2007-08-24

Using Fuzzy-Word Correlation Factors to Compute Document Similarity Based on Phrase Matching

Jun won Lee
p3man4@gmail.com

Yiu-Kai D. Ng
ng@cs.byu.edu

Original Publication Citation

Jun Won Lee and Yiu-Kai Ng. "Using Fuzzy-Word Correlation Factors to Compute Document Similarity Based on Phrase Matching." In Proceedings of the 4th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD'7), pp. 186-191, August 24-27, 27, Haikou, China.

Using Fuzzy-Word Correlation Factors to Compute Document Similarity Based on Phrase Matching

Jun won Lee
Computer Science Department
Brigham Young University
Provo, Utah 84602, U.S.A
p3man4@gmail.com

Yiu-Kai Ng
Computer Science Department
Brigham Young University
Provo, Utah 84602, U.S.A
ng@cs.byu.edu

Abstract

One of the Web information Retrieval (IR) problems these days is to identify redundant information that exist in (replicated) Web documents. These documents can easily be found in several forms, such as documents in different versions, small documents combined with others to form a larger document, etc. As the Web is becoming more and more popular, the number of documents on the Web is increasing on a daily basis, and filtering redundant ones among this huge number of documents becomes a more difficult and an urgent task. As one of the solutions to this problem, we present a new method that identifies similar documents based on phrase matching using the fuzzy-word correlation factors among words in phrases. Since phrases can be treated as sequences of words in a sentence in any document, we consider the correlation factors of different words in any two phrases of two different documents to determine the degree of similarity of the phrases, which in turns can determine the similarity of the documents based on the number of matched phrases/sentences in the documents. Experimental results show that our phrase-matching approach is accurate and outperforms the word-based similarity matching approach.

1 Introduction

Due to the huge number of documents on the Web these days, the storage and retrieval of Web documents has become a difficult problem to handle by information retrieval (IR) systems. To make the matter worse, the accumulation of similar documents degrades the information retrieval efficiency and increases the end user's dissatisfaction caused by flooding of replicated/similar documents that share identical or very similar information. As one of main design goals in IR is to retrieve as many relevant documents as possible and minimize the number of irrelevant documents with respect to the user's need, retrieving replicated information is undesirable, since retrieved relevant documents should be informative, non-replicated (in terms of their contents) documents. As semantically duplicated documents can easily be found on the Web these days[1], such as news articles and different versions of a document that are modified/updated on a regular basis, detecting similar/duplicated documents in order to minimize redundant ones and thus speed up the retrieval process is an urgent task for IR system designers.

The fuzzy set IR model [3, 6, 10] has been developed, which measures the similarity between related words in a document and a user's query. The fuzzy set IR model has been proven [10] to be a very effective model for detecting similar documents, since it can detect related semantic content. The fuzzy set IR model in [10] uses word connection factors, which represent the degrees of similarities between words in any two documents, to determine the degree of similarity of the documents. Since the standard fuzzy set IR model is based on single-word similarity matching, the orders of words appeared in any two documents are not considered at all. However, it is possible that two documents (or sentences) which have exactly the same words but in different order deliver semantically different contexts. For example, consider the sentences "They jog for thirty minutes and walk for an hour every Sunday," and "They jog for an hour and walk for thirty minutes every Sunday." If we ignore the order of words in the two sentences and simply consider the single-word matching, these sentences are considered identical, even though they are semantically different.

In this paper, we introduce a phrase-based similarity detection approach using fuzzy-word correlation factors to solve the inaccurate problem introduced by single-word matching on detecting similar documents. The proposed approach analyzes the content of documents by computing the degree of similarity among different phrases in sentences of

[1] According to [1], a third of the Web documents are near-duplicates of the remaining two-third.

Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007)
0-7695-2874-0/07 $25.00 © 2007 IEEE

different documents. Our approach begins with eliminating stopwords and stemming words[2]. This process is crucial because it reduces the total number of phrases in a document D to be considered and thus minimizes the space and time complexity in document comparisons. Hereafter, we construct the *Document Index Graph* (DIG) G of D to capture different phrases in D. Each node in G represents a non-stop, stemmed word[3], and two nodes in G are connected if they are consecutive words in D. Using this structure, every possible phrase in any pair of sentences of two documents can be discovered by tracing all nodes in their corresponding DIGs when the latter are overlaid. The correlation factors of phrases in different sentences from two documents yield the degree of similarity of the documents using the correlation factors of words in phrases.

We proceed to present our phrase-matching approach as follows. In Section 2, we introduce our phrase-matching approach in computing the degree of similarity of any two Web documents. In Section 3, we present the experimental results on a phrase query which verify the accuracy of our phrase-matching approach in detecting similar documents. In Section 4, we give a concluding remark.

2 Our phrase-matching approach

Phrases have been widely used in document retrieval [5]. In this paper, we treat a *phrase* as a consecutive sequence of words as in [2]. For instance, given the three words A, B, and C as a sequence of words, the corresponding phrases are A, B, C, AB, BC, and ABC. In general, for any n ($n \geq 1$) distinct words in a sequence, the number of possible phrases is $\frac{n\times(n+1)}{2}$. We modify the DIG in [2] to simplify the DIG of a Web document. The DIG proposed in [2] gives different weight to each keyword based on its tag location in an HTML structure (e.g., if a word appears in a <title> or <section> tag, the highest weight is given); however, we do not consider this weighting scheme, since we do not focus on HTML structure, even though we can detect similar HTML documents as well. Besides exact matching as in [2], we match phrases according to the correlation factors of corresponding words in the phrases, which we believe is a more accurate approach in performing phrase matching than the simple exact matching.

2.1 The word-to-word correlation factors

Prior to applying our phrase-matching approach, we first construct the word-correlation matrix by (i) removing all the stopwords and replace non-stop words in different sample Web documents by their stems using the Portor's algorithm [7] and (ii) computing the word-to-word correlation factors using the remaining words. The set of sample Web documents was the Oct 20, 2005 Wikipedia database dump [8]. The database dump contains 880,388 documents, 74,663,883 sentences, 46,861,448,677 words, and 2,389,984,085,254 characters for a total size of 4.6 GB. Of course the most ideal set of documents to compute the word-to-word correlation factors would be the set of all possible Web documents. However, this set is impractical and impossible to obtain as it is not feasible to retrieve all documents on the Web and the size of such a set would be extremely huge. The best alternative is the set of documents that is *representative* of such a set. If a set of documents includes too many documents on a given topic, then the set is not representative, since documents on other topics are either under represented or not represented at all. The size and nature of Wikipedia, a free on-line encyclopedia, ensures that a variety of topics are covered. For example, Wikipedia covers topics from "apples" to "Yahweh," and from "cooking" to "zebras." One might claim that the set of Wikipedia documents was retrieved from one source and thus is biased. Our counter argument is that it is not bias because the downloaded Wikipedia documents were authored by more than 850,000 people [9]. The diversity of the authorships of these documents leads to a representative group of documents with different writing styles and a diversity of subject areas. No one person's style or preferences have defined the set of documents. As a result, the set of Wikipedia documents is an effective representative set of documents that is appropriate for computing the general correlation factors among words. Furthermore, the word-to-word correlation factors are computed only *once*, which is an off-line preprocessing step and can be used hereafter without affecting the real-time computational complexity for detecting any similar (new) Web documents on-line.

