Math.comput.sci. 2 (2008), 333–353

1661-8270/020333-21, *published online* December 5, 2008
DOI 10.1007/s11786-008-0057-3

Mathematics in Computer Science

Relevance Ranking and Hit Description in Math Search

Abdou S. Youssef

Abstract. As math becomes available in digital libraries and on the Web, math search has been receiving some research attention. To be effective and useful, math search systems must not only recognize math symbols and structures in queries and contents, but also present the search hits in a form that enables the user to identify quickly the truly relevant hits. To meet the latter requirement, the hits must be sorted according to domain-appropriate relevance criteria, and each hit ought to be accompanied with a query-relevant brief description, or summary, of its target.

In conventional text information retrieval systems, hits are ranked using relevance metrics that rely mostly on keyword frequencies and document sizes. Such metrics are inadequate in math search. Therefore, new relevance measures must be defined, which take into account math-specific factors. In this paper, new relevance metrics are defined for math search, methods for computing and implementing them are discussed, and comparative performance evaluation results are presented.

Query-relevant hit-summary generation is another factor that enables users to quickly determine the relevance of the presented hits. Hit titles accompanied by several leading sentences from the target document are often inadequate to convey to the user the relevant contents of the hit. This paper presents alternative query-relevant hit-summary generation methods, outlines implementation strategies, and presents performance evaluation results.

Mathematics Subject Classification (2000). 68N99.

Keywords. Fragmentation, hits, hit descriptions, math search, relevance metrics.

1. Introduction

Digital libraries of math, physical sciences, and engineering, contain many equations, graphs, tables, numerous embedded mathematical expressions, and text.

This work was done in part at the National Institute of Standards and Technology, USA, as part of the DLMF Project.

Therefore, users will need specialized, math-aware search systems to find and locate quickly the math information that is most relevant to their information needs. A number of search systems have been built, such as the NIST DLMF search system [15, 18, 19], the Design Science's Mathdex [14], and the formula search system MathWebSearch of Kohlhase et al. [8].

Math search systems must not only recognize math symbols and structures in queries and contents, but also present the search hits in a form that enables the user to identify quickly the truly relevant hits. To meet the latter requirement, the search system must sort the hits according to domain-appropriate relevance criteria, and provide with each hit a query-relevant summary of the hit target.

The standard relevance measures in text search, which rely mostly on keyword frequencies and document sizes, are inadequate in math search. This is because, among other things, the size of a math object (such as an equation) and term-frequency of a keyword inside the math object have no bearing on how important the object is, whereas the nature of the terms (e.g., standard function names and standard math operations) that appear in an object and the nature of the object itself (e.g., definition, theorem, and proof) carry considerably more relevance to the users of math search. Therefore, alternative relevance measures must be defined, which give more weight to certain types of information than to others, such as definitions, theorems, "standard" functions and operators, and frequently referenced items. In this paper, new non-interactive relevance metrics are defined for math search, methods for computing and implementing them are discussed, and performance evaluation results are presented. It must be noted that different users may want different things (e.g., some users may be more after definitions than theorems, while others are after theorems, and so on). An interactive user-interface-based mechanism for influencing relevance ranking can be taken into account, and many of the techniques presented in this paper can be adapted for that purpose, but this interactive aspect is left for future work.

Query-relevant hit-summary generation, or simply *hit packaging*, is another factor that enables users to quickly determine the relevance of the presented hits, and thus determine the most relevant hits. The hit title, accompanied by a few leading sentences from the target document, forms a fast and simple way for hit packaging. However, this often fails to convey to the user the true relevance of the target document. Since users should not be required to pursue many of the hits and read significant portions of their contents to locate the wanted information, automated generation of better hit summaries must be provided. This paper presents query-relevant hit-summary generation methods, outlines implementation strategies, and shows substantiating illustrations.

2. Background and related work

Three types of math search have received attention and/or have been built. The first is field based, widely deployed in several mathematics databases and content providers' web sites such as Zentralblatt's ZMATH [20] and MathDi [11],

the Jahrbuch Database [7], and AMS's MathSCiNet [13]. These search systems, however, are conventional text search, lacking awareness of math notation and structure. The second is formal-math search, such as the search systems developed and researched by Guidi et al. [5,6] and MoWGLI of the Helm project [16]. Formal-math search tools are in a basic form, and thus-far intended for the developers in the formal-math community. The third type is math-aware fine-grain search such as the DLMF search system [15,18,19], the Design Science's Mathdex Web search system [42], Kohlhase's search engine MathWebSearch [8], Mathematica search system (mathematica.com), the integral-table lookup tool TILU [4], and several Web-accessible rudimentary math term-occurrence search capabilities, such as Ask Dr. Math (http://mathforum.org/dr.math/). This type of math search is intended for general use by students, educators, researchers, and professionals, in mathematics, physical sciences, and engineering. It is this kind of search that requires further investigation for relevance ranking and hit packaging, which are the focus of this paper.

