MWDM: Minimum Weighted Displacement Model

Javad Paksima, Alimohammad Zareh Bidoki, Vali Derhami, Sajad, Zarifzadeh

Department of Electrical and Computer, University of Yazd, Yazd, Iran[1]

Abstract
Based on researches on search engines, the majority of user queries is more than one term. For queries with more than one term, two models can be used. The first model assumes that query terms are independent of each other but the second model considers a location and order dependency between query terms. Experiments show that in majority of queries there are dependencies between terms. One of the parameters that can specify dependencies between query terms is the distance between query terms in the document. In this paper, a new definition of distance based on Minimum Weighted Displacement Model (MWDM) of document terms to accommodate the query terms is presented. Also because most of the ranking algorithms use the TF (Term Frequency) to score documents and for queries more than one term, there is no clear definition of these parameters; in this paper, Phrase Frequency and Inverted Document Frequency are defined according to the new distance concept. Also algorithms to calculate them are presented. The results of the proposed algorithm compared with multiple corresponding algorithms shows a favorable increase in average precision.

Keywords:
Search engine, ranking, distance, term dependency, Proximity

1. Introduction

Finding high-quality web pages is one of the most important tasks of search engines. The relevance between the documents found and the query searched depends on the user observation and increases the complexity of ranking algorithms. The other issue is that users often explore just the first 10 to 20 results [1] while millions of pages related to a query may exist. So search engines have to use suitable algorithms with high performance to find the most relevant pages.

The ranking section is an important part of search engines. Ranking is a process in which the web page quality is estimated by the search engine. There are two main methods for ranking web pages. In the first method, ranking is done based on the content of documents (traditional rankings). Models, such as Boolean model, probability model and vector space model are used to rank documents based on their contents [2]. In the second method, based on the graph, web connections and the importance of web pages, ranking process is performed.

Manuscript received June 5, 2026
Manuscript revised June 20, 2026
https://doi.org/10.22937/IJCSNS.2026.26.6.14

In traditional ranking, search engine tries to find the relevance between documents and query in order to perform the ranking process. In these algorithms for each query, documents with more content similarity to query terms will have the higher scores. For example, TF-IDF [3] and BM25 [4] algorithms are two common examples for this ranking type.

Web includes a large number of unstructured documents which are connected to each other and generate a very large graph. Usually number of terms in queries are small (2.4 terms per query [5]) but the total set of terms is very large. In contrast, the majority of information retrieval algorithms usually have acceptable performance with small number of documents and large number of query terms [6]. A phrase, is a list of terms that have a certain order. For example, "search engine" is a phrase including two terms that its first term is "search" and the second term is "engine". Most of the phrases are two-terms long and less than one percent of the phrases are longer than six terms [7]. In an analysis over half a million queries in Excite search engine, it's found that 8.4 percent of the queries include quotation marks (' or ") [8]. Another study showed that the users' intentions in about 40 percent of queries with two or more terms without any quotation marks were to find the whole phrases, not any of the single terms [9].

The main method to search a phrase is to use the Inverted Index [10]. Each record in the Inverted Index contains a term with a list of ordered ternaries including document number, term frequency in the document and the locations in which the term appears in the document as follows:

$$<d, tf_{d,t}, [\ p_1, p_2, \ldots.]>$$

Where d is the document ID that contains the term, tf_{dt} is the t frequency in document d and p_i shows the t locations in that document. The following list is a sample of this ternary:

$$<101, 4, [5, 11, 25, 77]>$$

Using the Inverted Index list, the processing time for counting the number of phrases depends on the number of query terms and their frequencies in the document. In order to evaluate a phrase, the query terms locations must be first extracted and then counting is done by combining them.

For ranking documents, when the query is just one term long, TF and IDF are the most important parameters.

These parameters are used in TF-IDF and BM25 methods. If we want to use these methods to the search a phrase, similar parameters must be defined.

If in the phrase search, quotation marks are used, there is no ambiguity and the search could be done like a single-term search and multiple algorithms are proposed for this process [11] [12]. Without using any quotation marks, it's better to allocate the highest scores to the documents with highest phrase frequently with the same order and the least distance. For this reason, in this paper a similar parameter to TF called PF is defined as the phrase frequency and an algorithm to calculate the IDF based on PF is also proposed.

The remainder of this paper is organized as follows. In the next section, we present a review of related work. Then the used terminology are defined. In section four the proposed algorithm to calculate PF and IDF has been introduced. Finally, the comparison between the proposed algorithm and some other algorithms have been carried out.

2. Related Works

One of the oldest works on terms dependency is done by Rijisbergen (1977), which theoretically ranked the documents based on the proximity of search query terms [13]. In the proposed method, the relevance of query terms was determined using the bigram data and then it is used to recover the maximum spanning tree (MST). Years later, in 2002, Alan and Nallapati proposed their idea on how to determine the relevance of query terms [14]. To reduce the time, they suggested that instead of using document statistics, sentences statistics could be used to build the maximum spanning tree.

In 1991, Keen released his experimental results. He suggested that using information on term positions can help narrow down search results, by screening out irrelevant results. For the first time, Keen defined the concept of distance between two terms. By his definition, distance is the number of non-matching terms between the first and the last matching in a sentence. He offered seven different approaches about using the terms' proximity. In his study, the best results were achieved when the algorithm rewards the lower distance between the query terms [15].

