MASFA: MASS-COLLABORATIVE FACTED SEARCH FOR ONLINE COMMUNITIES

by

Seth Cleveland, B.S.

A thesis submitted to the Graduate Council of
Texas State University in partial fulfillment
of the requirements for the degree of
Master of Science
with a Major in Computer Science
December 2013

Committee Members:

Byron J. Gao, Chair

Anne H.H. Ngu

Yijuan Lu

FAIR USE AND AUTHOR'S PERMISSION STATEMENT

Fair Use

Duplication Permission

ACKNOWLEDGEMENTS

I acknowledge my advisor Dr. Byron J. Gao. I am grateful for his presence, willingness, and support during my stint at Texas State. I especially acknowledge him for setting a lofty goal to publish a paper based on this work. We did it. A paper was published. I also appreciate my thesis committee Dr. Anne H.H. Ngu and Dr. Yijuan Lu. Their participation and support was integral towards completing my Masters thesis. Thank you. I also acknowledge the people who participated in the user study, Cullen Fouts, Karl Fleddermann, Jennifer Foster, Raul Sieberauth, Rick Jones, Terry Penner, Zulma Gregory, Laramie Gorbett, Justin Cleveland, and James Creel. I would also like to acknowledge Vicky Wang for her early support with MASFA. Finally, I want to acknowledge my managers for their support while working and completing my Masters degree and thesis. They were Trent Johnson, Danita Day, Uwe Seelig, and Chris Smouse. Thanks. I stand in gratitude and appreciation to all.

TABLE OF CONTENTS

Page

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

LIST OF ABBREVIATIONS

Abbreviation	Description
RSS – Rich Site Summary	A web standard electronic feed for publishing frequently updated information
JSON – Java Script Object Notation	An electronic data format used in web applications
API – Application Programming Interface	A description of functionality that defines interfaces between software components
JDK – Java Development Kit	Java programming language compiler and APIs
XML – Extensible markup language	A document machine encoding language
AJAX - Asynchronous JavaScript and XML	A web development technique for asynchronous web applications
URL – Uniform resource locator	A web address
HTML – Hyper text markup language	A language for describing webpages

ABSTRACT

Faceted search combines faceted navigation with direct keyword search, providing exploratory search capacities allowing progressive query refinement. It has become the de facto standard for e-commerce and product-related websites such as amazon.com and ebay.com. However, faceted search has not been effectively incorporated into non-commercial online community portals such as craigslist.org and medhelp.org. This is mainly because unlike keyword search, faceted search systems require metadata that constantly evolve, making them very costly to build and maintain. In this thesis, we propose a framework, MASFA, which takes a human-machine approach to build and maintain effective faceted search systems free of cost. In MASFA human users, i.e. community members, contribute to the system in a mass-collaborative manner; and machines assist humans based on a set of non-domain-specific techniques. The MASFA approach is completely portable and can be deployed to any application domain supporting a direct search interface. To demonstrate its utility we implemented, deployed, and experimented with MASFA on a subset of Craigslist categories and made it open to public access.

1. INTRODUCTION

The term facet means “little face” and is often used to describe one side of a many-sided object such as a cut gemstone. In information science, facets are metadata that define alternative hierarchical categories for the information space, where each facet (e.g., Make, Model, Manufacturer, Color, Price) is a taxonomy structure that can be used to organize information, corresponding to a dimension in an OLAP (Online Analytical Processing) system (J. Teevan, 2008). Unlike traditional categories, facets allow a document to exist simultaneously in multiple overlapping taxonomies (K.-P. Yee, 2003). While a single organizational structure is too limiting, multiple independent facets enable flexible access by providing alternative ways of getting to the same information.

Faceted search adds structured browsing, or faceted navigation, to direct keyword search, supporting interactive and progressive query refinement. More formally, faceted search systems are a general knowledge management model based on a multi-dimensional classification of heterogeneous data objects and are used to explore/browse complex information bases in a guided yet unconstrained way through a visual interface (Tzitzikas, 2009). Faceted search well addresses weaknesses of conventional discovery oriented search paradigms. It emerged as a foundation for interactive information retrieval. User studies demonstrate that faceted search interfaces are intuitive and easy to use, providing more effective information seeking support than conventional search paradigms (M. Hearst, 2002) (Karger V. S., 2005). Faceted search has become increasingly prevalent in online information access systems and is currently the de-facto

standard for e-commerce and product-related websites, such as amazon.com, ebay.com. walmart.com, bestbuy.com, homedepot.com, and carmax.com.

Despite the prominent success in e-commerce, faceted search has not been effectively incorporated into non-commercial online community portals such as craigslist.org and medhelp.org. This is mainly because, compared to keyword search, faceted search systems require structured metadata. In addition, such metadata constantly evolves following the life cycles of products or topics. Community data are mainly free texts and unstructured. Possibly, there are several ways to obtain the necessary metadata and facilitate faceted search:

- Hiring and training employees as for e-commerce businesses. However, community portals are usually not-for-profit and cannot afford the monetary cost.
- Forcing community members to publish structured data, e.g., by filling out forms. However, this is not practical in general cases, as it would significantly increase publishing costs. Popular community portals such as Craigslist only ask users to input very simple metadata (e.g., category and price for products), which are far from sufficient to support effective faceted search. Time-consuming browsing is still the dominating pattern of search activities for Craigslist users.
- Automating generation of metadata by using text mining and named entity recognition and classification techniques. However, facets and the metadata are domain-specific. The facet structures organizing cars differ from the ones organizing clothes. Existing techniques generally assume domain knowledge of facet structures and make use of domain-specific, hand-crafted rules or machine

learning models, which are costly to generate and update, and not portable across domains (Sekine, 2007).

Today, thriving online communities have re-defined modern society and transformed the way day-to-day activities are conducted. People spend more and more time participating in various online communities on a daily basis. For example, craigslist.org accounts for nearly 2% of global internet traffic. Despite the economic and technical challenges, there is an increasing need to take on the challenges and facilitate faceted search for online communities that enable more effective use and management of community data.

In this thesis, we explore a novel direction in enabling faceted search for online communities, utilizing the power of mass-collaboration or crowdsourcing (A. Doan, 2011). In particular, we introduce MASFA, the first framework for mass-collaborative faceted search that can be deployed and operated free of cost. MASFA takes a human-machine partnership approach, where humans, i.e., community members, contribute to the faceted search system while using it, and machines assist humans in this process based on non-domain-specific tools. The MASFA approach is completely portable and can be deployed and maintained in any application domain. It can be highly effective at significantly reducing user search time. Porting requires a web API for querying and retrieving documents. The documents can come from any domain.

A MASFA prototype was implemented and deployed using a subset of Craigslist categories based on a Craigslist RSS feed (www.craigslist.org/about/rss). The prototype is open to public access and Figure 1 shows a screenshot of it. The left-hand panel presents a set of facets, i.e., taxonomies. The right-hand panel presents refined search

results (Craigslist ad items) for a given query that satisfy the condition specified by the selected facet values. A community member can edit the facets by adding, deleting, and modifying facet names and values.

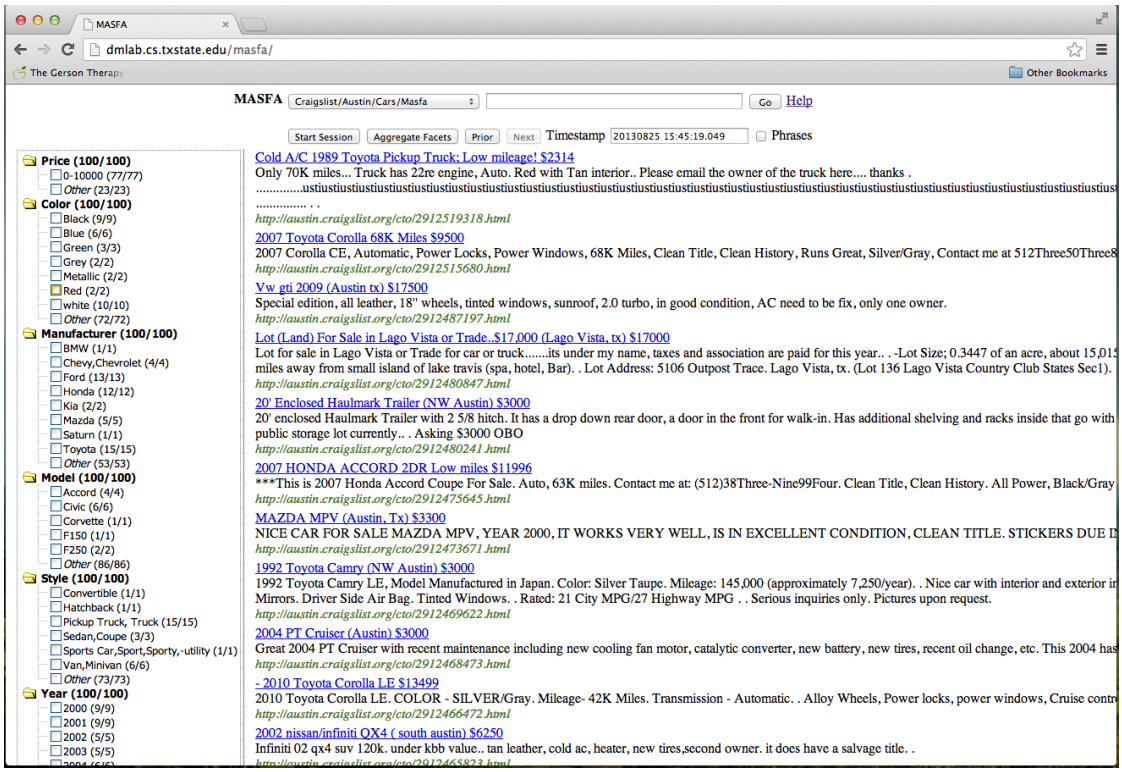

Figure 1: MASFA Screenshot

In MASFA, the metadata (labels for Craigslist ad items) are "generated" in an implicit, cost-free, and non-domain specific manner. Unlike conventional faceted search, a facet value in MASFA is a set of positive and negative phrases that represent a Boolean formula. The items satisfying the formula (covered by the facet value) are the ones that contain any of the positive phrases and do not contain any of the negative phrases. This corresponds to an implicit labeling of the satisfying items with the positive and negative phrases. For example, "$P_1, P_2, P_3, -N_1, -N_2$" covers the items that contain either the phrase P_1 or P_2 or P_3, but not N_1 nor N_2 Suppose a user is interested in sports cars (small

cars designed for performance) but not sport utility vehicles (special purpose vehicles for towing with on and off road capabilities), they can create a facet value of "sports car, sporty, -sport utility" under the "Style" facet. During faceted navigation, a user can select multiple facet values from multiple facets. MASFA implements the CNF semantics for the selected facets, where they form a conjunction of disjunctions.

In MASFA, community members can arbitrarily edit the facets. Such edits are recorded in a temporal database (Snodgrass, 1999), so that the facets can be brought back to any previous version for a given timestamp. By doing so MASFA actually provides an implicit and public way of personalization, where a user can retrieve a preferred version of facets by memorizing and specifying a timestamp.

In MASFA, machines collect historical data and generate frequent phrases, which can be used to suggest addition or removal of facet values. Machines also contribute to the formation of a robust, aggregated version of facets from the numerous human-edited versions based on their life span and usage statistics. The aggregation incorporates clustering techniques and is expected to smooth out noise and turbulence that are common in mass collaboration tasks.

Contributions.

- We propose the first mass-collaborative framework MASFA to facilitate faceted search for not-for-profit online community portals.
- MASFA is highly effective, yet completely portable and can be deployed and operated free of cost.
- We implement, deploy and experiment MASFA on selected categories of Craigslist, demonstrating its utility and promise.

2. BACKGROUND

In this section, we discuss the background research and technologies required to build, comprehend, and motivate MASFA.

