Interactive Exploration for Domain Discovery on the Web

Yamuna Krishnamurthy
yamuna@nyu.edu

Kien Pham
kien.pham@nyu.edu

Aécio Santos
aecio.santos@nyu.edu

Juliana Freire
juliana.freire@nyu.edu

Tandon School of Engineering
New York University

ABSTRACT

As the volume of information on the Web grows, it has become increasingly difficult to find web pages that are relevant to a specific domain or topic. In this paper, we explore the general question of how to assist users in the domain discovery process. Domain discovery entails the translation of a user's information needs and conceptual view of a domain into a computational model that enables the identification and retrieval of relevant content from the Web. We discuss the challenges and propose an initial approach based on exploratory data analysis that combines techniques from information retrieval, machine learning and data mining to streamline domain discovery. We implemented the approach in an open-source tool and present the results of a preliminary evaluation.

Keywords

Exploratory data analysis, exploratory search, human interaction, visualization, information retrieval, text mining, focused crawling

1. INTRODUCTION

Domain discovery is the process through which a user identifies and retrieves information and sources from the Web that are relevant for a specific information need. Consider the following scenario. Analysts at a law enforcement agency that is tasked with investigating and preventing the illegal use and trafficking of firearms regularly search the Web to discover and track potentially illicit activities. They want to find suspicious brokers and online stores, forbidden weapons for sale, reports of stolen weapons, and leads into trafficking activities. While they have a clear idea of the information they need, finding this information on the Web is challenging. They often start by issuing queries to Google or Bing using keywords such as "AR15" or "no paperwork", which based on their prior knowledge, provide a good indication of illegal weapon sales. While search engines provide broad coverage of the Web, for domain specific searches they have an important drawback: they return a very large number of irrelevant results. Figure 1 shows results from Google for the queries `ar15 no paperwork` and `sell ar15 no paperwork`. Most of these results are not related to the sale of the weapons with no paperwork.

The experts need to analyze the results of the search either by reading the snippets returned by the search engine or the actual pages. When they identify a relevant page which contains information like phone numbers, user ids in forums and images, they bookmark or save it locally. As they perform multiple investigations, it is easy to lose the search context. Moreover, content on the Web is very dynamic: existing pages change or are deleted, and new pages are added at a high rate. Thus, just keeping track of URLs visited is not sufficient. To maintain the information up-to-date and discover new relevant content, the expert must continuously query the search engine. This process is time consuming and tedious.

Another challenge lies in formulating keyword queries. While these queries are simple, selecting the right terms for a domain-specific search can be daunting. The representation of a given domain on the Web can differ from what an analyst expects, and the analyst may not be aware of certain nuances. During exploration, by examining pages returned by a search engine, the analyst can discover other related terms. For example, as illustrated in Figure 1, another term that appears in the search results that is potentially related to illegal activity is `background check`. Currently, analysts have to read the pages, manually record the keywords of interest they discover, and later use these keywords as additional queries. This clearly does not scale.

The simplicity of keyword queries is a strength and also a limitation. In theory, an analyst could improve the relevance of the results by issuing more specific queries. For example, an analyst could search all forums and user ids associated with the sale of a particular illegal weapon. Or when she finds a user in a forum who posted an ad for a gun without paperwork, she would like to check whether this user is active in other forums. Such queries cannot be expressed using the interfaces supported by search engines.

These challenges are commonplace in many different tasks, from tracking criminal activities to understanding how research areas evolve over time.

Contributions. To address these challenges, we developed a visual analytics framework for interactive domain discovery that augments the functionality provided by search engines to support analysts in exploratory search. The framework (1) supports exploratory data analysis (EDA) [27] of web pages, and (2) translates the analyst's interactions with this data into a computational model of the domain of interest. By organizing and summarizing the search results,

KDD 2016 Workshop on Interactive Data Exploration and Analytics (IDEA '16)
August 14th, 2016, San Francisco, CA, USA.

