Contextual Ranking of Keywords Using Click Data

Utku Irmak #1, Vadim von Brzeski #2, Reiner Kraft #3

#*Yahoo! Inc.*
701 First Avenue Sunnyvale, CA USA
1uirmak@yahoo-inc.com 2vadimv@yahoo-inc.com 3reiner@yahoo-inc.com

***Abstract*— The problem of automatically extracting the most interesting and relevant keyword phrases in a document has been studied extensively as it is crucial for a number of applications. These applications include contextual advertising, automatic text summarization, and user-centric entity detection systems. All these applications can potentially benefit from a successful solution as it enables computational efficiency (by decreasing the input size), noise reduction, or overall improved user satisfaction.**

In this paper, we study this problem and focus on improving the overall quality of user-centric entity detection systems. First, we review our concept extraction technique, which relies on search engine query logs. We then define a new feature space to represent the interestingness of concepts, and describe a new approach to estimate their relevancy for a given context. We utilize click through data obtained from a large scale user-centric entity detection system – Contextual Shortcuts – to train a model to rank the extracted concepts, and evaluate the resulting model extensively again based on their click through data.

Our results show that the learned model outperforms the baseline model, which employs similar features but whose weights are tuned carefully based on empirical observations, and reduces the error rate from 30.22% to 18.66%.

I. Introduction

The problem of determining the most *relevant* and *interesting* entities and concepts - the *key concepts* - in a given document is of growing importance due to the ever increasing amount of information available on the World Wide Web. The problem is not simply one of determing the set of all named entities and phrases in a document - a natural language problem which has been attacked extensively via rules, dictionaries, and machine learning algorithms [1] - but of refining that set to a smaller set of highly relevant and interesting concepts that accurately capture the topic of the document and are interesting to a large base of users.

In this paper, we focus on this problem and present a novel technique for determining and ranking the key concepts in a document. Extracting the key concepts from documents plays a major role in a variety of applications, including contextual advertising systems, search engines, and user-centric entity detection systems. Therefore all these applications can potentially benefit from the techniques that we present in this paper.

A. Applications

Contextual advertising systems attempt to determine the most relevant textual ads for a given document (e.g. webpage), and embed those ads into the document. Examples of such systems are Yahoo's Contextual Ads and Google's AdSense programs. These systems first attempt to discover the relevant keywords in a document, and then find the ads that best match the set of keywords [2]. It has been shown that reducing a document to a small set of key concepts can improve performance of such systems by decreasing their overall latency (processing time) without a loss in relevance [3].

Text summarization is an important component of Web search engines. Search engines can potentially return a large number of search results (URLs) for a given query, so in addition to each result, a small snippet of text is included that attempts to summarize the contents of the page behind the URL. Users scan these snippets to understand which URLs contain the most relevant results for their queries. Therefore, providing effective summaries via key concepts can increase the overall user satisfaction. Furthermore, effective web page summarization is critical in handheld and mobile web applications where the display real estate (screen size) is smaller than in conventional personal computers [4].

User-centric entity detection systems, where the primary consumer of the detected entities is a human (not a machine), benefit from detecting just the key concepts in document by improving the overall user experience. User-centric entity detection systems not only detect entities and concepts within text (e.g., web pages), but also transform those detected entities into actionable, "intelligent hyperlinks". These hyperlinks provide additional relevant information about the entity in a single-click, e.g. detecting an address and showing a map to it, detecting a product name and showing an ad, etc. In these cases, the quality (not necessarily the quantity) of detected entities matters, since not every single entity or concept is necessarily relevant to the topic of the document, nor is it necessarily interesting. Examples of user-centric entity detection systems are *IntelliTXT*[1], *Vibrant Media*[2], *Kontera*[3], and our own *Contextual Shortcuts*.

B. Approach

In previous work [5] around leveraging context in user-centric entity detection systems we proposed 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 showed that leveraging surrounding context can greatly improve the performance of such systems in all three dimensions. In the past months we collected a large amount of click-through data from users using our Contextual Shortcuts platform. Given all

[1]http://www.intellitext.com
[2]http://www.vibrantmedia.com
[3]http://www.konterea.com

this user interaction data we were motivated to see how we can leverage it to further improve the overall detection quality in regards of relevancy and interestingness.

This paper presents a novel approach for determining the most interesting and relevant set of entities and concepts in a document (the key concepts), using implicit user feedback in the form of click data collected in Contextual Shortcuts. Specifically, given a document and a candidate set of entities extracted “naively” from the document, the algorithm will return the entities ranked in decreasing order of interestingness and relevance. An application can then choose the top N entities from this ranked list.

We believe the user click data is a good measure of relevance and interestingness since each click on an entity or concept means that a user took a detour from his or her current task to investigate the entity or concept further. The assumption is that the more relevant an entity is to the topic of the document and the more interesting it is to the general user base, the more clicks it will ultimately get. Intuitively, we define an entity to be relevant to a document if a shortened synopsys of the document cannot leave the entity out without losing a key idea. Interestingness can be thought of as a measure of how likely a reader is to want further information regarding the entity (either see a map of a place, search/news results for a person, etc.). Combined, relevance and interesingness determine the likelihood of a person actually clicking on the entity. For example, consider the following snippet,

President Bush’s position was similar to that of New York Sen. Clinton, who argued at a debate with Obama last week in Texas that there should be no talks with Cuba until it makes progress on releasing political prisoners and improving human rights.

In the above snippet, *President Bush*, *Sen. Clinton*, *Obama*, and *Cuba* are all relevant and interesting entities to the reader of the above text. However, *Texas* is not at all relevant to the topic of the snippet (President Bush’s, Sen Clinton’s, and Obama’s positions on talks with Cuba), and even though it may be interesting to some users, we expect the click numbers for Texas to be much lower compared to the other entities. Another way to look at relevance is via a substitution argument: if the entity can be replaced with another entity of the same class/type without losing the meaning of the text, the entity is not relevant, e.g. *Texas* can be replaced with *Ohio* in the snippet above without changing main idea of the text.

The click data we use is in the form of click-through-rate (CTR) data mined from user interactions with Contextual Shortcuts in documents published on Yahoo! News [4]. We use CTR data, along with the context surrounding the entity, in a machine learning approach to build a ranking function that ranks a set of entities found in a document. The feature space consists of attributes related to the relevance of an entity in a context, as well as attributes related to how interesting an entity is to the general base of users. Features related to relevance are primarily derived from analysis of document corpora, whereas features related to interestingness are derived from analysis of search query logs.

