Scalable Web Content Understanding Framework

Yang Sun, Hyungsik Shin, Sayandev Mukherjee, Ronald Sujithan, Hongfeng Yin, Yoshikazu Akinaga and Pero Subasic

DOCOMO Innovations, Inc.
Emails: {ysun, hshin, smukherjee, rsujithan, hyin, akinaga, psubasic}@docomoinnovations.com

Abstract—**The contextualization of an unknown web page is a fundamental need in many online applications. We propose a new framework known as the Content Understanding Engine (CUE) that allows computational stages to be composed with different technologies to contextualize an unknown URL. We describe how this computation pipeline interfaces with our Big Data infrastructure and how this approach simplifies deployment to private or public cloud environments. The implementation details of this framework are provided along with a use case to demonstrate the value of the CUE. We provide the results from our evaluation of this pipelined architecture with a wide range of URL from different topics.**

***Keywords*–contextual tagging; advertising; content understanding engine.**

I. INTRODUCTION

Many online applications heavily rely on automated systems to analyze the context of web pages. These contextual systems are typically required to take URLs, fetch web pages, parse content, extract keywords, classify text and find the most relevant tags. The resulting contextual profiles will not only present the opportunities for advertisement matching, but also unveil personal preferences, interests and trending concepts [1][2]. For example, to optimize advertising campaigns, advertisers often develop models based on analyzing historical contextual consumption of users and then match product offerings to specific contextual profiles. Contextual advertising systems also rely on contextual information to match advertisement with web pages.

In practice, a number of different techniques drawn from diverse fields, such as Text Mining, Natural Language Processing, Machine Learning and Information Retrieval need to be combined into a pipelined architecture to meet these requirements. However, current frameworks do not handle the whole process from URL crawling to the semantic analysis. Our proposed framework integrates existing technologies and handles the end to end complexity of the contextual profiling process.

Many components are required to build a scalable and useful contextual profiling system, including (1) a scalable web content fetching module for billions of URLs, (2) a fast web page parsing and ad removing module, (3) a scalable document storage and processing module, (4) a content understanding module to extract and tag web pages with brief and representative text, and (5) a tag generation module that finds the most relevant tags from the representative text. Each module has unique requirements in terms of response time, scalability and accuracy.

Our proposed system framework has the following contributions to the community:

- A Software Architecture for a scalable computation pipeline that can be deployed to any private or public cloud infrastructure;
- A staged architecture, allowing the use of different technologies for each stage, and an interface with big data infrastructure for truly large scale processing;
- A Wikipedia-based contextual tagging solution that reflects the latest events and trending concepts; and
- A uniform way of communicating the contextual tags to all players involved in the marketing process.

The rest of this paper is organized as follows. In Section II, we describe the related work done in the areas of distributed workflow engines and prior research on Contextual Profiling Systems. In Section III, we describe the stages of our Content Understanding Framework in detail. In Section IV, we briefly describe the cloud deployment of our Content Understanding Engine (CUE). In Section V, we describe a compelling use case for the CUE and discuss evaluation results. Finally, we state our conclusions and describe further work.

II. RELATED WORK

The contextualization of the contents of any web page, including content tagging using a controlled vocabulary, is a fundamental task in many online applications. Existing approaches focus on three aspects of the context understanding problem: (1) the free-text approach – using free text labels to tag articles [2][3]; (2) the classification approach – classifying articles to a well-organized hierarchical taxonomy of topics [1][4]; and (3) the semantic approach – using semantic analysis to determine advertising needs [5]. Unfortunately, none of these by itself is suited to our task, as discussed below.

The first approach summarizes articles with free text that is rich enough to represent the meaning as well as abstract enough to fit to specific applications. The feature space of free text has dimension in the millions. Therefore, it is typically difficult for advertising systems to use. The second approach maps complex concepts to well-structured categories. These categories typically have very general terms, so that specifics of the articles are not represented in the approach. The third approach, semantic analysis, is an evolving field that has potential, but does not at present provide a mature solution to the content understanding problem.

