

Leveraging Context in User-Centric Entity Detection Systems

Vadim von Brzeski
Yahoo! Inc.
2811 Mission College Blvd
Santa Clara, USA
vadimv@yahoo-inc.com

Utku Irmak
Yahoo! Inc.
2811 Mission College Blvd
Santa Clara, USA
uirmak@yahoo-inc.com

Reiner Kraft
Yahoo! Inc.
2811 Mission College Blvd
Santa Clara, USA
reiner@yahoo-inc.com

ABSTRACT

A *user-centric* entity detection system is one in which the primary consumer of the detected entities is a person who can perform actions on the detected entities (e.g. perform a search, view a map, shop, etc.). We contrast this with *machine-centric* detection systems where the primary consumer of the detected entities is a machine. Machine-centric detection systems typically focus on the *quantity* of detected entities, measured by precision and recall metrics, with the goal of correctly identifying every single entity in a document.

However, the simple precision/recall scores of machine-centric entity detection systems fail to accurately reflect the *quality* of detected entities in user-centric systems, where users may not necessarily want to “see” every possible entity. We posit that not all of the detected entities in a given piece of text are necessarily relevant to the main topic of the text, nor are they necessarily interesting enough to the user to warrant further action. In fact, presenting all of the detected entities to a user may annoy the user to the point where he decides to turn this capability off completely, an undesirable outcome. Therefore, we propose to measure the quality and utility of user-centric entity detection systems in three core dimensions: the accuracy, the interestingness, and the relevance of the entities it presents to the user. We show that leveraging surrounding context can greatly improve the performance of such systems in all three dimensions by employing novel algorithms for generating a concept vector and for finding concept extensions using search query logs.

We extensively evaluate the proposed algorithms within Contextual Shortcuts – a large-scale user-centric entity detection platform – using 1,586 entities detected over 1,519 documents. The results confirm the importance of using context within user-centric entity detection systems, and validate the usefulness of the proposed algorithms by showing how they improve the overall entity detection quality within Contextual Shortcuts.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CIKM 2007 Lisboa, Portugal

Copyright 200X ACM X-XXXXX-XX-X/XX/XX ...\$5.00.

Categories and Subject Descriptors

H.0 [Information Systems]: General

General Terms

Algorithms

Keywords

Entity Detection, Content Syndication, Contextual, Context, Contextual Shortcuts, Information Extraction

1. INTRODUCTION

The field of entity detection is embedded within the bigger area of natural language processing (NLP) and information extraction. Much work has been done in this area over the past 40 years using various techniques, including rules, dictionaries, and machine learning algorithms [13].

In the majority of these cases the implicit consumer of the detected entities was a machine, e.g. an algorithm whose goal was to correctly identify every single entity in a document *regardless of the surrounding context*. Thus precision and recall were the natural measures in such systems, with maximum precision and maximum recall being the ideal. We refer to these works as *machine-centric* entity detection systems.

Within the past 5 years we have seen an emerging trend of systems (e.g., *IntelliTXT*¹, *Vibrant Media*², *Kontera*³, *Ontok*⁴, and our own *Contextual Shortcuts*) that not only detect entities or concepts within text (e.g., web pages), but also transform those detected entities or concepts into actionable, “intelligent hyperlinks”. These hyperlinks provide additional relevant information in a single-click, e.g. detecting an address and showing a map to it, detecting a product name and showing an ad to it, or detecting a movie title and showing local show times. We refer to these as *user-centric* entity detection systems, because their intended target audience is a user interacting with the detected, actionable entities or concepts.

When it comes to *user-centric* entity detection systems, we claim that the simple goals of maximum precision and maximum recall are not optimal since they measure only the *quantity* but not the *quality* of the detected entities

¹<http://www.intellitext.com>

²<http://www.vibrantmedia.com>

³<http://www.konterea.com>

⁴<http://www.ontok.com>

or concepts. Quality matters, because users (people) may not necessarily want to see every possible entity in the text “tagged” (converted into an actionable hyperlink) for the following reasons:

- Not every single entity or concept is necessarily relevant to the topic of the document, nor is it necessarily interesting. Tagging such irrelevant or uninteresting entities or concepts is a nuisance and a distraction.
- A named entity may be contained within a larger *concept*. A concept can correspond to something in the real world, but is not necessarily a named entity, e.g. “Betty Crocker’s famous cheesecake”. Solely tagging “Betty Crocker” is at best not optimal, or at worst simply wrong.

To summarize, the consumer of these detected entities or concepts is a human (a user), therefore, these entities need to be of real interest in order for him to take any action. One of the challenges of user-centric entity detection systems is that users typically focus on a specific task at hand (e.g., reading an article), while the system presents actionable entities embedded in the document. Not only do those detected entities need to be interesting, they must also be interesting enough to divert the user from his current task and investigate further.

Given the above issues we propose to evaluate the quality (utility) of user-centric entity detection systems along three core dimensions:

Boundary Correctness (Accuracy): Are the boundaries of the presented entities or concepts correctly selected?

Relevance: Are the presented entities or concepts relevant to the topic of the text?

Interestingness: Are the presented entities or concepts interesting enough to the reader to warrant further action or investigation?

The above three metrics can be seen in the following text snippet taken from a recent news story:

By DAVID ESPO, AP Special Correspondent WASHINGTON

Anti-war Democrats in the Senate failed in an attempt to cut off funds for the Iraq war on Wednesday, a lopsided bipartisan vote that masked growing impatience within both political parties over President Bush’s handling of the four-year conflict.