2.1 Direct Search

In the context of web search, information retrieval attempts to mitigate information overload by using a query-based technique. This technique is made popular by many commercial search engines such as Google, Bing, Yahoo, etc. An inverted index is built based on terms within a document collection and a user browses the collection using queries. A typical user refines their query terms progressively and issues multiple queries to search for relevant documents. Additionally, each query is matched against the inverted index to best satisfy the present need. Unfortunately, the iterative query based approach requires cognitive effort to develop a query that defines the users need (Ryen White, 2009). Moreover, this process loses context surrounding the user's information need because each query is viewed as a single transaction against the index. Direct search systems provide a baseline system for faceted search interfaces. There are two significant direct search methods in existence – Boolean retrieval and ranked retrieval.

Boolean Retrieval. Early search engines were based on a Boolean set-retrieval method. They are different from modern search engines, which typically support both Boolean and ranked retrieval. The Boolean retrieval model derives its name from the query operators it supports. Basic Boolean retrieval systems support AND, OR, and NOT operators. However, over time, the syntax was extended to support additional operators (Tunkelang, 2009). Example extensions include keyword location in

document, keyword synonyms, wildcard operators for partial word matching, and operator grouping for improved result filtering. Boolean retrieval methods were a popular retrieval method, however, as the world-wide-web came forward, the systems were replaced with a better method called ranked retrieval (Christopher D. Manning, 2008).

Ranked Retrieval. Ranked retrieval simplifies the information seeking process by allowing the user to use unstructured queries instead of constructing structured queries. The method uses words to match and rank documents according to how relevant the documents match the words. This method alleviates returning a precise set of results to the user for simpler queries that return more relevant results (Tunkelang, 2009). Unfortunately, this flexibility loses precision and still challenges the user to identify the keywords necessary to return desired documents within the ranked results.

2.2 Exploratory Search

An alternative approach to direct search that requires less cognitive effort and further simplifies the information seeking process is called exploratory search. Exploratory search changes the information seeking process by presenting a user with relevant information to search and navigate a document collection. Additionally, studies show users can easily alter behaviors using an exploration technique (Bill Kules B. S., 2007). One form of exploration is provided through a faceted interface. The user is visually guided through an iterative process of query refinement and expansion, ideally never encountering situations with zero results (Giovanni Maria Sacco, 2009). Facets define the collection being searched and they are typically mutually exclusive. In other words, the results covered by each facet are disjointed. A user progressively selects

facets to narrow in on their information need and further refine their query. This approach also lends itself to maintaining context around an information need. Early faceted interfaces were based on a parametric search method.

Parametric Search. The parametric search method leverages the Boolean search method by providing facets and allowing users to visually specify facet value constraints to build queries. A query is typically an AND of ORs: values selected within a single facet are combined using a logical OR, whereas constraints associated with different facets are combined using a logical AND (Tunkelang, 2009). The key difference between parametric search and Boolean search is that parametric search performs set retrieval over structured data, facets, instead of unstructured data, free-form text. Unfortunately, the parametric search often leads the user to either too many or too few results. Furthermore, the interface lacks a clear method to support exploring the search results with additional queries.

Faceted Navigation. Faceted navigation provides the method for exploring results. Faceted navigation allows the user to elaborate a query progressively, seeing the effect of each choice in one facet on the available choices in other facets (Tunkelang, 2009). Faceted navigation facilitates exploring, however, it doesn't support searching unstructured textual data.

Faceted Search. Faceted search is an application of exploratory search that provides a keyword search interface with a method for navigation. Faceted search is also the dominant technique applied in e-commerce sites like Amazon.com, Ebay.com, Shopping.com, etc. For example, suppose a user wishes to purchase a vehicle on an e-commerce site. A typical system presents a direct search interface with facet categories

that directly relate to the vehicle domain. Such categories are vehicle make, model, year, or mileage. However, to support this interface, businesses require significant effort to maintain structured meta-data and build detailed lexicons. Considering the vehicle domain as an example, by using common phrases and terms, a machine can readily identify relevant patterns like Chevrolet, Toyota, Malibu, and Corolla. However, these machines do not make good hierarchies with their facets (Hearst M. A., 2006). In the aforementioned vehicle example, facets would ideally be organized with Chevrolet/Toyota under make and Malibu/Corolla under model.

2.3 Mass-collaboration

An alternative technique to building hierarchies, which supplements the strengths of machines, is mass-collaboration. Humans more readily know the organization of facets based on social context and knowledge. Mass-collaboration (Williams, 2006), human computation, and crowdsourcing are synonyms describing a phenomenon where a multitude of humans are enlisted to help solve a wide variety of problems (A. Doan, 2011). Such systems are abundant on the World-Wide Web. Prime examples include Wikipedia, Linux, Yahoo Answers, and Mechanical Turk-based systems. MASFA uses a mass collaborative technique to support organizing facets in a hierarchy.

2.4 Information Extraction

Information extraction encompasses a set of tools, techniques, and tasks for extracting structured knowledge from unstructured documents. Typically, unstructured documents are computer readable in a format like XML, HTML, TXT, etc. The most common form of unstructured documents consists of written natural language. Information extraction attempts to find semantic data in the natural language and build

structured data that can be easily used by computers. Information extraction identifies information in texts by taking advantage of their linguistic organization. Any text in any language consists of a complex layering of recurring patterns that form a coherent, meaningful whole (Moens, 2006). Because natural language patterns may change with different discourses, extraction may be either domain specific or domain unspecific. Domain specific extraction builds tools and techniques to extract specific patterns assuming the text is only in one domain, for example extract noun phrases from only financial documents. The tools may work well for their specific domain, however, they do not apply to other domains. When working with the open-web, because documents span multiple domains, domain unspecific information extraction techniques are essential. MASFA applies a domain unspecific method to extract named entities from noun phrases within natural language text. This is also called named entity recognition. MASFA uses a form of domain unspecific named entity recognition to build a vocabulary that supports users with potential facets. Additionally, MASFA applies information extraction to build an inverted index for supporting the user interface.

Phrase Extraction. A phrase is a sequence of one or more words that exists within a phrase boundary. Phrase boundaries are punctuation marks or a non-natural language element like an HTML tag. MASFA uses a suffix tree based approach (Ukkonen, 1995) developed for clustering search engine results (Zamir, 1999). A suffix tree is a trie of all suffixes that exist in a given sequence. A sequence can be characters, words, numbers, etc. The phrase extraction process consists of first building a suffix tree using Ukkonen's algorithm, then walking the tree and extracting phrases. The suffix tree algorithm consists of building a series of implicit suffix trees as nodes. The algorithm

additionally builds suffix-links between nodes for common suffixes. The algorithm has an amortized linear runtime. The building process annotates edges and nodes with the word offsets in the original document. The offsets are used for future extraction. After building the suffix tree, to extract phrases, walk the tree from root to each leaf node. Each edge that doesn't end in a leaf node becomes a potential phrase for extraction. The following figure shows the suffix tree built from three sequences "cat ate cheese", "mouse ate cheese too", and "cat ate mouse too." The star represents the root node of the tree. The dollar sign represents leaf nodes of the tree. Everything between dashes is a non-leaf node. The extraction algorithm identifies the following phrases, "mouse", "ate", "cat ate", "cheese", and "ate cheese."

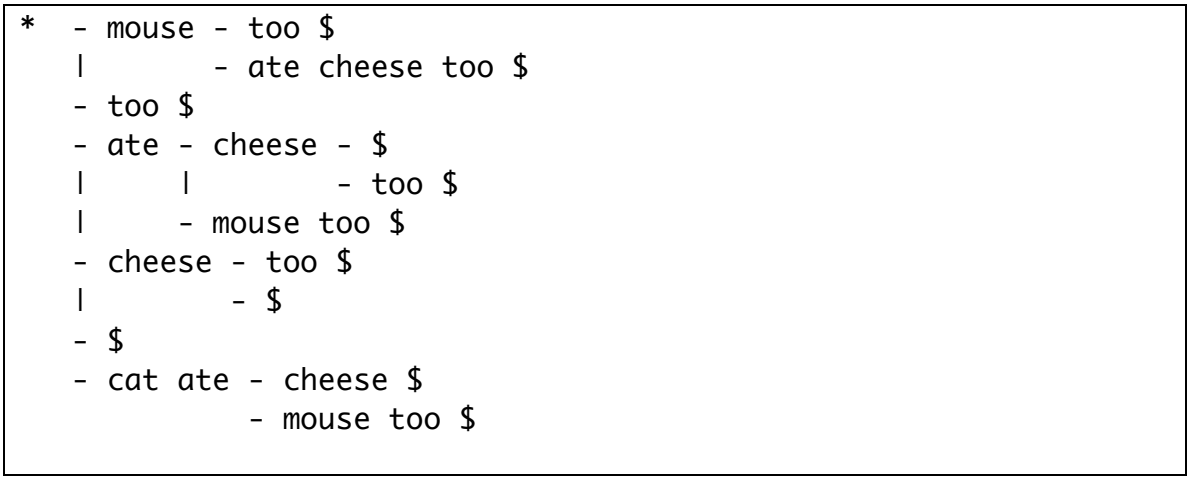

Figure 2: Example Phrase Suffix Tree

2.5 Inverted Index

The inverted index is a dictionary that maps words to document postings within a collection. The index supports satisfying ranked and Boolean queries. Building an inverted index involves collecting the documents, tokenizing the document text, performing linguistic processing which become the indexing terms, and building a

dictionary and postings index (Christopher D. Manning, 2008). For example, assume the following three documents for building an inverted index.

Table 1:Example Documents

Document Id	Document Text
1	I am your father.
2	You should listen to your father.
3	Your mother listens to your father.

Tokenizing the documents turns the text in the documents into a list of tokens, consisting of a word and its document id. Next, linguistic processing is applied to normalize and filter tokens. Typical processing includes case normalization (for example making the token word lower-case), stemming, and stop word removal. Stemming normalizes plural words to singular and attempts to algorithmically find each words root form. Stop words are common words that provide little statistical significance. Assume the following stop words: "i, to, am, your" for this example. Finally, the processed tokens are inverted to build an inverted index. The following table shows the inverted index built from the example documents.

Table 2: Example Inverted Index

Word	Document Postings
father	1, 2, 3
listen	2, 3
mother	3
should	2

MASFA uses the inverted index to support word to document id lookup for rapid phrase matching. Additionally, MASFA uses the Boolean query retrieval to support a user study.

3. LITERATURE SURVEY

3.1 Faceted Search

Faceted search augments direct keyword search capabilities with faceted navigation that enables interactive and exploratory query refinement. For many search tasks an initial query is sufficient, and faceted search can be used to further describe what to look for (Jonathan Koren, 2008). Faceted navigation is essentially a form of set retrieval model related to Boolean parametric search and advanced search, which allows users to formulate queries by specifying a set of constraints on the facet values.

Parametric search suffers from the million or none problem, where under-specified queries return too many results and over-specified queries return no results. It offers expressivity, but not guidance through the space of possible queries (Tunkelang, Faceted Search, 2009). In addition, users are either ineffective at forming complex queries or unwilling to take the effort. Most users do not use advanced search, and the average query length is 2.4 words according to a study based on 60,000,000 searches (Inan, 2006). Users prefer to specify as little as necessary in their query to find what they are looking for (Edward Cutrell, 2006) (Doug Downey, 2008) (Dumais, 2009). Rather than fully specifying their target upfront, they prefer to interact with the results to refine their query as necessary. Faceted navigation fills in the piece that is missing in parametric search: guidance. While parametric search requires users to express an information need as a query in one shot and make selections across all facets of interest, faceted navigation allows users to elaborate a query progressively (Tunkelang, Faceted Search, 2009).

