Leveraging User Feedback for Automated Web Page Inline Linking

Adam Oest
Department of Computer Science
Rochester Institute of Technology

Manjeet Rege
Graduate Programs in Software
University of St. Thomas

Abstract

In this paper we propose and test a system that indexes a large collection of HTML documents (i.e. an entire web site) and automatically generates context-relevant inline text links between pairs of related documents (i.e. web pages). The goal of the system is threefold: to increase user interaction with the site being browsed, to discover relevant keywords for each document, and to effectively cluster the documents into semantically-significant groupings. The quality of the links is improved over time through passive user feedback collection. Our system can be deployed as a web service and has been tested on offline datasets as well as a live web site. A distinctive feature of our system is that it supports datasets that grow or change over time.

1. Introduction

Prior to the Web 2.0 movement of the early 2000's, most web sites used only static HTML pages supporting a minimal level of user interaction. The movement led to such widespread evolution that today's Internet is now dominated by frequently-updated dynamic content presented in a highly-interactive environment. Despite all this change, however, the hyperlink continues to be a core component of any document on the web. Hyperlinks define the flow of web traffic, as they dictate visitor activity by sending users to additional resources that are relevant to the current document. A good linking scheme helps users discover everything that a given web site has to offer, and it keeps them engaged by providing a means to proceed to additional related content; frequent linking to additional related pages is especially important on

modern web sites with large amounts of dynamic content. However, manual linking, while effective, is not always feasible for large or ever-expanding collections of documents.

In this paper, we present a system that analyzes a collection of HTML documents (such as the pages found on a web site) and automatically generates inline links between related pages in the collection. The system works in three phases. First, it indexes each web page and extracts candidate keywords that could potentially be replaced with links on the page. Second, it clusters the pages based on the keywords that have been identified. Finally, it identifies the keywords that are most relevant within the cluster of each page, and converts those keywords into links each pointing to a different but related page within the cluster. The system can be deployed as a web service usable by virtually any type of web site with only a minimal installation process.

Our system is built using a client-server web service architecture. Javascript code communicates with the server side and seamlessly displays automatically-generated inline links on pages on which it is deployed. The server handles page indexing and link generation. What makes our system unique is that the server is able to collect real-time user feedback based on clicks and time spent on landing pages. This feedback is then used to improve the relevance of generated links. Our system is also capable of incorporating a growing number of documents into its auto-linking scheme as they are added, without any type of supervision. The ultimate goal of our system is to provide an unsupervised way to discover keywords relevant to web documents, to increase user interaction on a web site, and to identify web documents that are related to each other through semantic analysis.

2. Related Work

Automated linking as well as web document clustering has already been employed in a variety of ways and for a variety of purposes. In the simplest automatic linking system, an administrator might specify a list of keywords as well as a list of corresponding landing pages. Words or phrases within paragraphs would then be converted into links, with their frequency being determined by some heuristic. Such systems are trivial to implement and are available as plug-ins for popular blogging and content management software, such as Joomla. Advertising companies such as Vibrant Media, Skimlinks, and VigLink use scripts that convert text into inline advertisement links based on advertiser-specified keywords that are discovered on publisher web sites. More-interesting and more-complex approaches have been proposed by researchers. For example, Camacho-Guerrero et al. use clustering and latent semantic analysis to group related web pages and then add links within the clusters [1]. Similarly, Yang and Lee cluster pages in order to generate a list of relevant documents that mimics product recommendations on modern shopping sites [2]. Zeng proposes an automated linking scheme which performed keyword extraction using information retrieval techniques [3]. Heer and Chi, and many others have analyzed web traffic statistics in order to cluster web pages and determine user interests [3].

The work most similar to ours is that of Zeng [3]. In his paper, the author uses the Vector Space Model to discover similar paragraphs across a variety of documents. Similar keywords are then converted into links that point to related paragraphs, though the keywords do not necessarily need to appear in the related paragraphs. The author suggests incorporating user feedback as part of future work. His system is tested on a closed dataset and the results indicate successful keyword extraction that is approximately 64% similar to hand-picked keywords. While Zeng's results are promising, the author does not offer conclusive evidence about the relevance of the automatically-generated links. His most significant contribution to our work is the idea of using not only individual words, but also phrases as possible inline link text.

To the best of our knowledge, no previous automatic linking scheme factors in real-time user feedback in order to dynamically improve its semantic understanding of documents and the links that are generated. Furthermore, we believe our work is the first to support a growing dataset of arbitrary size that can be deployed on live web sites without any supervision. Most prior research, including [2], does not work with inline links. Advertising services often fail to extract the context of a keyword, and thus their inline links suffer from low relevance and sub-optimal conversion rates tied with end user frustration (though in this case, their performance is further constrained by the keywords of advertisers and their budgets). It is clear that this prior work can be built upon in order to deliver relevant inline links which respond to user feedback.

Our long-term goal is to make our system available to the public through a web service. The deployment of our web service would be similar in principle to existing services such as LinkedWithin.com [13], which would allow web publishers to easily adapt our system with minimal programming knowledge and setup time.

3. Background Information

Our system uses methodologies from a number of different fields, including data mining, information retrieval, and natural language processing. Background information required to understand the design of our system is outlined below.