The named entities and concepts in this snippet are: *David Espo, Washington, Democrats, Senate, Iraq, Wednesday, and President Bush*. First, “*David Espo*”, “*Washington*”, and “*Wednesday*” are probably of little utility to the user, and thus should not be tagged and converted into actionable links. For example, it would be a distraction to users to see *Washington* turned into a link leading to a map of the city. Second, “*Iraq*” is a relevant and potentially interesting named entity. However, it is contained in a more relevant concept, namely the “*Iraq war*”, which is contained in an even more relevant and potentially very interesting concept, namely the “*funds for the Iraq war*”.

In order to improve the overall user experience, we believe that the goals for a high quality user-centric entity detection system are as follows. First, the system should be able to present new, interesting, and relevant entities and concepts to the user, rather than just static named entities. Second, to avoid potential under-selection issues, the system should also “extend” entities to cover the most specific (and presumably most interesting) concepts, a technique we call *concept extension*.

Under-selection of entities is not only an issue for relevance or interestingness – it can also lead to gross errors, making for a poor user experience. To illustrate this further consider the case shown in Figure 1 of a popular entity detection and content syndication system. The detected entity was “Garland”, a clear case of under-selection from “Kelly Garland”, further exacerbated by the offer of more information about “Garland, Texas”.

The major contributions of this paper are listed below, and address the aforementioned problems by leveraging *context*:

We distinguish between *user-centric* and *machine-centric* entity detection systems, and argue that for the former the simple precision/recall scores fail to accurately reflect the quality of detected entities. Therefore we propose to measure the quality of detected entities within user-centric entity detection systems in the three core dimensions mentioned above.

We analyze the global context of the document to generate a *concept vector* - a list of entities and concepts in the document, ranked by their relevance to the rest of the document. Candidate concepts are derived from periodic analyses of (time-varying) search query log data from Yahoo! Search⁵. The concept vector is another source of potentially interesting and relevant concepts that may not have yet been found by the entity detection process. The ranking of the concept vector is used to filter out the noisy and irrelevant concepts. For example, popular queries like “I Love You” may or may not be relevant - depending on whether the document is about the “I Love You” computer virus or whether “I Love You” is a simple greeting.

We present a novel algorithm for finding *concept extensions* which uses the concepts in the concept vector to address the problem of *under-selection*. Candidate concepts in the concept vector are compared against the raw detected entities in search of possible extensions.

We demonstrate our contributions by showcasing them in our “Contextual Shortcuts” platform, an instance of a user-centric entity detection system (which we describe in detail in Section 4). The Contextual Shortcuts platform bridges the gap between *entity detection* and *user-interaction* with the detected entities, in a customizable and scalable framework. The platform comprises a series of components for various tasks in the entity detection and presentation pipeline: HTML parsing, tokenization, entity detection (series of specialized entity detectors), context processing, entity filtering, and output generation of annotated (tagged) actionable entities in a variety of formats (e.g., HTML, XML, JSON).

In Section 2 we define the terminology used throughout the paper. Unless explicitly stated otherwise, when we refer to *entities* we typically also mean *concepts*. The remainder

⁵<http://search.yahoo.com>



Figure 1: Underselection issues in a popular user-centric entity detection system.

of the paper discusses the related work in this area, provides a system overview of Contextual Shortcuts, and presents the proposed algorithm for finding concept extensions and generating a concept vector in depth. We then present an extensive evaluation of the algorithms within Contextual Shortcuts, and show how they improve the overall quality of detected entities within our system. We discuss the results and conclude the paper with a summary and an outlook on future work.

2. DEFINITIONS

This section briefly defines the terminology we are using in this paper.

- **document**: A single piece of text, of any length, structured or unstructured, in either plain text or HTML format. Examples are news articles, email messages, blog postings, instant message conversation transcripts, advertisements, etc.
- **entity**: Refers to a piece of text of which is perceived or known or inferred to have its own physical existence (living or nonliving) in the real or virtual world. Examples of named entities are persons, organizations, or places, but also things like physical (street) addresses, phone numbers, email addresses, URLs, etc.
- **concept**: A piece of text that refers to an abstract thought or idea, which is not an entity. For example, “car insurance”, “Chinese restaurants in San Francisco”, “justice” are concepts.
- **concept extension**: A more specific concept that has been derived from a broader entity or concept. For example, “Betty Crocker’s famous cheesecake” is a concept extension of “Betty Crocker”.
- **under-selection and over-selection problems**: These problems occur when the boundaries of the presented entity or concept are not selected correctly. In the under-selection case, only a part of the larger concept

is detected, such as selecting only “Betty Crocker” within “Betty Crocker’s famous cheesecake”. In the over-selection case, some unrelated text is also included as part of the entity or concept, such as detecting “in California” versus “California” alone.

- **context**: A piece of text, of varying length, surrounding an entity or concept; can be used to infer more semantics about an entity or concept (e.g., for disambiguating the meaning of an entity).

3. RELATED WORK

We categorize the related work in this area into three categories. First, we discuss machine learning approaches, such as named entity detection and keyword extraction algorithms, and point out the differences in our system. Second, we describe other user-centric entity detection systems. Third, we discuss some recent studies, which also leverage search engine query logs.

3.1 Machine Learning Approaches

There is a significant amount of work in the field of named entity recognition (NER), where the goal is to locate and classify parts of free text into a set of predefined categories. The first named entity set had 7 categories: names of persons, organizations, locations, date, time, money and percent expressions [10]. However, most recent research has focused on recognizing person, organization and location names, since finding expressions for other categories are shown to be much simpler [16]. The approaches include manually generated regular expressions (see e.g., [1, 2]), and machine learning techniques (see e.g., [5, 6, 4, 3]). In general, the target consumers of the NER systems are machines, and common applications include question answering, summarization, and automatic correction of missing case information or misspellings in text. However, since our main goal is to identify the most “interesting” entities or concepts in the given text, rather than detecting all the entities that fall into predefined categories, our problem is rather different than that of traditional NER systems. Therefore we focus

on the quality of the detected entities in our system.