A. Principal Challenges

There are several challenges in the domain of content understanding systems: (1) such systems need to function well in the presence of "noise", such as spelling and grammatical errors, different languages, markup errors, handling boilerplate content, etc.; (2) they need to create features from the extracted text so that the features represent the original content in a satisfactory form; (3) they need to map the representative content extracted from the page to a set of known vocabulary terms; and (4) they need to be able to scale and deal with petabyte-scale volumes on model cloud infrastructures such as the now-ubiquitous Amazon Web Services.

In order to address these challenges, many different technologies are used in practice. These technologies arise from many different related fields, such as Text Mining, Natural Language Processing, Machine Learning and Information Retrieval. A number of pipelined architectures have been developed to solve the problem of applying these different technologies to solve the end-to-end problem.

Our work is closely related to work on web usage mining, automatic discovery and analysis of patterns in click streams, user transactions, and other associated data collected or generated as a result of user interactions with Web resources [6][7][8]. In particular, our work is related to web mining systems that use other sources of knowledge: either semantic domain knowledge from ontologies (such as product catalogs, concepts and categories) or a more generic knowledge base such as the freely-available Wikipedia concepts and categories [9]. However, the existing approaches focus primarily on challenges 1, 2 and 3, but do not address challenge 4 as an integral part of the solution. While building on this early work on web mining, we needed a workflow solution that can scale to process petabyte-scale volumes that can be deployed and operated as a cloud-based service.

B. Existing Technologies and Frameworks

Hadoop [10], the open source implementation of the MapReduce framework, has become the dominant environment for building an architecture for solving the scalability problems described in the previous section. In particular, systems such as Oozie [11] and Azkaban [12] provide a workflow abstraction on top of Hadoop. Both support defining a workflow as a Directed Acyclic Graph (DAG) [13] made up of a composition of individual steps. In Azkaban the job type, any parameters, and any dependencies are specified. However, Azkaban does not have any notion of a self-contained workflow, so a job can depend on any other job in the system. In Oozie, a workflow is defined in an XML file, which specifies a start action. However, Oozie is tightly coupled with Hadoop and HDFS, thereby making it harder to deploy computational stages that use other technologies. Cascading [14] is a popular workflow engine for building flexible enterprise data processing solutions without having to worry about how to distribute the workload.

Recently, real-time workflow engines that overcome the batch oriented nature of Hadoop, such as Storm [15] and Spark [16], have become very popular. Storm is a distributed realtime computation system that provides a set of general primitives, Spouts and Bolts, for composing computation stages. However, Storm flow is based on individual items flowing through the system as a stream. Spark is a MapReduce-like cluster computing framework designed to support low-latency iterative computations. Spark aims to overcome the batch-oriented nature of Hadoop by distributing the data in slices and storing it in memory, thereby gaining a significant performance boost. However, Spark maintains a tight coupling with Hadoop and HDFS, making the integration of non-Hadoop distributed systems non-trivial.

C. Why we design our own framework

After evaluating the frameworks mentioned above, we decided to develop our own content understanding engine from first principles for a number of reasons. Firstly, our requirements are such that we need a flexible architecture to combine very different technologies into a uniform pipeline. Secondly, we need the ability to produce detailed output at each stage and carefully study the results. Thirdly, we need the ability to substitute a completely different technology for a given stage without impacting other stages or the final results. Finally, we wanted to leverage best-of-breed Open Source technologies at each stage so that we can focus on the broader solution we need to build.

III. Framework

A. Requirements

The keys to the web content fetching module are flexibility and scalability. The module is typically required to fetch content for millions or even billions of URLs. It has to be flexible enough to handle many exception cases such as invalid URL, redirects, connection timeout, lost connection and so on. It also has to be scalable at the same time in order to fetch from billions of URLs in a reasonable time period.

The key requirements for the web page parsing module and document storage module are scalability, processing speed and accuracy. With billions of web pages, the parsing module and document storage module have to be able to process web pages fast and be scalable at the same time. To process 1 billion web pages with 100 machines where each machine can process a page in 100 ms, the system needs more than 11 days to finish. The parsing module also has to be very accurate in extracting core content from the web page markup to eliminate irrelevant tags, scripts, and ads.