The concept of faceted search dates back to 1933, when Ranganathan introduced the colon classification scheme (Ranganathan, 1933) and developed the first library

classification scheme based on facet analysis. The earliest efforts in the 1990s that catalyzed faceted navigation include dynamic queries (Shneiderman B. , 1994) and view-based search (Karger V. S., 2005). The former built a FilmFinder prototype to enable exploration of a movie database (Shneiderman C. A., 1994). The latter built a HIBROWSE prototype for a collection of documents from Lexis-Nexis.

In the mid 1990s, Marti Hearst developed Scatter/Gather, a cluster-based approach to browsing large document collections (Mufti A. Hearst, 1995). She subsequently worked on the well-known Flamenco project (flamenco.berkeley.edu) (Hearst M. , 2000), developing faceted search tools and performing usability studies with them. They developed an open-source faceted search engine supporting hierarchical facets (sourceforge, netprojects, flamenco), and also researched issues such as automating (domain-specific) metadata creation (Hearst E. S.) (E. Stoica, 2007).

The Relation Browser project (Marchionini, 2008) (Brunk, 2003) was originally developed for the US Bureau of Labor Statistics to improve searching and navigating its web site by a preview-oriented interface. The mSpace project (mspace.fm) (m. c. Schraefel M. K., 2003) developed the mSpace Classical Music Explorer (m. c. Schraefel S. A., 2005) that improves access to the classical music domain, especially for those who do not have domain knowledge. The Parallax project (Karger D. H., 2009) introduced the set-based browsing paradigm that lets users traverse the Web graph in an efficient manner.

While faceted search continues to receive attention from the information retrieval community (Osma Suominen, 2007) (Hearst M. A., Design recommendations for hierarchical faceted search interfaces, 2006) (Debabrata Dash, 2008) (B.-Y, 2008) (Bill

Kules R. C., 2009), database researchers have recently studied exploratory faceted search over databases. Such efforts include minimum-effort driven dynamic faceted search in structured databases (Senjuti Basu Roy H. W., 2008), building dynamic faceted search systems over databases (Senjuti Basu Roy H. W., 2009), and automatic extraction of facet hierarchies from text databases (Ipeirotis, 2008) (Wisam Dakka P. G., 2005).

Endeca (www.endeca.com), recently acquired by Oracle, delivers faceted search for enterprises. It is most known for providing faceted search for e-commerce sites including walmart.com and homedepot.com. Apache Solr (lucene.apache.org/solr), Sphinx (sphinxsearch.com) and Drupal (drupal.org) are popular open source faceted search engine libraries. Solr has powered the new FCC.gov site, Netflix and CNET.

While these efforts have resulted in huge success in e-commerce, none of them provide complete portable and cost-free (in terms of faceted metadata generation and maintenance) solutions for non-commercial online community portals. (Jonathan Koren, 2008) studied "personalized" interactive faceted search, but it concerns customization of presentation of facets on the search interface, instead of generation of facets and metadata as in our case.

Faceted search is a relatively new research field. It is part of the broader field of human-computer information retrieval (HCIR) that applies interactive techniques to a broad spectrum of information-seeking tasks (Tunkelang, Faceted Search, 2009). Note that the huge success of faceted search in online retail may have overshadowed other domains where it can be valuable. Some futurists even start to enthusiastically discuss the feasibility and challenges of applying faceted search to the open Web (Jaime Teevan, 2008).

3.2 Mass-collaboration

Mass-collaboration has been applied to information retrieval as in social search. In contrast to established algorithmic or machine-based approaches, social search determines the relevance of search results by considering the interactions or contributions of users. Example social search engines include Google social search (googleblog.blogspot.com/2009/10/introducing-google-social-search-i.html) and "community powered" Eurekster Swiki (www.eurekster.com). Previous work Rants (Byron J. Gao, 2010) and ClusteringWiki (David C. Anastasiu, 2011) attempted to establish a mass-collaborative framework for tackling information retrieval tasks by soliciting valuable inputs via motivated personalization. MASFA follows the same concept but is applied to faceted search interface, instead of list interface as in Rants and clustering interface as in ClusteringWiki.

3.3 Named Entity Recognition

If the facets are known, then the problem of obtaining faceted metadata boils down to the problem of named entity recognition and classification (NERC), an important sub-task of information extraction (IE). Existing NERC techniques make use of handcrafted rules or machine learning methods, which are costly and not portable across domains (Sekine D. N., 2007). (Kosseim, 2001) tested some systems on both the MUC-6 collection composed of newswire texts and a proprietary corpus made of manual translations of phone conversations and technical emails. They report a drop of 20% to 40% in precision and recall for rule-based NERC systems.

4. MASFA FRAMEWORK

Researching MASFA included building a web application, building a static dataset with craigslist data, and evaluating the application with the data set. The web application provides the necessary elements to present and manipulate facets using mass-collaborative techniques. Additionally, the application supports requesting live data from craigslist and any data set providing a web API. In this section, we discuss the MASFA design. The design covers the architecture, interface semantics, facet editing, and phrase extraction.

4.1 MASFA Architecture and Overview

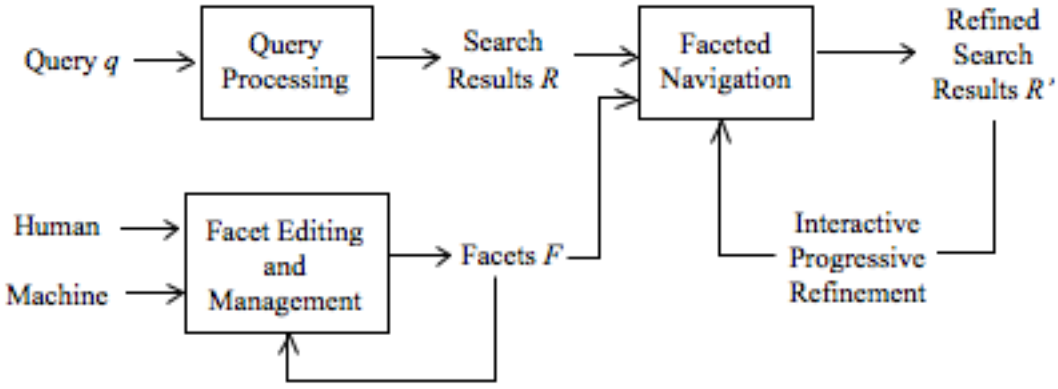

Figure 3: MASFA Architecture

The figure shows the main architecture of MASFA. For each data source category, e.g., Cars, MASFA maintains a set of facets that evolve over time. For a given keyword query q within a selected category, the Query Processing module produces a set of search results, R. Throughout the thesis, search results are often referred to as items that can be product descriptions or Craigslist ads. Then, based on a chosen version of

facets F, the Faceted Navigation module allows the user to interactively and progressively refine the search results and produce R', a refined set of items.

The Facet Editing and Management module takes human and machine efforts to build and maintain facets. A community member (user) can start an editing session by clicking the "Start Session" button. Then they can edit the facets by adding, deleting, and modifying facet names and values. A successful editing session will result in a new version with an assigned timestamp. The module has a temporal database back end that records the entire evolving history of facets. A temporal database (Snodgrass, 1999) is a database with built-in time aspects that are able to store different database states. In MASFA, specifying a version's timestamp in the "Timestamp" box brings its version back. There are also "Prior" and "Next" buttons that can be used to navigate through all the recorded versions.

MASFA does not provide explicit personalization, which would typically require login access control and significant overhead in managing the numerous personal profiles (versions). Based on the observation that facets are mainly descriptions of intrinsic product or topic features instead of decentralized personal interests or opinions, the temporal database design in MASFA actually provides a lightweight personalization mechanism with full flexibility. A user can easily retrieve a preferred version of facets by memorizing and specifying a timestamp.

By clicking the "Aggregated Facets" button, users can obtain a synthetic set of facets. The aggregation utilizes clustering techniques by analyzing the numerous human-edited versions considering their life span and usage statistics. It can effectively smooth out noise and turbulence that are common in crowd sourcing tasks.

4.2 Interface and Semantics

In a typical faceted search interface, there are sets of facets or taxonomies. Each facet has a name, which can be of different types such as nominal, ordinal, interval, ratio, or free-text. A facet is associated with facet values that are exhaustive (collectively covering all the items) and mutually exclusive (not covering any item in common). For example, facet Make would have Toyota, Chevrolet ... as its facet values. The values within a facet can be a flat or hierarchical list.

In MASFA, facet values appear as a flat list. This is a design, not a technical option. In general, faceted search works better with a broad taxonomy that is relatively shallow, as this lets users combine more perspectives rather than get stuck in an eternal drill down, which causes fatigue (www.uie.com/articles/faceted search/). Many commercial sites such as Linkedin people search (linkedin.com) and the Costco wireless phone shopping site (membership- wireless.com/index.cfm) use flat lists for clarity of interface and logic. MASFA implements a mass-collaborative framework, where it is particularly important to avoid unnecessary confusions and complications, making sure that contributors share the same or similar understanding about the system and have a common ground to work on collaboratively. Flat lists are much easier to visualize, comprehend, and edit.

While the conventional exhaustiveness and mutual exclusiveness constrains provide clear classification of items, their enforcement would incur significant difficulty for community members to construct facets. MASFA relaxes these constraints by allowing incomplete and overlapping coverage of items. A special value “Other” is added whenever necessary to collect the items not covered by the sibling values within a facet.

Reasonable overlapping of items will not be purposely ruled out. For example, if a car has both black and blue colors, then the corresponding car ad item would be covered by both "Black" and "Blue" values under the facet "Color". In practice, a well-behaved MASFA faceted search system would be nearly exhaustive and nearly mutually exclusive. This relaxation does not compromise the utility of the system, yet successfully avoids significant building and maintenance costs.

Unlike conventional faceted search, a facet value in MASFA is not a single value, but a set of positive and negative phrases separated by commas. It represents a Boolean formula and covers the items that satisfy the formula. Let *V* be a facet value, where $P_1, P_2, \cdots, P_m$ constitute the set of positive phrases and $-N_1, -N_2, \cdots, -N_n$ constitute the set of negative phrases. Then *V* corresponds to a Boolean formula of $(P_1 \cup P_2 \cup \cdots \cup P_m) \cap \neg(N_1 \cup N_2 \cup \cdots \cup N_n)$. The items that satisfy the formula are the ones that contain any of the positive phrases and do not contain any of the negative phrases. This interpretation corresponds to an implicit way of generating metadata, where the satisfying items are "labeled" by a set of positive phrases $P_1, P_2, \cdots, P_m$ and a set of negative phrases $-N_1, -N_2, \cdots, -N_n$. Note that this implicit named entity recognition and classification mechanism is not domain-specific. It can be utilized in any application domain and does not incur maintenance or update cost.

In practice, the positive phrases are usually different mentions of the same (or similar) target feature, for example, Chevrolet and Chevy. The negative phrases are used to weed out different features that happen to have the same or similar mentions to those of the target feature. For example, if the target feature is sport cars (small cars designed for performance), then we may want to weed out sport utility vehicles (special purpose

vehicles for towing with on and off road capabilities) by using a negative phrase "-sport utility".

During faceted navigation, multiple facet values maybe selected from multiple facets. MASFA implements the CNF semantics for the selected facets, where they form a conjunction of disjunctions. For example, if V_1 (e.g., Ford) and V_2 (e.g., Honda) are selected from facet F_1 (e.g., Manufacturer) and U_1 (e.g., Black) and U_2 (e.g., Blue) are selected from facet F_2 (e.g., Color), then the compound Boolean formula will be $(V_1 \cup V_2) \cap (U_1 \cup U_2)$ (e.g., cars made either by Ford or Honda that are either black or blue in color). The refined search results R' will contain all the items from R (the original search results for query q) that satisfy the compound formula.

In MASFA, each facet value V' is associated with an item count in the form of *x/y*, where *y* is the total number of original results for query q that are covered by V. The *y* number is a function of q and V and will not change dynamically throughout the progressive refinement process for query q.