3.1 Web Document Format

While documents served on the web can be of just about any type, most web pages are built using HyperText Markup Language (HTML) code that is displayed inside a web browser application. HTML is a markup language derived from XML, though it does not necessarily conform to the same validity requirements that XML imposes. Each HTML document contains a node tree with a variety of different types of nodes, each of which has a different semantic meaning, and each of which can take on different attributes. Some nodes in the document are used to display content, while others may supply metadata or simply add structure to the document. HTML includes a number of formatting options, and thanks to supporting technologies such as Cascading Style Sheets (CSS), designers can style them with hardly any limitations. Furthermore, HTML documents can be made dynamic through browser-based scripting, which is currently carried out using JavaScript in all modern browsers. Images and other media can of course also be embedded.

A distinctive feature of web documents is their ability to contain hyperlinks, which are blocks of text or images that point to other web documents using a URL. When clicked, the user is taken to the linked document. The HTML <*a*> tag is used for links.

When indexing web pages, our system flattens web documents and extracts keywords from the resulting plain text. Once links have been generated, a client-side script is used to replace a given string with a hyperlink. We refer to such links as inline links, as the clickable text is always a part of the original document.

3.2 Web Page Clustering

Clustering is an unsupervised data mining technique that has seen a great deal over research during the past two decades. A clustering system arranges a set of n documents into k clusters such that the grouping that is output minimizes some overall distance measure. The number of clusters, *k*, is input by the user, while the distance metric used generally depends on the type of clustering algorithm being employed. Traditional clustering algorithms place each document into just one cluster, though "fuzzy" clustering techniques that can assign documents into multiple clusters also exist.

Web page clustering involves taking a sizable group of web pages as an input and assigning them into groups based on similarity, without necessarily drawing any additional conclusions about the content itself. Thus, the main result of clustering is discovery of web pages that are related to each other in some way. This can later help aid the discovery of relevant subject matter or keywords.

In this work, we employ the k-means++ clustering algorithm [5] to cluster web documents. This algorithm seeks to minimize the mean square distance between the documents found in each cluster. The distance calculation is carried out on a term-frequency vector using the euclidean distance formula. The algorithm picks initial centroids "proportionally to the square of the distance between each successive choice" [6], assigns documents to the one closest centroid, and calculates new centroids for each generated cluster. The last two steps are repeated until no change is made to the centroids, at which point the clusters are returned.

```
1) Pick a single centroid randomly from the input data
2) For each data item i, calculate the value of the distance metric d(i) between i and the nearest chosen centroid
3) Add an additional centroid j randomly, such that the probability of selecting j is proportional to the square of d(j)
4) Do steps 2-3 until k centroids are selected
5) Cluster using k-means
```

Algorithm 1: K-means++ pseudocode

K-means++ has been shown to outperform traditional k-means for the purpose of web document clustering [6]. The only difference between k-means++ and the well-known k-means algorithm is in how the starting centroids are chosen. Once initialization is complete, k-means++ proceeds just like k-means.

3.3 Data Preparation

Data cleaning and data preparation is a key step in many types of data mining and information retrieval tasks that deal with web data. Due to the heterogeneous nature of web documents, however, there is no single method for cleaning web data.

The goal of data cleaning is to remove the "noise" in raw data and convert it into a format that is friendly to data mining and analysis. Simplifying the raw dataset is important, as it reduces the complexity of later processing. Data with high dimensionality may be too sparse to be effectively analyzed. However, when choosing the right attributes or making simplifications, great care has to be taken during this step, as over-simplification could also lead to loss of relevant data.

In section 4 we describe the specific data preparation tasks that were carried out on our dataset.

4. System Implementation

Our auto-linking system is designed with scalability in mind, as it is meant to be compatible with web sites that contain hundreds or even thousands of documents. Since the system must be able to work in real time, we have built it as a RESTful web service that can be queried for links using a page URL as the only input. The web service runs on a server accessible via a public URL.

4.1 Client-Side Code

In our case, the server is queried by JavaScript client code that runs inside the user's web browser. The client code is executed as soon as the requested web page is loaded. The webmaster must include this code on every page that he or she wishes to use with the system. While this may seem like a tedious task, it is in fact quite trivial as long as the target web site supports templating or server-side includes. In order to include the client script, it is enough to place a single HTML tag in the *<head>* section of the target page, as shown in Figure 1. The *src* attribute should of course be prefixed with the URL of the actual web service installation.

```
<script type="text/javascript" src="al_main.js">
</script>
```

Figure 1. The HTML code enabling auto-linking on a web page

The web service interface is very simple, as it consists of only two methods that the client can call. The first method, *getLinks*, takes a page URL as input and returns a list of links as well as a hash of the indexed page content (or nothing if the page has not yet been indexed). This will allow the client to display any links that have been generated and determine whether or not the page needs to be re-indexed. The second method, *indexPage*, takes a page URL, content, and title as input, indexes the page on the server, and returns nothing unless an error occurred. Both methods are called using POST via the HTTP protocol, and all output data is encoded using JSON, a lightweight data serialization format. The client code does not interact with the server in any other way except by calling these two methods. In other words, these two methods are sufficient to keep the system's index updated and display the latest links that have been generated on the server side (excluding user feedback collection, to be discussed later). Figure 2 summarizes the complete client-server interaction in our system.