The word-to-word correlation factor of any two words w_1 and w_2 is computed according to the *frequency of occurrences* and *distances* between w_1 and w_2 in any Wikipedia document to measure their degrees of similarity. An unnormalized correlation factor $P_{i,j}$ of words w_i and w_j is defined as

$$P_{i,j} = \sum_{k_i \in D} \sum_{k_j \in D} \frac{1}{r(k_i, k_j)} \quad (1)$$

where D is the set of Wikipedia documents, k_i (k_j, respectively) is an occurrence of (a stemmed version of) word w_i (w_j, respectively) in any document in D, and $r(k_i, k_j)$ is the number of words between (i.e., separating) k_i and k_j plus 1, which insures that the distance between k_i and k_j is always non-zero. $r(k_i, k_j) = 1/\infty = 0$, if k_i and k_j are in different documents. Thus, words that co-occur closer together yield

[2] A *stopword* is any word that is used very frequently (e.g., "him," "with," "a," etc.) and is typically not useful for analyzing informational content, whereas a *stemmed word* is the lexical root of other words (e.g., "driven" and "drove" can both be reduced to the stemmed word, "drive").

[3] From now on, unless stated otherwise, whenever we use the term "word (phrase, respectively)" we really mean "non-stop, stemmed word (phrase without stopwords nor non-stemmed words, respectively)."

0-7695-2874-0/07 $25.00

higher correlation values than words that co-occur farther apart, and words in separate documents do not affect their correlation values at all, since their distance values are zeros. The normalized correlation factor $C_{i,j}$ is given by

$$C_{i,j} = P_{i,j}/(N_i \times N_j) \quad (2)$$

where $P_{i,j}$ is the un-normalized correlation factor as defined in Equation 1, and N_i (N_j, respectively) is the number of times k_i (k_j, respectively) appeared in the set of Wikipedia documents.

2.2 Building document index graphs

Prior to computing the degree of similarity between any two given documents D_1 and D_2 using the word-to-word correlation factors constructed in Section 2.1, we represent D_1 and D_2 by their DIGs. The DIG in [2] is a directed graph, which represents the order of words and its relationships in sentence phrases in the corresponding document. Each *node* in a DIG corresponds to a single indexed term (i.e., word). Two nodes are connected by an edge if their corresponding words appear consecutively in a document. In a DIG [2], there is a *document table* and an *edge table* for each *node*. The edge table of each node contains the *location* of its corresponding word in a sentence of a document and is being pointed to by each corresponding document entry in the document table of the node, whereas for each entry (i.e., document) D in the document table, it contains the *term frequency* of the corresponding word in sentences of D and the pointer to the edge table. Our proposed DIG is similar to the DIG in [2]; however, in our DIG model, document and edge tables of a node N are combined into the document table of N, which further simplies the DIG structure as proposed in [2]. More importantly, as mentioned earlier, we consider not only *exact* matching of words/phrases represented in the node structures of a DIG, but also *inexact* matching (i.e., degrees of similarity) of words/phrases, which is the uniqueness of our phrase-matching approach.

Example 1 In the document table of node A in Figure 1, two E are contained in the next word list of document Doc 1, and an E is in the next word list of document Doc 2, since E is preceded by A in both Doc 1 and Doc 2. The first A in Doc 1 is the 1^{st} word followed by E in the first sentence, which is represented as (1, 1, E), whereas the second A in the same document occurs as the sixth word in the first sentence, which is represented as (1, 6, E). In addition, since A is in the second sentence in Doc 1, which is not followed by any word, a NULL is assigned as the 3^{th} component of an NEXT list to indicate that A is the last word of the (second) sentence. As for node B, the NEXT field in Doc 2 is empty, since Doc 2 does not have any B, and thus its term frequency is zero. The document tables of other nodes are created accordingly. □

2.3 Degrees of resemblance among different documents

In the standard fuzzy set IR model, term (word-to-word) correlation factors are used for document comparisons. In our phrase-matching approach, word-to-word correlation factors are used to compute *phrase correlation factors* for phrases of the same length. At a high level, a sentence can be treated as a group of phrases arranged in a particular order. The following phrase correlation factor, denoted $c(phrase_i, phrase_j)$, defines the extent of similarity between any two phrases $phrase_i$ and $phrase_j$ of length m ($m \geq 1$), where $phrase_i$ is in a sentence of one document and $phrase_j$ is in a sentence of another document. (As mentioned earlier, for each sentence S with n words, there are $\frac{n\times(n+1)}{2}$ phrases of various lengths, i.e., from 1 to n, that can be created from S.)

$$c(phrase_i, phrase_j) = \frac{\sum_{k=1}^{m} C_{phrase_i[k],phrase_j[k]}}{m} \quad (3)$$

where $C_{phrase_i[k],phrase_j[k]}$ is the word-to-word correlation factor (as defined in Equation 2) of the k^{th} ($1 \leq k \leq m$) word in $phrase_i$ and $phrase_j$.