Algorithms and methods play a key role in representing complex articles via short text labels. Web pages are typically written or edited with long natural language text for readability. Contextual profiling and advertising systems cannot directly utilize the web page content because it contains many HTML and scripting tags and commonly used words that do not contribute to the core meaning of the page. Content extraction and summarization systems typically scrape the web pages and convert the natural language text into much shorter words and phrases.

In contextual advertising, marketers typically prefer well-defined categories over free text for reasons of communication, consistency and measurement. Well-defined contextual elements, such as tags and categories, are easy to communicate from marketers to advertising operators. It is also easy to measure the campaign complexity and potential gain and cost with well-organized categories and tags. However, predefined category hierarchies cannot be changed dynamically as new concepts become available. It is also hard for category structures to capture the meaning of articles. On the other hand, free

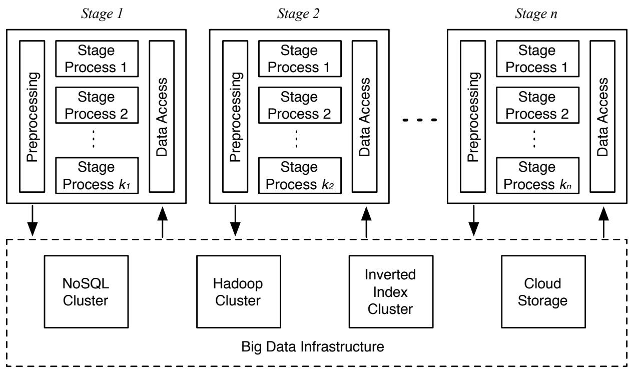

Figure 1. The Framework: separation of Computational Workflow from the Big Data Infrastructure allowing each module to scale separately in a cloud environment.

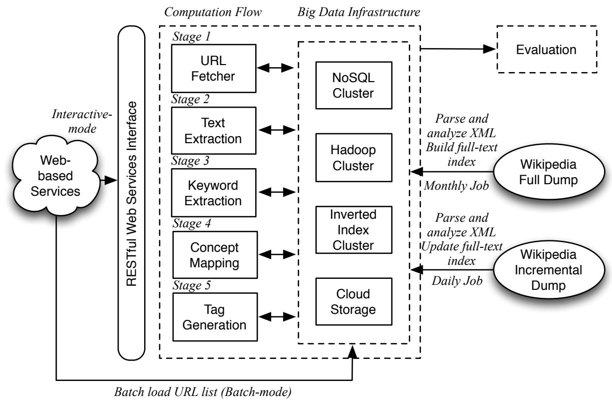

Figure 2. The Content Understanding Engine: internal modules and interfaces.

text summarization provides richer contextual information than categories. However, the huge dimension size makes it harder for advertisers to prepare marketing strategies.

B. Architecture

To balance the granularity of information representation and the ease of management, we propose a Wikipedia based concept system to extract and match contextual tags from news articles to the most relevant Wikipedia categories and concepts. Instead of free text summaries, the CUE generates a set of tags that map the contextual meanings of news articles to user-defined and user-maintained concepts and categories.

We now describe our architecture in detail. Our architecture separates computation-tier from the data-tier. As shown in Fig. 1, the computation-tier consists of a sequence of stages where each stage takes a predefined set of inputs and produces some predefined output. Each stage can be configured to interface with a data-tier cluster to access data-at-rest as well as to distribute large-scale data crunching. Our Heterogeneous Big Data Infrastructure allows us to use different technologies, such as NoSQL Database (e.g., HBase [17], Hadoop, Inverted Index (Solr [18] or ElasticSearch [19]) and Cloud Storage.

The same pipeline can be run in batch-mode (a list of URLs), interactive-mode (one URL in near realtime) or service-mode. In service-mode, a RESTful Web Service interface is provided so that a run can be initiated posting a URL. When initializing the pipeline, a configuration can be provided to specify the sequence of stages that need to be run and the additional parameters needed by each stage. This framework allows us to perform comparative evaluation of different technologies and to understand their relative merits.