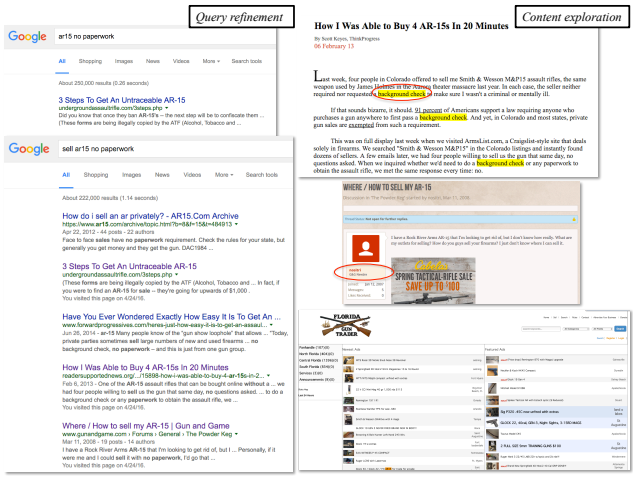


Figure 1: Searching for assault rifles sold with out proper paperwork

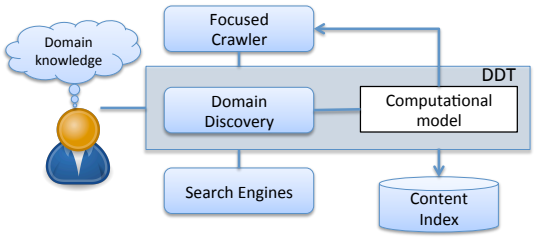


Figure 2: Interactive Domain Discovery

the framework helps users better understand and analyze the retrieved content as well as provide feedback. By clustering the retrieved pages and grouping similar pages together, it simplifies the process of selecting and annotating the pages. It also automatically extracts important keywords and phrases in the pages. These not only serve as a summary of the content, but also as suggestions for new queries to be issued. In the course of exploration, the search context is maintained: queries and their results are persisted, allowing users to revisit and analyze the content. The annotations provided by the users regarding the relevance of pages is used to build the domain model, a computational specification of the domain, which can then be used to configure a focused crawler [2, 4]. The focused crawler, in turn, provides a scalable mechanism to retrieve additional pages which feeds back into the domain discovery process. Figure 2 illustrates the interactive domain discovery process.

We have implemented the framework in the Domain Discovery Tool (DDT),[1] an open-source system that implements these mechanisms. We also report the results of a preliminary user evaluation.

2. RELATED WORK

Search user interfaces have been extensively studied and implemented. Hearst [8] provides a comprehensive summary of work on search interface design. She also discusses the broader problem of *sensemaking* [16, 20, 21], the "iterative process of formulating a conceptual representation from a large volume of information", and argues that "the standard Web search interface does not do a good job of supporting the sensemaking process". Tools such as Sandbox [28] provide an interface for advanced analysis of the information gathered, from various sources, by allowing free-form organization of retrieved results. However, it misses an important step in sensemaking: the collection of a good set of resources, representative of the domain. Search is an integral part of this domain discovery process which enables analysis and information extraction. The framework we propose provides this missing step towards sensemaking.

Vertical search engines focus on a specific segment of online content. For example, Google Scholar[2] stores information about scientific publications and Yelp[3] helps users find information about local businesses. These systems have several benefits over general search engines. Notably, because of their limited scope, they return more relevant results and lead to higher precision. In addition, they support domain-specific tasks. For example, in Google Scholar, it is possible to search for papers written by a given author. These vertical engines, however, are expensive to build and hence they are available only for broad topics of wide interest. Our framework allows exploration of any given domain, and is a step towards lowering the costs of building vertical search engines for any domain available on the Web.

Focused crawling [2, 4] has been proposed as a scalable mechanism to gather data about specific domains from the Web. In order to bootstrap a focused crawler, it is necessary to provide a set of seed URLs that serve as the starting points for the crawl, and a page classifier that can decide whether a retrieved page is relevant or not. While these systems are effective and address many of the challenges discussed previously, they require substantial human input. Collecting a set of positive and negative examples to train classifiers that recognize the target concept is time consuming; and as new pages are obtained by the crawler, the classifier needs to be iteratively refined. Not surprisingly, focused crawlers have not been widely adopted. The framework proposed in this paper helps to solve the crawler bootstrapping problem by helping the user to acquire seed URLs and build models to classify Web pages.

Domain discovery requires exploration of text corpora gathered from the Web. *Interactive text mining* techniques help address this problem. Over the years there has been substantial research on various aspects of interactive text mining, for both web and other documents, such as clustering [5, 11, 12, 15, 17, 23, 30], topic modeling [10, 29], and semantic analysis [25]. Interactive applications like STREAMIT [1] and i-GNSSMM [14] have attempted to bring some of this work together to analyze web documents. However, STREAMIT assumes the existence of an external continuous source of documents. The user cannot add documents to this source during exploration using STREAMIT. It does not allow users to annotate the documents and create their own clusters – users can only tweak certain parameters to adjust the system's clustering algorithm. i-GNSSMM extracts the topic graph from a collection of web pages. Although this could be a useful representation of the content it is not always appropriate for a user's information seeking needs.

[1]DDT is available at https://github.com/ViDA-NYU/domain_discovery_tool. For demos see: https://youtu.be/XmZUnMwI10M, https://youtu.be/YKAI9HPg4FM, https://youtu.be/HPX8lR_8QS4.

[2]http://scholar.google.com
[3]http://www.yelp.com

There are also a number of text mining software packages.[4] Since these require technical expertise to configure and use them, they are out of reach for domain experts without training in computing.

Recent work in *dynamic search* [31] improves search over time by learning from the user's interaction with the system. This work is complementary to our effort. We may leverage dynamic search to improve the search and filtering of documents in our framework. More closely related to our framework is the *intent modeling* work by Ruotsalo et al. [19]. But this work uses only the feedback of important keywords to model the intent of the user. It does not allow the users to provide feedback on the relevance of documents or group them as they see fit.

