to develop a Persuasive NUTrition Management System (PNUTS). PNUTS seeks to shift current research and clinical practices in nutrition management toward persuasion, automated nutritional information extraction and processing, and context-sensitive nutrition decision support.

PNUTS is based on a nutrition management approach inspired by the Fogg Behavior Model (FBM) [8], which states that motivation alone may not be insufficient to stimulate target behaviors such as nutrition intake recording or blood tests. Even a motivated user must have both the ability to execute a behavior and a trigger to engage in that behavior at an appropriate place and time. Many nutrition management system designers assume that consumers and patients are either more skilled than they actually are or that they can be trained to obtain the required skills. Since training is difficult and time consuming, a more promising path is to make target behaviors easier and more intuitive to execute.

PNUTS makes proactive nutrition management easier and more intuitive by utilizing the relative advantages of mobile and cloud computing to improve nutrition information comprehension and retention and to automate real-time vision-based NL analysis and nutrition intake recording [9, 10]. In this paper, we focus on effective NL use on smartphones that addresses four out of six major factors that impede nutrition information retention and comprehension by consumers. While vision-based nutrition intake recording is beyond the scope of this paper, we give a sketchy overview of how it works to give the reader a broader background on the system.

B. Barcode Scanning

In PNUTS, there are two kinds of nutrition information extraction algorithms: barcode localization and scanning and nutrition information extraction from NLs. In this section, we give a sketchy overview of both algorithms. Interested readers are referred to [9] and [10] for technical details and experiments.

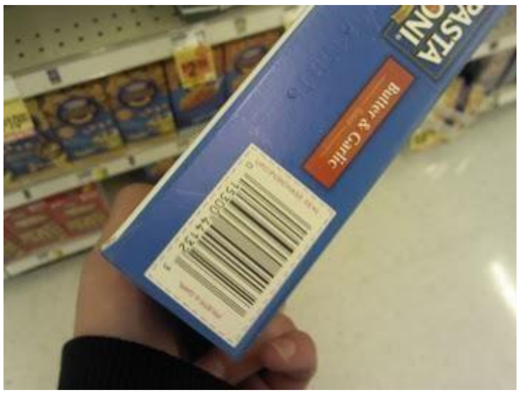


Figure 1. Skewed barcode

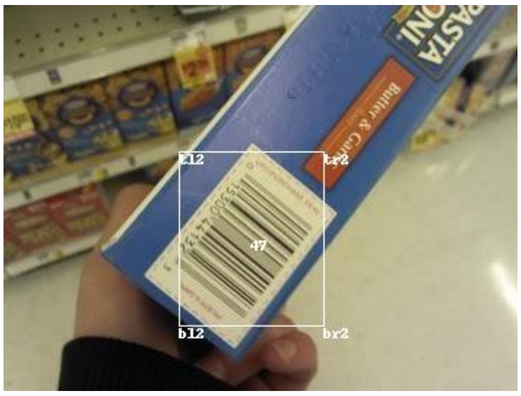


Figure 2. Localized skewed barcode

Recognized barcodes are used to retrieve NLs from a database of NLs. In our previous research [11], we presented an eyes-free algorithm for vision-based localization and decoding of aligned barcodes. The algorithm was based on the assumption that simple and efficient vision techniques, when augmented with interactive user interfaces, ensure that the smartphone camera is horizontally or vertically aligned with the surface on which a barcode is sought.

In [10], we presented two algorithms that relaxed the horizontal or vertical alignment constraints, which may not always hold, to localize skewed barcodes in frames captured by the smartphone's camera, as shown in Figures 1 and 2. The first algorithm localizes skewed barcodes in captured frames by computing dominant orientations of gradients (DOGs) of image segments and collecting smaller segments with similar DOGs into larger connected components.