Also related to our work are the keyword extraction algorithms where the goal is to find the most distinctive or representative terms in text. These algorithms are typically based on machine learning techniques [8, 21, 12, 11]. However, compared to our user-centric entity detection problem, they are more suitable for applications such as text summarization, or automatically retrieving relevant documents based on the extracted keywords.

3.2 User-centric Entity Detection Systems

In early user-centric entity detection applications [17, 15], the user selects a piece of text, and the system’s goal is to recognize (meaningful) entities within that text to enable useful operations on them (e.g., adding events to the calendar, or starting a phone book application for phone numbers). These systems rely on regular expressions for the detection, whereas [17] also uses simple lookup tables. One of the main differences in our approach is that we automatically create actionable, intelligent hyperlinks in the text without requiring any interaction (i.e. manual selection of text) by the user. Secondly, we utilize context to achieve a higher quality of detection. Thirdly, we do not limit the system to simple rules or dictionaries, but rather use periodic a priori analyses of massive query logs to identify the entities that may be interesting to users in the context. This gives our system the capability of capturing real-world interests that are sensitive to time, allowing our system to dynamically evolve and not be limited to static rules or manually generated dictionaries.

In the area of targeted advertising, there are commercial offerings from the aforementioned companies such as Intel-liTXT, Vibrant Media, Kontera, and Ontok. These solutions focus exclusively on the *in-text advertising* problem, specifically on finding keywords in a web page that best match an advertiser’s ad to the content of the page, and then linking those keywords to mini-ads that lead to the advertiser’s main site. Although they are similar in spirit to our work from a system perspective, these solutions only address the advertising use case.

Another example that falls into this category is Microsoft’s SmartTags™ technology available in products such as Outlook XP™ for email. This technology offers a similar interaction model to ours, using the type of a detected entity to trigger different actions, e.g. address book for peoples’ names and map displays for addresses. However, currently this is only a proprietary technology tightly coupled to Microsoft’s desktop products, and the details of the system are not publicly available to external publishers.

3.3 Leveraging Search Query Logs

The idea of leveraging search query logs has been explored by many researchers to obtain improvements in various applications. The most closely related work to our approach is described in [9]. The goal in this work is to find keywords or phrases (implicit queries) in an email message that, once submitted to a search engine, would yield results relevant to the topic of the message. First, the authors investigate a simple $tf*idf$ based approach as a base system, which is also discussed in [7]. Then the authors show that if the implicit queries returned by this base system are restricted to those commonly found in the logs, then results improve significantly. For this restriction, the authors used the top 7.5

million most common queries in MSN Search, and discarded those generated queries that do not appear in this list. In the rest of their work, the authors also investigated machine learning techniques and used this restriction as a feature. In [20], a similar machine learning approach is employed to identify the best set of advertising keywords in web pages, i.e. keywords in a web page that are most interesting to advertisers. One of the main differences in our work is that we use query logs not as simple constraints as above, but instead utilize them to enrich our dictionaries. In addition, we further utilize query logs for concept extensions, which help avoid the under-selection problem.

4. CONTEXTUAL SHORTCUTS

The Contextual Shortcuts entity detection platform is an example of a *user-centric* entity detection system which aims to provide a pluggable architecture for entity detection and content syndication. It is designed to be highly scalable to support the detection of hundreds of millions of entities per day, and has already been successfully deployed on various Yahoo! network properties (e.g., Yahoo! Mail⁶, Yahoo! Personal Finance⁷, Yahoo! Travel⁸, Yahoo! News⁹).

Once entities have been detected (e.g., on a web page) Contextual Shortcuts provides a framework that allows publishers to associate relevant content and services to be displayed in context at the “point of inspiration”. For example, show a map for a place, add an email address to an address book, or show related information. The paper focuses on the entity detection aspects of Contextual Shortcuts, not the content syndication or presentation part.

The major components of the platform are shown in Figure 2:

Pre-processing: HTML parsing, tokenization, sentence, and paragraph boundary detection.

Entity Detection: Use of specialized detectors for detecting entities of various predefined types (e.g., places, persons, organizations, identifiers like URLs, email addresses, phone numbers, etc.), concept extension, and detectors that utilize Yahoo! search engine query logs to identify other terms or phrases.

Post-processing Collision detection between overlapping entities, filtering, and annotation.

Contextual Shortcuts uses *context* to improve its overall detection quality. We describe two algorithms that are leveraging context:

Generating a Concept Vector: This algorithm extracts concepts using data from search engine query logs.

Finding Concept Extensions: The concept extensions algorithm uses the concept vector, and its goal is to eliminate under-selection errors on the detected entities.

In below sub sections, we describe the major system components and the algorithms in detail.