3. DESIDERATA OF INTERACTIVE DOMAIN DISCOVERY

This work was originally motivated by challenges of domain specific search that were encountered as part of the DARPA Memex program.[5] In this project, we have interacted with experts with a wide range of information needs in different domains, including human trafficking, sale of explosives, illegal weapons and counterfeit electronics, micro cap fraud and patent trolls. In what follows, we discuss the desiderata for domain discovery based on our interactions with these experts, their feedback on existing state-of-the-art tools, and information needs.

Translation of conceptual definition of a domain into a computational model. Domain definition and discovery can be viewed as the iterative process of mapping an expert's conceptual view of a domain into a set of artifacts available on the Web (e.g., web sites, web pages, terms, phrases and topics). Since the human is clearly the biggest bottleneck in this process, we need usable and scalable mechanisms that guide and support the user. Capturing the domain definition as a computational model enables this process to scale: with such a model, automated processes can be deployed to retrieve relevant information.

Data gathering. Analysts use various information retrieval mechanisms to collect relevant data for subsequent analysis. Some of the common mechanisms include, but are not limited to, web searches to locate new information and uploading already known relevant web pages. As they identify relevant pages, they often crawl forward and backward in an attempt to find additional content. A tool for domain discovery should support these mechanisms and make them easy to use.

Maintaining search context and capturing user feedback. Search engines treat each query independently. While there is a notion of *session* which either refers to a specific period of time or the linear chain of links followed, in domain discovery the context should take the domain into account. This would be an aggregation of sessions of exploration of that domain. The history and bookmarking mechanisms allow users to save some of the search context, but it is hard to reason about and revisit previously viewed content. This is a major roadblock for domain discovery. The search context should include queries issued, pages retrieved, indications provided by users regarding the relevance of both pages and keywords extracted from them. This information should be readily available and easily interrogated.

Summarizing search results. The simple list of links with snippets provided by existing search engines is not sufficient for quick analysis and annotation of pages especially when the number of results returned is large. The list fails to provide an overview of the results. As we discuss in Section 4, we explored different techniques to better summarize the information.

Streamlining annotations. An important component of domain discovery is user feedback regarding the relevance of pages and sites. This feedback is essential to: Guide users in the process of understanding a domain and help them construct effective queries to be issued to a search engine; and configure focused crawlers [4] that efficiently search the Web for additional pages on the topic by using the feedback to build page classifier models and gather seed URLs.

Exploring and Filtering Results. Once a set of pages is gathered for a particular domain, the experts, as part of their investigation, benefit from exploring subsets of these results. Useful filtering mechanisms include, for example, filter by keywords or specific time period.

Minimal setup and configuration. Analysts working on domain discovery do not necessarily have technical expertise to setup and configure tools and applications. They usually require systems that have a simple, intuitive, visual and interactive interface that has a very low learning curve with minimal or no configuration required.

4. DOMAIN DISCOVERY TOOL

Informed by the desiderata described in Section 3, we designed a framework to support domain discovery. The framework aims to support users in the construction of a computational model of their conceptual view of the domain. To achieve this, it includes several mechanisms that aid analysts to explore, interact with and learn about the domain from the Web content retrieved, and that also gather user feedback . The mechanisms, which we describe below, combine techniques from data mining, machine learning and information retrieval, and their results are presented to the expert through interactive visualizations. They were implemented in Domain Discovery Tool (DDT), whose user interface is shown in Figure 3.

4.1 Data Gathering and Persistence

Search context is maintained by persisting it in an index[6], created for each domain, where all the domain specific exploration activities are stored. Domain experts can use a variety of methods to make pages of interest available for analysis through DDT.

Querying the Web. DDT allows users to query the Web using Google or Bing. They can leverage the large collections already crawled by the search engines to discover interesting pages across the Web using simple queries. Since search engines only return the URLs and associated snippet, DDT downloads the HTML content given the URLs and stores it in the selected domain's index. This content can be used later for analysis of the domain and also as seeds for fo-

[4]https://en.wikipedia.org/wiki/List_of_text_mining_software

[5]http://www.darpa.mil/program/memex

[6]Our prototype makes use of an elastic search index: https://www.elastic.co/products/elasticsearch