The CUE is architected from first principles to address our specific needs. Our approach allows us to combine disparate technologies into a unified pipeline as a sequence of stages with clearly defined input and output for each stage. The first stage in our pipeline is a custom Fetcher that can take a list of URLs and fetch the HTML content of the web page. In the next stage, we extract the main textual content from the web page using a Support Vector Machine (SVM) model trained from many different kinds of web pages (e.g., blogs, news articles, product reviews, etc.). Once the plain text is extracted, we perform sentence and phrase detection followed by keyword extraction in the next stage.

As part of this project, we have built an inverted index of the Wikipedia content, including the titles (concepts) and categories. Thus, in the next stage, we search the inverted index using the extracted keywords (and phrases) from the previous stage. This provides us with the list of Wikipedia concepts and categories related to the original content of the web page. In the final stage, we propagate the relevance scores attached to the concepts through the category hierarchy and find the most relevant categories representative of the original article. Next, we describe the stages of the pipeline in detail.

C. Web Service Interface

As shown in Fig. 2, the CUE offers a RESTful Web Service interface so that it can be integrated into other services. This service can be invoked either in interactive mode or bulk mode. Several applications as mentioned before can be integrated with this service to gain a contextual understanding of a web page. In interactive mode, the service provides a browser interface to view the detailed output from each stage of the pipeline. In bulk mode, a batch of URLs (e.g., from a weblog) can be posted to the service to be processed in the background. For processing very large URL batches, a file location can be provided as input. Files can be local, network-mounted or cloud-based (e.g., Amazon S3). With the latter approach, a large URL batch file can be uploaded to a cloud storage first and CUE can be requested to process this file and send the final output to another local, network-mounted or cloud-based file system.

D. URL Fetcher

This stage (Stage 1 in Fig. 2) takes a list of URL as input and stores the contents of that page in original form including the HTML or other markup present in the document. The fetcher can be configured to deal with the complexities involved in crawling a web page (such as setting the user agent, dealing with scripts, images, etc.). We use the well known techniques for fetching the contents given a set of URLs [6][20]. The harvested raw web content is stored in the NoSQL cluster with the URL plus timestamp as a key and the raw content as the value.

E. Text Extraction

This is an important stage (Stage 2 in Fig. 2) in the pipeline where we wanted to evaluate and use different technologies for extracting text from the downloaded page content from the Fetcher. After evaluating several solutions in this space [6][21][22], we settled on the open source project Boilerpipe [23] since this library yielded the best results for our purposes. The Boilerpipe library provides us a way to strip out all the boilerplate content, such as irrelevant tags, scripts, and ads from the web page and extract only the main content (e.g., the main article of a news page) with high accuracy [24]. This library, under the covers, uses an SVM classifier trained from thousands of web pages to extract the main content of a web page.

F. Keyword Extraction

This stage (Stage 3 in Fig. 2) performs sentence detection, tokenization, phrase detection and keywords extraction. First, we segment the extracted text from the previous stage into sentences by applying several rules to detect the sentence boundary. Then we tokenize each sentence and detect the most frequent one to three word phrases. We experimented with several ways of finding the significant phrases from the extracted text including TF-IDF scores. From this sorted list of phrases, we select the top N (with $N = 20$, say) as the extracted keywords from the original text document. This stage can be configured with the rules for sentence segmentation, stemming or stopword removal [25].

G. Concept Mapping

This stage (Stage 4 in Fig. 2) constructs a query based on the extracted keywords and performs a full-text search. We utilize the Open Source search engine ElasticSearch that uses the Lucene [26] toolkit to build a full-text index of a document collection. Using the Wikipedia Plugin provided by ElasticSearch we have built a full-text index of the entire Wikipedia content. This provides us the ability to search a set of keywords and find the matching Wikipedia Articles and retrieve the Concepts (Titles) and Categories with the highest scores. The underlying Lucene Engine provides scores for the matching concepts that we use as a way of ranking the results. We apply a boosting factor to keyword matches in article titles in order to improve the accuracy of the search results.

H. Tag Generation

This stage (Stage 5 in Fig. 2) finds the related higher-level Wikipedia Categories from the highest ranked Wikipedia concepts from the previous stage. The Wikipedia categories are first cleaned to remove categories that do not represent a meaningful concept (e.g., lists of famous dead people) and to remove cycles. We also perform case normalization since categories for the same concept are present with different capitalization (e.g., *Computer science* and *Computer Science*). After these preprocessing steps, we get a clean graph representation of the Wikipedia category hierarchy.