⁶<http://mail.yahoo.com>

⁷<http://finance.yahoo.com/personal-finance>

⁸<http://travel.yahoo.com>

⁹<http://news.yahoo.com>

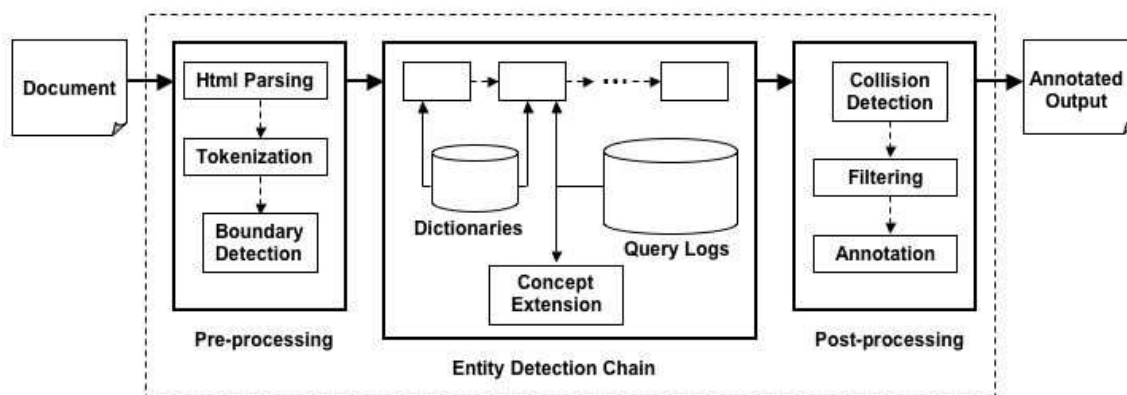


Figure 2: Contextual Shortcuts Platform

4.1 Pre-Processing

The pre-processing step expects plain text or HTML input, which is parsed by a highly optimized HTML parser and tokenized. The expected encoding format is UTF-8. Finally, as part of tokenization, a sentence and paragraph detection step is performed to aid in the downstream detection of entities. The processed text serves as the input for the set of entity detectors.

It can be seen that this pre-processing step provides a foundation for the accuracy of the detection. For example, HTML parsing problems, incorrect tokenization, or sentence/paragraph boundaries result in a severe degradation of detection quality.

4.2 Entity Detection

The Contextual Shortcuts platform supports a pluggable architecture, allowing a variety of entity detectors to be employed in the back-end. The goal is to allow the easy integration of additional detectors.

The components for detecting particular types of entities are the *detectors*. Detectors work on the parsed and/or tokenized input, which allows each detector to be simple and avoids re-parsing and re-tokenization.

In the current implementation we use a few detectors that are based on regular expressions for entities such as US phone numbers, URLs, and email addresses. The regular expression detectors work solely on the parsed input. The tokenized input is used with detectors such as the location detector (e.g., street addresses, places of interest) and the keyword detector (e.g., people, organizations).

Currently, all entity detectors are run consecutively, with each detector adding its entities to a global hit collection. This hit collection represents the input for the post-processing steps (e.g., ranking, filtering). Each entity returned by any detector has several properties, including the text of the entity as it appears in the input, the starting and ending offsets (in characters) of the entity in the given input before parsing, the type of the entity (phone number, street address, person, organization, etc.), and the surrounding context. The surrounding context has been extracted to help in finding concept extensions - see Section 4.2.2. The amount of extracted context is configurable. We currently set it to a fixed size window of tokens before and after the entity.

By modularizing the detectors we can easily run different configurations for different clients by building individual chains of detectors. The platform provides a Web Service API that allows publishers to control various aspects of entity detection, such as limiting the number and types of entities detected, as well as display and annotation formats.

Some of the detectors used require additional data to help detect entities. In particular, the keyword detector and the location detector use data-packs that are pre-loaded into memory to allow for high-performance entity detection.

The Contextual Shortcuts platform uses its own taxonomy of entity types. The current system consists of a handful major types, such as *people*, *organizations*, *places*, *events*, *animals*, *shopping and products*, and various kinds of *identifiers*. Each of these major types contains sub-types as shown for *places* and *identifiers*.

```

/places
  /places/<country code>/country
  /places/<country code>/town
  /places/<country code>/street
  ...

/identifier
  URL
  email
  phone_number/
    <country code>
  ups_tracking
  ...
  
```

Some of the types of entities are provided as editorially reviewed dictionaries, e.g. *people* and *organizations*. For keyword detection, we employ a set of dictionaries that contain categorized terms and phrases according to the taxonomy described above. We currently use 265,000 entities across 120 different types. It is possible that a term can be a member of multiple types, such as the term *jaguar* which exists in */products/auto* and in */flora_fauna/animals/predators*.

We use geo-spatial data to detect locations. This allows detection of complete addresses, place names like cities, counties, countries and places of interest. For some regions, where available, the data-pack contains enough information to also provide street address validation, i.e., it can detect

whether the house number provided is a valid address. Only those portions of the address that can be properly validated are detected, preventing the over-selection of location entities.

The data-packs are used to associate type information and meta-data to detected entities. In the case of keywords, the meta-data contains type information as listed in the entity taxonomy. In the case of locations, an entity can contain meta-data in the form of geo-location information. For example, it contains a breakdown of the address into its individual components and additional information such as longitude-latitude coordinates and aerial information. Passing such data along is beneficial to any service provider, e.g. for locations it frees the service from having to parse an address string again.

Although there are quality issues related to the tuning of regular expressions these detectors typically achieve very high accuracy and overall interestingness. We are therefore focusing our overall efforts on the quality on keyword-based entities.

To detect things of interest that go beyond editorially reviewed terms and phrases (e.g., peoples names, artists, company names, etc.), we employ terms and concepts derived from Yahoo! search engine query logs. Below, we first describe our *concept vector* generation algorithm in detail in Section 4.2.1. We then describe how we use the concepts detected in the concept vector in our *concept extensions* algorithm, which is crucial in dealing with under-selection errors.

4.2.1 Algorithm: Generating a Concept Vector

As mentioned previously, our named entity detection utilizes dictionaries that are maintained editorially and a data-pack with geo-spatial information. However, a very important resource for us are query logs, which allow the system to adapt to the current real-world interests in a timely fashion. In this section we describe how we generate a *concept vector* from query log data, which enhances the naked dictionary entities with new and interesting concepts.