The second algorithm localizes skewed barcodes by growing edge alignment trees (EATs) on binary images with detected edges. Since our experiments showed that the DOG algorithm outperformed the EAT algorithm [9], the current version of PNUTS uses the DOG algorithm for barcode localization. The localized barcodes are scanned without any image rotation. Our current barcode scanning algorithm handles both UPC and EAN formats. Unlike other barcode scanning solutions (e.g., https://github.com/zxing/zxing, http://redlaser.com), our algorithm does not require the user to align the smartphone's camera with the barcode and can detect skewed or aligned barcodes anywhere in the image.

C. Nutrition Label Segmentation

We have also been working on a vision-based algorithm to localize NLs on grocery product packages and to segment them into text chunks for subsequent optical character recognition (OCR) [9]. The algorithm captures frames in video mode from the smartphone's camera, localizes horizontally or vertically aligned NLs (see Figure 3), and segments the NLs into single- or multi-line text chunks, as shown in Figure 4 (right). Each text chunk is given to an OCR engine. While we have been using free open source GOCR (jocr.sourceforge.net) and Tesseract (http://code.google.com/p/tesseract-ocr/) engines, other OCR engines can be used as well.

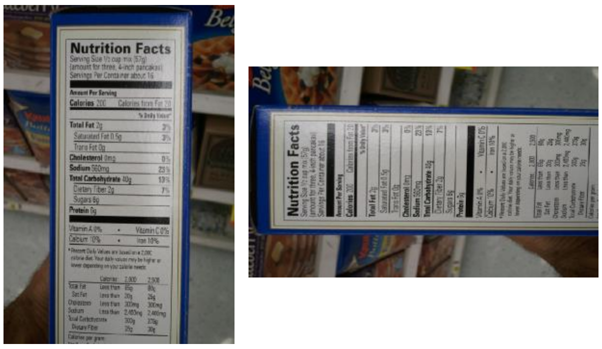


Figure 3. Horizontally and vertically aligned NLs

Images captured from the smartphone's video stream can be divided into foreground and background pixels. In general, foreground pixels are defined as content-bearing units in a domain-dependent manner. For example, content can be defined as black pixels, white pixels, pixels with specific luminosity levels, specific neighborhood connection patterns (e.g., 4-connected, 8-connetected), etc. Background pixels are those that are not foreground.

The horizontal projection of an image (HP) is a sequence of foreground pixel counts for each image row. The vertical projection of an image (VP) is a sequence of foreground pixel counts for each column in an image. Figure 4 shows the vertical projection of an NL image after edge detection, which is done in our system with the Canny Edge detector [11].

In detecting NL boundaries, three assumptions are currently made: 1) an NL is present in the image; 2) the NL present in the image is not cropped; and 3) the NL is horizontally or vertically aligned. The detection of NL boundaries proceeds in three stages. Firstly, the first approximation of the vertical table boundaries is computed. Secondly, the vertical boundaries computed in the first stage are extended to the left and to the right. Thirdly, the upper and lower horizontal boundaries are computed.

The objective of the first stage is to detect the approximate location of the NL along the horizontal axis. This approximation starts with the detection of horizontal lines in the image, which is accomplished with a horizontal line detection kernel (HLDK) that we developed in our previous research and described in our previous publications [12, 13]. It should be noted that other line detection techniques (e.g., Hough transform [14]) can be used for this purpose. Our HLDK is designed to detect large horizontal lines in images to maximize computational efficiency.

Let HLFI be a horizontally line filtered image, i.e., the image put through the HLDK filter or some other line detection filter (see Figure 4 left). The projections of white pixels can then be computed for each column of HLFI. The right image in Figure 4 shows the vertical projection of the HLFI on the left. A threshold is chosen, which in our application is set to the mean count of the white foreground pixels in columns. In Figure 4 (right), the threshold is shown by a red line. It can be observed that the foreground pixel counts in the columns of the image region with the NL are greater than the threshold.