With the concepts as the leaf-nodes of this graph structure, we propagate the Lucene scores associated with each concept to find the most significant categories that are present at the intersection of several concepts [9]. These categories will be taken as representative of the contents of the page, providing the semantic meaning of the original page.

I. Evaluation Module

Evaluation is the core component of the framework that bridges the automated system and the business applications. A flexible framework has to be able to support variety of evaluation methods. The architecture design of our framework sets evaluation module as a pluggable component which communicates with modules via RESTful services.

IV. Deployment

We deployed our Web Content Understanding Framework to different cloud environments and compared our experiences. For this part of our evaluation, we deployed our system on the following cloud services and conducted experiments with various workloads: (1) Internal Private Cloud, (2) Amazon Web Services, and (3) HP Cloud Services.

Our development environment is an internal private cloud and the computational stages are distributed as shown in Fig. 2. Each stage can be scaled by adding more nodes as necessary. For instance the ElasticSearch server used in the Concept Mapping stage is shared across four nodes to index the entire Wikipedia dump and search matching articles given a set of keywords. With this configuration we were able to reduce the search time to be $< 100\,\text{ms}$. Moreover, we are able to deploy this architecture to an AWS cloud using medium-powered EC2 instances as well as to deploy to an HPCS cloud using a similar server configuration. We are able to deploy and be operational within a day and are able to scale both horizontally and vertically depending on the workload.

V. Use Case: Profiling Weblogs

In this section, we will describe a use case for the CUE - web usage mining to analyze the behavioral patterns and profiles of users interacting with web sites. With the continual growth and proliferation of mobile devices and the Internet of things, the volume of interaction logs generated by web-based service providers has reached several petabytes in size.

Analyzing such data can help organizations determine the life-time value of customers, design cross-marketing strategies, evaluate effectiveness of campaigns and personalize content to visitors. This type of analysis involves the automatic discovery of meaningful patterns and relationships from very large collection of weblogs collected from various operational data sources. Such weblogs typically contain a list of URLs accessed by each user along with information such as (1) the user's IP address, (2) the user's authentication name, (3) the date and time stamp of the access, (4) the HTTP request, (5) the response status, (6) the size of the requested resource, and optionally, (7) the referrer URL and (8) the user's browser identification.

Mobile service providers are collecting these logs from multiple sources. We are using CUE in several projects to enhance user experience with the goal of increasing customer retention and revenue (ARPU) (see Fig. 3). URL collections are processed through CUE to contextualize each URL with concepts and categories. This information can then be associated with user profiles to ascertain the interests of the users.

A. Evaluation

Evaluating the effectiveness of a contextual service is a difficult task. Unlike traditional evaluation of text classification

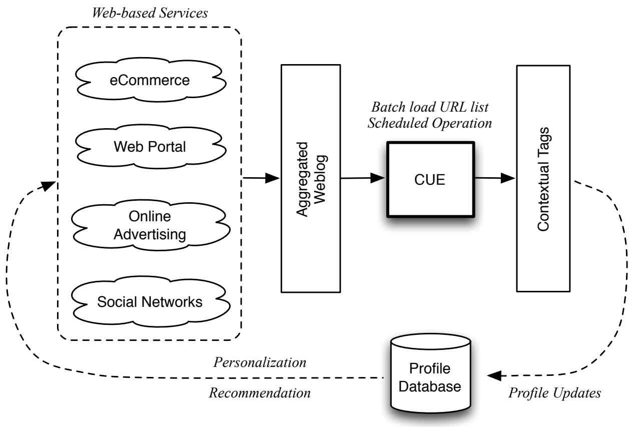

Figure 3. The Content Understanding Engine (CUE) in Context. The figure shows how CUE is deployed along with existing web based services.

systems where the metric is the accuracy of articles being classified to a predefined category, the effectiveness of a contextual service can be measured from different perspectives when they are used for different goals. Accuracy does not always translate to marketing strategies and goals directly. In digital advertising, marketing strategies and goals lead to different measurements of successfulness.