First, given a document, a term vector is generated with a *tf*idf* score [19] of each term using a term dictionary which contains the term-document frequencies (i.e. the number of documents of a large web corpus containing the dictionary term). Our corpus in this case consists of all the web documents that are indexed by Yahoo! Search. The stop-words are removed and the remaining terms' weights are normalized so that they are between 0 and 1. The weights of terms that fall under a certain threshold are punished (their *tf*idf* score is decreased), and the resulting *tf*idf* scores below another threshold are removed from the term vector.

Second, a *unit vector* is generated of all the *units* found in the document. A unit, described in [18, 14], is simply a multi-term entity in the query logs which refers to a single concept. Briefly, units are constructed from query logs in an iterative statistical approach using the frequencies of the distinct queries as follows. In the first iteration, all the single terms that appear in queries are considered to be units. In the following iterations, the units that frequently co-occur in queries are combined into larger candidate units. The validation of these units is performed based on statistical measures, including mutual information, whose formal definition is shown below:

$$I(x, y) = \log \frac{p(x, y)}{p(x)p(y)} \quad (1)$$

where $p(x, y)$ is the joint probability distribution function of x and y , and $p(x)$ and $p(y)$ are the marginal probability distribution functions. Informally, mutual information compares the probability of observing x and y together as a query with the probabilities of observing x and y independent queries. If there is an association between x and y , then the score $I(x, y)$ will be higher.

For units, mutual information helps to identify those terms that frequently co-occur in user queries. Similar to the term vector scores, unit scores are also normalized so that they are between 0 and 1. Again, the weights of units that fall under a certain threshold are punished, and low scoring units are removed.

Finally, the term vector is merged with the unit vector to obtain the concept vector. We have the following possible cases:

1. A term appears in the term vector, but not in the unit vector: This suggests that the term did not appear as a popular query, so we add it to the concept vector, but punish its term vector weight.
2. A term appears in the unit vector, but not in the term vector: We simply add this term to the concept vector with its unit weight.
3. A term appears in both term and unit vector: We add this term to the concept vector, and we sum its term vector and unit vector weights.

We then inspect the merged concept vector, and perform the following additional step on the multi-term concepts. To the weight of the multi-term concept calculated in step two, we add both the unit vector and term vector scores of each individual term it contains. Thus, the maximum final concept weight possible is equal to two times the number of terms a multi-term concept contains (this would happen if each one of single terms it contains has unit vector and term vector scores of 1, which is not possible in practice). This way more specific concepts eventually bubble up in the overall rank. As an example, we list top five concepts in the news snippet shown in Section 1 with their concept vector scores:

```
<termvector id="concept">
  <item term="david espo" weight="1.4403">
  <item term="special correspondent" weight="1.2075">
  <item term="iraq war" weight="1.1833">
  <item term="president bush" weight="1.1549">
  <item term="political parties" weight="0.6147">
  ...
</termvector>
```

All steps and thresholds are configurable by setting various parameter values within a configuration. For example, there exists a threshold for a weak unit score. These threshold values are set empirically.

Note how a concept vector captures both the interestingness and relevance of the concepts found in the document. First, since unit vector scores are derived from query logs, they are a proxy for interestingness, i.e. a popular query

leads to an “interesting” (high unit vector weight) concept. Second, term vector scores are based on the all terms in the document with respect to a corpus, so they are a proxy for relevance, i.e. a “relevant” term has a high term vector weight. A concept vector is a merge between a unit and a term vector, and thus can be thought of measuring both interestingness and relevance.

4.2.2 Algorithm: Finding Concept Extensions

We argue that the problem of accurate selection of entities is very important in user-centric entity detection systems. In particular, we would like to avoid concept *under-selection* problems. In the example shown earlier where we have “Betty Crocker’s famous cheesecake” we would like to detect that whole concept, not only fragments like “Betty Crocker”. To prevent under-selection we present an algorithm for finding concept extensions by leveraging the concepts contained in the concept vector (see above).

The algorithm works as follows. For each candidate entity detected, we consider the concepts appearing in the local context surrounding the entity. If the entity is contained in a more specific concept (the entity’s text boundaries fall

```

/*
Input: S = E ∪ C, where
      E = {e0, e1, ..., en-1} (Set of entities)
      C = {c0, c1, ..., cm-1} (Set of concepts)
Output: modified S when concept extension is applied
*/
for each s in S
  for each concept c in the context of s
    if (s is fully contained in c) {
      remove s from S;
    }
return S

```

Figure 3: Algorithm for finding Concept Extensions

4.3 Post-processing

After all detectors are finished, the entity boundaries are inspected and collisions are resolved in cases of overlapping entities. Then a filtering step is performed to remove “uninteresting” concepts. Finally, the original text, annotated with the set of entities, is output in a variety of possible formats.

Collision Detection: The platform allows simple detectors to be tailored towards very narrow and precise detection capabilities. Therefore, it is possible to encounter collisions between entities of different detectors. Suppose one of the detectors detects the institution “University of Rome”, whereas a location detector returns part of it, “Rome”, as a place. We consider returning “Rome” alone as a case of under-selection. To avoid such problems, we favor the more specific hit in cases where one of the entities fully contains another. For other cases (e.g. partial overlap), we employ simple heuristics such as making a decision based on the order of the entities in the text.

Filtering: Since we use query log data as a resource, the detected entities may either be junk (uninteresting random concepts) or may contain certain sensitive (e.g., illegal, con-

troversial, or adult) terms. The filtering step consists of removing concepts that fall below a certain concept weight - see Section 5.3 - and also of removing blacklisted terms. Similarly, stop-words and certain ambiguous terms contained in our data-pack are also filtered out.