[4]http://news.yahoo.com

C. Contributions

In this paper, we study the problem of determining and ranking the key concepts in a document, which plays an important role in a number of applications. The major contributions of this paper are listed below:

- We propose to use implicit user feedback in the form of click data to determine the most interesting and relevant concepts in a context via a machine learning approach.
- We describe a feature space pertinent to the interestingness of a concept, and present algorithms to identify relevance of a concept in a given context.
- We evaluate the proposed techniques extensively using click data, an editorial study, and an analysis on production system. The results show significant improvements.
- We provide a detailed description of a framework that enables efficient implementation of the proposed techniques in a production system.

The remainder of the paper provides an overview of Contextual Shortcuts, describes the learning methodology and the space of features. We then present our results, describe the framework and conclude the paper with a summary and an outlook on future work.

II. Contextual Shortcuts

The Contextual Shortcuts entity detection platform is an example of a *user-centric* entity detection system which provides a framework for entity detection and content syndication. It is designed to detect interesting named entities and concepts (the *key concepts*) in unstructured text, and annotate them with intelligent hyperlinks (shortcuts) - thus the name Contextual Shortcuts. Clicking on a Shortcut results in a small overlay window appearing next to the detected entity, which shows content relevant to that entity, e.g. a map for a place or address, or news/web search results for a person, etc. The system is highly scalable, detecting hundreds of millions of entities per day, and has been successfully deployed on various Yahoo! network properties (e.g., Yahoo! Mail[5], Yahoo! News[6]).

The major components of the platform are as follows; for more details, please refer to [5]. A sequence of pre-processing steps handles HTML parsing, tokenization, sentence, and paragraph boundary detection. Next, specialized detectors discover entities of various predefined types (e.g. places, persons, organizations, identifiers like URLs, emails, phone numbers, etc.), as well as abstract concepts derived from Yahoo! search engine query logs. Finally, a sequence of post-processing steps handles collision detection between overlapping entities, disambiguation, filtering, and output annotation.

[5]http://mail.yahoo.com
[6]http://news.yahoo.com

A. Entity Detection

This paper focuses on improving the relevance and interestingness of detected entities, so we now give a brief summary of the entity detection process. Entities fall into three types:

1) Pattern based entities (emails, URLs, phones, etc.)
2) Named entities (names of concrete people, places, organizations, products, etc.)
3) Concepts (abstract entities, e.g. “auto-insurance”, “science fiction movies”, etc.)

Pattern based entities are primarily detected by regular expressions. To provide a level of consistent behavior to the end user, pattern based entities are not subject to any relevance calculations are always annotated and shown to the user. Although there are quality issues related to the tuning of regular expressions, they typically achieve very high accuracy and overall relevance. We therefore focus our overall efforts on the quality of named entities and concepts.

Named entities are detected with the help of editorially reviewed dictionaries. The dictionaries contain categorized terms and phrases according to a pre-defined taxonomy. The current system consists of a handful major types, such as *people*, *organizations*, *places*, *events*, *animals*, *products*, and each of these major types contains a large number of subtypes, e.g. *actor*, *musician*, *scientist*, etc. It is possible that a named entity can be a member of multiple types, such as the term *jaguar*, in which case the entity is disambiguated. The named location detector (places and street addresses) also uses data-packs that are pre-loaded into memory to allow for high-performance entity detection. The data-packs are used to associate type information and meta-data to detected entities, such as the type information as listed in the entity taxonomy. In the case of locations, the meta-data contained geo-location information (longtitude, latitude).

Concepts are detected using data from search engine query logs, thus allowing the system to detect things of interest that go beyond editorially reviewed terms. Below we describe our *concept vector* generation algorithm; for more details on the algorithm and Contextual Shortcuts, please see [5]:

B. Generating a Concept Vector

Given a document, first a term vector is generated with a *tf*idf* score [6] 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 [7], [8], 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:

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

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 to be 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 may have the following 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 I 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>
```

III. Methodology

We now outline our general approach to the entity ranking problem using user click data. As mentioned in section II, Contextual Shortcuts is integrated into various Yahoo! properties, one of them being Yahoo! News. This means that each news story shown on Yahoo! News is pre-processed via Contextual Shortcuts prior to being published. The interesting entities and concepts in each story (as determined by the current production version of Contextual Shortcuts - the *baseline* version) are annotated with Shortcuts (intelligent hyperlinks). In randomly sampled news stories (due to data volume constraints), the Shortcuts also include tracking information that allows us to capture whether a reader of the story actually clicked on the entity. This click data is aggregated and is mined to provide the following reports on a weekly basis for each sampled news story:

- the text of the news story
- a list of the entities annotated in the story, along with metadata for each entity (taxonomy type, position in text)
- the number of times each entity was viewed
- the number of times each entity was clicked

Note that for a given news story, the number of times each entity was viewed on that page is the same for all entities on that page, and that number equals the number of times the news story itself was viewed.

The view and click data above constitute the training and test data in our machine learning approach. Our goal is: given the candidate set of entities produced by the baseline system, learn a ranking function which orders the entities by their interestingness and relevance. In this case, click-through-rate (CTR), defined as the number of clicks divided by the number of views, is our proxy for interestingness and relevance - the assumption being that entities receiving higher CTR values are more interesting and relevant to users.

We split the problem into two parts: (a) determining whether the entity is relevant to the given context, and (b) determining whether the entity is interesting outside of the context. In order to produce a system that can be deployed in production and that can adhere to strict runtime constraints, we initially focus our efforts on a large, but finite set of entities, namely the set of named entities in our dictionaries plus a large subset of all the concepts available to us from query logs - see section II. For each entity and concept in this set, we first pre-compute (offline) a static set of features (for relevance and interestingness) from the training data - see section IV. For relevance, we construct a model for each entity and concept, and use this model at runtime to compute a *relevance score* for a given entity with respect to a (new) given context. For interestingness, we build a single model that given a set of entities, returns the entities ranked in decreasing order of interestingness. Note that the relevance score computed for each entity can by itself be used to rank the entities, but it can also serve as a feature in an overall (learned) model for interestingness and relevance. The final ranking is determined by this overall model.

We use an implementation of ranking SVM to learn a ranking function between pairs of instances. An open source library for ranking SVM [9] is also available in SVM^{light} [7]. [10] describes an open source library [8] for large-scale linear classification which can also be used to learn ranking functions without kernels. In our case, each instance consists of the entity/concept along with its associated features, and the label of each instance is its CTR value. The specific features used in the system and data pre-preprocessing steps are described in Section IV.

IV. Feature Space

In Section II, we reviewed our approach to extract a large set of general concepts in a given document. However, as mentioned earlier, our goal in this paper is to identify the key concepts in this candidate set through ranking. Such concepts would be highly interesting in general (independent of the context) and/or highly relevant in the context. In this section, we first define features that we use to estimate the interestingness of a concept, and then define a new approach to estimate the relevance of a detected concept in the context. Finally, we provide a discussion on other possible features and provide more insights on some practical issues.

A. *Interestingness of a Concept*

With interestingness, we would like to measure if a concept would be appealing to a broad user base in general. At this point, we do not focus on any personalization or profiling aspects, but rather try to capture how interesting a concept would be to a large population of users. For example, concepts like "global warming" or "tom cruise" can be examples of highly interesting concepts. Note that under this definition, the interestingness of a concept does not really depend on the context in which it appears (although context is fundamental for relevance). In this subsection, we define a feature vector to capture the interestingness of a concept; the contributions of the individual fields will be determined through machine learning techniques. The complete feature vector constructed for each concept is shown in Table I. Below, we categorize the features and describe them in detail.

Search Engine Query Logs: Our first intuition is that the search engine query logs should provide a strong correlation with the interestingness of a concept. In other words, if a concept is interesting in general, there should be a quite large number of queries submitted to an engine for that concept. We used following features mined from query logs: (1) freq_exact: number of queries received that are exactly same as the concept, (2) freq_phrase_contained: number of queries that contain the concept as a phrase, (3) unit_score: mutual information of the terms in the concept, as described in Section II.

When we construct our features from query logs, we essentially focus on the frequencies; we do not perform any categorization to understand their intentions such as *navigational*,

[7]http://svmlight.joachims.org
[8]http://www.csie.ntu.edu.tw/~cjlin/liblinear

	Features
1	freq_exact
2	freq_phrase_contained
3	unit_score
4	searchengine_phrase
5	concept_size
6	number_of_chars
7	subconcepts
8	high_level_type
9	wiki_word_count

TABLE I
FEATURE SPACE FOR INTERESTINGNESS (AFTER FEATURE SELECTION)

transactional or *informational* queries (see [11]), although there might be potential benefits in doing so.

In addition to these three features, we tried with a number of variations. These include considering queries and concepts as bags of words (which do not require any ordering), and define a cosine similarity threshold to identify similar queries to the concept. During our feature selection process, none of these features prove to improve upon the features mentioned above, and therefore we decided not to include them here. A more detailed discussion on how to measure similarity between two very short segments of text (such as queries and concept), can be found in [12].

Search Engine Result Pages: We submit the concept to the search engine as a phrase query, and use the number of result pages returned as a feature ((4) searchengine_phrase). The intuition behind this is to get an estimation on the specificity of the concept; very specific concepts would return fewer results than the more general concepts. This approach is also used in [13] for a different problem. During our feature selection, a variation which submits the concept as a regular query is eliminated, therefore it is not included in Table I.