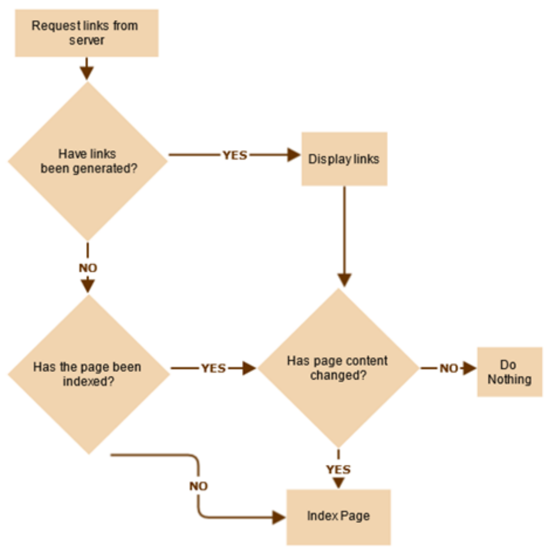

Request links from server
Have links been generated?
YES
Display links
NO
Has the page been indexed?
YES
Has page content changed?
NO
Do Nothing
YES
NO
Index Page

Figure 2. A high-level overview of client-server interaction in our auto-linking system

The client-side code starts by querying the server for links based on the URL of the current page (by calling the *getLinks* method). If the server responds with links, those links are immediately displayed. The client will then compare the hash of the indexed content on the server to the hash of the current page content, and if the two values differ, the client will tell the server to re-index the page using the *indexPage* method. If the server does not respond with a hash of the content, then the client will also call *getLinks* to get the page indexed. If the page is indexed but there are no links to be shown and the hashes match, the client does nothing.

4.2 Server-Side Code

The server-side code is what lies at the heart of our auto-linking system and drives the entire web service. It contains all of the logic needed to clean the input data, extract keywords, generate clusters, and output links. Its key components are shown in Figure 3 and described in the sections that follow.

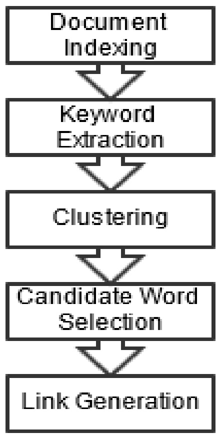


Figure 3. A high-level overview of the server-side component of our auto-linking system

Our implementation of the auto-linking system is written in PHP and uses a MySQL database to store pages, keywords, links, and usage statistics. A relational database data layer was chosen for its performance and scalability.

4.2.1 Document Indexing

Before any further processing can be done, the system must build up an index of all the pages that are to be processed. Whenever the client requests that a page be indexed, the server stores the page's URL, title, and content in the database, and assigns a unique id to the tuple by which the page will be referenced in later stages of analysis. The page content can either be pushed by the client or fetched by the server using a separate web request. If a page's content is empty or below a user-defined length, however, the server will not index it. Furthermore, if a page has the same content as another page but the two URLs differ, the server will merely mark the new URL as an alias of the original page in order to avoid duplicating the content.

The actual content to be indexed is manually identified by the webmaster; all HTML nodes having some predefined class attribute will be considered by our system. While automated data-rich section extraction has seen much research recently [7], we do not employ such techniques for the sake of simplicity and accuracy. Manual content selection ensures that all text data sent to our server is in fact real content, and not supporting navigational or visual markup. Regardless, in a real-world scenario, the end user will most likely not only find it trivial, but also preferable to manually add these class attributes using a page template system.

Clearly, HTML can contain more than just plain text nodes. In order to ensure that the content stored on the server contains no markup, we only keep the text nodes, however.

4.2.2 Keyword Extraction

The goal of the keyword extraction phase is to detect words and phrases that could potentially be relevant to each body of text that is analyzed. This is one of the most important parts of the system, as the quality of the candidate keywords directly influences the performance of clustering and the possible links that can be generated.

We approach this problem by starting with every word in the text and then eliminating words that we can almost certainly say would be poor keywords. In other words, we try to eliminate as many false positives as possible without drawing any conclusions about false positives. In addition to storing individual words, we also generate phrases up to a certain user-defined length (in our experiment, we limit the phrase length to 3 words). As we will show in section 5, multi-word phrases are often just as good if not better than single-word candidate keywords.

The keyword extraction phase starts by splitting the input text using spaces and line brakes as word boundaries. We then iterate through the array of words and flag all sentence boundaries based on the presence of punctuation marks such as periods, commas, exclamation points, colons, parentheses, etc. Once the sentence boundaries are known, we add to our array all n-grams up to the configured phrase length n.

As we cannot make any conclusions about the nature of the content that is being indexed, we cannot justify stripping any special characters from the keywords we generate. Similarly, because the words we generate may later be replaced with links on the web page from which they came from, we cannot alter them in any way. However, we of course do not want duplicate keywords being created simple due to the presence of trailing or leading punctuation marks such as periods or quotation marks. As long as we only alter the prefix or suffix of a word, the root should still be found on the page, so such changes can be made. Therefore, we filter punctuation as follows: if a punctuation mark is found at the beginning or end of a word, then it is stripped. Otherwise, it is kept. For example, the word "people." would be converted to "people", while the phrase "6.8 billion people" would be kept unchanged.

Once we have an array of cleaned terms, we can start to decide which terms to keep and which ones to discard. In order to do so, we employ three information retrieval techniques: stemming [8], part of speech identification [9], and stopword elimination [10]. We chose a general-purpose English stopword list for its universal applicability, and two efficient and accepted algorithms for the other two components.