The effectiveness of the contextual profiling from web usage logs can be examined from different perspectives when the results are used in different applications. The contextual profiling of web users is typically used for online advertising. Advertisers can design marketing strategies based on users' content consumption and trends. For the advertising application use case, we implemented three evaluation modules, consistency, accuracy, and usefulness, to measure the effectiveness of the system.
We use three metrics to evaluate our CUE system:

1) *Consistency*, the ability of a tagging system to produce similar results for similar contents, is one of the most important measures in advertising applications. The wording of news articles can be significantly different for different news sources. The free text based approach may generate very different summaries for the same news content from different sources, while the classification based approach typically fails at capturing the details of the content and the model can rapidly become outdated.
   We want the output of CUE conceptual tags of, for example, the "Crimea Crisis" news story from New York Times and from CNN to be similar, so that advertisers are assured that users on different websites are consuming the same content.
   For a given topic, the consistency evaluation module collects news articles from several news sources. Label these articles $1, \dots, K$, say. For the kth article, let A_k denote the set (actually, ranked list) of conceptual tags output by CUE from this article. The CUE outputs the same number n $(= 10$, say) of conceptual tags from each article, so $|A_k| = n$ for all k. For each topic, we compute two consistency metrics:
   a) The *overlap consistency* for each pair (i, j): the Sørensen-Dice similarity of A_i and A_j,
   $$C_{i,j}^{\text{overlap}} = D(A_i, A_j) = \frac{2|A_i \cap A_j|}{|A_i| + |A_j|}. \quad (1)$$
   The overlap consistency shows the degree of overlap of tags for a given topic.
   b) The *average rank correlation*: the average of the Kendall tau rank correlation coefficient $\tau(A_i, A_j)$ over all distinct tags for the topic,
   $$\overline{C}^{\text{rank}} = \frac{\sum_{i=1}^{K-1} \sum_{j=i+1}^{K} \tau(A_i, A_j)}{|\cup_{k=1}^{K} A_k|}. \quad (2)$$
   The average rank correlation shows the consistency of the ranking of tags for a topic.
2) *Accuracy* is defined as the degree of agreement between automatically generated tags and professional editors. This metric reflects how accurate the CUE tagging system is when processing news articles. We collect K Wikinews articles with editor-labeled conceptual tags and measure the accuracy as
   $$\text{Accuracy} = \frac{1}{K} \sum_{k=1}^{K} D(A_k, B_k), \quad (3)$$
   where for the kth article, A_k is the set of CUE tags, while B_k is the set of editor-assigned tags.
3) *Meaningfulness* is defined as how a general news reader agrees with the automatically generated tags. To measure the meaningfulness of the CUE system, a traditional expert rating evaluation module is attached to the system. The module takes the output of CUE system, randomly selects and ranks URLs and corresponding conceptual tags and then presents the results to experts. Experts rate each conceptual tag for the level of relevance to the article pointed by the corresponding URL. The rating results are summarized by the module and the relevance score will be presented automatically.

B. Experiments

To measure consistency, we use Google news as our data source. In Google news, articles from different news sources are grouped together. We collected 425 news topics with 2,571 articles. CUE system generated contextual tags for each articles. The consistency score distribution comparing to random scoring is shown in Fig. 4 and Fig. 5. The relatively low consistency scores are due to the strict matching rule implemented in the module where only the exact concept matchings are counted as success. For example, "variance" and "standard deviation" are considered not a match although they are conceptually close. More relaxed matching algorithms should get a higher score.