Text Based Features: We specify the following features derived from simple text analysis. (5) concept_size: number of terms in the concept; (6) number_of_chars: number of characters in the concept; (7) subconcepts: number of subconcepts contained in the concept that have more than two terms and have a unit score of larger than 0.25.

Taxonomy Based Features: If the concept exists in one of the editorially maintained lists, then the high level type, such as place, person, organization, etc., is used as a feature for (8) high_level_type.

Other: As Wikipedia continues to grow, we are able to find entries about many concepts in Wikipedia. Although measuring the quality of the articles in Wikipedia is an open problem, the length of the articles is shown to be a useful feature [14], and we decided to use the length of the article as a feature. (9) wiki_word_count: number of words in the Wikipedia article returned for the concept, and 0 is used if no article exists. We also tested with features that utilize idf (inverse document frequency) value of the individual terms that appear in the concept, however, these features were not useful and eliminated during feature selection process.

B. Relevance of a Concept in a Context

Relevance is quite important in ranking the concepts in a given context, and it applies to all of the applications mentioned above. Clearly, text summarization techniques would not be helpful if the summaries returned were not relevant to the overall content. The studies described in [15], [3], [16] indicate that the ads related to the page content receive more clicks. In user-centric entity detection systems, relevance plays an important role and those concepts with low relevance usually do not attract users to take further action unless they are extremely interesting.

In this subsection, we propose a mining approach to obtain a good relevance scoring mechanism. Our main goal with this approach is obviously to identify those concepts that are most relevant in the context. However, we would like to note that the initial candidate set may contain very general or low quality concepts (such as “my favorite”, “the other”, “what is happening”, etc). The reason for having them as candidates usually relates to their high unit scores, as described in Section II. Similarly, they might even get ranked higher due to the noisy fields in the interestingness feature space (such as search query logs, etc). However, we would expect the relevance scoring mechanism to address this issue since these concepts should almost never get a high relevance score in any context. In other words, our second goal with relevance is to provide a score that can be used as a safety net for such concepts. As described in the following sections, we employ machine learning techniques to learn the best rules (weights) to identify such cases. In summary, we expect the relevance score to play an important role in conjunction with the interestingness features in the ranking of the candidate concepts.

Our main idea for computing the relevance score of a concept in a context is quite simple. Assume for now that we have constructed a set of relevant context keywords for that concept in advance. In other words, assume that we have extracted, say the top hundred most distinctive terms from contexts in which the concept was known to be relevant. Now, given a new context containing that concept, we would like to find out the relevancy of the concept in the context. We argue that a reasonable approximation for the relevance of that concept can be computed based on the co-occurrences of the pre-mined keywords and the given concept in the context. If none of the keywords occur, we can conclude that the concept is not relevant in the context, and it is likely that the model will rank that concept much lower based on this fact. Below we describe our approach for mining the relevant keywords, and the resources we use in more detail.

1) Mining Relevant Keywords for Each Concept: We now formally define the problem that we address through preprocessing, which is crucial for the relevance score of a concept in a context. We are given a large set of concepts of size n:

$$C = \{c_1, c_2, ..., c_n\} \tag{2}$$

For each concept c_i, we would like to mine the top m (100 used in practice) relevant context keywords, and their scores:

$$relevantTerms_i = \{(t_{i,1}, s_{i,1}), ..., (t_{i,m}, s_{i,m})\} \quad (3)$$

Here, the score would indicate the importance of the relevant term for the concept, i.e. our confidence about this term. As mentioned above, for concept c_i the relevant keywords $\{t_{i,1}, ..., t_{i,m}\}$ are mined from contexts in which the given concept is presumed to be relevant. In this work, the contexts in which the concept is presumed to be relevant come from three sources: (a) search engine result snippets, (b) Prisma, a tool for query refinement, and (c) related query suggestions. No manual (human) analysis of relevance is performed. Intuitively, the more frequently the term appears in those contexts, the higher this score should be. Similarly, the more important the term is in general, the higher this score should be. Below we describe the resources we used for this task, and the exact scoring metric we employed for each resource in detail.

Before proceeding, we would like to note the relevant terms that we are working with are all stemmed [17], all characters are lower cased and the surrounding punctuation characters are removed.

Search engine snippets: For the task of mining the relevant keywords for a given concept c_i in C, we first propose using search engine snippets. These short text strings are constructed from the result pages by the engine, and they usually provide a good summary of the target page since they help users in making the decision of whether to visit the target page or not. We use Yahoo! Search Engine snippets for our task, and obtain them through the Yahoo! Developer Network [18]. We submit the concept to this API and use the snippets retrieved for the first hundred results. In our setup, we pretend that the returned snippets constitute a single document and then use a bag-of-words model. For each unique term that appears in this document, we compute its *tf*idf* score, where *tf* stands for the term frequency and *idf* stands for the inverse document frequency [6]. Then we choose top *m=100* terms based on this score, and use them as the relevant keywords for concept c_i. So for term $t_{i,j}$, we define score $s_{i,j}$ to be *tf*idf* value. Note that this score can successfully provide the two desired features, which is discussed above, through *tf* and *idf* values.

Prisma query refinement tool: Prisma is a tool which assists users to augment or replace their queries by providing feedback terms [19]. The feedback terms are generated using a pseudo-relevance feedback approach [19], [20] by considering the top 50 documents in a large collection, based on factors such as count and position of the terms in the documents, document rank, occurrence of query terms within the input phrase, etc. When Prisma is queried, it returns top twenty feedback concepts for the submitted query (the reason for this limitation is that the tool is designed for assisting users, and providing more feedback terms is not desirable from users' perspective; however, for our application having a larger feedback set would certainly be beneficial). In order to obtain the set $relevantTerms_i$ for concept c_i, we follow the same approach as we did for snippets: We construct a single document from the concepts returned by Prisma for concept c_i, and compute scores $s_{i,j}$ based on the *tf*idf* values.