Also in 1991, for the first time, Croft et al. [16] proposed a method for using the inference network as for adjacent terms called InQuery. They also deleted the terms with high DF [1] from the query, for example, the term "city" was removed from the "recreation places of Shiraz city" query. They achieved an acceptable precision on the first pages of the search, but in general the precision was decreased. This issue was confirmed later in 2005 by Metzler and Croft [17]. Dang et al. followed the Croft work. They calculated the document scores based on the terms appear in the window that is the focus of the query terms as well as the user feedback [18].

A variety of language models for the dependence of the query terms are presented. For the first time in 1999, Song and Croft [19] developed the standard unigram model using the interpolation of bigram model. In a small scale experiment, using bigrams improved the results. Gao et al. in 2004, proposed the dependence language model (DLM) [20]. In their model, they defined the connection between the distribution of unigram language model and the documents including the adjacent terms in a window with a length of three. They used a series of documents to estimate the most similar linkage that sequentially connects all query terms using every query term once. Only term-pairs in this linkage were considered in the model. The algorithm could not be used in practice because the extraction of full link structures for a search query was so time-consuming [21].

Another group of researchers suggested ways to increase the term dependencies in probability models. For example Rasolofo and Savy developed the BM25 method based on the term dependencies [22]. They replaced the parameter TF with a distance function. In their method, the bigram distances are calculated in a five terms window. He et al. in 2011 used a window to count the n-gram frequency in a document and changed the BM25 that uses the frequency to calculate the score. Distance criteria in their proposed model was the minimum number of terms that separate a sequence of terms including all query terms [21]. In the proposed algorithm, this idea has been used and the phrase frequency parameter (PF) is defined to be used in ranking algorithms instead of TF. Eickhoff et al. used a statistical tool called Copulas to develop their statistical model [23]. They used Copulas to find the co-occurrence possibility of query terms. The main advantage of this method is its high speed.

Büttcher et al. proposed the aggregation model that calculates the proximity score for each query term separately [24]. They algorithm was implemented based on inverted index files. If a term changes during the posting list processing for the query terms, the distance will cumulatively increase. Similar works was done by Tao et al. [25]. They cumulatively calculated the five characteristics of the distance of the terms and then used them as the weight of the terms in content-based methods. The best results were obtained when the weight criterion considered as the minimum gap between all query terms. Based on the results obtained by Tao and Zhai, in 2009, Zhao and Yun proposed the proximity language model (PLM) [26]. In their approach, the shortest distance between all query terms in the content of the document was obtained and the sum of minimal distances was used to calculate the scores.

CRTER model was proposed by Zhao et al. [27] [28]. They used the intersection of query terms to define the dependency. Closer terms were receiving larger weights. To calculate the weight, some functions were considered for each term and then the intersection point of two adjacent term was combined with BM25.

There are other related issues like searching an exact phrase, which means when a user specifies a phrase using the quotation marks. Another issue is the query expansion. For example, Miao et al. in 2012, the improved the query expansion using the concept of term proximity and user feedback [29].

3. Terminology used

Distance: Minimum displacement (shift) of the terms in the document to match the query terms. For example, if in the 'a c b' document, we search the 'a b' phrase (without quotation marks), the distance will be one because the 'b' in the document must be shifted one place to match the 'a b' query (for simplicity, we assumed each Latin alphabet as a term, means that the above document has three terms a, b and c). Searching the 'b a' phrase in the above document, needs two displacement to match the query, thus its distance is equal to two. The zero distance means that there exists an exact match without any displacement. For example, searching for 'c b' in the above document yields to a zero distance.

The term distance, has different definitions in different methods mentioned in the previous section. Some researchers have considered the minimum distance between bigrams and some other used the total distance and other methods to calculate the distance.

MWDM model divides a web document into three parts: body, text and anchor texts. This model considers different term weights in different parts and while moving the terms from one part to another, these weighs are also effective in calculating the distance. Figure 1 shows a view of these three parts and the imaginary distance between them.

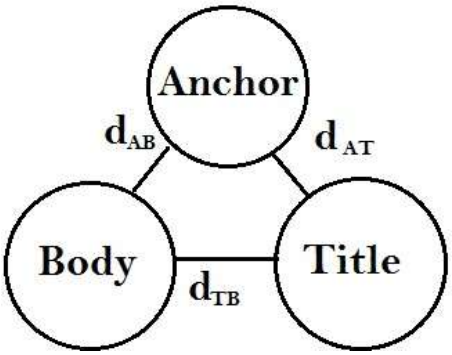

Figure 1: A view of different parts of the document and the imaginary distance between them

The s(d) function: s(d) is a function that converts the distance to score. It takes the distance as its input. This function should be decreased while increasing the distance, and has zero value in distance equal to one. It means that:

$s(0)=1$
If $d_1 > d_2$ then $s(d_1) < s(d_2)$
For example, $s(d)=1/(d+1)$ has the above-mentioned properties and can be used in phrase evaluation [30]. Or in [22] scores are calculated as the inverse square of distance or in [25] a logarithmic equation has been used.

PF: to match the number of phrases in a document with the methods that use TF to calculate score, a parameter called PF which stands for Phrase Frequency is defined. TF is always an integer number but PF can be a decimal number. PF is calculated as follows:

$$PF = \sum_{i=1}^{n} s(d_i) \quad (1)$$

Where n is the number of query phrases found in the document and d_i is the distance of i-th phrase. In the optimal case, the combination of phrases must be chosen such that maximize the PF.