We start by executing part of speech identification on the cleaned text, such that every term gets tagged with its part of speech. We intend on primarily converting nouns into links, so we only store whether or not each term is a noun. Once we have this information, we then loop through each term in our array and eliminate terms for which algorithm 2 returns a value of false. We only keep multi-word phrases if a algorithm 2 returns a value of true for a majority of the words that it is comprised of, and if it returns true for the first and the last words.

```
Data: A word or phrase
Result: Whether or not to keep the word
if word is shorter than 2 characters then
    return false;
end
if word is longer than preset maximum length then
    return false;
end
if word is a listed stopword then
    return false;
end
if word is not a noun then
    return false;
end
if stem of word is a listed stopword then
    return false;
end
if word contains no letters then
    return false;
else
    return true;
end
```

Algorithm 2: Filtering out stopwords

Once we have determined which words should be kept, we count their overall frequencies in the text. Before returning the final result, however, all multi-word phrases that are substrings of other phrases are removed in order to avoid redundancies. For example, if the phrase "executive board meeting" existed in our processed output, we would eliminate the phrases "executive board" and "board meeting".

The final filtered output is stored in our SQL database and linked to the page to which it corresponds. In order to improve clustering performance, we also perform stemming on the entire filtered output, merge it with title word frequencies (which we multiply by a constant factor to increase their weights), and store this information separately. The original keyword frequencies will later be used for keyword selection during link generation, and the stemmed keyword frequencies will be used for clustering in the following section.

4.2.3 Clustering

The K-means++ clustering algorithm that we use takes an array of term-frequency vectors as its input, and outputs arrays of page identifiers, each of which represents a different cluster. In order to reduce the computational complexity of this clustering operation, we limit the terms included in the term-frequency vector to those that fall above a pre-configured minimum frequency and maximum frequency. We choose to ignore words that occur on fewer than 5% of the indexed pages as well as those that occur on more than half the pages. These values can of course be adjusted.

The only attribute that we use for clustering is term frequency. When user feedback is available, it is used to adjust these frequencies.

The number of clusters is determined semi-automatically based on two configuration parameters. The first is the minimum average number of pages per cluster. If this value were set to 100 given a dataset of 1000 pages, at most 10 clusters would be generated, although each cluster would not necessarily contain 100 pages. The second is the ratio of total keywords over keywords in the term-frequency vector times some user-supplied constant factor. If 1000 pages had 3000 total keywords, 100 were used for clustering, and the user set the factor to 1, then the base number of clusters from this formula 30. Our system takes the minimum of the first and second values (in this case, the minimum of 10 and 30) and aims to generate that many clusters. This approach ensures that the number of clusters remains proportional to the number of relevant keywords while never being below a user-supplied lower limit that scales with the number of pages in the system.

Once enough pages have been indexed and keywords have been generated, our system is ready to begin the clustering process. Clustering can occur whenever the *getLinks* interface method is called. However, it is only carried out if the number of new or updated pages in the system exceeds some user-supplied constant, or if a certain amount of time passes since the last clustering. This ensures that new pages are considered in a timely fashion, and it also ensures that user feedback is regularly incorporated into the system. If the system is due for re-clustering and a clustering operation is initiated, other incoming *getLinks* requests will not initiate re-clustering, and the old links will be returned (if any) until re-clustering is complete.

Every time new clusters are generated, all existing links in the system are deleted.

4.2.4 Keyword and Link Generation

Link generation occurs once per page per clustering operation. If a user requests links but none have been generated, new links are generated on-the-fly as long as clustering has already been carried out. Otherwise, previously-generated links are fetched from the database.

Before we can select the keywords and landing pages for our links, we must obtain a list of keyword frequencies within the cluster of the current page. We query the database for these frequencies, and give multi-word phrases a slight boost compared to single words (a configuration parameter controls this). Then, all words with frequencies below a configured threshold are stripped. This helps encourage that only keywords that are prominent in the current cluster are used as links.

Next, we generate a list of keyword frequencies for the current page itself. We do not remove any words, even if their frequency is 1, but we do boost the relative frequencies of multi-word phrases as we did with the cluster keywords. After both lists are obtained, the latter is sorted by frequency in descending order, and links are generated using algorithm 3.

In order to encourage link diversity, we do not allow for cyclical links, such as links having the same keyword on two different pages that point to each other. Furthermore, while we put faith in the reliability of our clusters, it is very possible that some page keywords will not exist within the current cluster. This could be caused by poor clustering of the current page or the rarity of a given keyword [11]. When this occurs, we choose a page outside of the current cluster as the link's target, though we require that the keyword frequency on the target page is higher than the frequency on the source page by some pre-configured constant factor.

```
Data: Keyword frequencies for the current page and
      cluster
Data: Maximum number of links to generate $m$
Result: Array of links
$a \leftarrow$ empty array;
$p \leftarrow$ current page;
$c \leftarrow$ cluster of $p$;
forall the keywords $k$ on $p$ do
    $pf \leftarrow$ keyword frequency of $k$ on $p$;
    $cf \leftarrow$ keyword frequency of $k$ on $c$;
    if $cf$ greater than 0 and $cf$ greater than $pf$ then
        append to $a$: a new link using keyword $k$ and
        target as the page having the highest frequency of
        $k$ within $c$, other than $p$ or any page already
        having a reciprocal link to $p$ using $k$;
    else
        $f \leftarrow$ maximum page keyword frequency of $k$
        outside of $c$;
        if $f$ greater than ($pf$ + constant factor) then
            append to $a$: a new link using keyword $k$ and
            target as the page having frequency $f$;
        end
    end
    if (length of $a$) greater than or equal to $m$ then
        break;
    end
end
return $a$;
```

Algorithm 3: Filtering out stopwords

Once algorithm 3 returns an array of links, these links are stored in the database for future use; they are also output immediately to the user. Algorithm 3 takes just a few milliseconds to execute, so the end user will likely not be able to tell if links were fetched from the database or if fresh links were generated.