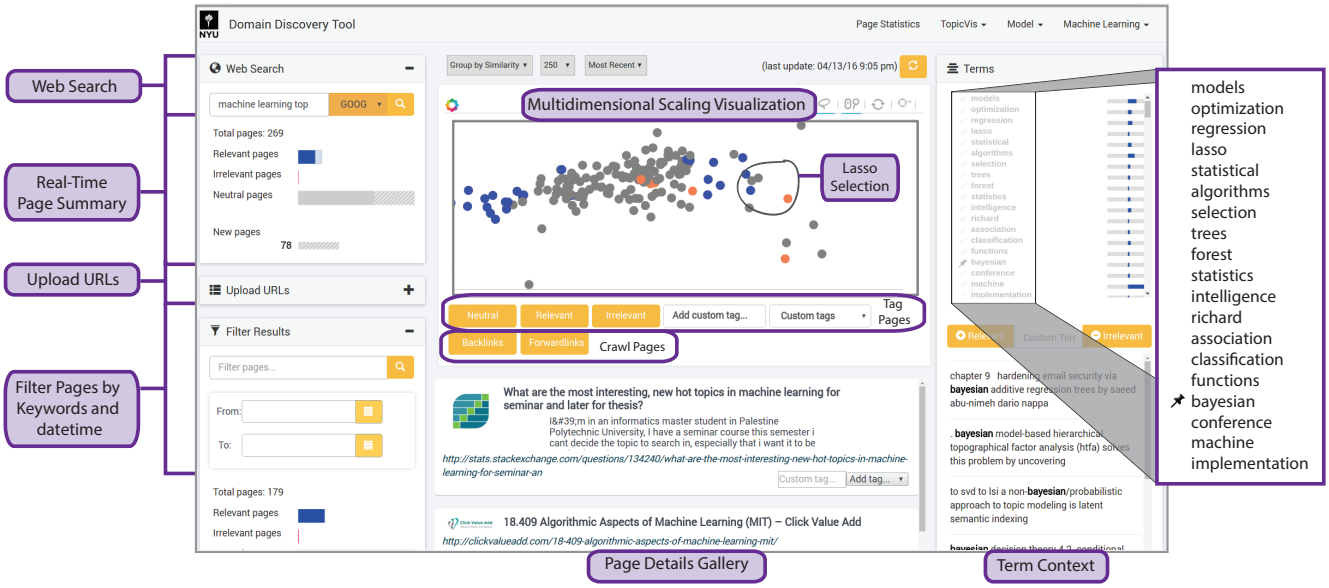


Figure 3: Domain Discovery Tool Interface Components

cused crawlers. Since downloading a large number of pages (including the raw HTML content) takes significant time, DDT performs this operation in the background.

Uploading URLs. In our interviews with experts and use cases we explored, experts often have a set of sites (or pages) they know are relevant. Therefore, it is important to provide a mechanism for incorporating this background knowledge. DDT allows users to provide URLs either through the input box provided or by uploading a file containing a list of URLs. DDT then downloads the pages corresponding to these URLs and makes them available through its interface.

Forward and Backward Crawling. While users can manually follow links forward and backward from the pages they explore, this process is tedious. DDT automates these tasks. Given a page, crawling backwards retrieves the backlinks (the resources that contain a link to the selected page) of that page and then downloads the corresponding pages. Forward crawling from a selected page retrieves all the pages whose links are contained in that page. The intuition behind the effectiveness of these operations is that there is a high probability that the backlink of a page and the page itself will contain links to other pages that are relevant.

4.2 Visual Summary of Search Results

To provide the analyst an overview of the pages they have explored, DDT summarizes them visually in different ways.

4.2.1 *Multidimensional Scaling*

DDT uses multidimensional scaling (MDS) (see Figure 3) for visualizing the retrieved content. Instead of displaying just a list of snippets, DDT applies MDS to create a visualization of the retrieved pages that maintains the relative similarity and dissimilarity of the pages. This allows the user to more easily select (e.g., using lasso selection), inspect and annotate a set of pages.

MDS is currently achieved by principal component analysis (PCA) [26] of the documents. Since initially all pages are unlabeled we need an unsupervised learning method to group the pages by similarity. Note that other unsupervised clustering methods such as K-Means [7] and hierarchical clustering [18] can be used, and we plan to explore these in future work.

To improve scalability, instead of using the sparse *document*×*term* (words in a document) matrix of TF-IDF [22] as the input to the scaling algorithms, we use Google's word2vec [13] pre-trained vectors that were trained on part of Google News dataset (about 100 billion words). The model contains a 300-dimensional vector for each word in a set $W2V$ of 3 million words and phrases. The archive is available online as GoogleNews-vectors-negative300.bin.gz[7].

We convert each document using the word vectors as follows. Let $D = \{d_1, d_2, ..., d_n\}$ be the set of documents to be scaled. $\forall d \in D$ *let* $W_d = \{w_1, w_2..., w_m\}$ where W_d is the set of all words in the document (after removing stopwords). The word2vec archive provides a 300 dimension vector $V = \{v_1, v_2, ..., v_{300}\}$, $\forall w \in \{W_d \cap W2V\}$. So $\forall d \in D$ the vector corresponding to $d = \frac{\sum V_w,\ \forall w \in \{W_d \cap W2V\}}{|\{W_d \cap W2V\}|}$. This generates an input matrix of dimension $n \times 300$, where n is the number of documents, which is much smaller than the original *document* × *term* matrix. By mapping words to word vector representations, we saw a significant improvement in the speed of scaling computation, and also got the benefits of a word vector representation trained on a large text corpus.

4.2.2 *Descriptive Statistics*

Real-time Page Statistics. As new pages are retrieved, DDT dynamically updates the following statistics:

- *Total pages* - total number of pages in the domain
- *Relevant pages* - number of pages marked as relevant
- *Irrelevant pages* - number of pages marked as irrelevant
- *Neutral pages* - pages that have yet to be annotated

[7]https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit?usp=sharing

- *New pages* - number of pages downloaded in the background since last update. This indicates that there are new pages yet to be analyzed.