QTM²: Most search engines specify a maximum number of query input terms. For example, in the Google search engine, the maximum number of query terms is 32 [31]. QTM specifies the maximum number of terms allowed in a query.

4. The proposed algorithm

In the proposed algorithm we follow two issues, first calculating the distance and second, calculating the PF. Furthermore, the IDF value is determined by PF definition. During the distance calculation time we assume that the terms have appeared in the document only once and then find a place to gather the terms that minimize the transitions.

4.1. The distance calculating algorithm

When matching a phrase in a document distance is calculated based on the locations and weights of query terms. Here the distance is equal to the smallest weighted displacement of query terms in the document to build the query. For example, if we search the 'a b c' query in the 'a d f c d b e' document, in one case (assuming an equal weight for all terms) 'a' can be fixed in its place while we move 'b' and 'c'. In this case the number of displacement for 'b' and 'c' are 4 and 1, respectively, resulting in a total displacement (distance) equal to 5. But if we consider 'c' as the fixed one, the total displacement would be equal to 4 since 'b' and 'a' need 3 and 1 displacement.

Another example is that if we want to search the 'a b c' query and 'a', 'b' and 'c' are located in the 100th, 300th and 200th locations with weightings equal to 2, 2 and 3, respectively, (a document like '...a ...c … b…'); using Lemma 1 and 2 and theorem 1 it is proved that if we fix the center of mass and displace the other terms, we can minimize the displacement. In the above example, 'c' should be fixed and then we just have to displace 'a' and 'b' to make the query. In this case, the displacement cost would be equal to 398. Because 'b' should be displaced 101 times to be placed before 'c' and due to its weight, its displacement cost would be equal to 202. Also 'a' needs 98 displacement to be placed before 'b' and again due to its weight, its displacement cost would be 196. Thus the total cost to create the query would be equal to 398.

Also, if the query terms appear in more than one part, two other issues should be considered. The first issue is an imaginary distance between the parts and the other one is that we should consider the specific weight for every term in each part. In order to reduce the computational load, some formulas are presented to calculate the distance. Also in order to determine the imaginary distance between the parts and the weight given to the terms in different parts, one of the learning methods must be used. Here the genetic algorithm is used to calculate the distances and optimal weights.

In the following lines, it is proved that the center of mass for the weights and the location of the terms in the document, is the best place to be selected as the pivotal displacement point and all the terms must gathered there.

Lemma 1: the gathering place for the terms of a document in MWDM model, could not be an intermediate location of the terms found in the document (the gathering place for terms is the location of one of the query terms in the document).

Proof: The aim of this Lemma is to limit the places which may be considered as the gathering place of terms. For proof, it is assumed that the query terms are arranged according to their position in the document. P_i is assumed as the i^{th} term location with a w_i weight in the document (the i^{th} query term does not necessarily coincident with the place of the i^{th} location of the term found in the document). Figure 2 shows a view of query terms matching in the document.

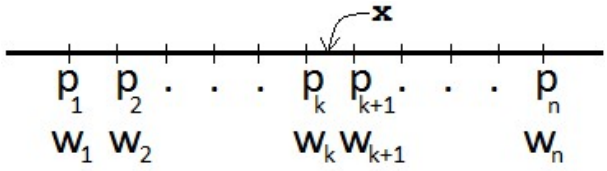

Figure 2: Location of the query terms in the document (p) and their weights (w)

If we consider the d_x as the displacement cost of the terms, to gather them at the location x in the document, then it can be calculated as follows:

$$d_x = w_1(x - p_1) + w_2(x - p_2) + \cdots + w_k(x - p_k) + w_{k+1}(p_{k+1} - x) + \cdots + w_n(p_n - x)$$

And also d_{x+1} as the displacement cost of the terms to be gathered at the location (x + 1) in the document will be as follows:

$$d_{x+1} = w_1(x + 1 - p_1) + \cdots + w_k(x + 1 - p_k) + w_{k+1}(p_{k+1} - x - 1) + \cdots + w_n(p_n - x - 1) \quad (3$$

We show the difference between d_x and d_{x+1} as Δ_x and will be calculated as follows:

$$\Delta_x = d_{x+1} - d_x = w_1 + \cdots + w_k - w_{k+1} - \cdots - w_n \quad (4)$$

Equation 4 shows that the Δ_x value for the places between the consecutive locations of query terms in the document is constant and so d_x is a line segment and for a line segment the smallest value is located at the beginning of it (p_k) or at its end (p_{k+1}).

Lemma 2: d_x function is almost parabolic shape and is similar to Figure 3.

Proof: According to Lemma 1 it is clear that d_x is a line segment in the intermediate points and thus its shape only depends on their values at the p_is.