If a sibling value V' within the same facet has been selected, then *x* indicates the maximum (not exact, because MASFA allows overlapping coverage among sibling facet values) number of the items that can possibly be added (removed) to the refined results if V is selected (de-selected). This is because MASFA implements CNF semantics for selected facet values and selected sibling facet values are OR-connected. Suppose under the facet "Color", "Black" (9/9) has been selected and there are 9 items in the set of refined results. The subsequent selection of "Blue" (5/5) would add at most 5 items to the refined results if there is no overlapping between "Black" and "Blue". If one car is black

and blue (containing both words in the ad), then only 4 items will be added to the refined results.

If none of the sibling values of *V* has been selected, then *x* indicates exactly the number of items that will appear in the refined search results if *V* is selected. This is the case even when some facet values from other facets have been selected because MASFA implements CNF semantics for selected facet values and selected values from different facets are AND-connected. If none of the sibling values but *V* has been selected, then *x* only indicates the current number of refined results covered by *V* (which is also the total number of refined results since *V* is the only value selected within the facet) and cannot be used to predict the change of number of refined results once *V* is deselected. This is due to a convenient but incorrect convention in faceted search: it is considered all values within a facet are selected if none of them is selected.

Item count numbers contain very important information for progressive query refinement. They provide a preview of the refined search results before a facet value is actually selected.

Each facet name in MASFA is also associated with an item count in the form of x'/y', where y' indicates the total number of original search results for the initial keyword query q, and x' indicates the total number of refined results for the selected facet values. Obviously, all the facets will share the same x'/y' at all times. Such numbers provide summative information for the progressively refined search results.

4.3 Facet Editing and Management

A community member can start an editing session and edit the facets by adding, deleting or modifying facet names and facet values. The refined results as well as item

counts will be updated immediately and automatically after each edit. No login is required. All edits in MASFA are available through context menus. The editing session will expire in certain period of time (10 minutes) unless renewed. A successful editing session with valid edits will result in a new version of facets to be created.

Machine-extracted phrases can assist with human editing. For example, phrases Toyota, Honda, Audi, and BMW can be moved into one facet labeled with “Make”. Phrases Wagons, Convertibles, and Pickup Trucks can be moved into one facet labeled with Vehicle Type.

User edits are valuable contributions. It is important to keep the historical edits, instead of only the current version of facets, for multiple beneficial purposes such as aggregating user contributions and personalizing user preferences.

In MASFA, a temporal database (Snodgrass, 1999) is used to store all the user edits, where addition and deletion (a modification is equivalent to a deletion plus an addition) timestamps are recorded. Specifically, the facet trees are decomposed into pairs of labels that correspond to edges, and the pairs are the actual stored database objects. Addition timestamps and deletion timestamps together form a composite key in a relational table. Note that in (David C. Anastasiu, 2011), we have used root-to-leaf paths as the editing and storage unit, instead of the much simpler edges (label pairs). This is because such paths are guaranteed to be unique, while edges are not and there are many repeating edges in the hierarchical clustering interface. However, in the faceted search scenario, edges rarely repeat and we can safely assume their uniqueness.

Our current prototype does not stress concurrency control. It implements a simple policy that only one user can edit the facets at a session. Community members can edit any version of facets, not necessarily the current one.

The personalized and collaborative framework in (David C. Anastasiu, 2011) provides explicit personalization for logged-in users, which incurs significant overhead in managing the numerous personal profiles. We observe that in the scenario of faceted search, facets mainly describe intrinsic product features or properties. Users usually share centralized common cognition and comprehension about the representation and structure of such features with few deviating personal preferences. Thus MASFA facilitates version navigation and retrieval of an implicit and personalization mechanism.

Algorithm: *Facet aggregation*

Input: All facet versions, their life span, and usage statistics

Output: *F*: an aggregated version of facets

1: Decompose each version into a facet set to obtain combined facet set $\mathbb{F}$
2: Rank facets in $\mathbb{F}$ according to their life span and usage statistics
3: Compute pair-wise similarity for facets in $\mathbb{F}$
4: $\mathbb{C} = \emptyset$ //initial cluster set
5: **while** ($|\mathbb{F}| \neq 0$)
6: Remove $f_i \in \mathbb{F}$ with highest rank
7: Compute similarity of f_i to each cluster in $\mathbb{C}$
8: Select $c_j \in \mathbb{C}$ with largest similarity
9: **if** ($similarity(f_i, c_j) \geq threshhold$) **then**
10: Assign f_i to c_j
11: **else**
12: Create new cluster with f_i in $\mathbb{C}$
13: **end if**
14: **end while**
15: For each cluster in $\mathbb{C}$ choose the highest ranked member and insert into *F*

Figure 4: Facet Aggregation

In particular, a user can create a preferred version of facets based on a close one and easily retrieve it in the future by specifying the corresponding timestamp. This version is personal, but not private. Any user can access it by specifying the timestamp or using her/his own alias of it. Privacy is not of critical concern in faceted search. The users can also navigate through all the previous versions of facets by using the "Prior" and "Next" buttons. Our current prototype does not stress spamming issues. However, the embedded personalization mechanism actually provides a very effective way of avoiding spamming.

Facet Aggregation. Given the open nature of crowdsourcing systems, noise and turbulence are common due to differences in preferences and understanding, unintentional execution errors, or malicious spamming. Statistics-based aggregation can effectively smooth out such noise and turbulence. In MASFA, by clicking the "Aggregated Facets" button, users can obtain a synthetic set of facets. The aggregation utilizes clustering techniques by analyzing the numerous human-edited versions considering their life span and usage statistics.

Figure 4 presents the pseudo code for the facet aggregation algorithm of MASFA. The idea of the algorithm is to decompose the facets of all versions and perform clustering on them. Each cluster represents an equivalence class consisting of different versions of the same facet. For each cluster, the best member is chosen and inserted into *F*, the aggregated set of facets.

In line 2, facets are ranked according to their life span and usage statistics. In general, facets with longer life spans and more usage tend to be more robust, enduring and popular, and they are ranked high.

In line 3, pairwise similarity of facets is computed using modified Jaccard coefficient. Let F_1 and F_2 denote two facets, each consisting of a set of facet values, then $similarity(F_1, F_2) = F_1 \cap F_2 / \min(|F_1|, |F_2|)$.

We use min $(|F_1|, |F_2|)$ to replace $|F_1 \cup F_2|$ as in the standard Jaccard coefficient to boost the similarity measure. This is because in practice, different facets (e.g., "Color" and "Make") rarely share facet values. In case two facets do share a few in common, it would be a strong indication that they are actually different versions of the same facet and should be clustered together. The cut-off threshold (line 8) is set to 10% in MASFA but it is tunable.

In line 8, the similarity between a facet F_i and a cluster C_j of facets is computed by computing the similarity between F_i and the closest facet in C_j using the modified Jaccard coefficient.

Frequent Phrases. Frequent phrases can be used to suggest addition or removal of facet values, serving as building blocks in facet construction and organization and reducing editing workload of community members. It can also be used to add a layer of machine supervision to reduce turbulence. The extracted phrases can be considered as a superset of the common facet values. They are mixed, not unorganized into facets. However, they are ranked according to frequency and made clickable, which makes them useful even in progressive query refinement.

In a centralized category of documents, such as Car in Craigslist, facet values (Toyota, Chevy, Blue, and Power Window) are frequently used as feature descriptors. Thus, potentially a frequency-based, non-domain-specific approach can be used to extract such facet values. In MASFA, a category of documents is collected and pre-processed.

Then, a suffix-tree-based algorithm is used to extract frequent syntactic phrases. Then, simple cleansing heuristics are applied to remove noisy phrases. The phrase extraction techniques in MASFA are completely non-domain-specific and portable.

Pre-processing. In the beginning, duplicate items are removed using near-duplicate detection techniques. This is important for frequency-based phrase extraction because duplication can inflate the frequency of semantically meaningless long phrases leading to added noise. A typical example of such noise is "Call me only if you interested to buy this car 512-501-xxxx no message or email". After removing duplicates, the remaining documents are pre-processed by following standard tokenization, case folding, and stemming procedures.

Implementation-wise, we used SpotSigs, a near-duplicate detection package (www.mpi-inf.mpg.de/~mtb/). SpotSigs identifies duplicate documents by first extracting signature phrases in all documents, and then identifying documents with overlapping signatures.

Syntactic Phrase Extraction. Common facet values are frequent semantic phrases (not vice versa) because they are frequently used to describe product features. Existing techniques for named entity recognition and classification use costly handcrafted rules that are not portable across domains (David Nadeau, 2007). A syntactic phrase is a continuous sequence of words. The words may not be semantically related to form a coherent meaning. For example, "phrase is a continuous" is a syntactic phrase. Semantic phrases are also syntactic phrases but not vice versa. Since semantic phrases tend to be used frequently, and common facet values are frequent semantic phrases, frequency is a strong indicator for facet values.

MASFA uses a suffix tree-based approach to extract frequent syntactic phrases. In particular, it uses a generalized suffix tree as a compressed trie to identify word sequences inside a document collection. The suffix tree is built with Ukkonen's algorithm (Esko, 1995) over all tokens. Then, the syntactic phrases are extracted by traversing the tree depth first. The phrase length is calculated by summing the edge span from the root to a node. The phrase postings are collected from the edge postings. The postings are aggregated from a child node back to the suffix tree root. Phrase extraction is explained in (Zamir, 1999).

The frequency cut-off thresholds depend on the length of phrases. Smaller thresholds are used for longer phrases. While MASFA currently determines the thresholds heuristically, more sophisticated machine learning approaches can potentially be used to learn the thresholds.

To remain effective, facets have to evolve along with community data, where new products (e.g., Blu-ray players) or new features (e.g., 3D for TV) of products are introduced continuously. MASFA performs phrase extraction periodically to identify emerging frequent phrases, where previous frequent phrases are removed before building the suffix tree.

Cleansing. The extracted syntactic phrases are a superset of facet values. To improve accuracy, MASFA adopts heuristics to remove noise. The main heuristic is to make use of a set of stop words containing common verbs, adverbs, and adjectives. Any syntactic phrase starting or ending with a stop word will be removed. This simple heuristic works well in our experience, yet it remains as an important future work direction to further improve the extraction accuracy.

5. MASFA IMPLEMENTATION

The MASFA implementation discusses the front end and back end software components. MASFA is implemented as a web service in Java using JDK 1.7. In this section, the implementation details for the front end, back end, clients, and APIs are presented.

5.1 MASFA Back End

The MASFA back end provides a Craigslist client, facet data, and business logic to manage the data. The back end is implemented using the Spring Model-View-Controller package (MVC). Spring MVC provides custom presentations, JSON serialization, database access, and custom business logic. The back end serves all facets and processed data to the front end as JSON using AJAX (www.json.org/).

Craigslist Queries. Craigslist provides data as Really Simple Syndication feeds (RSS) (www.craigslist.org/about/rss). RSS is essentially XML that follows a schema. The XML data contains information for Craigslist ad-items such as title, timestamp, URL, description, etc. MASFA implements a client that requests and parses the RSS feeds using the open source library Rome (http://rometools.org). The URL, title, and description information is used for processing. Craigslist provides RSS feeds by city and category. For example, Craigslist provides 100 items for Austin cars and trucks for sale by owner at http://austin.craigslist.org/cto/index.rss. Craigslist also provides query capabilities. The following URL provides 25 items for the same category that contains "honda": http://austin.craigslist.org/search/cto?query=honda&srchType=T&format=rss.

Query Result Processing. To process query results retrieved from Craigslist, the title and description fields are first extracted and then processed to build an inverted

index. From each craigslist result the text fields are scanned into tokens using Jflex version 1.4.3 (http://jflex.de/), where the following token types are identified: word, number, punctuation, and end of content. Next, MASFA normalizes each token by applying a case filter (turning all tokens into lower cases), a stemmer (using the snowball stemmer at http://snowball.tartarus.org/), and a stop word filter to extract terms. MASFA uses the union of two stop words sets at http://snowball.tartarus.org/algorithms/english/stop.txt and http://www.webconfs.com/stop-words.php. Finally, MASFA builds an inverted index using the extracted terms. The inverted index tracks document ids, postings, and text positions. It supports the front end by mapping facet values to terms and their documents.