We also collected 278 articles from Wikinews with categories assigned by editors. Our CUE system has an average accuracy of 35.25% (see Eq. 3). Finally, 5 experts are hired to evaluate the meaningfulness of tags generated by CUE from 411 news articles. An average of 41% of tags are considered to be meaningful to the articles by the experts. The results show that the CUE system is fairly useful to identify key concepts despite the limitations of our evaluation methods. The accuracy and meaningfulness are greatly influenced by the strict

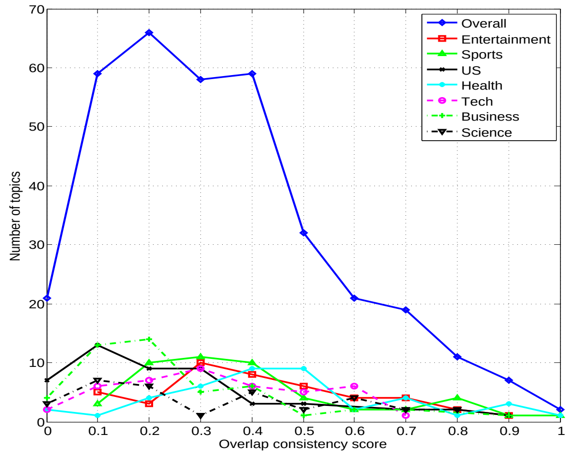

Figure 4. Overlap consistency score distributions for the top categories of news articles.

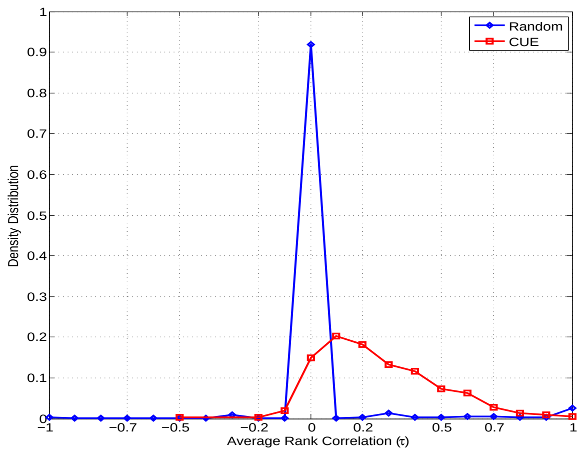

Figure 5. Rank correlation consistency distribution for CUE compared to randomly-assigned tags from top categories.

matching rule and binary judgments, where closely connected or similar tags are not considered a match. We are working on more fair and sophisticated matching algorithms that are appropriate for complex hierarchical matching problems.

VI. Conclusions and Future Work

This paper presented a scalable and modular architecture of a web Content Understanding Engine (CUE) for finding contextual tags for potentially large collection of URLs. There are two key aspects to the software framework presented in this paper: (1) computational flow that allows different technologies to be composed into a unified pipeline and (2) Big Data infrastructure that allows the use of different technologies such as NoSQL Database, Hadoop, Full-text Index or Cloud Storage. We are using CUE in several projects to enhance the user experience of web-based services with the goal of increased customer retention and incremental revenue. One specific example use of CUE is to generate trending concepts on a daily basis from URLs harvested from popular news aggregators and to leverage this information to produce better recommendations. Our further work is to explore alternative technologies for each stage of the computation flow to optimize the end-to-end scalability and performance of the CUE.

References

[1] A. Addis, G. Armano, and E. Vargiu, "Profiling users to perform contextual advertising," in Proc. 10th Workshop dagli Oggetti agli Agenti (WOA), Jul. 9–10, 2009, Parma, Italy, 2009, URL: http://cmt.math.unipr.it/woa09/papers/Addis2.pdf [accessed: 2014-05-01].

[2] G. Armano, A. Giuliani, and E. Vargiu, "Experimenting text summarization techniques for contextual advertising," in Proc. 2nd Italian Information Retrieval Workshop (IIR) Jan. 27–28, 2011, Milan, Italy, ser. CEUR Workshop Proceedings, vol. 704. CEUR-WS.org, 2011, Melucci, M., Mizzaro, S., and Pasi, G., Eds., ISSN: 1613-0073, URL: http://ceur-ws.org/Vol-704/12.pdf [accessed: 2014-05-01].

[3] G. Armano, A. Giuliani, A. Messina, M. Montagnuolo, and E. Vargiu, "Content-based keywords extraction and automatic advertisement associations to multimodal news aggregations," Studies in Computational Intelligence, vol. 439, Jul. 2011, pp. 33–52, ISSN:1860-949X.

[4] J.-H. Lee, J. Ha, J.-Y. Jung, and S. Lee, "Semantic contextual advertising based on the open directory project," ACM Trans. Web, vol. 7, no. 4, Oct. 2013, pp. 24:1–24:22, ISSN:1559-1131.