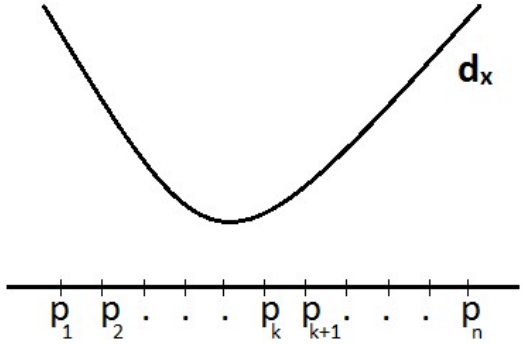

Figure 3: The approximate shape of d_x function

To present the d_x changes in the successive p_ks, we defined the φ_k as follows:

$$\varphi_k = d_{p_{k+1}} - d_{p_k} \tag{5}$$

It is sufficient to show that φ_k is descending at the beginning and then it becomes ascending. Substituting d in equation (2) we get:

$$\varphi_k = (w_1 + \cdots + w_k - w_{k+1} - \cdots - w_n) \times (p_{k+1} - p_k) \tag{6}$$

Given that in equation (6) the second coefficient is always positive, the φ_k sign only depends on the first coefficient. To find the φ_k flow, we acquire the sign of the first polynomial.

$$\begin{aligned} sign(\varphi_1) &= sign((w_1 - w_2 - w_3 - \cdots - w_n) \\ sign(\varphi_2) &= sign(w_1 + w_2 - w_3 - \cdots - w_n) \\ sign(\varphi_3) &= sign(w_1 + w_2 + w_3 - \cdots - w_n) \end{aligned} \tag{7}$$

As shown in equation (7), φ_k sign is negative at first and then step by step it increases and finally ends in a positive sign, so d_x is initially descending and then changes into ascending form, thus d_x has a parabolic shape.

Theorem 1: In the MWDM model, the smallest d_x value is obtained at center of mass of the query terms in the document.

Proof: According to Lemma 1, the optimal point for d_x is a place in the document that contains one of the query terms and according to Lemma 2, d_x value is initially descending and then changes into ascending form, and so in the moment that descending trend ends and the ascending trend begins, the smallest d_x value appears and φ_k is close to zero. If p_m considered as a point we have:

$$\begin{aligned} &\varphi_m = (w_1 + \cdots + w_m - w_{m+1} - \cdots - w_n) \cong 0 \\ &\Rightarrow\ w_1 + \cdots + w_m \cong w_{m+1} + \cdots + w_n \end{aligned} \tag{8}$$

As a result p_m is the point where the weight of terms is almost the same on both sides. In other words, the best place to gather the document terms is their center of mass.
The valuable result of theorem 1 is the significant reduction in the complexity of the ranking algorithm based on the MWDM model. In order to implement the algorithm, we must first calculate the φ_1 and then by adding the $2w_2$ to it, we can calculate the φ_2 and so on. As soon as the φ_k sign changes from negative to positive, the p_k will be returned as optimal point. Based on the above results of the theorem and lemmas, algorithm 1 is proposed to calculate the distance.

Algorithm Distance

01: **Input** Query *q*, Text *T*
02: **Output** *Distance*
03: **Assumption**
04: *q* include <$t_1, t_2, \ldots, t_m$>
05: *T* include <$(t_1,p_1), (t_2,p_2), \ldots, (t_m, p_m)$>
06: **Begin Algorithm**
07: sort <$p_1, p_2, \ldots, p_m$>
08: Mid← Center of mass of <$p_1, p_2, \ldots, p_m$>
09: **For** *i*=1 to m
10: move (t_i, p_i) to *Mid+i-1*
11: *Distance* ← *Distance* +abs(*Mid+i-1-*p_i)* w_i
12: **Return** Distance

Algorithm 1: distance calculation algorithm

In the algorithm 1 it is assumed that the query has m terms and terms are in a $t_1, t_2, \ldots, t_m$ form. Also we consider the location of the t_i term in the document as p_i (p_is are not necessarily arranged).

4.2. PF calculation algorithm

One of the algorithms that can calculate PF, is the brute-force search algorithm. In this method, all the combinations that can be used to create the phrase are extracted and then the best one with the higher total score is selected. In [32] and [33] some algorithms are proposed to extract the best matches for a phrase in a document that still have some drawbacks. For example, in all the proposed algorithms, searching 'a b c' query in 'b b a c' and 'b a c' documents, receives the same score although the first document is more relevant. So here the brute-force search is used to extract the best matches.

In the proposed method, first using the "AND" query, all the documents that contain the query terms are determined. Then, using the inverted index for all the phrase

terms, all the possible permutations for the phrase matches are investigated and for each permutation, the distance summation will be calculated. The main objective is to find the permutation that produces the maximum distance summation.

Algorithm 2 shows the method to calculate the PF. In this algorithm, the Max variable is used to hold the maximum distance summation value. The main work of this algorithm is to produce all the possible combinations of the query terms in the document.

Algorithm PF

01: **Input** Query *q*, Document *d*
02: **Output** PF
03: **Assumption**
04: *q* is <t_1, t_2... t_m>
05: *d* includes <t_1,(P_{11},P_{12},...)> , <t_2,(P_{21},P_{22},...)> ,..., <t_m,(P_{m1},P_{m2},...)>
06: **Begin Algorithm**
07: *Max*← 0
08: **Repeat**
09: Make a new permutation
10: *S*← Calculate Distance of all phrases
11: If S>*Max* Then
12: *Max*←*S*
13: **Until** no new permutation
14: **Return** Max

Algorithm 2: PF calculation algorithm for a phrase

In Algorithm 2, the P_{ij} variable is used to specify the i^{th} location query term is used in the document. Line 9 in the algorithm, produces different permutations of query terms matching with the equivalent terms of the document is responsible for the search term. For example, the first permutation can be selected as follows:

Phrase1: <t_1, P_{11}>, <t_2, P_{21}>… <t_n, P_{n1}>
Phrase2: <t_1, P_{12}>, <t_2, P_{22}>… <t_n, P_{n2}>
.