Example 2 We can use the DIG of each document to look for phrases of length k ($k \geq 1$) in the document. Let's consider Doc 1 and Doc 2 in Figure 1 and phrases of lengths 3 and 2. After A is detected in the first sentence of Doc 1, we examine the word after A in node A by considering the NEXT field of node A. Since E is the next word, we proceed to look for the successive word of E in the NEXT field of node E, which is B. Hence, by using one traversal, we determine the existence of a phrase of length three. If we look for a phrase of length two, then no traversal is needed. Let's consider Doc 1 and Doc 2 in Figure 1 again. Beginning with node A, the first phrase of length two we encounter in Doc 1 is AE. Likewise, we find EB in Doc 1. Once the location parameter reaches the $(n-k+1)^{th}$ position, where n ($n \geq k \geq 1$) is the number of words in a sentence, we increase the sentence parameter by one and reset location parameter to 1, i.e., we are at the last phrase in certain sentence and must switch to the next sentence to search for phrases in the next sentence. □

Using the phrase correlation factors computed by Equation 3, we obtain the degree of similarity of two sentences, which measures the extent to which the sentences match. To be exact, when we compute the degree of similarity between any two sentences, S_1 and S_2, by matching phrases of length k ($1 \leq k \leq MIN(|S_1|, |S_2|)$) in S_1 and S_2, where $|S_1|$ ($|S_2|$, respectively) denotes the number of words in $|S_1|$ ($|S_2|$, respectively), we first compute the *phrase-sentence correlation factor*, $\mu_{phrase_k,2}$, which is the correlation factor (i.e., the degree of similarity) of each $phrase_k$ in S_1

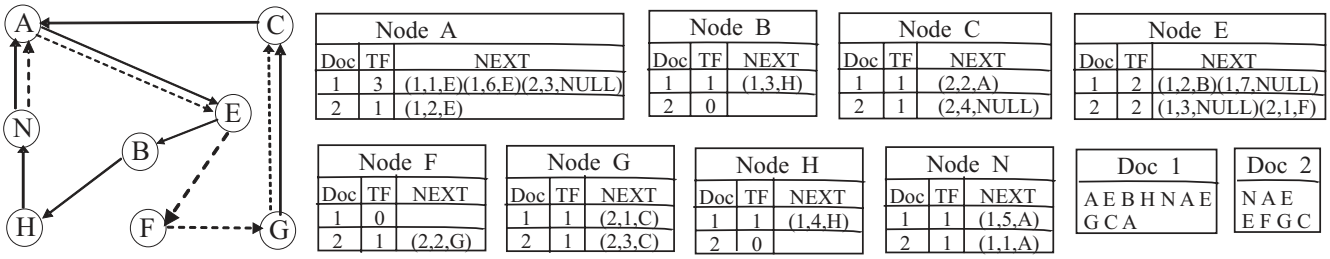

Figure 1. An illustrative example of our DIG and the document tables of nodes (i.e., words)

with respect to all the phrases of the same length in S_2:

$$\mu_{phrase_k,2} = 1 - \sum_{phrase_j \in S_2} (1 - c_{phrase_k,phrase_j}) \quad (4)$$

where $c_{phrase_k,phrase_j}$ (as defined in Equation 3) is the phrase correlation factor between $phrase_k$ in S_1 and any phrase $phrase_j$ of the same length in S_2. After we have calculated the *correlation factor* of each phrase of length k ($1 \leq k \leq MIN(|S_1|, |S_2|)$) in S_1 with respect to each of the phrases of length k in S_2, we obtain the *phrase-sentence correlation factor* of S_1 with respect to S_2. Hereafter, we define the *degree of similarity* of S_1 with respect to S_2.