The vertical boundaries of an NL are computed as follows. Firstly, the left boundary is extended to the first column to the left of the current left boundary, for which the projection is at or above the threshold, whereas the right boundary is extended to the first column to the right of the current right boundary, for which the vertical projection is at or above the threshold.

A typical NL includes text chunks with various types of caloric and ingredient information, e.g., "Total Fat 2g 3%." To optimize the performance of subsequent OCR, which is beyond the scope of this paper, these text chunks are segmented from localized NLs (see Figure 4). This approach is flexible in that segmented text chunks can be wirelessly transmitted to multiple cloud servers for parallel OCR. As can be seen in Figure 5 (left), text chunks are separated by black colored separators. Formally, text chunks are defined as image segments separated by horizontal black separator lines. The chunking algorithm detects such separators as well as areas with high concentration of corners between the separators, the theory being that text segments are areas with higher concentrations of corners. Figure 5 (right) shows text chunks detected in the localized NL shown in Figure 5 (left). These chunks are subsequently sent to an OCR engine [13].

D. Nutrition Label Crawler

Our original intention was to use only computer vision to extract NLs to populate the NL database. Both the localization and text chunking procedures outlined in the previous section showed robust performance levels in our experiments [13]. Unfortunately, when these techniques were coupled with open source OCR engines such as GOCR (jocr.sourceforge.net) and Tesseract (http://code.google.com/p/tesseract-ocr/), the number of OCR errors was high.

The crawler module was implemented to compensate for low OCR rates. We hope that, as OCR rates improve, the need for the crawler component will become less significant, because NL databases will be populated with computer vision. Another reason that we are currently working to improve OCR rates is that web site scraping may be unreliable in the long term in that a site that currently permits robots may prohibit them in the future. Additionally, some sites may contain inaccurate or obsolete nutrition information.

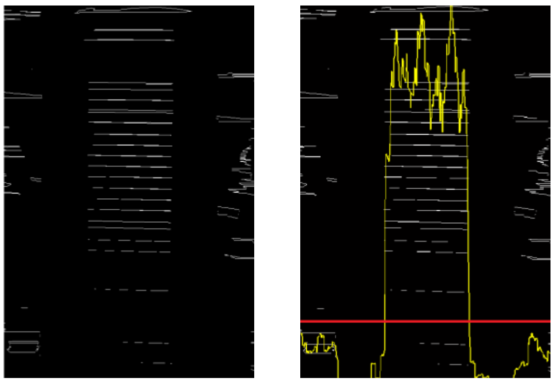
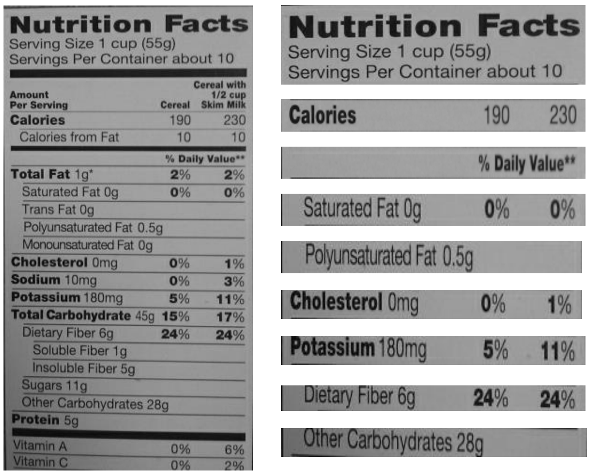


Figure 4. Edges detected in NL (left) and its vertical projection (right)