Page Statistics Dashboard. This dashboard, shown in Figure 4, displays various statistics over the entire content in the domain such as the distribution summary of sites, the distributions and intersections of the search queries issued, summary of page tags and their intersections and number of pages added to the domain over time. This provides the user a map of the domain represented by the retrieved pages.

Topic Distribution Dashboard. This dashboard, shown in Figure 5, visualizes the various topics contained in the domain. The topics are generated using the Topik[8] topic modeling toolkit. Topics can be generated using either LDA [3] or PLSA [9]. Visualization of the topics is done with LDAvis [24]. It shows the topics, the overlap of topics and the most frequent words contained in each topic.

4.2.3 *Keywords and Phrases Extraction*

The keywords and phrases extracted from the pages displayed in the MDS window are shown in the Terms window. They provide a summary of the content of the result pages from which the analyst can learn new information about the domain and use some of the keywords and phrases as new search queries to retrieve additional pages from the Web.

An example is shown in the zoomed region in Figure 3, which shows important terms for the "*Machine Learning*" domain. The initial set of keywords and phrases (bi-grams and tri-grams) displayed are the ones with high TF-IDF [22] in the retrieved pages. But as the pages are annotated, the keywords and phrases are selected from the pages that are annotated as relevant.

When the user hovers the mouse over a term, snippets of result pages that contain the corresponding keyword or phrase are shown below the Terms window. This helps to better understand the context in which the keyword and phrase appear.

4.3 User Annotations

DDT allows users to provide feedback for both pages and terms extracted. In addition to marking individual pages, users can select a group of documents for analysis and mark them as relevant (or irrelevant). Users may also annotate pages with user-defined tags. These tags are useful to define sub-domains, for example, in the "*Machine Learning*" domain we can have sub-domains like "*Deep Learning*" and "*Generative Models*"

Users can also mark the keywords and phrases extracted by DDT as relevant or irrelevant. Based on the relevant terms, the system re-ranks the untagged keywords and phrases, by relatedness to the relevant terms using Bayesian sets [6]. This brings in more related terms and phrases that help the user both understand the domain further and formulate new search queries. Given a query consisting of a few items of the cluster, the Bayesian sets algorithm retrieves more items belonging to that cluster. It achieves this by computing a score for each item, that indicates how related it is to the query cluster. We modeled our ranking of untagged terms and phrases, based on a few tagged terms and phrases, to this setting. The terms and phrases that are marked as relevant by the user make the query cluster. Each term or phrase is represented by a binary vector of all documents in the corpus. The binary value 1 indicates that the term or phrase occurs in the corresponding document and 0 otherwise. So we have two sparse binary matrices as inputs to the Bayesian sets algorithm, (1) *query terms* $\times$ *documents* and (2) *untagged terms* $\times$ *documents*. The output is a list of the untagged terms ranked in the decreasing order of their score. We chose to use Bayesian sets as it computes the score exactly using a single sparse matrix multiplication, making it possible to apply the algorithm to very large datasets, which in our case is the large vocabulary of the corpus.

[8]http://topik.readthedocs.io/en/latest/

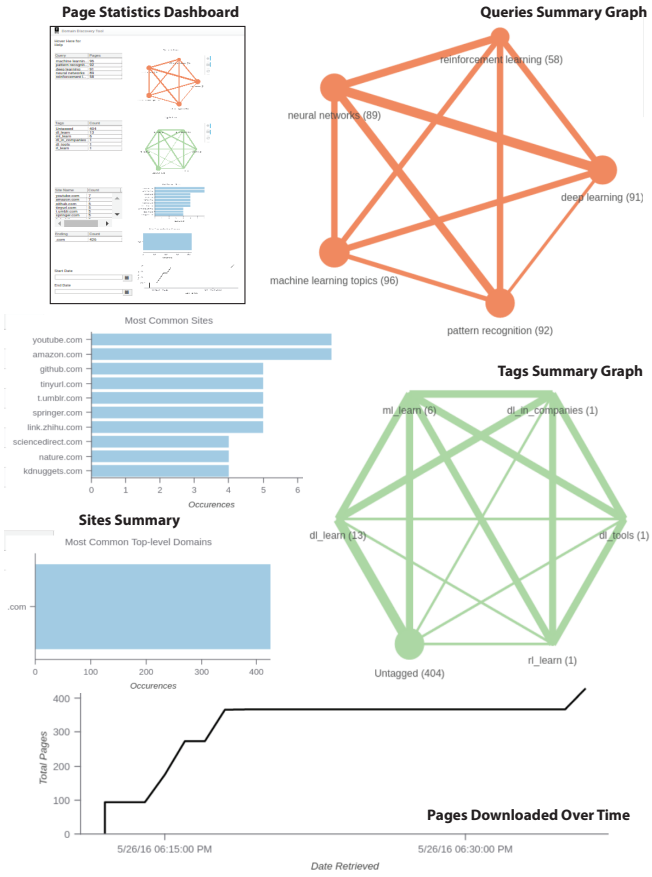


Figure 4: Page Statistics Dashboard

Users may also integrate background knowledge by adding custom keywords and phrases. To guide users and provide them a better understanding of the importance and discriminative power of the extracted terms, DDT shows the percentage of relevant and irrelevant pages the keyword or phrase appears in.