Temporal Database. The MASFA back end has a Java H2 Database engine that implements a temporal database. Java H2 (http://www.h2database.com/html/main.html) was chosen for its setup simplicity for testing and production deployment. Additionally, MASFA uses hibernate (http://www.hibernate.org/) to easily switch to other databases in the future. There are three tables in the database: Facet, Liveliness, and Session.

- Session: The Session table supports tracking of edits. It contains four fields: id, start, stop, and configuration, where start and stop are timestamps. Java supports converting timestamps to an elapsed number of milliseconds. The configuration field is a string describing a specific craigslist city and category.
- Facet: The Facet table records facet values and the facet names they belong to. It contains four fields: id, configuration, parent, and value. The configuration field identifies the craigslist city and category. The parent field records facet names for

facet values. The value field is a comma-separated list of positive and negative phrases.

- Liveliness: The Liveliness table records the life spans of facet names and values. It contains three fields: id activated timestamp and deactivated timestamp, where id is a foreign key referencing to the Facet table.

5.2 MASFA Front End

The MASFA front end provides dynamic user interface for editing facet trees using Javascript and HTML. MASFA allows the user to retrieve facets from the back end for a given timestamp, as well as processed Craigslist data for a given query. The processed Craigslist data is cached to facilitate dynamic updates to facet names, facet values, and their item counts. Each user edit to the facets during a session are sent to the back end for storage.

jQuery (http://jquery.com/) provides the core functionality for dynamic user interface based plugins and AJAX queries. The tree is presented using the Dynatree jQuery plugin (http://code.google.com/p/dynatree/). Dynatree provides an interface to dynamically build and manipulate trees in a Web browser. We used the hierarchical selection and checkbox feature of the plugin. A context menu jQuery plugin (http://abeautifulsite.net/2008/09/jquery-context-menu- plugin/) provides a user interface to support creation, update, and deletion of facet names and values as tree nodes. After editing a label, the item counts are updated dynamically.

To align the user labels with the terms and postings extracted from the Craigslist data, the phrases are stemmed using a Javascript implementation of the Snowball stemmer (http://code.google.com/p/urim/). The postings from phrases consisting of

multiple terms are merged using the inverted index provided by the back end for a given document set.

5.3 MASFA API

MASFA integrates the front-end and back-end through an API. In the Model-View-Controller design, the controllers provide the necessary APIs to the browser. MASFA provides three main controllers – craigslist, session, and home. In this section, we document the APIs provided by MASFA. The APIs are provided as HTTP methods. The craigslist and sessions controllers return JSON documents. The home controller returns HTML documents for the browser to interact with the craigslist and session controllers. The following sections document the craigslist and sessions JSON APIs in detail.

Craigslist Controller API. The craigslist controller provides an API to retrieve craigslist data, retrieve craigslist facets, and manipulate craigslist facets. Next, we document each craigslist method, their input parameters, and results.

- Get

  Get queries and processes a craigslist dataset. The method queries craigslist with the provided location and feed using the query terms. The query response from craigslist is processed. The processing includes token, term, and phrase extraction. Phrases may be excluded in the output. Including phrases increases method response time.

  Input Parameters

  Location, Feed, Query, Phrases

  Response Format

  documents: the documents url, title, summary, price, and timestamp

terms: inverted index build from the documents

tokens: text tokens, term, types, and their offsets in the documents

phrases: named entities in the documents when phrases are enabled

Example

http://dmlab.cs.txstate.edu/masfa/craigslist/get/austin/cto?query=vas&phrases=0

- Get Facets

  Get facets retrieves all facets for the craigslist location and feed that are alive at given timestamp.

  Input Parameters

  Location, Feed, Timestamp

  Response Format

  next: the next session version in the facet database

  prior: the prior session version in the facet database

  version: the version of the facets returns

  facets: a list of tuples containing a facet category and a list of position and negative labels

  Example

  http://dmlab.cs.txstate.edu/masfa/craigslist/get/facets/austin/cto

- Aggregate Facets

Aggregate facets return the most relevant facets across all facet versions using clustering and liveliness across versions. The method takes three parameters, a craigslist location, a craigslist feed, and the percent similarity to form a cluster. The percent similarity is not required and has a default value of 10%.

Input Parameters

Location, Feed, pSimularity

Response Format

facets: a list of facet tuples, each tuple contains a facet category and list of positive and negative phrases

Example

http://dmlab.cs.txstate.edu/masfa/craigslist/aggregate/facets/austin/cto

- Activate Facet

Activate facet asserts a valid session exists for the craigslist location and feed. If a session is not active, an error is returned. Otherwise, a new facet with provided category and label are marked alive in the temporal database.

Input Parameters

Location, Feed, Facet Category, Facet Label

Response Format

Empty JSON document on success either a document with a JSON error message

Example

http://dmlab.cs.txstate.edu/masfa/craigslist/activate/facet/austin/cto?category=Price&label=100-200

- Deactivate Facet

  Deactivate facet asserts a valid session exists for the craigslist location and feed. If a session is not active, an error is returned. Otherwise, the facet with provided category and label are marked not alive in the temporal database.

  Input Parameters

  Location, Feed, Facet Category, Facet Label

  Response Format

  Empty JSON document on success either a document with a JSON error message

  Example

  http://dmlab.cs.txstate.edu/masfa/craigslist/deactivate/facet/austin/cto?category=Price&label=100-200

- Deactivate Facet Category

  Deactivate facet category asserts a valid session exists for the craigslist location and feed. If a session is not active, an error is returned. Otherwise, all facets with provided category are marked not alive in the temporal database.

  Input Parameters

  Location, Feed, Facet Category

  Response Format

  Empty JSON document on success either a document with a JSON error message

  Example

  http://dmlab.cs.txstate.edu/masfa/craigslist/deactivate/facet/category/austin/cto?category=Price

- Rename Facet

  Rename facet asserts a valid session exists for the craigslist location and feed. If a session is not active, an error is returned. Otherwise, the facet with provided category and label is marked not alive in the temporal database. Additionally, a new facet is marked alive in the database with the same category and new label.

  Input Parameters

  Location, Feed, Facet Category, Facet Label, New Facet Label

  Response Format

  Empty JSON document on success either a document with a JSON error message

  Example

  http://dmlab.cs.txstate.edu/masfa/craigslist/rename/facet/austin/cto?category=Price&fromLabel=100-200&toLabel=100-201

- Rename Facet Category

  Rename facet category asserts a valid session exists for the craigslist location and feed. If a session is not active, an error is returned. Otherwise, all facets with provided category are marked not alive in the temporal database. Additionally, all facets with the old category are marked alive with the new category in the database.

  Input Parameters

  Location, Feed, Facet Category, New Facet Category

  Response Format

  Empty JSON document on success either a document with a JSON error message

  Example

http://dmlab.cs.txstate.edu/masfa/craigslist/rename/facet/category/austin/cto?fromCategory=Color&toCategory=Colour

Session Controller API. The session controller manages sessions within MASFA. The controller supports craigslist and any other potential client through a formatted configuration parameter. The controller API supports starting, stopping, extending, and testing sessions. The following documents the methods in detail.

- Start

  Start begins a new session for provided configuration. The method returns the starting and ending timestamp for the started configuration.

  Input Parameters

  Configuration

  Response Format

  Configuration: The configuration provided as input

  Start: The starting timestamp for a session

  Stop: The ending timestamp for a session

  Example

  http://dmlab.cs.txstate.edu/masfa/session/start/craigslist_austin_cto

- Stop

  Stop ends a session for provided configuration. In nominal conditions, the method should always return success, even when a session is not started.

  Input Parameters

  Configuration

  Response Type

HTTP Status Code

Response Format

HTTP Status Code = 200

Example

http://dmlab.cs.txstate.edu/masfa/session/stop/craigslist_austin_cto

- Extend

  Extends increases the sessions stopping timestamp for provided configuration. The stopping timestamp does not affect the facet version and all facets remain with the starting timestamp. The method will return an error if a sessions not started.

  Input Parameters

  Configuration

  Response Format

  Configuration: The configuration provided as input

  Start: The starting timestamp for a session

  Stop: The ending timestamp for a session

  Example

  http://dmlab.cs.txstate.edu/masfa/session/extend/craigslist_austin_cto

- Get

  Get retrieves the active session data for provided configuration. Get will return an empty document if a sessions is not started for the provided configuration.

  Input Parameters

  Configuration

  Response Format

Configuration: The configuration provided as input

Start: The starting timestamp for a session

Stop: The ending timestamp for a session

Example

http://dmlab.cs.txstate.edu/masfa/session/get/craigslist_austin_cto

5.4 MASFA Tools

Several tools were written to facilitate offline analysis of craigslist. Especially, extracting offline phrases and building a static data set using craigslist data. The tools developed range from downloading craigslist RSS feeds, extracting documents from the downloaded RSS feed, processing the extracted documents, and detecting duplicate documents. These tools are available after building MASFA. The tools are run from a Linux terminal.

CraigsListAggregator. CraigslistListAggregator downloads craigslist RSS documents using the provided location and feed. The location and feed can be comma separated. The tool also allows the user to set a download directory for the RSS documents.

```
CraigsListIndexFeedAggregator: download and cache craigslist rss index
feeds
Option                    Description
------                    -----------
-d, --directory <File>    the craiglist directory to save
                          results (default: craigslist/index)
-f, --feed                feed to pull (comma separated)
-h, --help                print this message
-l, --location            city to pull feeds from (comma
                          separated)
```

Figure 5: CraigsListAggregator Help

CraigsListProcessor. CraigsListProcessor extracts data from a download craigslist RSS document cache. The tool supports extracting JSON documents or summary documents. The JSON documents support loading into the MASFA data set. The summary documents support duplicate document detecting. The tool takes as input the directory for the cached RSS documents, a comma separated list of locations and feeds, and an optional list of document URLs to filter from the output.

```
CraigsListIndexProcessor: process downloaded craigslist data
Option            Description
------            -----------
-d, --dir <File>  craiglist dataset directory
-f, --feed        feed to process
--filter <File>   file containing urls to filter
-h, --help        print this message
-l, --loc         cities to process
-o, --output      output type (summary|json)
-t, --total <Integer> total documents to process
```

Figure 6: CraigsListProcessor Help

Process. Process supports extracting data from a collection of search results. The tools supports RSS feeds from craigslist as well as output from other search engines. The tools can extract tokens, phrases, and labels. The difference between phrases and labels is that labels are normalized and filtered after phrase extraction. See below for further help information.

```
Process: print information regarding cached search results
Option               Description
------               -----------
-d, --documents      display document content
-f, --file <File>    a file or a directory that contains
                     files to process
-h, --help           print this message
-i, --input          file input format (json|raw) (default:
                     raw)
-l, --labels         display labels extracted from all files
-o, --output         output format (csv|raw) (default: raw)
-p, --phrases        display phrases extracted from all
                     files
-t, --tokens         display document tokens
```

Figure 7: Process Help

SpotSigs. The SpotSigs tool executes the duplicate document detection algorithm over a collection of raw text documents. The SpotSigs author, (Martin Theobald, 2008), provided the original implementation of the tool. Minor modifications were made to the tool to execute in a JDK 1.7 environment.

process-dups.pl. Process-dups.pl is a Perl script that takes the duplicate document output from SpotSigs and generates a list of duplicated documents. The file with the highest size is not included in the output.

6. MASFA ADMINISTRATION AND USER GUIDE

This section explains how to administer and use MASFA. The administration section explains how to bring-up the MASFA software on a Linux computer. The user guide explains how to use the MASFA web interface.

6.1 MASFA Administration

This section explains how to build the software, install the application, and how to start/stop the MASFA service.