Note that "long"-phrase matching should be given preferences over "short"-phrase matching, since long-phrase matching reflects higher order of lengthy words than short-phrase matching, and the correct matching based on long sequences of words reflects higher similarity between the two corresponding sentences than short sequences of words in the same sentences. Therefore, we devise a model such that matched long phrases are giving heavier weight than matched short phrases, and the weights are proportional to the length of the matched phrases. To prevent longer phrases from being given excessive weight, each weight value is normalized by dividing the length of its corresponding phrase by the sum of all the lengths of other phrases that appear in the same sentence. Thus, the degree of similarity of S_1 with respect to S_2 is defined as

$$Sim(S_1, S_2) = \sum_{i=1}^{len} (w_i \times \frac{\sum_{i=1}^{|phrase_i|} \mu_{phrase_{i_k},2}}{|phrase_i|}) \quad (5)$$

where len = $MIN(|S_1|, |S_2|)$, i.e., the longest possible phrase (in terms of the number of words) that can appear in both S_1 and S_2, $|phrase_i|$ denotes the number of phrases of length i in S_1, $phrase_{i_k}$ ($1 \leq i \leq len, 1 \leq k \leq |phrase_i|$) is the k^{th} distinct phrase of length i in S_1, and w_i is the normalized weight of phrases of length i in S_1, which is defined as

$$w_i = \frac{i}{\sum_{k=1}^{len} k}, \text{ and } \sum_{j=1}^{len} w_j = 1 \quad (6)$$

Likewise, $Sim(S_2, S_1)$ can be computed accordingly.

Example 3 Let's consider the phrase correlation factors between any two phrases of the same length (between lengths 1 and 3) as created (for demonstration purpose) and shown in Table 1. The correlation factors of phrases (of all possible lengths) in the first sentence, denoted S, in Doc 1 and the first sentence, denoted T, in Doc 2 of Figure 1 are given in Table 2. The degree of similarity between S and T is calculated as

$$\begin{aligned} Sim(S,T) &= \frac{1}{6} \times 0.96 + \frac{2}{6} \times 0.88 + \frac{3}{6} \times 0.6 \\ &= 0.75 \end{aligned}$$

where $\frac{1}{6}$, $\frac{2}{6}$, and $\frac{3}{6}$ are the weights of the single-word, two-word, and three-word phrases, respectively. □

Using $Sim(S_1, S_2)$ and $Sim(S_2, S_1)$, which are not necessary the same, an EQ function is defined in Equation 7 to determine whether S_1 and S_2 should be treated as semantically the same.

$$EQ(S_1, S_2) = \begin{cases} 1 & \text{if } MIN(Sim(S_1, S_2), Sim(S_2, S_1)) \\ & \geq 0.825 \wedge |Sim(S_1, S_2) - Sim(S_2, \\ & S_1)| \leq 0.15 \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

where 0.825 is called the *permissible threshold value*, whereas 0.15 is called the *variation threshold value*. The permissible threshold is a value set to obtain the *minimal similarity* between any two sentences S_1 and S_2 in our phrase-matching approach, which is used partially to determine whether S_1 and S_2 should be treated as equal (EQ). Along with the permissible threshold value, the variation threshold value is used to decrease the number of false positives (i.e., sentences that are *different* but are treated as the *same*) and false negatives (i.e., sentences that are the *same* but are treated as *different*) in determining the equality of two sentences. The variation threshold value sets the the *maximal, allowable difference* in the number of *similar sentences* between S_1 and S_2, which is computed by calculating the difference between $Sim(S_1, S_2)$ and $Sim(S_2, S_1)$. The threshold values 0.825 and 0.15, which provide the *necessary* and *sufficient* conditions for