4.4 Domain Model and Focused Crawling

By using the pages marked relevant and irrelevant as positive and negative examples, respectively, DDT supports the construction of a page classifier which serves as a model for the domain. This classifier together with a set of seeds (relevant pages) can be used to configure a focused crawler. In DDT, we support the ACHE [2] crawler.

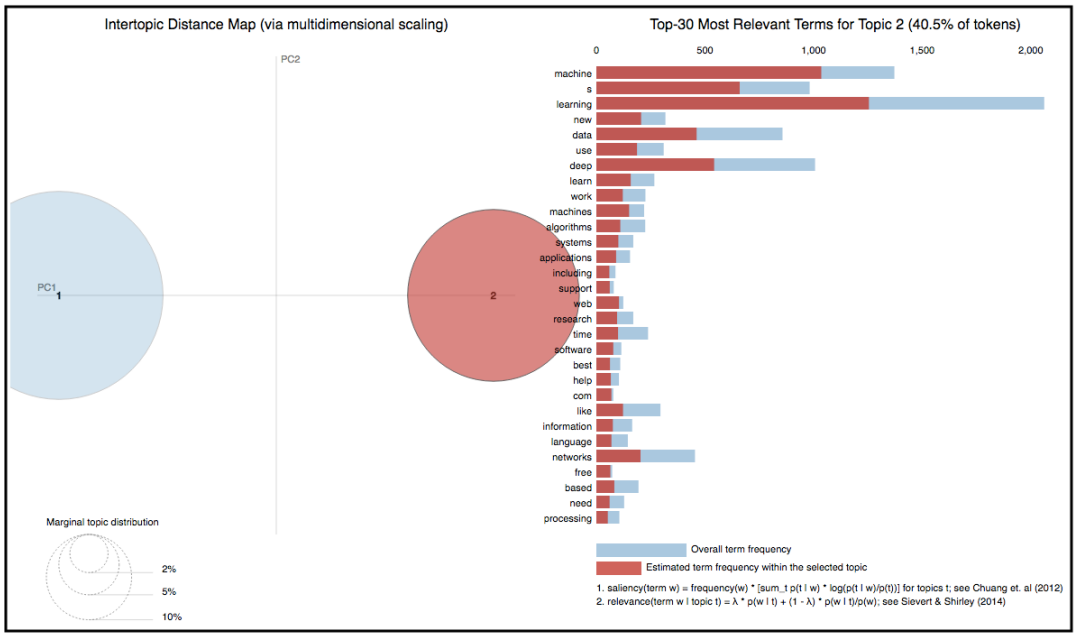


Figure 5: Topic Distribution Dashboard

4.5 Implementation

DDT is designed as a client-server model. The client is a web-based Javascript interface. This ensures that there is no client side setup as the analysts using the system could have a non-technical background.

The server is a Java and Python based cherrypy server that supports multiple clients simultaneously. The core features and functionality of DDT's domain search interface are shown in Figure 3. DDT is also packaged as a docker[9] container for easy deployment on the server.

5. USER EVALUATION

As an initial validation for our design decisions, we carried out a small-scale study. Since search engines are the most common tool used for gathering information on the Web, our study compares the effectiveness of DDT with that of Google for gathering information in a specific domain.

5.1 Experimental Setup

The evaluation involved six participants. The participants were graduate students or research associates with background in computer science. The two primary criteria for their selection was (1) that they should be very familiar with using search engines, especially Google and Bing, and (2) they should be capable of exploring information about a given topic on the Web. The users were given a demo of DDT and all its features, and they were allowed to use DDT to get familiar with it before the actual evaluation.

In order to keep the topics easy to understand and to ensure that the participants were not experts in the domain (as the goal here is for them to discover the topics), we selected topics from the *Ebola* domain in the TREC Dynamic Domain (DD) Track 2015 dataset[10]. The dataset for the *Ebola* domain consists of $\sim 143,044$ pages of which $\sim 5,832$ pages are labeled by humans into 20 topics.

Each user was then given the same 2 topics, in the same domain, and asked to find as many pages as they could for each of those topics using Google and DDT. While using Google the users annotated pages relevant to each topic by bookmarking them under corresponding folders. For DDT, the users annotated the pages for each topic with a custom tag corresponding to that topic. They were allowed 15 minutes for each topic on Google and DDT.