Building MASFA. Building the MASFA software requires a minimum environment configuration of JDK 1.7 on a Linux like system. The JDK is available through www.oracle.com. Install the JDK on a system using the instructions provided by oracle. The MASFA source code exists on the dmlab server at /var/www/masfa/masfa_source.tgz. The following steps explain how to build MASFA.

1) tar xzf masfa_sources.tgz
2) cd gummy
3) ./gradle clean build generateScripts
4) ./gradle generateScripts

After building, the MASFA web server exists in the directory: gummy-web-server/build/server/. Additional tools supporting this work also exist in the directory: build/bin.

Installing MASFA. After building the software, copy the files in gummy-web-server/build/server into a system directory. The system directory used on the dmlab server is /var/www/jetty/server. Also ensure all files in "/var/www/jetty" are owned by a non-root user. In this example we use jetty. Next, as root, install the following system

init script into /etc/init.d/jetty to ensure MASFA starts and stops on system startup and shutdown.

```
#!/bin/bash

# Some variables to make the below more readable
export MAFA_HOME=/var/www/jetty/server
export MASFA_USER=jetty
export MASFA_PORT=80

start()
{
        su -p -s /bin/sh $MASFA_USER -c "$MASFA_HOME/bin/startup
$MASFA_PORT > /dev/null &"
}

stop()
{
        su -p -s /bin/sh $MASFA_USER -c "$MASFA_HOME/bin/shutdown >
/dev/null &"
}

case "$1" in
        start)
                start
                ;;
        stop)
                stop
                ;;
        restart)
                stop
                start
                ;;
        *)
                echo $"Usage: $0 {start|stop|restart}"
esac
exit
```

Figure 8: MASFA init script

Once installed, MASFA is started as root with the following command – service jetty start. Next, using a web browser open this website http://HOSTNAME/masfa. HOSTNAME is the name of the system where MASFA's installed.

Adding a Craigslist Category. Adding a new Craigslist category requires updating an HTML template file in the source code. After adding the category the source code must be rebuilt and deployed. For example we will add the Austin furniture

category to the set of available categories in MASFA. First, edit the following source file: gummy-web/src/main/webapp/WEB-INF/views/home.ftl. Next, add the following option to the source select HTML element in the aforementioned file: '<option value="craigslist/austin/fua"> Craigslist/Austin/Furniture</option>'. Finally, rebuild and install craigslist as per the instructions outlined above. The figure below shows the select element with the added furniture option.

```
<select id="source" onchange="reset()">
 <option value="craigslist/austin/cto/masfa">Craigslist/Austin/Cars/Masfa</option>
 <option value="craigslist/austin/moa/masfa">Craigslist/Austin/Cell Phones/Masfa</option>
 <option value="craigslist/austin/sys/masfa">Craigslist/Austin/Computer/Masfa</option>
 <option value="craigslist/austin/cto">Craigslist/Austin/Cars</option>
 <option value="craigslist/austin/moa">Craigslist/Austin/Cell Phones</option>
 <option value="craigslist/austin/sys">Craigslist/Austin/Computer</option>
 <option value="craigslist/austin/fua">Craigslist/Austin/Furniture</option>
</select>
```

Figure 9: Adding Austin Furniture to Craigslist Categories

6.2 MASFA User Guide

The following describes how to use MASFA once the application's installed. Use the deployed version on the dmlab servers as reference (http://dmlab.cs.txstate.edu/masfa).

Category Selection. MASFA presents the user with a collection of datasets to query. One dataset is the MASFA dataset built from cached craigslist RSS feeds. The other data set is the live craigslist data. The following table describes the possible selections – their name, index, and source. The user selects a dataset using the drop-down box. MASFA will pull the top results using the query words in addition to the facets related to that data set and index.

Table 3: MASFA Categories

Name	Index	Data Source
Craigslist/Austin/Cars/MASFA	Austin, Cars	MASFA
Craigslist/Austin/Cell Phones/MASFA	Austin, Cell Phones	MASFA
Craigslist/Austin/Computers/MASFA	Austin, Computers	MASFA
Craigslist/Austin/Cars	Austin, Cars	Craigslist
Craigslist/Austin/Cell Phones	Austin, Cell Phones	Craigslist
Craigslist/Austin/Computers	Austin, Computers	Craigslist

Data Querying. MASFA supports direct search with keyword and Boolean query functionality. Put in a list of keywords and MASFA returns results with either keyword or both in descending time order. Boolean searching the MASFA dataset and Craigslist is different. Both datasets support Boolean operators and grouping. However, the syntax is different. The Craigslist data source uses &, |, and () for and-ing, or-ing, and grouping respectively. The MASFA data source uses AND, OR, and () for and-ing, or-ing, and grouping respectively. The two examples below show how to precisely find results for a 2010 or 2011 Blue Honda.

Table 4: Example Advanced Query Support

MASFA	Craigslist
(2010 OR 2011) AND blue AND honda	(2010 | 2011) & blue & honda

Facet Editing. After selecting the dataset with the desired craigslist category and submitting a query, the user may find the facets lack a desired facet label or category. In such a case, the user should start a facet editing session by pressing the button labeled "Start Session." If another user is actively in a session, MASFA will respond with an

error. Otherwise, a session is started. The timestamp is updated with the session version. The user can record that timestamp for personalization. After the sessions started the user has the following options:

- Create a Facet

  To create a facet, right-click the facet labeled "New Facet" on the hierarchy and select "Edit." A new facet is created for the user to provide a name. The user can press "esc" to cancel facet creation, or, hit "enter" to complete facet creation.

- Edit, Delete a Facet

  After a facet has been created, a user can delete or edit that facet. To accomplish this, using the mouse, right-click a facet. Then select either edit or delete. Both edit and delete affect the facet label value associations to the facet.

- Create a Facet Label Value

  After creating a facet, a user can add label values to the facet. To add a new value, right click the desired facet and select "New." Afterwards, enter the desired values. Press the "esc" key to cancel creation or press the "enter" key to create the value. The values are comma-separated list of positive or negative phrases. For example to create a value for Sports Cars and not Sports Utility Vehicles, the values would be "sports car, -sports utility." The positive phrases require no characters, however, the negative phrases require a leading '-'. The phrases are case-insensitive and are stemmed. Hence, there would be no need to add Sport or Sports to the values. MASFA also provides a special facet, price. A user can add a numeric range to the MASFA price to select results. For example

to create a price value for $0 to $1000.00, create a new facet in the price category with the value 0-1000.00.

- Edit, Delete a Facet Label Value

  After a facet's been created, a user can delete or edit that facet value. To accomplish this, using the mouse, right-click a facet value. Then select either edit or delete. Editing follows the same rules as creation.

After editing completion, the user stops the session with the button labeled "Stop Session." The user can always return to this version of the facets using the timestamp displayed.

Facet Aggregation. MASFA finds the best facet labels and categories through aggregation. Facet aggregation identifies the best facets by removing noise with short-lived facets. In other words, the facets that survived the longest and have the largest group exists in the aggregated set. The button, Aggregate Facets, on the MASFA UI returns said facets. The method also returns the timestamp the facet labels come from with the facet category name.

Facet Personalization. MASFA provides facet personalization through timestamp-based versioning. A user can record the timestamp during session editing or remember the date he created the facet version. The timestamp is encoded in the following format YYYYMMDD HH:mm:ss.SSS. YYYY is year, MM is month, DD is day, HH is hour, mm is minutes, ss is seconds, and SSS is milliseconds. For example, consider the following data October 5th 2012 at 4:00:00 PM. The timestamp would be 20121005 16:00:00.000. In addition to versioning, MASFA supports navigation. A user can go to the next or prior version based on a given version. MASFA will not allow

navigation past or before the latest and earliest version. The buttons on the UI labeled, "Prior" and "Next" support navigation.

7. EVALUATION

Evaluating MASFA covers three areas – correctness, efficiency, and utility. Correctness evaluation shows the system was built according to design and requirements. Correctness was measured through functional and unit software tests. Efficiency shows the system performs according to real-time user expectations. Efficiency was measured by timing pertinent system software components – data querying and processing, facet retrieval, facet aggregation, and facet tree setup. Finally, utility measures MASFA's ability to assist and satisfy user expectations. The study compares the new collaborative facet interface to a traditional keyword interface. The user study supports both objective and subjective measurements. The objective measurements show the system's ability to improve searching compared to a traditional keyword based system by timing task completion. The subjective measurement shows the user's preference to MASFA over keyword-based searching. MASFA provides a keyword-based interface emulating craigslist using the MASFA data to support this study. This section also discusses the MASFA data set used to support this study.

The MASFA application is maintained on a server with 2 Intel Xeon X5675 processors each having 6 cores @ 3.07GHz, 24GB memory, and 1.3TB disk storage, running Apache Tomcat 6.0.26. In this section, we evaluate the correctness and efficiency of MASFA. Experiments were performed on a PC with Intel Core i5 running Mac OSX with 8GB of RAM. Google Chrome version 23.0.1271.64 was used for all front-end efficiency and correctness measurements.

7.1 MASFA Dataset

A data set was built to support evaluating MASFA using static craigslist documents. The data set is also intended to support constructing offline phrases for the facets view and future research. The data set was built from the craigslist API producing RSS documents. The data was taken from the top results in a craigslist category and city roughly every 15 minutes over a period of 3 months. The data set contains 7 categories over 5 cities. The categories include: cars and trucks for sale by owner, computers, furniture, jewelry, cell phones, appliances, and electronics. The Texas cities include: Austin, Corpus Christi, Dallas, Houston, and San Antonio. The total raw compressed data downloaded as RSS files is 4.01 GB. The RSS documents are processed and stored in an inverted index. The inverted index supports keyword-based queries similar to craigslist and standard keyword query system. The data set also supports query operators AND/OR and grouping with parenthesis.

The inverted index technology used is ElasticSearch (http://www.elasticsearch.org/). ElasticSearch provides a full-text search engine, api, and schema-less storage using Apache Lucene. The api accepts documents as name value pairs in JSON. ElasticSearch comes configured with default text processing technologies (ex. stemming and case folding) for the values in each document. Because ElasticSearch accepts JSON documents and the craigslist api produced RSS feeds, an extraction script was written to extract the relevant fields and convert the RSS documents into JSON. Additionally, the SpotSigs duplicate document detection algorithm was used to remove redundant documents before loading them into ElasticSearch. The JSON fields extracted include: url, title, description, and date time. After processing, removing

duplicates, and loading the data into ElasticSearch, the dataset size reduced to 1 GB. Finally, the dataset was integrated into MASFA using ElasticSearch's api. The api is web-based, similar to craigslist, and provides the json documents for a given city and category. The integration amounted to developing a json parser and changing the url to use a hostname and method different than craigslists.

7.2 Correctness Evaluation

We applied formal software-testing techniques to ensure system correctness. Unit and Functional testing were applied. Unit testing guaranteed the individual software units work as designed. Example units include tokenization, phrase extraction, database querying, etc. Junit 4.X was used as a Java test framework (https://github.com/kentbeck/junit/wiki). A total of 203 unit tests were written and total execution time was 4.497 seconds. Additionally, 10 functional tests were manually executed to ensure correctness with respect to requirements. Before executing the test cases, the system was prepopulated with facets. We summarize the functional test cases as follows. All functional test cases executed successfully.

- Query Functions
    - Query with and without keywords
    - Aggregate facets
    - Time navigation
- Session Functions
    - Start and stop session
    - Session timeout with and without continue
- Edit Functions

- Add, modify and delete facet name
- Add, modify and delete facet value

7.3 Efficiency Evaluation

We evaluated the execution efficiency of MASFA prototype in terms of query processing, preprocessing, facet retrieval, facet tree setup, and facet aggregation. In summary, MASFA is efficient with satisfactory response time. Typical response time for most requests is within 1~2 seconds. Although the facet tree setup time does not scale well, it is sufficiently fast for practical sizes. Note that we used up to 320 labels (facet values) in the experiments for evaluation purposes. Practical faceted search systems are never as large. Excessive labels in a faceted search interface defeat its purpose of reducing information overload. There are further enhancements that can be made to improve response time. For example, the facets and data can be compressed. This is a trivial change, however, this is not implemented in the MASFA.