4.2.5 User Feedback Integration

Our approach to user feedback collection is a passive one. Rather than asking users directly or having them answer questions, we transparently analyze how often links are clicked, and how long users spend on each page. This approach to data collection is not invasive and it does not distract the user when interacting with the web content that he or she was originally interested in. Nevertheless, the feedback can still be used to improve the user's overall experience by helping us generate relevant links.

Two types of user feedback data are collected: the number of clicks that each link receives and the time spent on the landing page whenever an automatically-generated link is clicked. In addition to this, we log the number of times that each link is displayed.

Using this data, we can improve not only the relevance of the keywords being selected for links, but also the relevance of the landing pages that each links points to. While all three of our metrics are well-accepted in web analytics, we must still show their effectiveness experimentally. This is done in section 5.

Suppose a given page has 6 links pointing to 6 different pages, but there were 10 candidate keywords in the original keyword frequency array that we found in the previous section. As time progresses, we gather an increasing amount of click data. Once we have a sufficient sample size, we can use the number of link clicks to update the relative frequencies of each keyword for a given page. We do so by normalizing the click frequencies, subtracting the mean from each data term and dividing by the range. The frequency of each corresponding keyword is then multiplied by this ratio and rounded to the nearest integer. This effectively promotes keywords receiving many clicks while demoting keywords that do not attract much attention. Then, when re-clustering occurs, poor keywords might be replaced by other candidate terms.

We take a similar approach for the time users spent on each landing page. When the user leaves a given landing page, the number of seconds that have elapsed since the page was loaded is is transmitted via the JavaScript code to our server. We implement this as a standalone system in order to be able performs analysis in real time. Solutions relying on external data sources, such as server access logs [3] or third-party services such as Google Analytics require manual data cleaning and preparation and thus would be much more difficult to use in an automated fashion. Instead, our system gives us the information we need when we need it, without any additional overhead. Once we have enough data, we again normalize it and penalize or boost the keyword frequencies of the landing pages that are considered in algorithm 3. This can lead to changes in the target URLs for one or more of the keywords on a page.

The number of times that each links is displayed, or our impression data, can be used to analyze which links are being shown but never clicked. Once a link has reached a certain number of impressions without being clicked, it is removed in favor of a different link, or no link at all if no candidate keywords remain.

5. Analysis

We analyzed the performance of our system on two datasets: first, we used a test dataset to test and improve the system during initial development, and later, we deployed the system on a live web site to evaluate the effectiveness of the user feedback.

For our test dataset, we used a corpus of German web news articles compiled by the University of Edinburgh [12]. We chose this dataset because it included articles on a wide range of topics unified by a common theme, much like blog posts or news stories on real web sites that might choose to employ our system. We chose a random sample of 2,000 out of the more than 8,000 articles in this dataset and created simple web pages that included each article as well as our JavaScript client code. This large size would allow us to test the scalability of our linking system without being overwhelmed by an extraordinary number of keywords during manual analysis.

Our system has a number of user-defined configuration options and constants, such as the re-clustering interval, minimum cluster keyword frequency, maximum number of links, and minimum content length. In our experiments we used an upper bound of 50% for the maximum number of pages a keyword can be on, 5% for the minimum number of pages a keyword can be on, and 6% for the minimum absolute frequency (such that a keyword must appear multiple times on at least one page). The re-clustering interval was set to one day, and re-clustering would also be carried out if 50 or more pages changed.

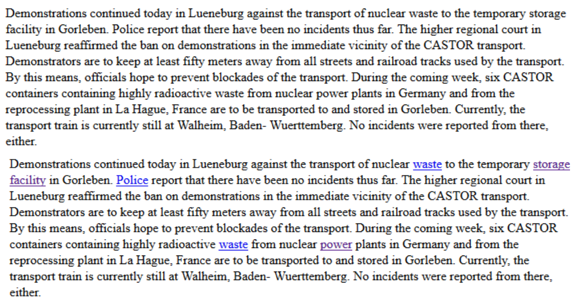
While optimization of all these parameters is outside the scope of this work, with the test dataset we found that having too few or too many keywords could greatly impact the intra-cluster relevance of different pages.

In order to definitively measure our system's performance, we later deployed it on a live web site. The site we chose posts daily news from the photographic industry and our system was deployed on some 1,000 articles.

5.1 Test Dataset

The test dataset showed us that the system worked with reasonable accuracy for both keyword generation and landing page generation. Figures 5 through 7 show an example of the system applied to two articles in this dataset.

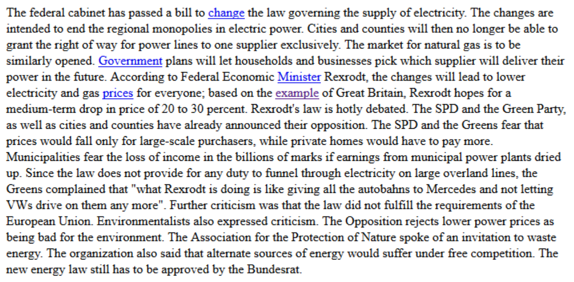
When limited to 7 links per article, our system generated approximately 8,800 links for the 2,000 pages listed. Approximately 75% of those links pointed to landing pages within the same cluster as the source page, while 25% pointed to pages outside of the cluster, as generated by algorithm 3.