Related query suggestions: Finally, we propose to use search engine query suggestions as a resource, which can be obtained from Yahoo! Developer Network [18]. This service provides query suggestions related to the submitted query. In our case, we submit the concept c_i to this service and obtain up to 300 suggestions. We also obtain the query frequencies of the suggestions (i.e how many times this suggestion was submitted to the engine as a query). Note that each unique term seen in these suggestions may appear in multiple suggestions (say, it appears in k suggestions). We consider the query frequencies of these suggestions, and define the score of the term to be $\sum_{i=1}^{k} ln(query\_freq_i) * idf(term)$. We again sort the terms based on their scores and choose top *m=100* terms to be used as the relevant keywords for concept c_i.

C. Discussion

In Section IV-A, we introduced a set of features that would potentially be useful in determining the overall interestingness of a concept. As stated earlier, our goal in this aspect is to be able to identify if a concept would be appealing to a broad user base. However, it is clear that interestingness is a subjective matter and can vary significantly for different individuals and backgrounds. In cases where the application supports a user login, we believe that personalization and collaborative filtering techniques can greatly improve this prediction for individuals by analyzing the history of actions taken. Secondly, the interestingness of a concept can change in time depending on the world's state as news breaks, trends change, etc. To identify this case, new features can be included to the space that can identify spikes or changes in news articles and/or query logs. We plan to address these issues in our future work.

In Section IV-B, we described a new approach to compute the relevancy scores, which essentially mines the relevant keywords for a given concept through preprocessing, and then uses them to estimate the relevance score of that concept in a new context. The approach proves to be quite effective according to our evalations, which will be presented in detail in Section V. However, we would like to first discuss an important goal of the relevancy as mentioned in Section IV-B: Ideally, those concepts that are very general or have low quality should almost never get a high relevance score in any context. Our approach actually addresses this issue implicitly. Assume for now that we are given a concept, and we are using the search engine snippets as the relevant contexts. If the concept is very specific, we observe that some distinctive terms (that have high *idf* scores) occur very often in the snippets and they get high *tf* scores; so these terms end up having quite large final (*tf*idf*) scores. On the other hand, if the concept is very general or have low quality, then we observe that the mined relevant keywords are usually very sparse, and no such clustering occurs, so the mined relevant terms usually end up having small final (*tf*idf*) scores. To illustrate this observation, we applied our approach on a large set of concepts, and for each concept we summed the final scores of its top hundred

Concept	Summation
methicillin resistant staphylococcus aureus	9544.3
motorola razr v3m silver	9118.7
egyptian foreign minister ahmed aboul gheit	9024.9
my favorite	2142.9
the other	1718.0
what is happening	1503.0

TABLE II
CONCEPTS AND THEIR SUMMATION VALUES, WHERE SUMMATION IS THE SUM OF SCORES FOR THE CONCEPT'S TOP HUNDRED RELEVANT KEYWORDS.

relevant keywords. Then we sorted the concepts based on these summations. The upper part of Table II shows three concepts ranked very high with quite large summations, and the lower part shows three concepts ranked very low with much smaller summations. As we can see, low quality concepts get much lower chance of getting identified as relevant in any context since their relevant terms end up having small scores.

Another issue that we would like to discuss is the ambiguous concepts (such as Madonna or Jaguar). If a concept is ambiguous, then the relevant keywords mined might have low final scores, as they would not cluster well globally. However, there would be some good local clusters, depending on the number of senses, and if such clusters can be identified then the scores can be boosted. A number of techniques, inluding ones that are based on latent semantic analysis [21], can potentially be useful for this problem, however this is beyond the scope of this paper.

V. Evaluation and Results

We performed a series of experiments: In Section V-A, we apply a cross-validation method to train and test our approach on news stories that were published and annotated by Contextual Shortcuts. Under this setup, we train a model based on the CTRs, and test the model on remaining unseen documents. For the evaluation, we propose two metrics and provide detailed analysis and comparison to the concept vector score. In Section V-B, we report results from an editorial study. The study is conducted by a team of expert judges, and the judges are asked to rate each entity or concept highlighted in the document in terms of its interestingness, and relevance. Finally, in Section V-C, we provide some highlights on the real world results as the approach is currently being used in production.

A. Cross Validation Approach

1) Data: As discussed in Section III, our approach utilizes randomly sampled news stories that were annotated by Contextual Shortcuts. Our information tracking system allows us to capture how many times these stories were viewed by the readers, and the number of clicks received by each concept that was detected in the stories.

In our evaluation, we used news stories that were published between November 26th, 2007 and January 27th 2008. However, in order to avoid noisy input in our experiments, we ignored the story (1) if the number of sampled views is less than 30, (2) if the story contained only one concept, or (3) if no concept has more than three sampled clicks on the page. After these simple rules, our final data set contained 870 stories which had 6420 concepts detected. These concepts received a total number of 16549 sampled clicks. To avoid the positioning bias inherent in working with user click data (i.e. the first entities in a document may get an unfair share of user attention), we partitioned large documents into windows of size 2500 characters. We made sure that consecutive windows overlap (with 500 characters) so that the neighboring concepts are not separated due to this appoach. Doing this produced a total of 947 windows. For query logs, we considered the most popular 20 million queries submitted to the engine in the week of November 17th – 23rd, 2007.

2) Evaluation Metrics: In our experiments, we use Contextual Shortcuts as a baseline system, which was described in Section II. This baseline system ranks the detected entities and concepts based on their concept vector scores. As another baseline method, we used random ordering. To evaluate the effectiveness of the proposed approach in this paper, we use two metrics. Below, we describe each metric in detail, and provide examples to give the reader some intuition.

Weighted Error Rate: A common metric used for evaluating ranking systems is the "error rate" of preference pairs. This metric or its variations have been used in [22], [23], [24].

Given a ranking (prediction) and the correct ordering, this metric considers all possible preference pairs in the prediction, and counts the number of mistakes made. Then it simply reports the ratio of mistakes to all possible preference pairs:

$$Error\ Rate = \frac{\mid Mistakenly\ Predicted\ Pairs \mid}{\mid All\ Pairs \mid} \tag{4}$$

To obtain an intuition, consider a perfect ranking of four concepts, [A,B,C,D], on a news article, based on the CTR values observed. Assume a system recommends ranking R1=[A,B,D,C], and another system recommends R2=[B,A,C,D], and we would like to evaluate which system performed better. Using this metric, the error rate for both systems are 16.67%, since both systems make a single pairwise mistake out of all six possible pairs.