Annotation: As a final step the remaining entities and concepts are returned as either a result set in XML or JavaScript Object Notation (JSON), or as annotated HTML. Each entity or concept is returned with a weight, a set of types, a context and (if available) meta-data, such as longitude-latitude coordinates for places. The XML output format is particularly suitable for applications that only need access to the detected entities.

5. EVALUATION AND RESULTS

We describe the methodology used to evaluate the utility of the Contextual Shortcuts platform along the three aforementioned criteria – **accuracy**, **interestingness**, and **relevance** – and present the respective results across a sample corpus of documents. Furthermore, we specifically highlight the roles and contributions of the algorithms mentioned in Sections 4.2.2 and 4.2.1 to the overall results.

5.1 Methodology

Our standard evaluation methodology consists of a team of expert judges rating the entities / concepts detected and presented by Contextual Shortcuts in a set of documents in a corpus. Using an interface specifically designed for this task, the processed set of documents is presented to the judges. A judge is asked to select a document from the pool of documents, and once selected, asked to read the document carefully prior to issuing any judgments. Each document shows the detected entities (through highlighting), along with the surrounding context. The judge is then asked to rate each entity or concept highlighted in the document in terms of its accuracy, interestingness, and relevance.

5.1.1 Question 1 : Accuracy

Question 1 asks if the detected entity or concept is correctly selected in the text in terms of its logical boundaries. We distinguish primarily between the two error cases of *under-selection* and *over-selection*. The judges can select from the following choices:

Yes – The highlighted entity or concept has been selected correctly, i.e., no under- or overselection occurred.

No – The boundaries of the highlighted entity or concept are incorrect. Additionally, the judge then indicates whether an under- or over-selection occurred, or whether the entity is simply a “bad term”, e.g. random nonsense.

5.1.2 Question 2 : Interestingness

Question 2 attempts to measure the degree to which the highlighted entity or concept is interesting, useful, or compelling enough to tear the reader away from the main thread of the document. Would the reader take time out from reading the document to actually click on the entity or concept and explore it further?

The judges can select from the following choices:

Interesting or Useful in General – It is very likely that the reader will find this entity or concept interesting or

useful enough to take immediate further action. Moreover, the entity or concept is interesting enough on its own, *regardless of the context* it is in.

Interesting or Useful Only in This Context – This entity or concept on its own would not pique a user’s curiosity, but because of the context its in, it does become compelling enough to investigate further. An example of this may be: “President Bush, otherwise known as The Decider, held a press conference...”. *The Decider* may not be interesting enough on its own, but in the context of “President Bush”, it becomes compelling.

Definitely Not Interesting or Useful – There is no plausible scenario in which the reader will find this entity or concept interesting or useful.

Keep in mind that an interesting entity or concept may not necessarily be *relevant*, i.e. central to the gist or topic of the document, or vice versa. In our guidelines the interestingness of entities or concepts is independent of their relevance to the meaning of the document.

5.1.3 Question 3: Relevance

Question 3 attempts to measure the degree to which the highlighted entity or concept is relevant to the main topic of the document. We can think of this as “summarizing” the document with the best set of entities, regardless of their interestingness.

The judges can select from the following choices:

Relevant – The entity or concept is core to the overall meaning or topic of the document. You could not accurately summarize the given text without mentioning this entity or concept. Example: The entity *Iraq* is a relevant entity in a story about the Iraq war – without it, you could not accurately relate the story.

Somewhat Relevant – This entity or concept fits into the overall sense of the document, but only peripherally. If you omit this entity in your “summary”, then you’ll still capture the overall sense of the text, but some minor details may be missing. Example: The document is a story about the Iraq war, and *M1-Abrams* is a found as an entity.

Not Relevant – This entity or concept is certainly not relevant to the meaning of the document. Example: The document is a story about the Iraq war, and includes a quote from a soldier that calls Ukiah, CA home – *Ukiah* is not a relevant entity in this case.

Each of the above questions also includes a “Can’t Tell” choice for those rare case when a judge can’t decide.

5.2 Concept Extensions Results

We experimented with our concept extension algorithm on a corpus consisting of 1,304 documents, randomly selected from various sources (549 from Yahoo! Answers Q & A’s¹⁰, 353 from the Enron mail corpus¹¹, and 403 from Yahoo! news stories¹²). This corpus yielded a total of 2,305 entities, 376 of which were extensions. The accuracy scores and

¹⁰<http://answer.yahoo.com>

¹¹<http://www.cs.cmu.edu/~enron/>

¹²<http://news.yahoo.com>

With Extensions	Qty	OK	US	OS	BT
Dictionary Entities	1929	1833	61	4	14
Concept Extensions	376	337	7	7	1

Table 1: Accuracy results for dictionary entities and concept extensions (OK = correct, US/OS = under/over-selection, BT = bad term). The overall accuracy of the system is 95.8%.

Without Extensions	Qty	OK	US	OS	BT
Dictionary Entities	2305	1840	405	4	15

Table 2: Accuracy results when concept extensions are not used. All entities are now plain dictionary entities. The number of under-selections increases, and the over-selections from table 1 are now correct. The overall accuracy of the system is 81.3%.

the effect of the concept extension technique are reflected in Tables 1 and 2.

As Tables 1 and 2 show, concept extensions play a large role in maintaining high accuracy - without them, overall accuracy drops from 95.8% to 81.3%, driven by the number of under-selections. The reason for this is that the 337 extended entities in Table 1, which were judged to be correct, are reduced to naked entities 2, thus becoming under-selections.