Demonstrations continued today in Lueneburg against the transport of nuclear waste to the temporary storage facility in Gorleben. Police report that there have been no incidents thus far. The higher regional court in Lueneburg reaffirmed the ban on demonstrations in the immediate vicinity of the CASTOR transport. Demonstrators are to keep at least fifty meters away from all streets and railroad tracks used by the transport. By this means, officials hope to prevent blockades of the transport. During the coming week, six CASTOR containers containing highly radioactive waste from nuclear power plants in Germany and from the reprocessing plant in La Hague, France are to be transported to and stored in Gorleben. Currently, the transport train is currently still at Walheim, Baden- Wuerttemberg. No incidents were reported from there, either.

Demonstrations continued today in Lueneburg against the transport of nuclear waste to the temporary storage facility in Gorleben. Police report that there have been no incidents thus far. The higher regional court in Lueneburg reaffirmed the ban on demonstrations in the immediate vicinity of the CASTOR transport. Demonstrators are to keep at least fifty meters away from all streets and railroad tracks used by the transport. By this means, officials hope to prevent blockades of the transport. During the coming week, six CASTOR containers containing highly radioactive waste from nuclear power plants in Germany and from the reprocessing plant in La Hague, France are to be transported to and stored in Gorleben. Currently, the transport train is currently still at Walheim, Baden- Wuerttemberg. No incidents were reported from there, either.

Fig. 5. An unaltered news article viewed in a web browser beside the same article with automatically-generated links



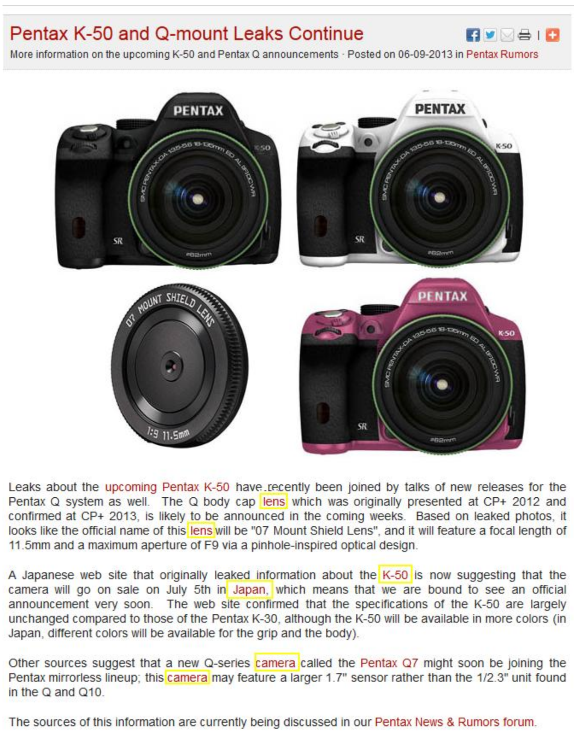
The federal cabinet has passed a bill to change the law governing the supply of electricity. The changes are intended to end the regional monopolies in electric power. Cities and counties will then no longer be able to grant the right of way for power lines to one supplier exclusively. The market for natural gas is to be similarly opened. Government plans will let households and businesses pick which supplier will deliver their power in the future. According to Federal Economic Minister Rexrodt, the changes will lead to lower electricity and gas prices for everyone; based on the example of Great Britain, Rexrodt hopes for a medium-term drop in price of 20 to 30 percent. Rexrodt's law is hotly debated. The SPD and the Green Party, as well as cities and counties have already announced their opposition. The SPD and the Greens fear that prices would fall only for large-scale purchasers, while private homes would have to pay more. Municipalities fear the loss of income in the billions of marks if earnings from municipal power plants dried up. Since the law does not provide for any duty to funnel through electricity on large overland lines, the Greens complained that "what Rexrodt is doing is like giving all the autobahns to Mercedes and not letting VWs drive on them any more". Further criticism was that the law did not fulfill the requirements of the European Union. Environmentalists also expressed criticism. The Opposition rejects lower power prices as being bad for the environment. The Association for the Protection of Nature spoke of an invitation to waste energy. The organization also said that alternate sources of energy would suffer under free competition. The new energy law still has to be approved by the Bundesrat.

Figure 6. The landing page when clicking on "power"

5.2 Live Dataset

Over the course of two weeks, our system indexed a total of 648 news article on a large web site (www.pentaxforums.com), grouped them into 18 clusters, and generated a total of 3172 links. 2336 of those links pointed to pages within the same cluster as the source page, while the remainder pointed to other pages. Those links received some 2400 impressions and 150 clicks during this time. Figure 8 shows an example of a web page containing automatically-generated links from our system. All non-organic links are highlighted for illustrative purposes.

In general, the k-means++ clustering performance was excellent. We analyzed intra-cluster page similarity by first identifying the topic of the majority of pages within the cluster, then manually counting how many of the pages in that cluster were relevant to this topic. Of the 648 pages clustered, we found 552 to be relevant within the clusters to which we were assigned, meaning that the intra-cluster similarity was a high 85.2%. An average of just 5 pages per cluster were incorrectly categorized.