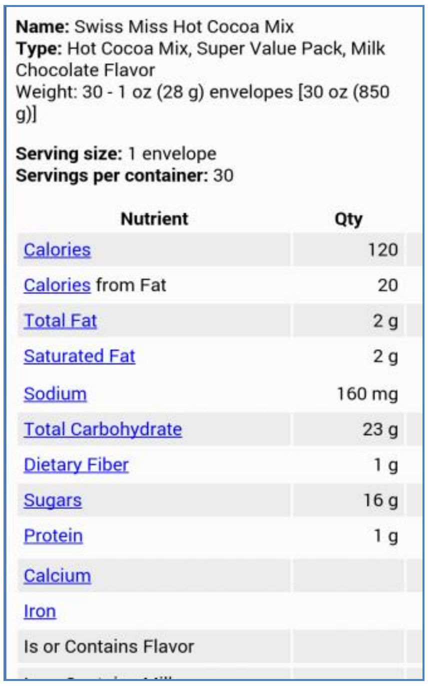


Nutrition Facts		
Serving Size 1 cup (55g)		
Servings Per Container about 10		
Amount Per Serving	Cereal	Cereal with 1/2 cup Skim Milk
Calories	190	230
Calories from Fat	10	10
	% Daily Value**	
Total Fat 1g*	2%	2%
Saturated Fat 0g	0%	0%
Trans Fat 0g		
Polyunsaturated Fat 0.5g		
Monounsaturated Fat 0g		
Cholesterol 0mg	0%	1%
Sodium 10mg	0%	3%
Potassium 180mg	5%	11%
Total Carbohydrate 45g	15%	17%
Dietary Fiber 6g	24%	24%
Soluble Fiber 1g		
Insoluble Fiber 5g		
Sugars 11g		
Other Carbohydrates 28g		
Protein 5g		
Vitamin A	0%	6%
Vitamin C	0%	2%

Figure 5. Segmented NL (left) and its text chunks (right)

The current version of the crawler scrapes three public web sites dedicated to nutrition, which we found helpful in our background research on public nutrition information sites: www.directionsforme.org, www.smithsfoodanddrug.com, www.digit-eyes.com. As we find other helpful public sites, we may add them to the crawler's list in the future. Permissions are verified for each web site before it is scraped.

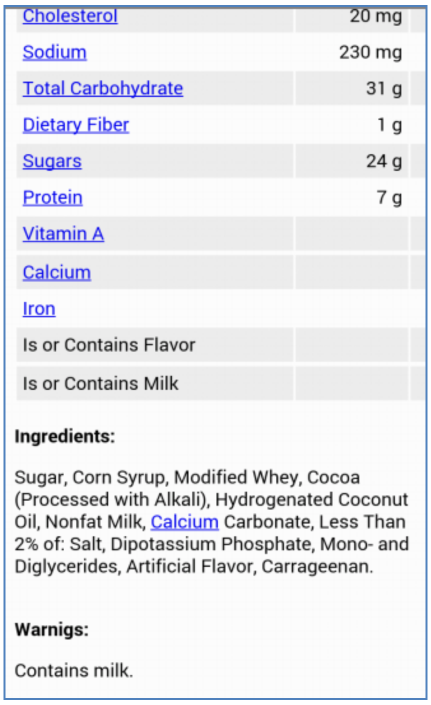
Each URL is parsed with the Python BeautifulSoup library (http://www.crummy.com/software/BeautifulSoup/) which is implemented on top of the popular Python XML and HTML parsers LXML (lxml.de) and HTML5LIB (http://code.google.com/p/html5lib/) . Each URL is parsed to obtain the NL, ingredients, warnings, and categories. When this information is extracted, a new HTML document is generated that contains not only the extracted information but also embedded Wikipedia links to all nutrition terms used in the tabular part of the NL. A path to this HTML document is saved in a database under a specific barcode. The NL database currently includes 200,000 products compiled from public web sites by the crawler.