For the next permutation, we can change the t_1 location in Phrase1 and consider P_{12} and similarly place the t_1 location in Phrase 2 in P_{11} to create a new permutation. Line 10 in the algorithm calculates the minimum displacement for every extracted phrase in each permutation and stores the total scores obtained in the variable S.

Although in the worst case the algorithm is linear due to limited QTM but because the importance of linear coefficient in mass searching, it must be optimized. To optimize this algorithm we can use the distance based pruning. One pruning approach is to use this distance concept.

It is clear that at a distance greater than a specified value, the connection between the terms is almost lost and with this assumption we can limit the distance of terms and prune a part of the tree. For example, if two terms are located at a distance equal to 1000, they didn't produce a significant score and therefore these cases can be eliminated [34].

Another way to optimize the above algorithm is to process the queries statistically and extract the borders for decision making based on tfs and idfs. Means that for the low idf value, small TF can be ignored.

4.3. IDF calculation

During the search for a phrase, IDF parameter must be defined appropriately to determine the correct significance of the phrase. IDF is a parameter which is directly related to the inverted logarithm of the DF and DF shows the number of documents that contain the intended phrase. The lower the number of the documents, the more important the phrase will be. One of the most famous definitions for the IDF is as follows:

$$idf_t = log\frac{N}{1+df_t} \tag{9}$$

Where n is the total number of documents and df_t is the number of documents containing the t term. If we want to define the IDF for a phrase in a similar way, we must calculate the DF for the phrase. To calculate the DF for a phrase we can divide the documents into two categories:

- Documents that have PF greater or equal to 1.
- Documents that have PF smaller than 1.

It is obvious that when counting the number of documents including the phrase, documents that have PF larger or equal to 1 must be counted.

Also the documents that have PF smaller than 1 must be counted, but with a coefficient less than 1. We define the equation (10) for this purpose as follows and also in equation (11) the DF value will be calculated.

$$F(d,q) = \begin{cases} 1 & PF \geq 1 \\ PF & PF < 1 \end{cases} \tag{10}$$

$$df_q = \sum_d F(d,q) \tag{11}$$

Now, using the DF value, We can simply use the equation (9) to calculate the idf. Algorithm 3 shows the IDF calculation steps.

Algorithm IDF

01: **Input** Query q
02: **Output** *IDF*
03: **Assumption**
04: q include $<t_1, t_2 \ldots t_m>$
05: N is total document in corpus
06: **Begin Algorithm**
07: $DF \leftarrow 0$
08: **For** every document d_i
09: $PF \leftarrow$ calculate PF(q,d_i)
10: **If** PF>=1 **Then**
11: $DF \leftarrow DF$+1
12: **Else**
13: $DF \leftarrow DF$+PF
14: $IDF \leftarrow$ Log(N/(1+DF))
15: **Return** IDF

Algorithm 3: IDF calculation algorithm for a phrase

With a simple change we can merge the algorithm 3 into algorithm 2 (PF calculation algorithm) and while calculating the PF values for all documents, we can also calculate the IDF values. Of course, the performance issue for these algorithms will be different from the traditional methods for extracting TF and IDF values. In traditional methods, the TF value for each term in each document is easily accessible and also the IDF value for each term in the index time can be calculated, stored and used in the ranking section during the search without any special processing task. But in this way, these two parameters must be calculated during the search with a processing task that makes it slower than the traditional methods.

4.4. Time complexity of algorithm

If you want to match a phrase, based on Theorem 1 we must find the center of mass of it. If you use the quicksort method to find the center of mass, the time complexity would be O(nlogn) that n represents the number of values we want to calculate their center of mass. As shown in section 2, search engines determine a maximum number of query terms that we've shown it with a constant QTM. So, the time complexity of finding the distance would be O(QTM*log(QTM)) and since the QTM value is a constant, this complexity would be O(1). The time complexity of PF calculation based on the proposed algorithm, has a direct relation with the number of phrase permutations in the document. The number of permutations of a query in a form of $t_1t_2 \ldots t_m$ will be calculated as follows:

$$T(m) = \prod_{i=1}^{m} tf(t_i) \tag{12}$$

For the time complexity of PF calculation, we have to find the three best, worst and average cases. It is obvious that the best case occurs when the TF value for all the phrase terms is equal to 1 and therefore the best case time complexity would be O(1). Also the worst case occurs when the document absolutely contains only query terms. It can be proved that the time complexity of PF calculation in the average case, would be O(1). In the following, using the Lemma 3 and Theorems 2 and 3, the time complexity of PF calculation in the worst and average cases are presented.

Lemma 3: if we consider $n = \sum_{i=1}^{m} a_i$ as a constant, then $\prod_{i=1}^{m} a_i$ would be maximized when all a_is are equal.
This Lemma can be easily proved using the Lagrange multipliers [35].

Theorem 2: the time complexity of PF calculation in the worst case would be $O(n^{QTM})$, where n is the number of the terms in the document.

Proof: Obviously, the worst case occurs when every term in the document included in the query terms, so $n = \sum_{i=1}^{m} tf(t_i)$. According to Lemma 3 in order to maximize the tfs product, all the $tf(t_i)$s must be equal. So in the worst case, we have:

$$tf(t_i) = \frac{n}{m} \tag{13}$$

Since the maximum value of m is equal to QTM, the time complexity of PF calculation in the worst case would be $O(n^{QTM})$.