Pentax K-50 and Q-mount Leaks Continue

More information on the upcoming K-50 and Pentax Q announcements · Posted on 06-09-2013 in Pentax Rumors

Leaks about the upcoming Pentax K-50 have recently been joined by talks of new releases for the Pentax Q system as well. The Q body cap lens which was originally presented at CP+ 2012 and confirmed at CP+ 2013, is likely to be announced in the coming weeks. Based on leaked photos, it looks like the official name of this lens will be "07 Mount Shield Lens", and it will feature a focal length of 11.5mm and a maximum aperture of F9 via a pinhole-inspired optical design.

A Japanese web site that originally leaked information about the K-50 is now suggesting that the camera will go on sale on July 5th in Japan, which means that we are bound to see an official announcement very soon. The web site confirmed that the specifications of the K-50 are largely unchanged compared to those of the Pentax K-30, although the K-50 will be available in more colors (in Japan, different colors will be available for the grip and the body).

Other sources suggest that a new Q-series camera called the Pentax Q7 might soon be joining the Pentax mirrorless lineup; this camera may feature a larger 1.7" sensor rather than the 1/2.3" unit found in the Q and Q10.

The sources of this information are currently being discussed in our Pentax News & Rumors forum.

Figure 7. Automatically-generated links on a live web page

The two best-performing clusters had perfect similarity; one cluster contained only news posts about interviews, while another cluster contained only news posts about software and firmware updates. The worst-performing clusters were very small in size (3-4 pages) and had no semantic intra-cluster similarity. The remaining clusters had similarity values very close to the overall mean. In many cases, the generated clusters closely resembled the editor-assigned categories for the pages even though our system had no knowledge of these categories. Table 1 shows the clustering effectiveness.

5.2.1 Impact of User Feedback

The relevance of each link keyword to the page it was on proved not to be as high as the overall keyword relevance. This is due to the constraint that each link must point to a landing page containing the same keyword. If no appropriate landing page exists, then the keyword cannot be used as in a link on the source page. Nevertheless, after analyzing the links that users elected to click on over the course of the test period, our system was able to increase the overall relevance of the keywords converted into links on each page. This increase is shown in table 2.

We believe that over time, as the number of user feedback data points increases, more optimal results can be achieved. Long-term data collection will be necessary to confirm this, and we plan to do so in the future.

Table 1: Intra-cluster content/similarity on live web site

Cluster & Subject	**Pages**	**Similar Pages**	**% Similarity**
1. Favorite User Photos	16	16	100
2. Pentax K-50/500 Cameras	4	4	100
3. (Varied)	3	0	0
4. Third-party SLR lenses	42	37	88
5. (Varied)	3	0	0
6. Pentax DSLRs	10	9	90
7. Software and firmware	17	17	100
8. Photo contests	87	75	86
9. Member profiles	25	24	96
10. Pentax X-5 camera	1	1	100
11. Camera reviews/news	209	168	80
12. Lens news	10	8	80
13. Photographic articles	47	40	85
14. Journeys into photography	2	2	100
15. Website news	44	32	72
16. Interviews	9	9	100
17. Favorite photo equipment	58	56	97
18. Pentax K-5 camera	61	54	89
(Overall)	***648***	***552***	***85***

We were only able to collect some 20 data points about landing page relevance; this sample proved to be much too small to even have an impact on the landing pages our system chose. We discovered that the mechanism we used to transmit this feedback to our server is blocked by many web browsers due to abuse by malicious coders. In the future, we plan on addressing this issue by transmitting usage information to the server on a set interval such that information is always transmitted before a user leaves the page.

Table 2: Effect of user feedback on relevance of link keywords to source page

	Before Feedback	**After Feedback**
# of links	290	290
Relevant link keywords	201	219
% Relevance	69	76

Despite this, we found that because user click feedback had a positive side-effect of improving the overall relevance of the link keywords on the source page, the relevance of the landing pages did go up as well, as evidenced by table 3.

Table 3: Effect of user feedback on relevance of source page to destination page

	Before Feedback	**After Feedback**
# of links	290	290
Relevant landing pages	167	182
% Relevance	58	63

While table 2 shows that the relevance of the source keywords went up overall, on some individual pages, user feedback actually worsened the results. This is because the keywords originally chosen were already being used in organic links, and thus no clicks were recorded for them in our system. In the future, we will need to detect the presence of organic links in order to avoid generating excessive auto-links.

5.3 Runtime Performance

We deployed our system on a dedicated dual-Xeon web server, a common hardware configuration for medium-to-large web sites. Trivial operations such as retrieving links or saving data in the database take but a few milliseconds to execute thanks to the efficiency of a well-indexed MySQL database. The actual generation of links is also a very quick process, and even with our test dataset with 2,000 pages, it took just fractions of a second for links to be generated for any one page.

Table 4: Clustering performance with 80 keywords in feature vector

# of Pages	**Clustering Time (s)**
25	6
250	45
500	300

Table 5: Clustering performance with 130 keywords in feature vector

# of Pages	**Clustering Time (s)**
25	7
250	50
500	492

Performance-wise, the main bottleneck in our system is therefore only the actual K-means clustering of pages. The k-means clustering algorithm runs in polynomial time and requires more time whenever the number of pages, number of attributes, or number of clusters increases. Thus, the more pages we have in our system, the longer the clustering operation will take. Because the clustering is not carried out very frequently (once per day would be a reasonable maximum), though, this is not a major concern. In order to minimize the load on our web server, however, it is still desirable to attempt to execute the clustering as efficiently as possible. It is for this reason that filtering out keywords that are too frequent or too rare is very important, as is proper identification of stopwords. Tables 4 and 5 show the clustering performance for two different feature vector sizes.