Name: Swiss Miss Hot Cocoa Mix
Type: Hot Cocoa Mix, Super Value Pack, Milk Chocolate Flavor
Weight: 30 - 1 oz (28 g) envelopes [30 oz (850 g)]

Serving size: 1 envelope
Servings per container: 30

Nutrient	Qty
Calories	120
Calories from Fat	20
Total Fat	2 g
Saturated Fat	2 g
Sodium	160 mg
Total Carbohydrate	23 g
Dietary Fiber	1 g
Sugars	16 g
Protein	1 g
Calcium	
Iron	
Is or Contains Flavor	

Figure 6. Upper part of NL with embedded Wiki links



Cholesterol	20 mg
Sodium	230 mg
Total Carbohydrate	31 g
Dietary Fiber	1 g
Sugars	24 g
Protein	7 g
Vitamin A	
Calcium	
Iron	
Is or Contains Flavor	
Is or Contains Milk	

Ingredients:

Sugar, Corn Syrup, Modified Whey, Cocoa (Processed with Alkali), Hydrogenated Coconut Oil, Nonfat Milk, Calcium Carbonate, Less Than 2% of: Salt, Dipotassium Phosphate, Mono- and Diglycerides, Artificial Flavor, Carrageenan.

Warnigs:

Contains milk.

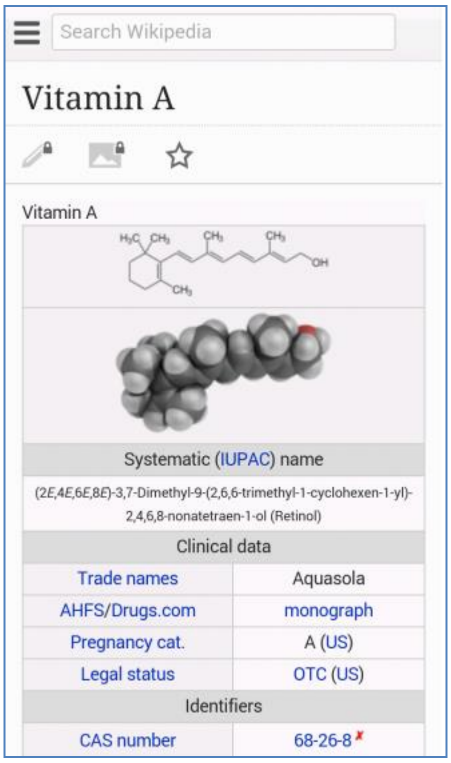
Figure 7. Lower part of NL with embedded Wiki links

E. Nutrition Label Display

When a barcode is recognized at the back end in an image sent to it from the smartphone, its HTML document (if there is one) is sent back to the smartphone and displayed on the touchscreen, as shown in Figure 6.

Consumers can use standard touch gestures (e.g., zoom in/out, swipe) for manipulating the label's surface size or browsing its contents. For example, Figure 7 shows the lower part of the NL displayed in Figure 6 after the user does a down swipe on the touchscreen. When the user clicks on an embedded link, a Wiki page for that nutrient is displayed, as shown in Figure 8.

This presentation method satisfies four out of the six recommendations made in the NL use literature to improve nutrition information retention and comprehension. As stated in Section I, there are six recommendations: 1) central positions of NLs; 2) nutrients sorted by health relevance; 3) explanation of nutrients; 4) reduced visual clutter around NLs; 5) increased visual salience through contrast and orientation; 6) increased surface size of NLs.



Search Wikipedia

Vitamin A

Vitamin A

Systematic (IUPAC) name	
(2E,4E,6E,8E)-3,7-Dimethyl-9-(2,6,6-trimethyl-1-cyclohexen-1-yl)-2,4,6,8-nonatetraen-1-ol (Retinol)	
Clinical data	
Trade names	Aquasola
AHFS/Drugs.com	monograph
Pregnancy cat.	A (US)
Legal status	OTC (US)
Identifiers	
CAS number	68-26-8 ✗

Figure 8. Wiki page of a nutrient (Vitamin A)

This presentation addresses the first recommendation by positioning NLs centrally on the touchscreen, as shown in Figures 6 and 7. The third recommendation is addressed through embedded Wiki links to the nutrients in the tabular component of each NL, as shown in Figure 8. The fourth recommendation is completely satisfied, because there is no visual clutter on the touchscreen around the displayed NL. The sixth recommendation is also addressed, because the user can use standard touch gestures to increase or decrease the actual size of the NL, which can be beneficial not only for regular users but also for low vision ones.