Note that this ranking metric punishes each pairwise mistake equally. However, in reality some mistakes are more crucial than others, and ideally a pairwise mistake should be punished relative to the preference difference between the two items. In our case, we know the actual CTR of each concept on a given document, and since CTRs usually reflect the strength of the preferences, we propose to punish mistakes according to their CTRs differences. So we define a weighted error rate metric:

$$Weighted\ Error\ Rate = \frac{\sum_{i=1}^{\mid mistakes \mid} CTR\ difference_i}{\sum_{i=1}^{\mid all\ pairs \mid} CTR\ difference_i} \tag{5}$$

So assume the CTR values for the four concepts mentioned above are as follows: [(A, 0.15), (B, 0.05), (C, 0.02), (D, 0.01)]. Using this metric, R1 and R2 predictions would get weighted error rates of 2.22% and 22.22% respectively, since the cost of (B, A) misprediction is much higher than that of (D, C). We will use the weighted error rate metric in our evaluations since we believe it provides a better comparison technique.

NDCG Score: In our evaluation, we also use the normalized discounted cumulative gain measure (NDCG) [25]. NDCG is a valuable metric for those applications that require high precision at top ranks. Such applications can typically leverage and incorporate the position information in the ranked output to provide a better user experience. In our case, following applications can benefit from a high precision ranking of concepts at top ranks: 1) News Alerts, which offers a notification service for key entities and concept in an article, 2) Advertisement Systems, which may leverage the rank positions of the key concepts detected in a document to improve the matching quality. On the other hand, all applications that consume key concepts in documents can potentially improve their overall user satisfaction by working with fewer high precision (quality) concepts at top ranks, and getting the noisy ones removed. Therefore, we decide to use this metric in evaluating the effectiveness of the proposed approach.

NDCG assumes each result has a label (score) assigned; it computes a gain function for each result, and applies a discount function depending on the rank of the result. The NDCG score for a sorted list of k concepts on $document_i$ can be computed as follows:

$$NDCG_{document_i} = N_i \sum_{j=1}^{k} \frac{2^{score(j)} - 1}{log(j+1)} \qquad (6)$$

where N_i is the normalization constant so that a perfect ordering of the concepts in $document_i$ will receive the score of 1.0, $score()$ is the gain function, and $log(j+1)$ is a discounting function to reduce the gain value as the rank increases. In our case, for $concept_j$, we define *score(j) = bucketNo(CTR(j))/100*, where *bucketNo()* simply returns a bucket number between 0 and 1000 considering all the CTR values observed in the system in increasing order. By dividing the bucket number by 100, we basically obtain a judgement score between 0.00 and 10.00.

To obtain an intuition for NDCG, let us again consider the two rankings discussed above: R1=[A,B,D,C] and R2=[B,A,C,D]. For simplicity, let us assume *score(j) = CTR(j)*10*. The ndcg@1 values for R1 and R2 are 1.0 and 0.23 respectively: R1 gets a perfect score as expected, and R2 a much lower score 0.23 ($(2^{0.05} - 1)/(2^{0.15} - 1)$). ndcg@2 values for R1 and R2 are 1.0 and 0.75, and ndcg@3 values for R1 and R2 are 0.98 and 0.76, respectively.

3) Baseline Methods and Methodology: In our experiments, we compare the proposed approach to two baseline methods: random and concept vector score based ordering. When we randomly rank the concepts in our document set, we observe that the error rate is 50.0%, as expected. Next, we use the concept vector scores as has been used in Contextual Shortcuts: With this ranking we observe that the error rate is 30.22%. We would like to note that in the case of ties, we assume a random ordering of concepts.

Technique	**Weighted Error Rate (%)**
Random	50.01
Concept Vector Score	30.22
Feature Set	
All Features	**23.69**
- Query Logs	24.50
- Taxonomy Based	24.47
- Search Results	23.80
- Other	23.78
- Text Based	23.73

TABLE III
WEIGHTED ERROR RATES WITH INTERESTINGNESS FEATURES

In the following experiments, we followed the five-fold cross-validation process: We randomly partitioned our document set into five subsets, used four subsets for training and the remaining subset for testing. We repeated this five times to ensure the learned model is tested on each unseen subset. For the ranking SVM, we test with both linear and the radial basis function kernels with the default parameters, and report the best result we obtain.

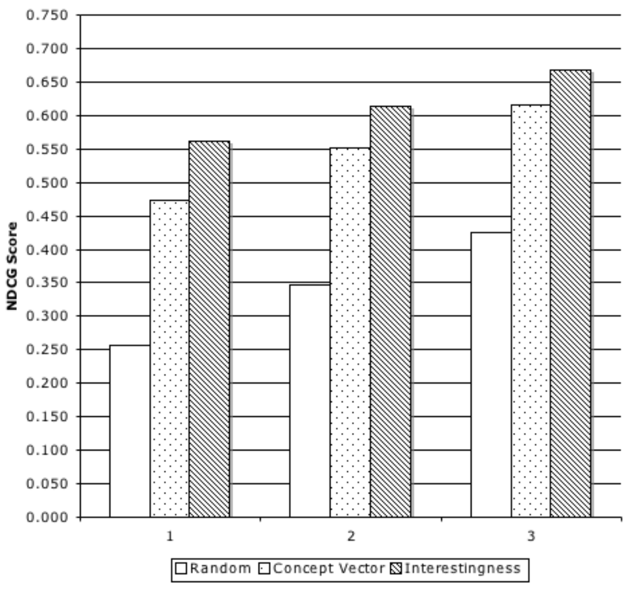

Fig. 1. NDCG scores for top k = {1, 2, 3} results when all interesting features are used in the model

4) Results with Interestingness Features: When we utilize all features shown in Table I in the model, we observe the error rate can be reduced to 23.69%. In order to understand the contributions of the features, we removed each feature set discussed in Section IV-A one at a time. As we can see in Table III, search engine query logs and the taxonomy information

Technique	Weighted Error Rate (%)
Random	50.01
Concept Vector Score	30.22
Best Interestingness Model	23.69
Resource	
Prisma	32.32
Query Suggestions	31.23
Snippets	**24.86**

TABLE IV

WEIGHTED ERROR RATES WITH RELEVANCE SCORE BASED RANKING

Technique	Weighted Error Rate (%)
Random	50.01
Concept Vector Score	30.22
Best Interestingness Model	23.69
Best Relevance	24.86
Interestingness + Relevance	**18.66**

TABLE V

WEIGHTED ERROR RATES WHEN ALL FEATURES ARE USED IN THE MODEL

play important roles, and when each one of them is removed, we see a big increase in the error rate.

Next, we present the NDCG results for positions at k = {1, 2, 3} in Figure 1. The trend seen for the random method actually confirms our initial assumption: Few concepts on a document actually get most of the clicks. The NDCG results are in line with the error rate metric and show that interestingness features can greatly help.

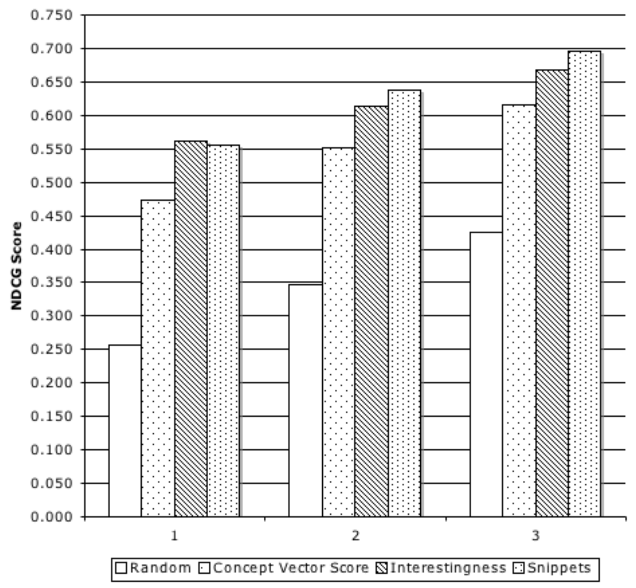

Fig. 2. NDCG scores for top k = {1, 2, 3} results when concepts are ranked based on their relevance scores

5) Results with only Relevance Score: In this subsection, we do not train a model but simply rank the concepts based on their relevance scores. In Table IV, we present error rates for each resource that we used for mining (see Section IV-B). The Prisma and related query suggestions perform worse than the baseline. However, snippets perform significantly better than either of them, and improve on the baseline. We believe snippets provide much better coverage of keywords compared to Prisma and query suggestions, and therefore allow for better clustering. We present the NDCG results in Figure 2.

6) Results with all Features: Finally, we train a model using all interestingness features and the relevance score based on snippets. As presented in Section V-A.5, the relevance score alone is quite useful in reducing the error rate. Therefore, in case of ties, we decided to favor concepts that have higher relevance scores. The error rate results are shown in Table V. As we can see, when all features are employed, the model performs better than any method discussed so far: The error rate is reduced to 18.66%, which is significantly lower than our baseline result of 30.22%. The NDCG results are shown in Figure 3.

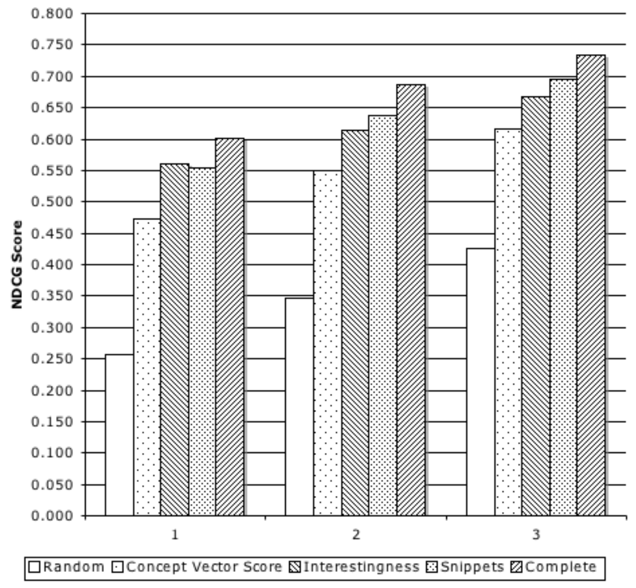

Fig. 3. NDCG scores for top k = {1, 2, 3} results when both interestingness and relevance features are used in the model

B. Editorial Evaluation