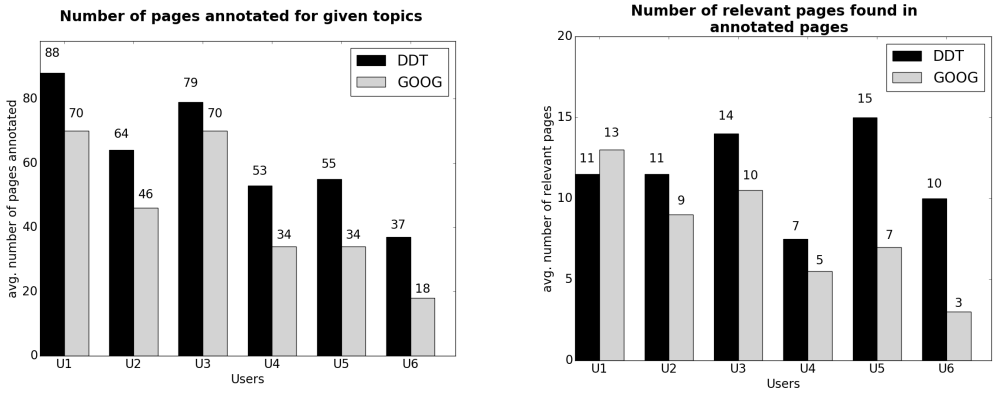
Since we used Google as a search engine in our experiments, we needed a "domain expert" that could consistently judge whether a page annotated by a user belonged to the given topic or not. Since we did not have access to such an expert directly, we instead built a multiclass SVM classifier[11], using the TREC DD data that was labeled by humans. The words (excluding stopwords) in the pages were used as features and the topic a page belonged to was the output class. The model was tested using cross validation which produced an average accuracy of 74.6%. Given the topic distribution, where the most frequent topic consisted of 700 pages, the model is still quite good, as a max baseline accuracy, if we labeled all samples with the most frequent topic label, would be $(700/5832) * 100 = 12\% << 74.6\%$.

5.2 Results

We measured the total number of pages that the users were able to annotate with Google and DDT. We executed the model on the annotated pages to find how many of them were actually relevant to the given topics. The results are shown in Figure 6.

Figure 6a plots the average number of pages annotated for the topics by each user. Users were able to annotate more pages using DDT than Google. Users reported that visualization and grouping of the pages by similarity made it easier for them to select and annotate a set of pages. Whereas on Google, they had to go through the list of results on multiple pages to be able to find the relevant pages and then bookmark them individually.

[9]https://www.docker.com/what-docker
[10]http://trec-dd.org/2015dataset.html
[11]We used LinearSVC from the scikit-learn library



(a) (b)

Figure 6: Evaluation: Google vs. DDT

Figure 6b shows the average number of relevant pages found by each user. The plot shows that the majority of the users were able to find more relevant pages with DDT than Google – in some cases 2-3 times more pages. This indicates that the features provided by DDT do help streamline domain discovery. The only exception was user $U1$. This user used the least number of features of DDT, which could explain the lower relevant pages found.

5.3 User Feedback

Users also completed a questionnaire about their experience with DDT. The following are the summarized positive and negative feedback we received. Given the duration of 15 minutes for each topic the users were not able to use all the features of DDT. The union of the set of features used by each user for this experiment were web search, MDS visualization window, backlinks and forward links, various filtering options and page tagging.

Positive.

- The users found the MDS visualization of the pages useful to see the similarity between the pages, analyze and annotate a group of pages
- The various methods to filter pages, such as by queries, tags and "more like this" (pages similar to a selected set of pages), facilitated finding and bringing in more pages related to the domain for analysis
- Ability to add user defined tags to annotate a set of pages allowed grouping them by topic
- Avoiding annotating the same pages multiple times as they are brought in through different queries
- Though none of the users was able to use the terms extracted due to the limited time of the test, the consensus was that the extracted terms were relevant to the domain and improved with page annotations

Negative.

- The feature for crawling forward and backwards from selected pages was difficult to use and led to a large number of irrelevant pages. This was especially true for the *Ebola* domain as most of the pages for this domain were news articles with links to different unrelated topics
- Although DDT was easy to use with little training, some aspects like the need for tagging extracted terms, the workflow (the sequence in which data gathering and analysis should be done) were not clear.

6. CONCLUSION AND FUTURE WORK

In this paper, we discussed the challenges in domain discovery on the Web and presented our first step towards building a solution to this problem. We proposed a new exploratory data analysis framework that combines techniques from information retrieval, data mining and interactive visualization to guide users in exploratory search. This framework was implemented and has been released as an open source system. We have also carried out a preliminary evaluation whose results are promising and indicate that the framework is effective.

The preliminary evaluation suggests that a framework like DDT can considerably improve the quality and speed of domain discovery. We have been able to achieve these results by using fairly simple mechanisms. In future work, we plan to explore more sophisticated interactive data mining techniques to leverage all the user feedback available to further improve the performance and accuracy of DDT, including interactive document clustering [11] and interactive topic modeling [10, 29].

An important feedback we received as part of the evaluation was the difficulty in using forward and backward crawling. This was because many documents, irrelevant to the domain of interest, were downloaded. We plan to use a classifier, created in an online fashion using the pages labeled by the user, to filter the downloaded pages and thereby considerably reduce the number of irrelevant documents that the analyst must analyze.

While our results are promising, we need to perform a comprehensive user study with a larger number of participants of diverse background. We would also like to conduct various evaluations of the effectiveness of DDT for non-Web text corpora.