III. Linux Node Cluster

To implement the back end of our system, we have built a Linux cluster out of four Dell computers for cloud-based computer vision and data storage. Each computer has an Intel Core i5-650 3.2 GHz dual-core processor that supports 64-bit computing. The processors have 3MB of cache memory. The machines are equipped with 6GB DDR3 SDRAM and have Intel integrated GMA 4500 Dynamic Video Memory Technology 5.0. All machines have 320 GB of hard disk space. Ubuntu 12.04 LTS was installed on each machine. We used JBoss (http://www.jboss.org) to build and configure the cluster and the Apache mod_cluster module (http://www.jboss.org/mod_cluster) to configure the cluster for load balancing. Our cluster has one master node and three slaves. The master node is the domain controller. The master node also runs mod_cluster and httpd. All four machines are part of a local area network and have hi-speed Internet connectivity. We have installed JDK 7 in each node.

The JBoss Application Server (JBoss AS) is a free open-source Java EE-based application server. In addition to providing a full implementation of a Java application server, it also implements the Java EE part of Java. The JBoss AS is maintained by jboss.org, a community that provides free support for the server. JBoss is licensed under the GNU Lesser General Public License (LGPL).

The Apache mod_cluster module is an httpd-based load balancer. The module is implemented with httpd as a set of modules for httpd with mod_proxy enabled. This module uses a communication channel to send requests from httpd to a set of designated application server nodes. An additional communication channel is established between the server nodes and httpd. The nodes use the additional channel to transmit server-side load balance factors and lifecycle events back to httpd via a custom set of HTTP methods collectively referred to as the Mod-Cluster Management Protocol (MCMP).

The mod_cluster module provides dynamic configuration of httpd workers. The proxy's configuration is on the application servers. The application server sends lifecycle events to the proxies, which enables the proxies to auto-configure themselves. The mod_cluster module provides accurate load metrics, because the load balance factors are calculated by the application servers, not the proxies.

All nodes in our cluster run JBoss AS 7. Jboss AS 7.1.1 is the version of the application server installed on the cluster. Apache httpd runs on the master with the mod_cluster-1.2.0 module enabled. The Jboss AS 7.1.1 on the master and the slaves are discovered by httpd.

A Java servlet for image recognition is deployed on the master node as a web archive file. The servlet's URL is hardcoded in every front end smartphone. The servlet receives images uploaded with http post requests, recognizes barcodes, and sends the appropriate HTML pages back to front end smartphones. The HTML files generated by the crawler are stored on the cloud via a shared directory implemented as a Network File System (NFS) on the cluster. No data caching is currently done on the servlet or the front end smartphones.

IV. Experiments and Results

We tested the robustness of the node cluster in a series of stress test experiments. The objective was to check the accuracy of our cluster configuration and load balancing. We took eight Google Nexus 4 smartphones from our laboratory running Android 4.3 or 4.4 and deployed a node cluster stress tester application on each of them.

The application would start a background service at startup. The background service would download a random 1024 x 1024 barcode image from an http server, upload it to the node cluster, and display the node cluster's response at the smartphone's action bar.