We describe the methodology used to evaluate the performance of our approach using two criteria as a measure of quality – the **interestingness** and the **relevance** of the detected entities. We present and analyze the results achieved by the algorithm when run on sample corpus of documents.

1) Methodology: Our standard evaluation methodology consists of a team of expert judges rating the entities detected by the algorithms 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 interestingness and relevance.

Interestingness: Here we 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:

- *Very Interesting or Useful* – It is very likely that the reader will find this entity or concept interesting or useful enough to click on it.
- *Somewhat Interesting or Useful* – This entity or concept is mildly interesting, and maybe compelling to some portion of the general population, but it is not immediately obvious that most users will click on it.
- *Definitely Not Interesting or Useful* – There is no plausible scenario in which the reader will find this entity or concept interesting or useful enough to click on.

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.

Relevance: Here we 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.

Overall Results	**Concept Vector Score**		**Ranking Algorithm**	
Interestingness	News	Answers	News	Answers
Very Interesting	32.6%	35.9%	45.4%	41.6%
Somewhat Interesting	40.9%	35.4%	39.5%	40.3%
Not Interesting	26.4%	28.5%	15.1%	18.1%
Can't Tell	0.1%	0.2%	0.0%	0.0%
Relevance	News	Answers	News	Answers
Very Relevant	53.0%	50.3%	66.3%	61.3%
Somewhat Relevant	29.2%	29.1%	26.3%	28.1%
Not Relevant	17.7%	20.4%	7.4%	10.6%
Can't Tell	0.1%	0.2%	0.0%	0.0%

TABLE VI
INTERESTINGNESS AND RELEVANCE SCORE COMPARISON OF LEARNED RANKING ALGORITHM VS. RANKING BASED ON CONCEPT VECTOR SCORE

2) Test Results: The test corpus consisted of 1200 documents, composed of 800 snippets from Yahoo! Answers Q & A's [9], and 400 full length stories from Yahoo! news [10]). This corpus yielded 4995 entities from News stories and 2826 from Answers snippets. Given all the entities detected, we identified top three and top two entities in News stories and Answers snippets respectively, using the ranking algorithm presented in this paper and the concept vector scores alone. The judgments on these entities are shown in Table VI. As we can see, the ranking algorithm significantly outperforms the concept vector score on both types of content. With the proposed ranking algorithm, the overall average percentage of non-interesting and non-relevant terms across both types of content decreased by 45.1%, from 23.3% to 12.8%. Also noteworthy is the ratio of Very Relevant to Somewhat Relevant scores in News articles, up 38.5% from 1.82 to 2.52.

C. Real World Results

The proposed approach is currently being used in Contextual Shortcuts, and has been in production in the last fifteen weeks. Under this setup, we annotate much fewer entities and concepts in News articles, and make sure they are ranked at top according to techniques described in this paper. As expected and intended, this dramatically reduced the number of annotations we performed. When we compare the outcome to what we observed in the preceding twenty weeks, we see that the number of average weekly views was reduced by 52.5%, and yet the number of average weekly clicks received was down by only 2.0%. This translates to an increase of 100.1% in CTR, which clearly indicates a significant improvement in user experience.

VI. FRAMEWORK

All the techniques described so far for building the interestingness vector and mining relevant keywords for concepts are achieved through preprocessing and are therefore offline procedures. However, the final system, which detects and ranks the concepts in a given document, needs to be quite efficient

[9]http://answer.yahoo.com
[10]http://news.yahoo.com

as this will be done in real time. This sets computational as well as memory limitations.

In this section, we describe our framework and discuss the implementation details. Our user-centric entity detection system supports quite a large number of concepts as it employs search query logs. For simplicity, assume that we would like to support 1 million concepts in our system. The framework is illustrated in Figure 4.

Stemmer: As mentioned earlier, the Ranker works on the stemmed input, so we need to include a Stemmer component. When a document is received, the stemmed version is created first and stored for later usage, as described shortly.

Interestingness: In Section IV-A, we presented 9 features that are used in the interestingness vector. For each concept we have in the system, we first compute the values for these features in the offline process, and employ a normalization that would fit each field to two bytes (this causes a minor decrease in granularity). So the interestingness vectors for 1 million concepts would cost 18MB in memory; with the use of efficient data structures, such as hash tables, the vectors for the detected concepts can be retrieved in constant time.

Relevance: Supporting relevance as described in Section IV-B is trickier, since for each concept in the system, we need to store up to hundred relevant keywords with their scores. In the implementation, the relevant keywords are represented by unique term ids (perfect hashes). So this means that for each concept we actually need to store up to hundred term ids (TIDs) and their scores. In the process of computing the relevance scores, the system uses a global hash table (Global TID Table) which simply maps a given term to its TID (if that term is used by at least one concept). It is clear that the matching of these TIDs with those of relevant terms for the detected concepts can be achieved quite efficiently through a good implementation.

Now, we discuss the memory cost of this approach. Our analysis shows that the total number of unique terms stored in the Global TID Table decreases as we increase the number of concepts in the system. This is certainly as expected since many relevant keywords (TIDs) are shared among related concepts. Therefore, the largest TID value we need to support in the system is not too large and can easily fit into 22 bits. We normalize the scores of the relevant terms to be in the range of 0 and 1023, so that they can fit in 10 bits. So for each concept, we need 400 bytes to store its top 100 (TID, score) pairs, since each pair can be stored in 32 bits, combined. This means that, in addition to the Global TID Table, we need about 400MB for 1 million concepts to store their relevant terms with scores. Note that this cost can be even further reduced through: 1) exploiting the fact that many TIDs are shared by related concepts, 2) using integer compression techniques, such as Golomb Coding [26].

Performance: In order to evaluate the running time performance of the system, we performed the following experiment on a Linux Machine with Dual Core AMD Opteron Processor 275 (1808 MHz) and 1MB cache. The machine had enough space to load all the necessary hash tables into the main memory. In the test, we used 1445 randomly chosen documents with an average size of 2.5KB, and each document contained 6.45 detections on average. The total running time of the stemmer and ranker components were 0.457 sec and 1.519 sec, respectively, which translates to processing rates of 7.9MB/sec and 2.4MB/sec.

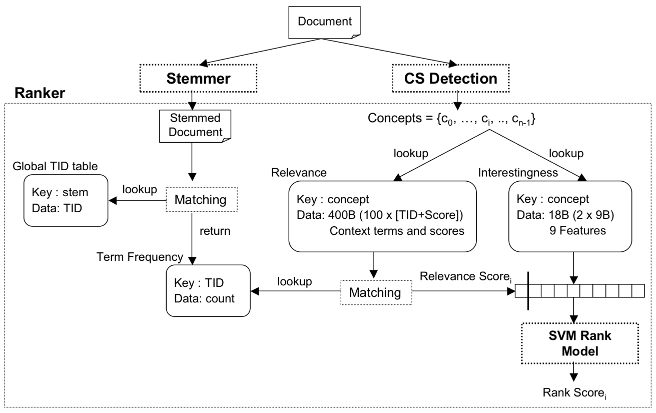

Fig. 4. Framework overview (rounded rectangles represent hash tables)

VII. Related Work

There is a significant amount of work in the field of named entity recognition (NER), where the goal is usually to recognize person, organization and location names in free text. 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 key entities or concepts in the given text, rather than detecting all the entities that fall into predefined categories, our focus is rather different than that of traditional NER systems.