Craigslist Query Processing. Craigslist API returns at most 100 search results per query. We measured the processing time for Craigslist queries over multiple executions for various Craigslist categories such as Austin cars and Austin cell phones. On average, it takes MASFA 436ms to process the queries. The average data size of the RSS feed was 90K.

Preprocess Craigslist Results. In MASFA, Craigslist search results are preprocessed to generate tokens, terms and phrases and to build an inverted index. On average, it takes MASFA 9.73ms for this preprocessing step. The average data size of the RSS feed was 90K.

Facet Retrieval. To evaluate facet retrieval efficiency, we measured the time required to retrieve facet names and values from the temporal database back end. The measurement did not include the facet serialization time into JSON. The time was measured by increasing the number of labels (facet values), where the labels were generated randomly. The measurement starts from 0 facet names up to 32 with 10 facet values per facet. As shown in the figure below, facet retrieval in MASFA scales well.

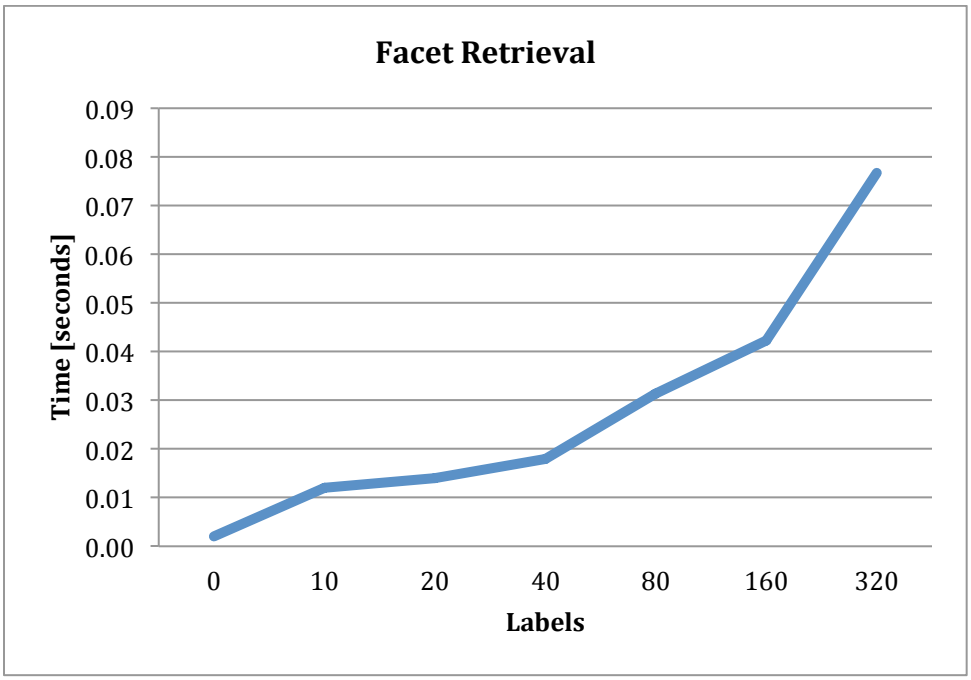

Figure 10: Facet Retrieval Time

Facet Tree Setup. To evaluate facet tree setup efficiency, we measured the time required to build facet trees on the front end. The measurement did not include any data retrieval time. The facet names and values were randomly generated. A total number of 320 values over 32 facets with 10 values per facet were generated. As shown in the figure below, the time to build facet trees grows exponentially. However, on average it took

only 0.2 or 0.5 second to build facet tress consisting of 50 or 100 labels, which is very acceptable. Practical effective faceted search systems never have more than 50 labels.

Nonetheless, further improvements for facet tree setup could be considered, e.g., by changing Javascript data structures and using a bit set to represent document ids instead of an array of integers. Additionally, different Javascript tree presentation libraries can be considered.

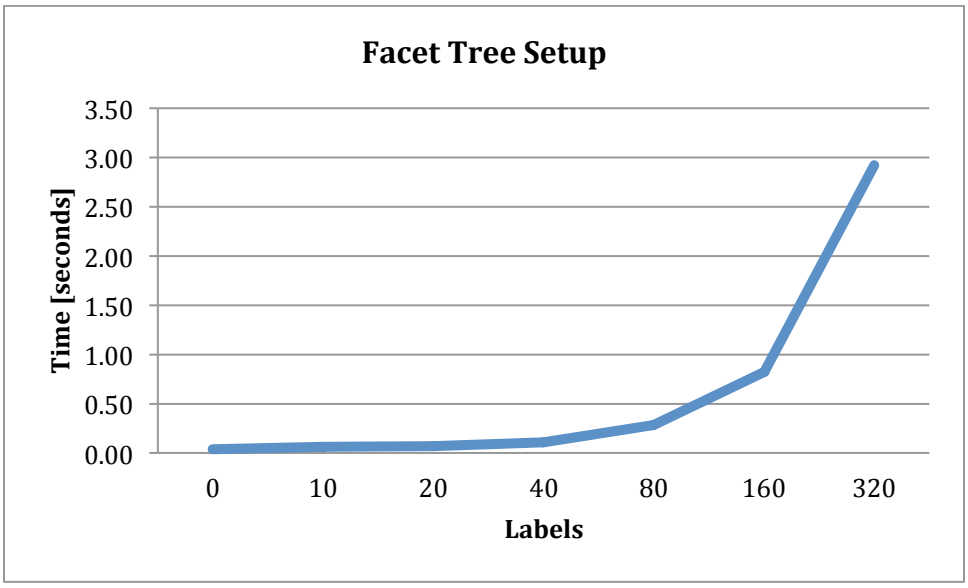

Figure 11: Facet Tree Setup Time

Facet Aggregation. To measure facet aggregation efficiency, we measured the time to obtain the aggregated version of facets from all historical versions. For the experiment, a version consisting of 10 facets with 10 values per facet was created. The measurements started with 10 versions up to 60. As shown in the figure below, the facet aggregation algorithm scales well with an increasing number of versions.

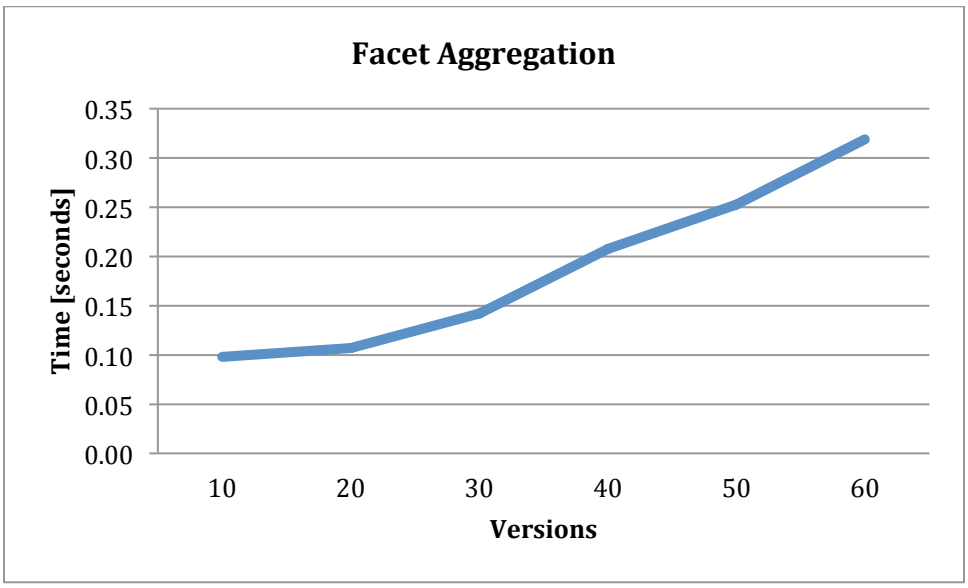

Figure 12: Facet Aggregation Time

7.4 User Study

To show MASFA improves utility compared to keyword searching we developed a user study. The user study consists of eight goal-oriented tasks using the MASFA data set. The tasks leverage three craigslist categories in the data set – cars, cell phones, and computers. Three tasks require the car category. Two tasks require the cell phone category. Three tasks require the computer category. The users were asked to perform the tasks using both MASFA and a keyword-based search interface to the MASFA dataset. The users were asked to record their answers and collect times required to complete the tasks. Additionally, the users were asked subjective questions regarding MASFA. The evaluation tasks, data collection table, and questionnaire are defined in the Appendix.

The user study was performed with ten people. Each person was provided with the evaluation form. The MASFA provides help through a website link on the front-page. The study was semi-moderated. Each person was briefly shown MASFA's basic

functionality. The functionality included – how to edit facets and how to query data. The people performing the user study came from diverse backgrounds. Their backgrounds included -- students, professionals, and doctors. Additionally, a presentation was given to the students. The presentation provided more detailed background knowledge and research regarding MASFA. In the following two sections we discuss the objective and subjective feedback collected from performing this user study.

Objective Evaluation. Objective user study evaluation of MASFA measures the time to complete the eight tasks. Completing the tasks includes query for the data, facet editing (MASFA only), and searching for the data in the results. The time data was collected and aggregated for each craigslist category. The timing data collected shows that MASFA improves task completion on average 52 seconds for an overall 25% improvement compared to the keyword based searching. The most improved MASFA category was the "Cell Phone" category. However, MASFA shows a negative change with the "Car" category. The "Car" category was the first task. Users spent time understanding the system during this task. However, users performance improved with subsequent tasks.

Table 5: MASFA vs. Keyword Task Timing

Category	Average MASFA Time	Average Total Keyword Time	Percent Improvement (MASFA vs. Keyword)
All	155	207	25%
Car	175	170	-3%
Cell Phone	122	237	49%
Computer	158	223	29%

Subjective Evaluation. After completing the tasks, the users were presented with a questionnaire to subjectively compare MASFA to a keyword-based search. The

users were asked four things. Three of the four questions were to rate MASFA. The questions were – is MASFA easy to use, Is MASFA easy to learn, and is MASFA helpful compared to keyword search. The answers were rated from a 1 to 5. A 1 answer means "No." A 2 answer means "Moderately No." A 3 answer means "Neutral." A 4 answer means "Moderately Yes." A 5 answer means "Yes." The fourth question was to provide any feedback, improvements, concerns, etc. to improve MASFA.

The table below presents the distributions for the rated questions. From the feedback, we can observe that most users found MASFA easy to use and learn. Additionally, most users found MASFA helpful compared to keyword searching.

Table 6: MASFA Feedback

	MASFA - Easy to Use	MASFA - Easy to Learn	MASFA- Helpful vs. Keyword
Yes	4	4	7
Moderately Yes	4	4	3
Neutral	1	1	0
Moderately No	1	1	1

In the feedback section, users shared a positive view of MASFA with areas to improve their experience. In general, the users felt the facets speed up searching especially after a complete facet tree is available. Users also found the online phrases useful. The phrases helped identify useful facets for finding results and building the facet tree. Users also stated the ability to add new facets were helpful compared to other existing faceted interfaces because it allows them to add new data that isn't modeled. Areas of improvement identified by the users, was to grey out facets with zero results instead of removing them. Users also expressed interest in having numeric ranges, for example, a facet for years from 2000 to 2010. The users also wanted check boxes to easily deselect

all facets. The last feedback was to highlight matching facets in the search result text to improve finding text in matching documents. An area of concern was that users experience an initial learning curve. This learning curve was noticed in the initial timing results with MASFA. However, after they understood the system, they state the system was useful.