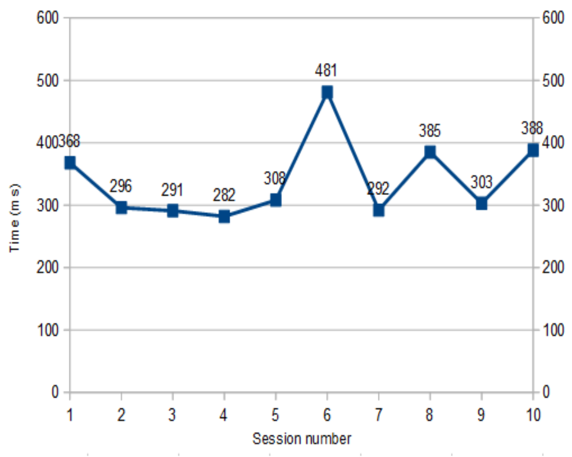


Figure 9. Node cluster request-response pairs

The applications on both smartphones were executed for ten sessions of 3,000 request-response pairs each and the average request-response time for each session was calculated. Figure 9 gives the graph of the node cluster's request-response times. The lowest average was 282 milliseconds; the highest average was 481 milliseconds.

Additional node cluster testing was done in an undergraduate mobile application development class taught by the first author at Utah State University in the fall 2013 semester. One of the assignments asked the students to write an image uploader application to stress test the node cluster. Specifically, fourteen students, each of whom had an Android smartphone, implemented and deployed this application on their smartphones and ran it for one week. Each application would submit three images per second for a total of forty images per second. Thus, during this week, the node cluster was tested with eight lab smartphones and fourteen student smartphones and received a total of sixty six 1024 x 1024 images per second (twenty four images from the lab smartphones and forty two images from the students' smartphones). The node cluster did not experience any failures and was able to handle and balance the load.

V. Conclusions

The R&D literature on NL use by consumers contains several recommendations for improving nutrition information retention and comprehension: 1) central positions of NLs; 2) sorting of nutrients by health relevance; 3) explanation of nutrients; 4) reduced visual clutter around NLs; 5) increased visual salience through contrast and orientation; and 6) increased surface size of NLs.

In this paper, we presented how our system, called PNUTS, addresses recommendations 1, 3, 4, and 6 to increase the effectiveness of NL use on smartphones. The system leverages vision-based barcode recognition to retrieve NLs for specific barcodes. Wikipedia links to important nutrition terms are embedded in NLs positioned centrally on the smartphone's touch screen. The user can follow the links to improve comprehension and retention of nutrition information. The system leverages the standard touch gestures (e.g., zoom in/out, swipe) to enable the user to manipulate the label's surface size and browse NLs. The NL database currently includes 230,000 products compiled from public web sites by a custom crawler.

PNUTS currently does not address recommendations 2 and 5. To address recommendation 2, user profiles will have to be added to the system. For example, if a user has Type II diabetes, the system can automatically sort the nutrients in each NL according to some relevancy taxonomy worked out in collaboration with a dietician. Alternatively, a smartphone UI can be designed to enable the user to specify the health relevance of nutrients for subsequent display.

A similar approach may turn out successful in addressing recommendation 5. For example, visual salience of displayed NLs can be increased by coding important NL components (e.g., carbohydrates, dietary fiber, sugar, etc.) with different colors or display them in a pie chart. It should also be possible to enable the user to choose a visual salience enhancement pattern at configuration time.

The average request-response time between the front end and the back end was three seconds. We expect additional reductions in request-response times to come from faster data communication plans (e.g., 4G) and adding additional nodes to the cluster. Additional time reductions will come from data caching both on the front end and the back end. In the current version of the system, no data caching is currently done on the node cluster or the smartphones. The smartphones can, for example, maintain a local cache of barcodes and retrieved NLs and display NLs without receiving them from the node cluster.

The presented system has an important implication for proactive nutrition management and food processing industry. A major implication for proactive nutrition management is that PNUTS enhances the user's ability to record and comprehend nutritional intake. The user is no longer required to manually enter either names or barcodes of consumed products. In the future, we plan to extract caloric information from NL HTML files or, in the longer term, from captured images automatically.