[5] B. Zamanzadeh, N. Ashish, C. Ramakrishnan, and J. Zimmerman, "Semantic advertising," CoRR, vol. abs/1309.5018, 2013, URL: http://arxiv.org/abs/1309.5018 [accessed: 2014-05-01].

[6] S. Chakrabarti, Mining the Web: Discovering Knowledge from Hypertext Data. Morgan-Kaufmann Publishers, San Francisco, 2003, ISBN: 978-1558607545.

[7] T. W. Yan, M. Jacobsen, H. Garcia-Molina, and U. Dayal, "From user access patterns to dynamic hypertext linking," Comput. Netw. ISDN Syst., vol. 28, no. 7-11, May 1996, pp. 1007–1014, ISSN: 0169-7552.

[8] R. Cooley, B. Mobasher, and J. Srivastava, "Web mining: information and pattern discovery on the world wide web," in Proc. Ninth IEEE Intl. Conf. Tools with Artificial Intelligence, Nov. 3–8, 1997, Newport Beach, CA, USA. IEEE, Nov. 1997, pp. 558–567, ISSN: 1082-3409.

[9] M. Strube and S. P. Ponzetto, "Wikirelate! computing semantic relatedness using wikipedia," in Proc. 21st Natl. Conf. Artificial Intelligence, Jul. 16–20, 2006, Boston, ser. AAAI'06, vol. 2. AAAI Press, Jul. 2006, pp. 1419–1424, ISBN: 978-1-57735-281-5.

[10] Hadoop. [Online]. Available: http://hadoop.apache.org/

[11] Oozie. [Online]. Available: http://oozie.apache.org/

[12] Azkaban. [Online]. Available: http://data.linkedin.com/opensource/azkaban/

[13] N. Christofides, Graph theory: An algorithmic approach. Academic press New York, 1975, vol. 8.

[14] Cascading. [Online]. Available: http://www.cascading.org/

[15] Storm. [Online]. Available: http://storm.incubator.apache.org/

[16] Spark. [Online]. Available: http://spark.apache.org/

[17] Hbase. [Online]. Available: http://hbase.apache.org/

[18] Solr. [Online]. Available: http://lucene.apache.org/solr/

[19] Elastic search. [Online]. Available: http://www.elasticsearch.org/

[20] C. C. Aggarwal, F. Al-Garawi, and P. S. Yu, "Intelligent crawling on the world wide web with arbitrary predicates," in Proc. 10th Intl. Conf. World Wide Web, May 1–5, 2001, Hong Kong, ser. WWW '01. ACM, May 2001, pp. 96–105, ISBN: 1-58113-348-0.

[21] S.-H. Lin and J.-M. Ho, "Discovering informative content blocks from web documents," in Proc. Eighth ACM SIGKDD Intl. Conf. Knowledge Discovery and Data Mining, Jul. 23–26, 2002, Edmonton, Canada, ser. KDD '02. ACM, Jul. 2002, pp. 588–593, ISBN: 1-58113-567-X.

[22] S. Debnath, P. Mitra, and C. L. Giles, "Automatic extraction of informative blocks from webpages," in Proc. 2005 ACM Symp. Applied Computing, Mar. 13–17, 2005, Santa Fe, USA, ser. SAC '05. ACM, Mar. 2005, pp. 1722–1726, ISBN: 1-58113-964-0.

[23] Boilerpipe. [Online]. Available: http://code.google.com/p/boilerpipe/

[24] C. Kohlschütter, P. Fankhauser, and W. Nejdl, "Boilerplate detection using shallow text features," in Proc. Third ACM Intl. Conf. Web Search and Data Mining, Feb. 3–6, 2010, New York, ser. WSDM '10. ACM, Feb. 2010, pp. 441–450, ISBN: 978-1-60558-889-6.

[25] C. D. Manning, P. Raghavan, and H. Schütze, Introduction to Information Retrieval. Cambridge University Press, New York, 2008, ISBN: 978-0521865715.

[26] Apache lucene. [Online]. Available: http://lucene.apache.org/