6. Conclusion and Future Work

In this paper we have presented a system for automated keyword extraction and inter-page link generation for collections of web pages. When deployed on a live web site, our system indexes the textual content of all pages to be considered, filters out unwanted words to generate candidate keywords, clusters the pages, and finally selects keywords to be converted into links, as well as landing pages for each keyword.

Our work is a proof of concept showing that it is not necessary to manually select keywords and landing pages in order to automatically generate web links to engage web site visitors. We've shown that such a system can collect feedback in real-time and seamlessly integrate it into a machine learning algorithm, and that the overall performance of the system is more than sufficient for real-world deployment. Finally, we've presented a non-invasive and anonymous yet effective methodology for user feedback collection in web environments.

User feedback proved to slightly improve both the overall relevance of the link keywords to each source page, and the relevance of the landing pages to the source page. More data collection will be necessary to definitively show the effectiveness of this feedback. During the two-week deployment of the system, no users of the live web site complained about the automatically-generated links.

With respect to our implementation itself, we have one outstanding issue that has not yet been addressed. If a piece of content is found on a standalone page as well as on an index page with other pieces of content, both pages will be indexed separately and the duplicate content may lead to pointless links as well as a degradation of clustering performance. More careful analysis of page content during indexing may therefore become necessary.

In the future, we plan to research ways to improve our system so that it can lead to a semantic understanding of the content that is indexed. The data that we have gathered could for instance be used to not only generate a list of keywords that are relevant to a page, but also identify synonyms and discover related terms that are not necessarily present on the page itself. This might also lead to the selection of keywords that link to pages without instances of that particular keyword. The candidate keyword selection process could potentially also be improved by factoring in keyword length as part of the keyword's relevance. Similarly, rather than stripping all HTML tags, we could make use of tags such as bold, italics, and images to identify keywords that are likely to be more relevant than others. As mentioned earlier, current organic links on each page should also be used to denote which keywords are of high relevance of a page, and to ensure that duplicate links are not generated by our system. With respect to our data collection, user engagements could be scrutinized more carefully to not only report the amount of time users spend on a page, but also the amount of mouse movement, highlighting, and scrolling that took place. Finally, cyclical links, which we currently avoid, could also be used to our advantage. Such links could strategically be placed in order to detect, over time (as click data is collected), if there exists directionality between the relationships of pairs of pages general or more specific than others on a related subject. We hope that this future work will help our system evolve into one that can over time reveal the semantics of the documents it indexes with high precision.

References

[1] Jose A. Camacho-Guerrero, Alex A. Carvalho, Maria G. C. Pimentel, Ethan V. Munson, and Alessandra A. Macedo. 2007. Clustering as an approach to support the automatic definition of semantic hyperlinks. In Proceedings of the eighteenth conference on Hypertext and hypermedia (HT ’07). ACM, New York, NY, USA, 81-84.

[2] Hsin-Chang Yang, Chung-Hong Lee, “A text mining approach for automatic construction of hypertexts”, Expert Systems with Applications,
Volume 29, Issue 4, November 2005, Pages 723-734.

[3] Jihong Zeng. “An Automatic Hyperlink Generation Approach for Content Management.” The Business Review, Cambridge 8.2 (2007): 243-50. ProQuest. Web. 31 Mar. 2013.

[4] Jeffrey Heer and Ed H. Chi. “Identification of Web User Traffic Composition using Multi-Modal Clustering and Information Scent.” Xerox Palo
Alto Research Center. 2001.

[5] D. Arthur and S. Vassilvitskii. “K-means++: the advantages of careful seeding”. Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, 2007, Pages 1027-1035.

[6] Bouras, C.; Tsogkas, V., “Assigning Web News to Clusters,” Internet and Web Applications and Services (ICIW), 2010 Fifth International
Conference on, pp.1,6, 9-15 May 2010

[7] Jiying Wang; Lochovsky, F.H., “Data-rich section extraction from HTML pages,” Web Information Systems Engineering, 2002. WISE 2002.Proceedings of the Third International Conference on , pp.313,322, 12-14 Dec. 2002

[8] Willett, P. “The Porter stemming algorithm: then and now”. Program: electronic library and information systems, 40 (3), 2006. Pages 219-223.

[9] Eric Brill. 1995. “Transformation-based error-driven learning and natural language processing: a case study in part-of-speech tagging”. Comput. Linguist. 21, 4 (December 1995), 543-565.

[10] Christopher Fox. 1989. “A stop list for general text”. SIGIR Forum 24, 1-2 (September 1989),19-21.

[11] Adam, G.; Bouras, C.; Poulopoulos, V., ”CUTER: An Efficient Useful Text Extraction Mechanism,” Advanced Information Networking and Applications Workshops, 2009. WAINA ’09. International Conference on, pp.703,708, 26-29 May 2009

[12] German-English Parallel Corpus "de-news", Daily News1996-2000, *http://homepages.inf. ed.ac.uk/pkoehn/publications/de-news/. Web. 18 April 2013.*

[13] Linked Within. http://www.linkedwithin.com. Web. 20 April 2013.