	N	A	E
A	1	1	1
E	1	1	1
B	0.5	0.5	0.5
H	0.5	0.5	0.5
N	1	1	1
A	1	1	1
E	1	1	1

(a) Single-word Phrases

	NA	AE
AE	1	1
EB	0.5	0.5
BH	0.5	0.5
HN	0.5	0.5
NA	1	1
AE	1	1

(b) 2-word Phrases

	NAE
AEB	0.5
EBH	0.5
BHN	0.5
HNA	0.5
NAE	1

(c) 3-word Phrases

Table 1. Correlation factors among phrases in the 1^{st} sentence of Doc 1 and Doc 2 in Figure 1

Phrases	Similarity of Phrases	Sum of Phrase Similarity	Correlation Factors Given in
A	$\mu_{A,T}$ = 1	$\frac{\sum \mu}{7}$ = 0.96	Table 1(a)
E	$\mu_{E,T}$ = 1		...
B	$\mu_{B,T}$ = 0.88		...
H	$\mu_{H,T}$ = 0.88		...
N	$\mu_{N,T}$ = 1		...
A	$\mu_{A,T}$ = 1		...
E	$\mu_{E,T}$ = 1		...
AE	$\mu_{AE,T}$ = 1	$\frac{\sum \mu}{6}$ = 0.88	Table 1(b)
EB	$\mu_{EB,T}$ = 0.75		...
BH	$\mu_{BH,T}$ = 0.75		...
HN	$\mu_{HN,T}$ = 0.75		...
NA	$\mu_{NA,T}$ = 1		...
AE	$\mu_{AE,T}$ = 1		...
AEB	$\mu_{AEB,T}$ = 0.5	$\frac{\sum \mu}{5}$ = 0.6	Table 1(c)
EBH	$\mu_{EBH,T}$ = 0.5		...
BHN	$\mu_{BHE,T}$ = 0.5		...
HNA	$\mu_{HNA,T}$ = 0.5		...
NAE	$\mu_{NAE,T}$ = 1		...

Table 2. Sum of similarity values of phrases in the 1^{st} sentence of Doc 1 and Doc 2 using the phrase correlation factors in Table 1

estimating the equality of two sentences, are determined by test sentences in the documents that were randomly sampled from TREC (http://trec.nist.gov/data/), Gutenberg (ftp://ftp.archive.org/pub/etext/), and a Web Text Archive, Etext.org (ftp://ftp.etext.org/pub/). A total number of 200 documents were downloaded, and 12.5%, 17.5%, 32.5%, and 37.5% of the 200 documents were chosen from TREC, Gutenberg, Etext.org, and a set of randomly selected Web archive documents, respectively. The threshold values, which are neither dominated by false positives nor false negatives, ensure that the number of similar phrases in S_1 and S_2 is significantly high enough to be treated as the same.

We define the RS function which computes the *degree of resemblance* between any two documents. This function computes as the sum of $EQ(S_{d_{1_i}}, S_{d_{2_j}})$ (i.e., the number of sentences in document doc_1 that appear, or are treated as the same, in document doc_2) and divide it by the total number of sentences in doc_1. $RS(doc_1, doc_2)$, which is the *degree of resemblance* of doc_1 with respect to doc_2, is defined as

$$RS(doc_1, doc_2) = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} EQ(S_{d_{1_i}}, S_{d_{2_j}})}{m} \quad (8)$$

where $S_{d_{1_i}}$ is the i^{th} sentence in doc_1, $S_{d_{2_j}}$ is a j^{th} sentence in doc_2, and m (n, respectively) is the total number of sentences in doc_1 (doc_2, respectively). Likewise, $RS(doc_2, doc_1)$ can be defined accordingly.