Theorem 3: the time complexity of PF calculation in the average case would be O(1).

Proof: for the average case we must find the expected value for tfs. It can be proved that the distribution of each term in any document is a Poisson distribution [36] and it is clear that the frequency of each term in any document is a binomial distribution. we can investigate this subject using TREC2003 and TREC2004 datasets [37]. So we have $E(tf) = \mu$ where μ is the expected value of the Poisson distribution.

Assuming tfs are independent from each other, the expected value of the product can changed into the product of expected values [38]. The upper bound of the number of permutations in the average case can be obtained from the Equation (13):

$$E\left[\prod_{i=1}^{m} tf(t_i)\right] = \prod_{i=1}^{m} E(tf(t_i)) = \mu^m \leq \mu^{MTQ} \tag{14}$$

Due to the constant values of μ and QTM, the time complexity of PF calculation in the average would be O(1).

5. Evaluation Results

To evaluate the proposed algorithm, a part of real data from the Parsijoo[3] search engine is used. This test set consists of approximately four hundred thousand web pages and around fifty evaluated Persian query. Also the precision measure at the n^{th} location (p@n) and the mean average precision (MAP) was used to compare the proposed algorithm with some other algorithms.
Precision benchmark in the n^{th} location or P@n indicates the ratio of the number of relevant documents in the first n final ranking documents for each query to n. The P@n formula is shown below:

$$P@n = \frac{1}{n}\sum_{j=1}^{n} r_j \tag{15}$$

Where r_j shows the relevance of the j^{th} document in the final ranking. Also the mean average precision (MAP) for any query is defined as the average of the P@n values for all relevant documents.

$$AP = \frac{1}{|D_+|}\sum_{j=1}^{N} r_j \times P@j \tag{16}$$

Where n represents the number of documents, D_+ shows the number of related documents and r_j shows the relevance of the j^{th} document; the mean average precision (MAP) is presented as the average AP values for all the queries. The MWDM model has been compared with two models: BM25F and CRTER models. The BM25F model is like the BM25 model but the BM25F model considers different weights for the terms found in different parts and we can adjust the model parameters for each part. In the CRTER model, the impact of terms on the adjacent terms is estimated using functions such as Gaussian, triangular, circular and cosine [39]. These functions are called kernel functions. Each kernel function must have the following characteristics:

- No negation.
- Continuous.
- Symmetrical.
- Uniform.
- having the zero value for the distance equal to 1.

The intersection of two q_i and q_j terms occurs when these two terms appear near each other in one document and thus their kernel function has a common point. Common point is a number between zero and one that the smaller the distance, the larger that number would be. Figure 4 shows an example of this intersection.

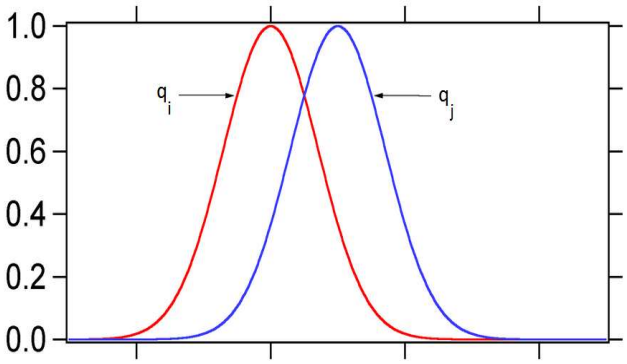

Figure 4: The intersection of two q_i and q_j terms based on two Gaussian kernel function

TF and DF parameters are used directly or indirectly in most cases. In the CRTER model, the new definitions for these two parameters are presented as follows:

$$tf(q_{ij}; D) = \sum_{m=1}^{tf_i}\sum_{n=1}^{tf_j} Kernel\left(\frac{1}{2}dist(Pos_m, Pos_n)\right) \tag{17}$$

$$df(q_{ij}) = \sum \frac{tf(q_{ij}; D)}{occur(q_{ij}; D)} \tag{18}$$

$$occur(q_{ij}; D) = \sum_{m=1}^{tf_i}\sum_{n=1}^{tf_j} 1\{Kernel\left(\frac{1}{2}dist(Pos_m, Pos_n)\right) \neq 0\} \tag{19}$$

In these equations, the dist function is used to calculate the distance between two terms in the document. In an experiment in [40] the best results were obtained when the triangular function has been used as a kernel function. Then the new TF and DF functions are used in BM25 function.

Table 1 shows the weight of the different sections of the document and also the parameter values that have been calculated on the basis of a given dataset for BM25F. Also the weight of the terms in different parts of the document and the imaginary distance between these parts for the MWDM model are given in Table 2.

Table 1: parameters and weights calculated for the BM25F

Parameter	value
k_1	**1.4**
b_{title}	**0.1**
b_{body}	**0.98**
b_{anchor}	**6**
v_{title}	**3.6**
v_{body}	**1**
v_{anchor}	**1.4**

Table 2: imaginary distance between different parts of the document and the weight of each section calculated for the MWDM model

parameter	value
d_{AB}	**1**
d_{TB}	**0**
d_{AT}	**5**
v_{title}	**6.6**
v_{body}	**1**
v_{anchor}	**10.3**

Figures 5, 6 and 7 show a comparison between the three search algorithms. In these figures, three different s(d) functions have been used.