5.3 Concept Vector Results

We investigated the relationship between the concept weight derived from the concept vector and the interestingness of the concepts in the vector. This was done using a corpus consisting of 352 Yahoo! Finance web pages, which yielded 2,099 entities (concepts) to be judged. Since interestingness is by nature a subjective measure, we needed to obtain multiple editorial judgments for each concept to gauge the level of agreement among editors. This is in fact what happened, and each of the 2,099 concepts received 2 or more judgments, with the majority of cases receiving exactly 2 judgments. Out of the 2,099 judgments, the editors disagreed on the interestingness rating in 1,060 instances. These 1,060 cases were not considered, leaving 1,039 judgments for further analysis. Table 3 shows the interestingness breakdown for these 1,039 cases.

The results in Table 3 show that the overall interestingness of concepts detected in this study is approximately 76%. However, this is not ideal, and we can improve on this by only considering concepts that meet a certain cutoff value, and filtering out the rest. The results in Table 3 already

Rating	Quantity	Percentage
Interesting in general	313	30.1%
Interesting in only in context	478	46.0%
Not interesting	248	23.9%

Table 3: Interestingness scores for detected concepts. All concepts have a minimum concept weight of 1.0.

reflect a minimum concept weight (cutoff) of 1.0. Figure 4 shows the effect on the overall interestingness and coverage of concepts as the minimum concept weight is increased past 1.0. Coverage is calculated as the number of concepts remaining at each cutoff value, as a percentage of the total number of concepts at the initial cutoff value.

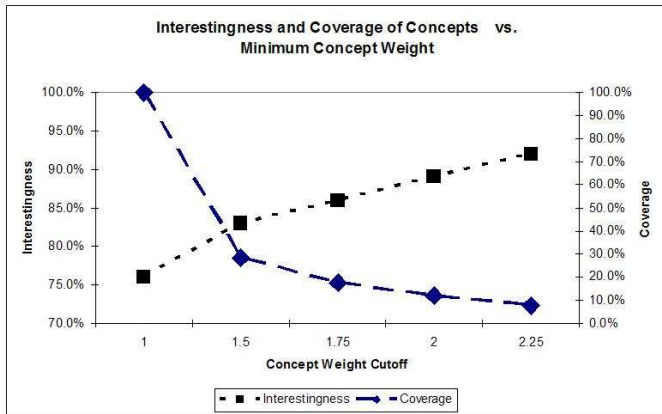


Figure 4: Overall interestingness as a function of concept weight cutoff. As the minimum concept weight is increased, the overall interestingness of the remaining concepts increases. The price for the overall increase in interestingness is a sharp decrease in coverage.

5.4 Overall Results

Having tuned the above algorithms with respect to concept extensions and minimum concept weight, we now put all the pieces together and evaluate the final system with respect to the overall accuracy, interestingness, and relevance. This is done on a fresh corpus, similar in flavor to the previous ones, but with new content. These results, shown in Table 4, are based on a set of 1,586 entities, generated from a corpus of 1,519 documents from various sources (Yahoo! Answers Q & A’s, the Enron mail corpus, Yahoo! news stories, newsgroup postings, and user product reviews). As Table 4 shows, the overall combination of concept extensions, concept vector generation, and concept filtering combine to yield an accuracy score of 97.3%, an interestingness score of 94.6%, and a relevance score of 90.1%.

5.5 Discussion

The use of dynamic search query logs to generate new interesting concepts and concept extensions can be of great value. However, given that search query logs contain many random and noisy entries, care must be taken when using them as a source of entities.

Concept extensions are an excellent tool to combat under-selection. There are cases where they intelligently identify extensions, such as extending the naked entity *Paris* to the concept *Paris fashion week* in the context “...as designer Marc Jacobs took Paris fashion week to its final phase...”. However, they can also miss, e.g. over-selecting the entity *Sinn Fein* to the concept *included Sinn Fein* in the context “...support at the last assembly election in 2003 because it included Sinn Fein in the Cabinet”. Mechanisms must ex-

Overall Results	Percent
Accuracy	
Correct Selection	97.3%
Underselection	2.0%
Overselection	0.3%
Bad Term	0.2%
Can’t Tell	0.3%
Interestingness	
Interesting in General	77.5%
Interesting Only in Context	16.8%
Not Interesting	5.0%
Can’t Tell	0.6%
Relevance	
Relevant	62.7%
Somewhat Relevant	27.4%
Not Relevant	9.9%
Can’t Tell	0.0%

Table 4: Overall accuracy, interestingness, and relevance scores for Contextual Shortcuts.

ist to filter out such incorrect selections, and we use the concept weight derived from the concept vector as one such mechanism. However, this is not without a price, since the concept weight by itself is a very “asymmetric” and sensitive filter. To borrow a term from economics, Figure 4 shows that concept weight is very *elastic* with respect to coverage (small changes in weight lead to large changes in coverage), whereas it is very *inelastic* with respect to interestingness (large changes in concept weight lead to small changes in interestingness). A deeper analysis of the generated concepts is required to maintain a high level of interestingness without greatly sacrificing coverage.

Finally, the goal of many user-centric entity detection systems (including Contextual Shortcuts) is to maximize the number of actions taken by users on the presented entities. For example, one such measure is the *click through rate (CTR)*, defined as the number of clicks (action) divided by the number of entities presented in a given time period. This measure is driven by the interestingness rather than the relevance score (see table 4), since interestingness measures a user’s propensity to click on an entity. Thus, in the extreme case, one trivial approach to boost interestingness and click-through-rate would be to simply inject some sensational pieces of text somewhere in each document, e.g. “Vaccine for AIDS Found!”, without regard to their relevance. However, this would be no different than spam, quickly leading users to ignore such “entities”. Thus the second challenge is to boost interestingness without also sacrificing relevance; one can think of the interestingness/relevance metrics of user-centric entity detection systems as the precision/recall metrics of machine-centric entity detection systems.