We uses the *odd ratio*, denoted *odds*, which captures the degree of overlapping between any two documents, is defined as the ratio of the probability (p) that an event occurs to the probability that it does not (i.e., $odds = \frac{p}{1-p}$), which is the *Dempster-shafer rule* [4]. The odd ratio can be used to compare the relative degree of similarity between any two documents d_1 and d_2. Using the degrees of resemblance $RS(d_1, d_2)$ and $RS(d_2, d_1)$, the odds ratio of d_1 and d_2 is

$$odds(d_1, d_2) = \frac{RS(d_1, d_2) \times RS(d_2, d_1)}{1 - RS(d_1, d_2) \times RS(d_2, d_1)} \quad (9)$$

Using the odd ratio, we determine the degree of similarity of any two documents. The *higher* the odd ratio of two documents is, the *lesser* the *difference* is between them, and the *greater* is the degree of similarity between them.

3 Experimental results

To verify the accuracy of our phrase-matching approach in detecting similar Web documents, we used a phrase query for retrieving Web documents and measured their degrees of similarity. The query is "order-sensitive" (i.e., rearrangement of words in the query yields different semantic meaning)[4] and was submitted to Google (www.Google.com) as a keyword query for retrieving test documents. Using the test set of documents retrieved by the query, we randomly selected five documents that can be categorized into two groups, i.e., *relevant* and *irrelevant* documents with respect to the phrase query. Hereafter, we apply our phrase-matching approach to the documents. Our hypothesis is that phrase-matching approach will detect more similar documents than its counterpart, i.e., the (single-)word-matching approach.

We chose the query "washington george president" as the test case of our phrase-matching approach. Among the top 50 documents retrieved by this query (as a keyword query) on Google, five documents were chosen and divided into $Group$ 1, which consists of documents describing George Washington, the first president of the United

[4]For instance, "fire truck" is a specialized truck that puts out fires. However, "truck fire" means a fire occurred on a truck.

Doc ID	Group ID	Description and Source
Doc 1	Group 1	Biography of George Washington
(whitehouse.gov/history/president/gw1.html)		
Doc 2	Group 1	First president of the USA
(lucidcafe.com/library/96feb/washington.html)		
Doc 3	Group 2	George W. Bush Wikipedia
(wn.wikipedia.org/wiki/George W. Bush)		
Doc 4	Group 2	George W. Bush
(usinfo.state.gov/products/pubs/bush43rdp)		
Doc 5	Group 1	George Washington
(gardenofpraise.com/ibdwash.htm)		

Table 3. The selected, retrieved documents according to the query, "washington george president"

Phrase-Based Matching

	Doc 1	Doc 2	doc 3	Doc 4	Doc 5
Doc 1	1	$\mathbf{1.6e^{-3}}$	$1.3e^{-5}$	$4.6e^{-4}$	$\mathbf{1.0e^{-3}}$
Doc 2	$1.6e^{-3}$	1	$1.9e^{-5}$	$2.9e^{-4}$	$3.1e^{-3}$
Doc 3	$1.3e^{-5}$	$1.9e^{-5}$	1	$\mathbf{1.2e^{-3}}$	$1.0e^{-4}$
Doc 4	$4.6e^{-4}$	$2.9e^{-4}$	$1.2e^{-3}$	1	$7.2e^{-5}$
Doc 5	$1.0e^{-3}$	$3.1e^{-3}$	$1.0e^{-4}$	$7.2e^{-5}$	1

Word-Based Matching

	Doc 1	Doc 2	Doc 3	Doc 4	Doc 5
Doc 1	1	$\mathbf{6.9e^{-3}}$	$4.3e^{-3}$	$\mathbf{7.1e^{-3}}$	$3.9e^{-2}$
Doc 2	$6.9e^{-3}$	1	$4.1e^{-4}$	$6.3e^{-4}$	$1.2e^{-2}$
Doc 3	$4.3e^{-3}$	$4.1e^{-4}$	1	$2.9e^{-2}$	$2.5e^{-3}$
Doc 4	$7.1e^{-3}$	$6.3e^{-4}$	$2.9e^{-2}$	1	$5.1e^{-3}$
Doc 5	$3.9e^{-2}$	$1.2e^{-2}$	$2.5e^{-3}$	$5.1e^{-3}$	1

Table 4. Phrase matching versus single-word matching on the query, "washington george president"

States, and $Group$ 2, which contains documents on George W. Bush, the 43^{rd} president of the United States. Table 3 provides a brief description of these documents, whereas Table 4 shows the result of our phrase-matching versus the word-matching approach.