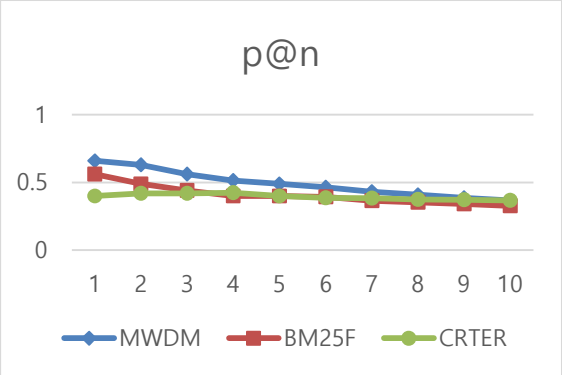

Figure 5: comparison between the proposed, CRTER and BM25F algorithms using the P@n benchmark in $s(d) = \frac{1}{d}$ state

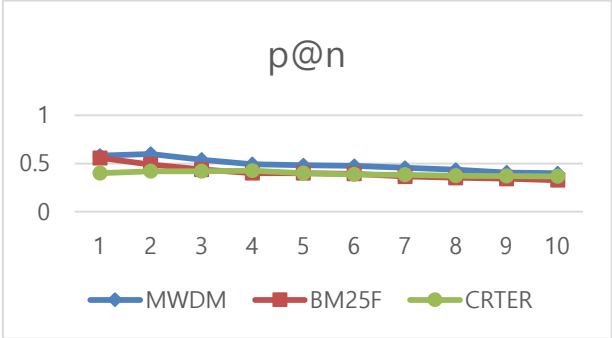

Figure 6: comparison between the proposed, CRTER and BM25F algorithms using the P@n benchmark in $s(d) = \frac{1}{d^2}$ state

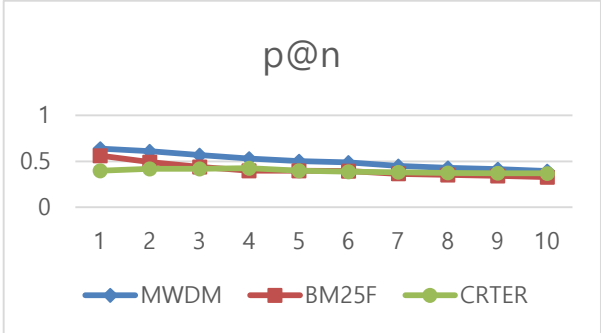

Figure 7: comparison between the proposed, CRTER and BM25F algorithms using the P@n benchmark in $s(d) = \frac{1}{d\sqrt{d}}$ case

Also Figure 8 shows that in the average case using the MAP benchmark, the proposed algorithm accuracy is higher than four other algorithms.

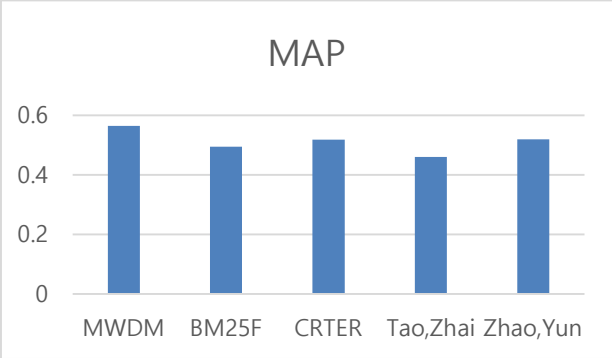

Figure 8: comparison between the proposed algorithm and two other algorithms using the MAP benchmark

6. Conclusion and future works

In this paper one of the search engine topics, searching a phrase without using the quotation marks, has been assessed. In searching a phrase in a document, it is possible to locate the phrase terms in different locations of the document, thus the concept of distance was defined that represents the minimum displacement needed to match a phrase. The obtained distance was then used in some algorithms calculate the PF and the IDF values and finally the proposed algorithm was compared with some other similar algorithms. In future works we can concentrate on accelerating the PF and IDF calculation algorithms. Also the parallelism can be investigated as a way to increase the computing speed in PF and IDF.

7. References

[1] A. Chuklin, P. Serdyukov, and M. De Rijke, "Modeling clicks beyond the first result page," in *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, 2013, pp. 1217–1220.

[2] R. Baeza-Yates, B. Ribeiro-Neto, and others, *Modern information retrieval*, vol. 463. ACM press New York, 1999.

[3] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Inf. Process. Manag.*, vol. 24, no. 5, pp. 513–523, 1988.

[4] S. E. Robertson, *Overview of the Okapi projects*, vol. 53, no. 1. MCB UP Ltd, 1997, pp. 3–7.

[5] Y. Zhang and A. Moffat, "Some Observations on User Search Behaviour.," *Austr. J. Intell. Inf. Process. Syst.*, vol. 9, no. 2, pp. 1–8, 2006.

[6] A. Z. Bidoki, "Effective Web Ranking and Crawling(in persian)," University of Tehran, 2009.

[7] D. Bahle, H. Williams, and J. Zobel, "Compaction techniques for nextword indexes," in *String Processing and Information Retrieval, International Symposium on*, 2001, p. 33.

[8] H. E. Williams, J. Zobel, and D. Bahle, "Fast phrase querying with combined indexes," *ACM Trans. Inf. Syst.*, vol. 22, no. 4, pp. 573–594, 2004.

[9] A. Doucet and H. Ahonen-Myka, "An efficient any language approach for the integration of phrases in document retrieval," *Lang. Resour. Eval.*, vol. 44, no. 1–2, pp. 159–180, 2010.