6. CONCLUSIONS AND FUTURE WORK

This work addresses the quality issues of detected entities in a user-centric entity detection system such as the Contextual Shortcuts platform. We show how we can leverage local and global context surrounding the entities to improve the overall quality and utility of the system. Specifically, we present two novel algorithms for using context.

First, we use query logs to discover additional concepts in the given document, thus forming a representation of the document as a ranked list of concepts - the concept vector. We then use the ranking of the concepts in the vector (the concept weight) as a measure of both the interestingness and relevance of the concept, and utilize the weight in a filtering mechanism to remove uninteresting and irrelevant concepts. Second, we use the concepts in the concept vector to avoid under-selecting entities in the text.

We argue that the way to measure the overall utility of detected entities in user-centric detection systems, where people can interact with the detected entities, is via the metrics of accuracy, interestingness, and relevance. We present results showing accurate, interestingness, and relevance scores in the 90%+ ranges, derived from an editorial evaluation of the Contextual Shortcuts platform.

As we move forward, we expect to expand our detection capabilities to cover a broader set of entities and concepts. Increasing the coverage of interesting concepts while maintaining high levels of relevance is one of our greater challenges. We hope to address this issue through a deeper analysis of the candidate concepts, potentially building a concept classifier. Disambiguation will also play a larger role in future versions, since determining the correct type (person, place, organization, event, things) of an entity is important to its overall utility.

7. ACKNOWLEDGMENTS

We are grateful to Farzin Maghoul for his helpful comments and suggestions. Furthermore, we would like to thank Jörg Meyer, Josh Coalson, our editorial team, and everyone else here at Yahoo! who helped developing and building the Contextual Shortcuts platform.

8. REFERENCES

- [1] D. Appelt, J. Hobbs, J. Bear, D. J. Israel, and M. Tyson. FASTUS: a finite-state processor for information extraction from real-world text. In *Proceedings of IJCAI-93*, 1993.
- [2] D. E. Appelt, J. R. Hobbs, J. Bear, D. Israel, M. Kameyama, A. Kehler, D. Martin, K. Myers, and M. Tyson. SRI International FASTUS system MUC-6 test results and analysis. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, pages 237–248, San Francisco, 1995. Morgan Kaufmann.
- [3] S. Baluja, V. Mittal, and R. Sukthankar. Applying Machine Learning for High Performance Named-Entity Extraction. *Computational Intelligence*, 16(4), November 2000.
- [4] O. Bender, F. J. Och, and H. Ney. Maximum entropy models for named entity recognition. In *Seventh Conference on Natural Language Learning (CoNLL-03)*, 2003.
- [5] D. M. Bikel, R. L. Schwartz, and R. M. Weischedel. An algorithm that learns what's in a name. *Machine Learning*, 34(1-3):211–231, 1999.
- [6] A. Borthwick, J. Sterling, E. Agichtein, and R. Grishman. Exploiting diverse knowledge sources via maximum entropy in named entity recognition. In *Proceedings of the 6th Workshop on Very Large Corpora*, 1998.
- [7] S. Dumais, E. Cutrell, R. Sarin, and E. Horvitz. Implicit queries (IQ) for contextualized search. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 594–594, New York, NY, USA, 2004. ACM Press.
- [8] E. Frank, G. Paynter, I. Witten, C. Gutwin, and C. Nevill-Manning. Domain-specific keyphrase extraction. In *Proceedings of the 1999 International Joint Conference on Artificial Intelligence*, pages 668–673, 1999.
- [9] J. Goodman and V. R. Carvalho. Implicit queries for email. In *Proceedings of the 2nd Conference on Email and Anti-Spam*, 2005.
- [10] R. Grishman and B. Sundheim. Design of the MUC-6 evaluation. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, pages 1–11, San Francisco, 1995. Morgan Kaufmann.
- [11] M. Henzinger, B.-W. Chang, B. Milch, and S. Brin. Query-free news search. In *Proceedings of the 12th International World Wide Web Conference (WWW)*, pages 1–10, 2003.
- [12] A. Hulth. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 216–223, 2003.
- [13] P. Jackson and I. Moulinier. *Natural Language Processing for Online Applications*. John Benjamins Publishing Company, 2002.
- [14] S. Kapur and D. Joshi. Systems and methods for generating concept units from search queries. *United States Patent 7051023*, May 2006.
- [15] B. A. Nardi, J. R. Miller, and D. J. Wright. Collaborative, programmable intelligent agents. *Communications of the ACM*, 41(3):96–104, March 1998.
- [16] D. Palmer and D. Day. A statistical profile of the named entity task. In *Proceedings of the Conference on Applied Natural Language Processing*, 1997.
- [17] M. Pandit and S. Kalbag. The selection recognition agent: Instant access to relevant information and operations. In *Proceedings of the International Conference on Intelligent User Interfaces*, 1997.
- [18] J. Parikh and S. Kapur. Unity: relevance feedback using user query logs. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, 2006.
- [19] G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. Technical report, Ithaca, NY, USA, 1987.
- [20] W. tau Yih, J. Goodman, and V. R. Carvalho. Finding advertising keywords on web pages. In *Proceedings of the 15th international conference on World Wide Web*, pages 213–222, New York, NY, USA, 2006. ACM Press.
- [21] P. Turney. Learning algorithms for keyphrase extraction. *Information Retrieval*, 2(4):303–336, 2000.