7.5 Discussion

Evaluating MASFA successfully shows the system performs per specification, effectively, and well compared to a keyword search interface. Considering the described architecture and implication, there are many ways to further improve the system. Areas to improve include providing more machine recommendations for facet building, spam filtering, and creating more facet types.

This study did not include detailed analysis on the spam filtering technique employed. Future work would be to analyze the technique against hypothetical spamming to the tree. Possible spamming includes very long facet values, multiple versions with spammed facets, or foul language. Alternative techniques to remove spam would be implemented and evaluated.

The current method provides minimal machine support for building a facet tree. The offline technique could be extended to make suggestions to adding values to facets. Theoretically, a facet represents a classification and the facet values represent named entities. A facet tree and their values could be used to identify frequent patterns for extracting new facet values. The patterns would be extracted and applied offline. Example patterns are words before and after a facet value. The new values would be

presented in the UI as possible expansions to a facet with a confidence value. Users could then selectively approve or deny any value.

Another limitation of the current system regards continuous valued facets. As identified through the user study, users created facets with numeric ranges and asked for the feature in future versions. Continuous values are common in commerce sites. For example, cars have mileage, year, and horsepower. Patterns could be developed by users or extracted for identifying continuous valued facets from unstructured text.

8. CONCLUSION

Faceted search has become the de facto standard for e-commerce and product-related websites. However, it has not been effectively incorporated into non-commercial online community portals such as craigslist.org due to economic and technical difficulties. In this thesis, we proposed MASFA, the first mass-collaborative framework that takes a human-machine partnership approach to build and maintain effective faceted search systems. The MASFA approach is completely portable and can be freely deployed in any application domain. We have implemented, deployed, and experimented MASFA on selected categories of Craigslist to demonstrate the utility of our approach.

There are several interesting directions for future work based on this initial development. For example, instilling more sophisticated concurrency control, incorporating optional human management, increasing portable machine contribution, and addressing facet spamming.

APPENDIX

User Study Evaluation Form

Objective: Compare MASFA faceted search with a direct keyword search using Craigslist data, demonstrating the utility and advantages of MASFA.

Task Procedure

1) Go to website,
   Direct Search: http://dmlab.cs.txstate.edu/masfa/craigslist/
   MASFA: http://dmlab.cs.txstate.edu/masfa/
2) Select category using drop-down box
3) Input queries to complete the task
4) Write time duration and answer in table

Tasks

1) How many blue or green 2010 Toyota Tundra are available?
   Category: Craigslist/Austin/Cars/Masfa

2) How many 2000 to 2006 Chevrolet Corvettes are available?
   Category: Craigslist/Austin/Cars/Masfa

3) Collect five contact phone numbers for a 2000 to 2003 325i BMW.
   Category: Craigslist/Austin/Cars/Masfa

4) How many HTC EVO 4G cells phones new or like new are for sale?
   Category: Craigslist/Austin/Cell Phones/Masfa

5) What's the cheapest 4g black Iphone for AT&T networks?
   Category: Craigslist/Austin/Cell Phones/Masfa

6) How many Apple laptops are for sale with 4 to 16 GB of memory and 1 or more TB of disk space?
   Category: Craigslist/Austin/Computer/Masfa

7) What's the cheapest 17 inch HP Pavillion laptop for sale?
   Category: Craigslist/Austin/Computer/Masfa

8) What's the most expensive Intel I7 Acer gaming laptop for sale?
   Category: Craigslist/Austin/Computer/Masfa

MASFA: Evaluation Form

Name: _____

Task Results

Task	MASFA Answer	MASFA Time (e.g. 1 min 10 sec)	Keyword Answer	Keyword Time (e.g. 2 min 5 sec)
1				
2				
3				
4				
5				
6				
7				
8				

MASFA Questionnaire

MASFA is easy to use

Disagree	1	2	3	4	5	**Agree**

MASFA is easy to learn

Disagree	1	2	3	4	5	**Agree**

MASFA is helpful with tasks compared to keyword search

Disagree	1	2	3	4	5	**Agree**

Comments (concerns, issues, suggestions):

REFERENCES

A. Doan, R. R. (2011). Crowdsourcing systems on the world-wide web. *Commun. ACM , 54* (4), 86–96.

B.-Y, e. a. (2008). Beyond basic faceted search. *Proceedings of the international conference on Web search and web data mining (WSDM)* .

Bill Kules, B. S. (2007, 7 24). Users can change their web search tactics: Design guidelines for categorized overviews. *ScienceDirect* , 463–484.

Bill Kules, R. C. (2009). What do exploratory searchers look at in a faceted search interface? *Proceedings of the 9th ACM/IEEE-CS joint conference on Digital libraries (JCDL)* .

Brunk, G. M. (2003). Towards a general relation browser: A gui for information architects. *Journal of Digital Information , 4*.

Byron J. Gao, J. J. (2010). Rants: A framework for rank editing and sharing in web search. *Proceedings of the 19th International World Wide Web Conference (WWW)* .

Christopher D. Manning, P. R. (2008). *Introduction to Information Retrieval.* Cambridge: Cambridge University Press.

David C. Anastasiu, B. J. (2011). Clusteringwiki: personalized and collaborative clustering of search results . *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)* .

David Nadeau, S. S. (2007). A survey of named entity recognition and classification. *Lingvisticae Investigationes , 30* (1), 3-26.

Debabrata Dash, J. R. (2008). Dynamic faceted search for discovery-driven analysis . *Proceedings of the 17th ACM conference on Information and knowledge management (CIKM)* .

Doug Downey, S. D. (2008). Understanding the relationship between searchers' queries and information goals. *Proceedings of the 17th ACM conference on Information and knowledge management (CIKM)* .

Dumais, S. T. (2009). Faceted search. In *Encyclopedia of Database Systems* (pp. 1103-1109).

E. Stoica, M. A. (2007). Automating creation of hierarchical faceted metadata structures . *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)* .

Edward Cutrell, D. R. (2006). Fast, flexible filtering with phlat. *Proceedings of the SIGCHI conference on Human Factors in computing systems (CHI)* .

Esko, U. (1995). On-line construction of suffix trees. *Algorithmica , 14* (3), 249-260. Giovanni Maria Sacco, Y. T. (2009). *Dynamic Taxonomies and Faceted Search: Theory, Practice, and Experience* (Vol. 25). Springer.

Hearst, E. S. (n.d.). Nearly-automated metadata hierarchy creation. *Proceedings of HLT-NAACL 2004: Short Papers* , 2004.

Hearst, M. A. (2006). Clustering versus faceted categories for information exploration. *Communications of the ACM , 49* (4), pp. 59-61.

Hearst, M. A. (2006). Design recommendations for hierarchical faceted search interfaces. *Proceedings of SIGIR 2006 Workshop on Faceted Search* .

Hearst, M. (2000). Next generation web search: Setting our sites. *IEEE Data Engineering Bulletin, Special issue on Next Generation Web Search* (23), 38-48.

Inan, H. (2006). *Search Analytics: A Guide to Analyzing and Optimizing Website Search Engines.* Book Surge Publishing .

Ipeirotis, W. D. (2008). Automatic extraction of useful facet hierarchies from text databases . *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering (ICDE)* .

J. Teevan, S. T. (2008). Challenges for supporting faceted search in large, heterogeneous corpora like the web. . *The Second Workshop on Human-Computer Interaction and Information Retrieval (HCIR)* .

Jaime Teevan, S. T. (2008). Challenges for supporting faceted search in large, heterogeneous corpora like the web . *Second Workshop on Human-Computer Interaction and Information Retrieval (HCIR)* .

Jonathan Koren, Y. Z. (2008). Personalized interactive faceted search. *WWW* .

K.-P. Yee, K. S. (2003). Faceted metadata for image search and browsing. *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI),* .

Karger, D. H. (2009). Parallax and companion: Set-based browsing for the data web. *WWW Conference* .

Karger, V. S. (2005). Magnet: supporting navigation in semistructured data environments . *Proceedings of the 2005 ACM SIGMOD international conference on Management of data* .

Karger, V. S. (2005). Magnet: supporting navigation in semistructured data environments. *Proceedings of the 2005 ACM SIGMOD international conference on Management of data (SIGMOD)* .

Kosseim, T. P. (2001). Proper name extraction from non-journalistic texts. *Computational Linguistics in the Netherlands* , 144-157.

M. A. Hearst, D. K. (1995). Scattergather as a tool for the navigation of retrieval results . *Working Notes of the AAAI Fall Symposium on AI Applications in Knowledge Navigation and Retrieval* .

m. c. Schraefel, M. K. (2003). mspace: interaction design for user-determined, adaptable domain exploration in hypermedia . *AH2003: Workshop on Adaptive Hypermedia and Adaptive Web Based Systems* .

m. c. Schraefel, S. A. (2005). The mspace classical music explorer: Improving access to classical music for real people . *V MUSICNETWORK OPEN WORKSHOP: Integration of Music in Multimedia Applications* .

M. Hearst, A. E.-P. (2002). Finding the flow in web site search . *Commun. ACM* .

Marchionini, R. G. (2008). The relation browser tool for faceted exploratory search. *Proceedings of the 8th ACM IEEE-CS joint conference on Digital libraries (JCDL)* .

Martin Theobald, J. S. (2008). SpotSigs: Robust and Efficient Near Duplicate Detection in Large Web Collections . *SIGIR* .

Moens, M.-F. (2006). *Information Extraction: Algorithms and Prospects in a Retrieval Context.* Belgium: Springer.

Mufti A. Hearst, D. K. (1995). Scattergather as a tool for the navigation of retrieval results . *Working Notes of the AAAI Fall Symposium on AI Applications in Knowledge Navigation and Retrieval, .*

Osma Suominen, K. V. (2007). User-centric faceted search for semantic portals. *Proceedings of the 4th European conference on The Semantic Web: Research and Applications (ESWC)* .

Ranganathan, S. (1933). *Colon Classification.* Madras, India: Madras Library Association.

Ryen White, R. W. (2009). *Exploratory search: beyond the query-response paradigm* (Vol. 1). Morgan & Claypool Publishers.

Sekine, D. N. (2007). A survey of named entity recognition and classification. *Linguisticae Investigationes , 30* (1), 3-26.

Senjuti Basu Roy, H. W. (2009). Dynacet: Building dynamic faceted search systems over databases . *Proceedings of the 2009 IEEE International Conference on Data Engineering (ICDE)* .

Senjuti Basu Roy, H. W. (2008). Minimum-effort driven dynamic faceted search in structured databases. *Proceedings of the 17th ACM conference on Information and knowledge management (CIKM)* .

Shneiderman, B. (1994). Dynamic queries for visual information seeking. *IEEE Softw.* , 11:70-77.

Shneiderman, C. A. (1994). Visual information seeking: tight coupling of dynamic query filters with starfield displays. *Proceedings of the SIGCHI conference on Human factors in computing systems: celebrating interdependence* .

Snodgrass, R. T. (1999). Developing Time-Oriented Database. *Applications in SQL* .
Stanislaw Osinski, D. W. (2005, 5). A Concept-Driven Algorithm for Clustering Search Results. *IEEE Intelligent Systems* , pp. 48-54.

Tunkelang, D. (2009). *Faceted Search.* Morgan & Claypool Publishers.
Tzitzikas, G. M. (2009). Dynamic Taxonomies and Faceted Search: Theory, Practice, and Experience. *Springer Publishing Company, Incorporated* .

Ukkonen, E. (1995). On-Line Construction of Suffix Trees. *Algorithmica , 14* (3), 249-260.

Williams, D. T. (2006). *Wikinomics: How Mass Collaboration Changes Everything.* Portfolio Hardcover.

Wisam Dakka, P. G. (2005). Automatic construction of multifaceted browsing interfaces. *Proceedings of the 14th ACM international conference on Information and knowledge management (CIKM)* .

Zamir, O. E. (1999). *Clustering web documents: A phrase-based method for grouping search engine results.* University of Washington.