Acknowledgments. We would like to thank the Continuum Analytics team for their help with packaging the DDT software for easy deployment. We would also like to thank Cesar Palomo who designed the original interface of the system, Jean-Daniel Fekete for his valuable inputs and Ritika Shandilya for her contributions to the DDT client interface. We also thank the Memex performers and collaborators for their feedback and suggestions to improve DDT. This work was funded by the Defense Advanced Research Projects Agency (DARPA) MEMEX program. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA.

7. REFERENCES

[1] J. Alsakran, Y. Cen, Y. Zhao, Y. Jing, and D. Luo. STREAMIT: Dynamic visualization and interactive exploration of text streams. In *IEEE Pacific Visualization Symposium*, pages 131 – 138, 2011.

[2] L. Barbosa and J. Freire. An adaptive crawler for locating hidden-web entry points. In *Proceedings of WWW*, pages 441–450, 2007.

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, Mar. 2003.

[4] S. Chakrabarti. Focused web crawling. In *Encyclopedia of Database Systems*, pages 1147–1155. Springer, 2009.

[5] A. Dayanik, D. D. Lewis, D. Madigan, V. Menkov, and A. Genkin. Constructing informative prior distributions from domain knowledge in text classification. In *Proceedings of ACM SIGIR*, pages 493–500, 2006.

[6] Z. Ghahramani and K. A. Heller. Bayesian sets. In *Advances in Neural Information Processing Systems (NIPS)*, 2005.

[7] J. A. Hartigan and M. A. Wong. A K-means clustering algorithm. *Applied Statistics*, 28:100–108, 1979.

[8] M. A. Hearst. *Search User Interfaces*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.

[9] T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of ACM SIGIR*, pages 50–57, 1999.

[10] Y. Hu, J. Boyd-Graber, B. Satinoff, and A. Smith. Interactive topic modeling. *Machine Learning Journal*, 95:423–469, 2014.

[11] Y. Huang and T. M. Mitchell. Text clustering with extended user feedback. In *Proceedings of ACM SIGIR*, pages 413–420, 2006.

[12] T. Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 200–209, 1999.

[13] T. Mikolov, W. tau Yih, and G. Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT-2013)*, pages 746–751, 2013.

[14] G. Neumann and S. Schmeier. A mobile touchable application for online topic graph extraction and exploration of web content. In *Proceedings of the ACL System Demonstrations*, pages 20–25, 2011.

[15] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning Journal*, 39(2-3):103–134, May 2000.

[16] P. Pirolli and S. Card. The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis. In *Proceedings of International Conference on Intelligence Analysis*, volume 5, pages 2–4, 2005.

[17] H. Raghavan, O. Madani, and R. Jones. Interactive feature selection. In *Proceedings of IJCAI*, pages 841–846, 2005.

[18] L. Rokach and O. Maimon. *Chapter 15, Clustering Methods in Data Mining and Knowledge Discovery Handbook*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[19] T. Ruotsalo, J. Peltonen, M. Eugster, D. Glowacka, K. Konyushkova, K. Athukorala, I. Kosunen, A. Reijonen, P. Myllymäki, G. Jacucci, and S. Kaski. Directing exploratory search with interactive intent modeling. In *Proceedings of ACM CIKM*, pages 1759–1764, 2013.

[20] D. M. Russell, M. Slaney, Y. Qu, and M. Houston. Being literate with large document collections: Observational studies and cost structure tradeoffs. In *Proceedings of IEEE HICSS*, volume 3, pages 55–, 2006.

[21] D. M. Russell, M. J. Stefik, P. Pirolli, and S. K. Card. The cost structure of sensemaking. In *Proceedings of the INTERACT and CHI*, pages 269–276, 1993.

[22] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, Aug. 1988.

[23] U. Scaiella, P. Ferragina, A. Marino, and M. Ciaramita. Topical clustering of search results. In *Proceedings of WSDM*, pages 223–232, 2012.

[24] C. Sievert and K. E. Shirley. LDAvis: A method for visualizing and interpreting topics. In *Proceedings of the Workshop on Interactive Language Learning, Visualization, and Interfaces*, pages 63–70, 2014.

[25] S. Soliman, M. F. Saad El-Sayed, and Y. F. Hassan. Semantic clustering of search engine results. *The Scientific World Journal*, 2015.

[26] M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society, Series B*, 61:611–622, 1999.

[27] J. W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, 1977.

[28] W. Wright, D. Schroh, P. Proulx, A. Skaburskis, and B. Cort. The sandbox for analysis: concepts and methods. In *Proceedings of SIGCHI*, pages 801–810, 2006.

[29] Y. Yang, D. Downey, and J. Boyd-Graber. Efficient methods for incorporating knowledge into topic models. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, pages 308–31, 2015.

[30] H.-J. Zeng, Q.-C. He, Z. Chen, W.-Y. Ma, and J. Ma. Learning to cluster web search results. In *Proceedings of ACM SIGIR*, pages 210–217, 2004.

[31] A. J. Zhou, J. Luo, and H. Yang. DUMPLING: A Novel Dynamic Search Engine. In *Proceedings of ACM SIGIR*, pages 1049–1050, 2015.