Relevance scoring has received much research attention in text search for over three decades [1–3,17]. Although several relevance metrics have been developed and studied, most are elaborations and variations of one central metric, often referred to as the *tf∗idf metric* (term frequency inverse document frequency). Essentially, this metric is predicated on the assumptions that (1) the higher the relative frequency of a query keyword in a hit document is, the more relevant the document is, and (2) the more frequent a term is in the whole database, the less important its occurrences are. One implication is that if two documents have the same number of occurrences of the keywords but one is a smaller document than the other, the smaller document ranks higher because its relative term-frequency (i.e., number of keyword occurrences divided by the document size) is larger.

Such traditional considerations are inadequate in math search. For example, assuming the desired hits are to be equations and definitions, a smaller-size equation (or definition) is not necessarily more relevant than a larger-size equation (or definition), and the frequency of occurrence of a term in the equation/definition is much less important than the mathematical significance of that term. To illustrate, suppose the user query is "sin". Among the many hits are those presented in Figure 1. The ranking exhibited in Figure 1 is the standard ranking used in text search systems. However, most math search users would wish a different ranking altogether. For example, a novice in trigonometry is more likely to be expecting a very basic definition of sin, that is, hit 28 should be the first hit; and for a more advanced trigonometry student, hits 23–27 should rank ahead of hits 1–22, and hits 24–27 are arguably more important than hit 23 and should rank ahead. Furthermore, it can be argued that a user looking for the value of sin at a specific angle would specify the angle in the query; therefore, for nearly all math search users, the standard methods of ranking used in text search is the wrong ranking for math search.

Even in the context of pure text search, the shortcomings of traditional relevance scores were recognized in Web search, especially by Google. It was realized

1. $\sin 0 = 0$
2. $\sin \pi = 0$

... The next 20 hits are the values of sin at 20 specific angles

23. $\sin 2x = 2 \sin x \cos x$
24. $\sin^2 x + \cos^2 x = 1$
25. $\cos nx + i \sin nx = (\cos x + i \sin x)^n$
26. $\tan x = \frac{\sin x}{\cos x}$
27. $\sin x = \frac{e^{ix} - e^{-ix}}{2i}$
28. $\sin x$ is the ratio of the length of side facing the angle x in a right triangle, to the length of the hypotenuse of the triangle.

FIGURE 1. tf∗idf ranking of hits for query "sin".

that the importance, and thus relevance, of a document/page depends more on who publishes it, how many links point to it, how many times it is visited, and such, than on the "uninformed" statistics of term frequencies and document sizes.

These same considerations can be utilized in math search, but after significant adaptations and specialization to math contents. For example, the number of times a particular math entity (e.g., equation) is referenced in a document/site can be a very telling indication of the relative importance of that entity. In addition to cross-reference statistics, domain-specific term weighting can be taken into account in relevance scoring, with great expected benefits. For example, if a query includes among its keywords the term "BesselI" and the variable name "x", then intuitively the first term is much weightier than the second term.

The relevance metrics used in the current generation of mostly experimental math search systems, including MathDex, MathWebSearch and DLMF search, have been identical to the ones used in conventional text search, that is, the tf∗idf metric. In this paper, alternative metrics are developed and shown to yield significantly improved results.

The other subject of focus in this paper, which has an equal bearing on helping users find relevant information fast, is hit-description generation. Hit-description generation, or hit packaging, has never been viewed as a major issue in text search, and has thus been done in a rather simple way. Prior to Web search, text search systems often reported each hit as a document title, sometimes accompanied by a few leading sentences in the hit's document. In Web search, such as in Google, the hit package consists of the page title of the hit, accompanied with 2–4 lines of sentences or sentence fragments that contain the keywords of the query, usually highlighted.

As math search is still in its early experimental phases, where more pressing issues have had to be addressed first, the same methods used in text search are used by necessity, until more specialized alternatives are found. In Mathdex [14], the Web page title and the first couple of lines of the Web page contents of the hit are displayed with the hit. As a significant enhancement, a special button is

added next to each hit, which when moused over, shows one equation or math expression that made the page match the query. In early experimental versions of DLMF search, two hit-packaging methods were used, depending on the nature of the hit. If the hit target has a small amount of contents, such as equations or even graphs and small-size tables, the entire target content is presented in the hit itself, providing immediacy and directness. If, on the other hand, the hit target is a section of a chapter, the hit description consists of the section title and the chapter title. Mathematica search, like Google, offers with each hit about 2 lines of sentence fragments that contain query keywords. Mathematica's hit packaging may be adequate for Mathematica contents, which tend to be short descriptions of functions or portions of code mixed with some text, but it will not meet the need for general-purpose math search.