Our system also has implications for broader food policy. The recommendations for improving NL use appear to focus on product manufacturers. The central theme of these recommendations appears to be that the product manufacturers should be rationally or legislatively persuaded to change the NL design on product packages. However, in order to change the NL design on a physical package, the manufacturer must bear many costs such as disruptions in product recognition and, quite possibly, reduced advertisement space on the package. Moreover, even if product manufacturers adopt a different NL design, there is no guarantee that the one-size-fits-all approach will succeed with all consumers. There will always be consumers who may not like the new design and may prefer additional or different design customization.

The approach presented and advocated in this paper argues that the current NL design on product packages does not necessarily have to change to make NL use more effective. Rather, the strengths of mobile and cloud computing can be leveraged to increase the effectiveness of NL use. Consumers can use their smartphones to design their own NL presentation schemes suitable to their specific nutrition needs without requiring product manufacturers to change physical product packages. While it remains to be seen which of these two approaches will be more successful in the long run, the wide adoption of smartphones and cloud services by the public at large is an indicator that the approach presented in this paper has potential.

References

[1] World Health Organization. "Annual World Health Statistics." Avail. at http://www.who.int/gho/publications/world_health_statistics/en/.

[2] Rubin, A. L. Diabetes for Dummies. 3rd Edition, Wiley, Publishing, Inc. Hoboken, New Jersey, 2008.

[3] Nutrition Labeling and Education Action of 1990. Avail. at http://en.wikipedia.org/wiki/Nutrition_Labeling_and_Education_Act_of_1990.

[4] Food Labelling to Advance Better Education for Life. Avail. at www.flabel.org/en.

[5] Graham, D. J., Orquin, J. L., and Visshers, V. H. M. "Eye tracking and nutritional label use: a review of the literature and recommendations for label enhancement," Food Policy, vol. 32, pp. 378-382, 2012.

[6] Graham, D.J. and Jeffery, R.W. "Location, location, location: eye tracking evidence that consumers preferentially view prominently positioned nutrition information," J. Am. Diet. Assoc., vol. 111, pp. 1704–1711, 2011.

[7] Årsand, E., Tatara, N., Østengen, G., and Hartvigsen, G. 2010. Mobile Phone-Based Self-Management Tools for Type 2 Diabetes: The Few Touch Application. *Journal of Diabetes Science and Technology*, 4, 2 (March 2010), pp. 328-336.

[8] B. J. Fog. "A behavior model for persuasive design," In Proc. 4th International Conference on Persuasive Technology, Article 40, ACM, New York, USA, 2009.

[9] Kulyukin, V., Kutiyanawala, A., Zaman, T., and Clyde, S. "Vision-based localization and text chunking of nutrition fact tables on android smartphones," In Proc. International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV 2013), pp. 314-320, ISBN 1-60132-252-6, CSREA Press, Las Vegas, NV, USA, 2013.

[10] Kulyukin, V. and Zaman, T. "Vision-based localization of skewed upc barcodes on smartphones," In Proc. International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV 2013), pp. 344-350, pp. 314-320, ISBN 1-60132-252-6, CSREA Press, Las Vegas, NV, USA, 2013.

[11] Canny, J.F. "A Computational approach to edge detection." IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 8, 1986, pp. 679-698.

[12] Kulyukin, V., Kutiyanawala, A., and Zaman, T. "Eyes-free barcode detection on smartphones with niblack's binarization and support vector machines," In Proc. 16-th International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV 2012), vol. I, pp. 284-290, CSREA Press, July 16-19, 2012, Las Vegas, Nevada, USA, (pdf); ISBN: 1-60132-223-2, 1-60132-224-0Y, 2012.

[13] Kulyukin, V., Vanka, A., and Wang, W. "Skip trie matching: a greedy algorithm for real-time ocr error correction on smartphones," International Journal of Digital Information and Wireless Communication (IJDIWC), vol 3(3), pp. 56-65, ISSN: 2225-658X, 2013.

[14] Duda, R. O. and Hart, P. E. "Use of the hough transformation to detect lines and curves in pictures," Comm. ACM, vol. 15, pp. 11–15, January, 1972.