[10] I. H. Witten, A. Moffat, and T. C. Bell, *Managing gigabytes: compressing and indexing documents and images*. Morgan Kaufmann, 1999.

[11] D. Bahle, "Efficient Phrase Querying," *Sch. Comput. Sci. Inf. Technol. R. Melb. Inst. Technol.*, 2003.

[12] A. Fellinghaug, "Phrase searching in text indexes," no. June, p. 137, 2008.

[13] C. J. van Rijsbergen, "A theoretical basis for the use of co-occurrence data in information retrieval," *J. Doc.*, vol. 33, no. 2, pp. 106–119, 1977.

[14] R. Nallapati and J. Allan, "Capturing term dependencies using a language model based on sentence trees," in *Proceedings of the eleventh international conference on Information and knowledge management*, 2002, pp. 383–390.

[15] E. M. Keen, "The use of term position devices in ranked output experiments," *J. Doc.*, vol. 47, no. 1, pp. 1–22, 1991.

[16] W. B. Croft, H. R. Turtle, and D. D. Lewis, "The use of phrases and structured queries in information retrieval," in *Proceedings of the 14th annual international ACM SIGIR conference on Research and development in information retrieval*, 1991, pp. 32–45.

[17] D. Metzler and W. B. Croft, "A Markov random field model for term dependencies," in *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, 2005, pp. 472–479.

[18] E. K. F. Dang, R. W. P. Luk, and J. Allan, "A context-dependent relevance model," *J. Assoc. Inf. Sci. Technol.*, 2015.

[19] F. Song and W. B. Croft, "A general language model for information retrieval," in *Proceedings of the eighth international conference on Information and knowledge management*, 1999, pp. 316–321.

[20] J. Gao, J.-Y. Nie, G. Wu, and G. Cao, "Dependence language model for information retrieval," in *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, 2004, pp. 170–177.

[21] B. He, J. X. Huang, and X. Zhou, "Modeling term proximity for probabilistic information retrieval models," *Inf. Sci. (Ny).*, vol. 181, no. 14, pp. 3017–3031, 2011.

[22] Y. Rasolofo and J. Savoy, *Term proximity scoring for keyword-based retrieval systems*. Springer, 2003.

[23] C. Eickhoff, A. P. de Vries, and T. Hofmann, "Modelling Term Dependence with Copulas," in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2015, pp. 783–786.

[24] S. Büttcher, C. L. A. Clarke, and B. Lushman, "Term proximity scoring for ad-hoc retrieval on very large text collections," in *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, 2006, pp. 621–622.

[25] T. Tao and C. Zhai, "An exploration of proximity measures in information retrieval," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, 2007, pp. 295–302.

[26] J. Zhao and Y. Yun, "A proximity language model for information retrieval," in *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, 2009, pp. 291–298.

[27] J. Zhao, J. X. Huang, and B. He, "CRTER: using cross terms to enhance probabilistic information retrieval," in *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, 2011, pp. 155–164.

[28] J. Zhao, J. X. Huang, and Z. Ye, "Modeling term associations for probabilistic information retrieval," *ACM Trans. Inf. Syst.*, vol. 32, no. 2, p. 7, 2014.

[29] J. Miao, J. X. Huang, and Z. Ye, "Proximity-based rocchio's model for pseudo relevance," in *Proceedings of*

the 35th international ACM SIGIR conference on Research and development in information retrieval, 2012, pp. 535–544.

[30] C. L. A. Clarke, G. V Cormack, and E. A. Tudhope, "Relevance ranking for one to three term queries," *Inf. Process. Manag.*, vol. 36, no. 2, pp. 291–311, 2000.

[31] J. Klekota, F. P. Roth, and S. L. Schreiber, "Query Chem: a Google-powered web search combining text and chemical structures," *Bioinformatics*, vol. 22, no. 13, pp. 1670–1673, 2006.

[32] K. Sadakane and H. Imai, "Text Retrieval by using k-word Proximity Search," in *Database Applications in Non-Traditional Environments, 1999.(DANTE'99) Proceedings. 1999 International Symposium on*, 1999, pp. 183–188.

[33] X. Lu, A. Moffat, and J. S. Culpepper, "On the cost of extracting proximity features for term-dependency models," in *CIKM 2015*, 2015, pp. 293–302.

[34] A. Kulkarni and J. Callan, "Selective search: Efficient and effective search of large textual collections," *ACM Trans. Inf. Syst.*, vol. 33, no. 4, p. 17, 2015.

[35] R. Courant, *Differential and integral calculus*, vol. 2. John Wiley & Sons, 2011.

[36] S. E. Robertson and S. Walker, "Some for Simple Effective Approximations to the 2 – Poisson Model Probabilistic Weighted Retrieval The," in *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, 1994, vol. 1994, no. 1, pp. 232–241.

[37] H. Zaragoza, N. Craswell, M. J. Taylor, S. Saria, and S. E. Robertson, "Microsoft Cambridge at TREC 13: Web and Hard Tracks.," in *TREC*, 2004, vol. 4, p. 1.

[38] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*. John Wiley & Sons, 2012.

[39] S. Robertson and H. Zaragoza, *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc, 2009.

[40] J. Zhao and J. X. Huang, "An enhanced context-sensitive proximity model for probabilistic information retrieval," in *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, 2014, pp. 1131–1134.