Closely related to our work are the keyphrase extraction algorithms where the goal is to find the most distinctive or representative terms in text [27], [28], [29], [30], [31]. However, the problem setup in these studies is rather different: the keyphrases are extracted from the body of the input document without using a predefined list. In our case, we use a large collection of concepts that are already mined from search engine query logs. So in a given document, our goal is to successfully rank these candidate concepts based on their interestingness and relevance. We argue that the click data collected in Contextual Shortcuts provides a good proxy for this, and we employ machine learning techniques to leverage this implicit user feedback.

The clickthrough data has been successfully used in a number studies to improve web search [32], [22], [33], as well as web page summarization [4]. All these applications rely on the implicit feedback received from the large user base and employ machine learning techniques for improvements.

VIII. Conclusions and Future Work

This work focuses on the problem of identifying the set of highly relevant and interesting key concepts in a piece of text. Our approach to this problem consists of using large scale user feedback to guide our algorithms. Users are presented with candidate entities and concepts in the form of Contextual Shortcuts in each Yahoo! News story that they read. Their subsequent actions (clicks), or lack thereof, provides us with a continuous stream of feedback on whether or not the presented entities are in fact relevant or interesting. Given this large set of "labeled examples", we proceed to build a feature space that includes attributes related to the relevance and interestingness of a concept in a document. The attributes and the labels are then used in a supervised machine learning approach to learn a ranking function over candidate pairs of concepts. We show that this approach substantially improves the performance of the Contextual Shortcuts platform. Finally, we present a framework for how this technique can be implemented in a production system.

In terms of future work, we would like to investigate an online version of this technique. In this scenario, the system would be able to respond to sudden fluctuations in click data, either boosting scores of low scoring concepts that are experiencing high CTRs, or punishing the scores of those experiencing low CTRs. This may allow the system to potentially react intelligently to world events in real time.

Acknowledgment

We are grateful to John Thrall for his helpful comments and suggestions. Furthermore, we would like to thank Farzin Maghoul, Josh Coalson, Omid Rouhani-Kalleh, Olivier Chapelle, our editorial team, and everyone else here at Yahoo! who helped developing and building the Contextual Shortcuts platform.

References

[1] P. Jackson and I. Moulinier, *Natural Language Processing for Online Applications*. John Benjamins Publishing Company, 2002.

[2] 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*, 2006.

[3] A. Anagnostopoulos, A. Broder, E. Gabrilovich, V. Josifovski, and L. Riedel, "Just-in-time contextual advertising," in *Proceedings of the 16th Conference on Information and Knowledge Management (CIKM)*, 2007.

[4] J.-T. Sun, D. Shen, H.-J. Zeng, Q. Yang, Y. Lu, and Z. Chen, "Web-page summarization using clickthrough data," in *Proceedings of the 28th annual international conference on Research and development in information retrieval*, New York, NY, USA, 2005.

[5] V. von Brzeski, U. Irmak, and R. Kraft, "Leveraging context in user-centric entity detection systems." in *CIKM*. ACM, 2007, pp. 691–700.

[6] G. Salton and C. Buckley, "Term weighting approaches in automatic text retrieval," Ithaca, NY, USA, Tech. Rep., 1987.

[7] 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.

[8] S. Kapur and D. Joshi, "Systems and methods for generating concept units from search queries," *United States Patent 7051023*, May 2006.

[9] T. Joachims, "Optimizing search engines using clickthrough data," in *Proceedings of the 8th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002.

[10] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.

[11] A. Z. Broder, "A taxonomy of web search," *SIGIR Forum*, vol. 36, no. 2, pp. 3–10, 2002.

[12] D. Metzler, S. T. Dumais, and C. Meek, "Similarity measures for short segments of text," in *Proceedings of the 29th European Conference on IR Research (ECIR)*, 2007.

[13] M. Najork, "Comparing the effectiveness of hits and salsa," in *Proceedings of the 16th Conference on Information and Knowledge Management (CIKM)*, 2007.

[14] M. Hu, E. Lim, A. Sun, H. Lauw, and B. Vuong, "Measuring article quality in wikipedia: Models and evaluation," in *Proceedings of the 16th Conference on Information and Knowledge Management (CIKM)*, 2007.

[15] P. Chatterjee, D. L. Hoffman, and T. P. Novak, "Modeling the click-stream: Implications for web-based advertising efforts," *Marketing Science*, vol. 22, no. 4, pp. 520–541, 2003.

[16] C. Wang, P. Zhang, R. Choi, and M. D'Eredita, "Understanding consumers attitude toward advertising," in *Proceedings of the 8th Americas Conference on Information Systems*, 2002.

[17] M. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, 1980.

[18] Yahoo! Developer Network, `http://developer.yahoo.com/`.

[19] P. Anick, "Using terminological feedback for web search refinement: a log-based study," in *Proceedings of the 26th Conference on Research and Development in Information Retrieval (SIGIR)*, 2003.

[20] J. Xu and W. B. Croft, "Query expansion using local and global document analysis," in *Proceedings of the 19th Conference on Research and Development in Information Retrieval (SIGIR)*, 1996.

[21] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman, "Indexing by latent semantic analysis," *Journal of the American Society of Information Science*, vol. 41, no. 6, pp. 391–407, 1990.

[22] T. Joachims, "Optimizing search engines using clickthrough data," in *Proceedings of the 8th international conference on knowledge discovery and data mining*, 2002.

[23] J. Xu, Y. Cao, H. Li, and M. Zhao, "Ranking definitions with supervised learning methods," in *Proceedings of the 14th international conference on World Wide Web*, 2005.

[24] R. Herbrich, T. Graepel, and K. Obermayer, "Support vector learning for ordinal regression," in *Proceedings of the 9th International Conference on Artificial Neural Networks*, 1999.

[25] K. Järvelin and J. Kekäläinen, "IR evaluation methods for retrieving highly relevant documents," in *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, 2000.

[26] I. H. Witten, A. Moffat, and T. C. Bell, *Managing Gigabytes: Compressing and Indexing Documents and Images*, 2nd ed. Morgan Kaufmann, 1999.

[27] 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*, 1999, pp. 668–673.

[28] P. Turney, "Learning algorithms for keyphrase extraction," *Information Retrieval*, vol. 2, no. 4, pp. 303–336, 2000.

[29] A. Hulth, "Improved automatic keyword extraction given more linguistic knowledge," in *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processcing*, 2003, pp. 216–223.

[30] 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)*, 2003.

[31] P. D. Turney, "Mining the web for lexical knowledge to improve keyphrase extraction: Learning from labeled and unlabeled data," *CoRR*, vol. cs.LG/0212011, 2002.

[32] E. Agichtein, E. Brill, and S. T. Dumais, "Improving web search ranking by incorporating user behavior information," in *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, 2006.

[33] F. Radlinski and T. Joachims, "Query chains: learning to rank from implicit feedback," in *Proceeding of the 11th ACM SIGKDD international conference on Knowledge discovery in data mining*, 2005.