Table 4 displays the odd ratio of any two documents in Table 3. The top table in Table 4 is the result by running our phrase-matching approach, whereas the bottom one in Table 4 is the result of the single-word matching approach. According to Table 3, Doc 1, Doc 2, and Doc 5 belong to $Group$ 1, whereas Doc 3 and Doc 4 belong to $Group$ 2. After computing the odd ratio of any two of these documents, we conclude that *phrase-matching* approach yields more accurate results in identifying similar documents than the *word-matching* approach, which uses the same set of equations (i.e., Equations 1 to 9) for single-word, instead of phrase, matching. For instance, on the phrase-matching table in Table 4, the first three documents that return high odd ratios with Doc 1 are Doc 1 itself (which is obvious), Doc 2 ($1.6e^{-3}$), and Doc 5 ($1.0e^{-3}$), which belong to $Group$ 1. Also, the document that yields the high odd ratio with Doc 3 (besides Doc 3 itself) in the phrase-matching table is Doc 4 ($1.2e^{-3}$), which belong to the same group (i.e., $Group$ 2). The word-matching approach, on the other hand, failed to detect the most similar documents. For example, the odd ratio $7.1e^{-3}$ of Doc 1 and Doc 4 is higher than the odd ratio $6.9e^{-3}$ of Doc 1 and Doc 2, even though Doc 1 and Doc 2 belong to $Group$ 1, whereas Doc 4 belongs to $Group$ 2.

4 Conclusions

In this paper, we present a phrase-matching approach in detecting the degree of similarity between any two Web documents using (i) the fuzzy-word correlation factors among all the words in the documents, (ii) *Document Index Graph* (DIG) to capture common phrases appeared in the documents, and (iii) odd ratio to determine the similarity of the documents based on phrase correlation factors. Experimental results show that our hypothesis is valid, i.e., the phrase-matching approach outperforms the word-matching approach in detecting similar documents and is accurate overall. Even though our phrase-matching approach is a slower process than the single-word matching approach due to the number of phrases to be compared (as needed), it still runs in polynomial-time.

References

[1] D. Fetterly, M. Manasse, and M. Najork. Detecting Phrase-Level Duplication on the World Wide Web. In *Proc. of ACM SIGIR*, pages 170–177, 2005.

[2] K. Hammouda and M. Kamel. Efficient Phrase-Based Document Indexing for Web Document Clustering. *IEEE Trans. on KDE*, 16(10):1279–1296, 2004.

[3] E. Kerre, R. Zenner, and R. de Caluwe. The Use of Fuzzy Set Theory in Information Retrieval and Databases: A Survey. *American Society for Info. Sci.*, 37(5):341–345, 1986.

[4] G. Luger. *Artificial Intelligence, Structures and Strategies for Complex Problem Solving, 5^{th} Ed.* Addison Wesley, 2005.

[5] M. Narita and Y. Ogawa. The Use of Phrases from Query Texts in Information Retrieval. In *Proc. of ACM SIGIR*, pages 318–320, 2000.

[6] A. Pereira and N. Ziviani. Retrieving Similar Documents from the Web. *Web Engineering*, 2:247–261, 2004.

[7] M. Porter. An Algorithm for Suffix Stripping. *Program*, 14(3):130–137, 1980.

[8] Wikipedia. http://en.wikipedia.org/wiki/Wikipedia:Database_download.

[9] Wikipedia. http://en.wikipedia.org/wiki/Wikipedia:Overview_FAQ 03Feb2006.

[10] R. Yerra and Y.-K. Ng. Detecting Similar HTML Documents Using a Fuzzy Set Information Retrieval Approach. In *Proc. of Intl. Conf. on Granular Comp.*, pages 693–699, 2005.