Clearly, much more representative and query-relevant descriptions of hit targets should be generated per math-hit. The reason is that the user will be able to judge faster the value and relevance of the hit without having to pursue many hits and read long passages in them before the valuable and truly relevant information is found. Techniques generating such descriptions/summaries are presented in this paper.

3. Relevance ranking in math search

Before the new relevance metrics are introduced and related considerations discussed, it is instructive to look closely at the standard tf∗idf metric. For a query q and a hit-target document d in some presumed database DB, the tf∗idf relevance metric value is:

$$Relevance_q(d) = \sum_{query\ terms\ t \in q} tf(t,d) \times idf(t)$$

where

$$tf\,(t,d) = \sqrt{\frac{frequency(t,d)}{|d|}}\,,$$

$$idf\,(t) = \log \frac{|DB|}{number\ of\ documents\ containing\ t\ in\ DB}\,,$$

$$frequency\,(t,d) = \text{the number of times the term } t \text{ occurs in the document } d\,,$$

$$|d| = \text{the number of non-stopwords in document } d\,,$$

and

$$|DB| = \text{the number of documents in the database } DB\,.$$

Note that the first factor, $tf(t,d)$, represents the frequency of a term in the document, normalized by the document size, and that the second factor, $idf(t)$, represents the inverse of the number of documents containing the term relative to the total number of documents in the database. The square root and the log are meant to attenuate the contributions of those factors to various degrees.

A deeper look into the formula reveals that the first factor attempts to capture the importance (or weight) of the term t with respect to the document d (and thus the relevance of the document relative to the term t), while the second factor attempts to capture the weight of the term t with respect to the database as a whole.

The paper will preserve this paradigm of expressing the relevance of a document to a term as a function of the weight of the term vis-a-vis the document and the weight of the term vis-a-vis the database. What will change is the way of measuring each of those factors; $tf(t,d)$ will be replaced by a general term-document weight function, and $idf(t)$ will be replaced by a term weight function, and a math object weight function will be introduced (where a math object can be a full document or some small items such as an equation or even a sentence); all such weight functions will be elaborated later. Furthermore, since various aspects will influence those factors, and some aspects are absolutely more important than others, a multidimensional relevance metric, which is then a *relevance vector*, or an equivalent weighted sum of the latter's components, form alternative ways of measuring relevance and sorting the hits. Several static (i.e., query-independent) and dynamic (i.e., query-dependent) aspects for relevance computation will be taken into account, as explained next.

3.1. Static weight information

Many math terms have intrinsic importance due to what they stand for, and some terms have more intrinsic importance than others. For example, *special function* names (like "sin" and "BesselI") stand for much more than a moot variable name. Similarly, certain operators, such as integration ($\int$), exponentiation and division, are more important than variable names. This type of intrinsic importance of terms in themselves is called *categorical importance*; it is a primary determinant of the term-weight function $\mathbf{Weight}(t)$. This function can be defined as follows:

$$\mathbf{Weight}(t) = \mathbf{Quantify}\big(\mathbf{Type}(t)\big)\,,$$

where

- **Type** is a mapping that maps a term to a category based on some typology or taxonomy of terms from a term-importance perspective. For example, the term categories can be
  - "operator" (such as arithmetic or logic binary/unary operations and relational operations, like $+, -, *, /, =, <, >$, and functional operators like $\int$ and differentiation);
  - "special-function" (such as the Bessel functions, Jacobi functions, trigonometric functions, and so on); and
  - "regular" (for everything else, such as variable names, arbitrary function names, numerical constants, and so on).

  Note that this typology just presented is only one out of possibly many, and is by no means exhaustive or universal. Different system designers may have different taxonomies that will suit their applications and intended purposes.

- **Quantify** is a mapping that maps a term-type into a positive real number associated with that type, where the more important a type is, the larger its associated number is. For example, one can have **Quantify**(regular) = 1, **Quantify**(operator) = 2, and **Quantify**(special function) = 4, which is what we used in our testing and performance evaluation on the DLMF system. Again, for different taxonomies of terms, system designers can customize their **Quantify** mapping.

Much like terms, math objects (e.g., equations, graphs, tables, or full documents) have intrinsic importance irrespective of the query. Several aspects feed into that importance:

1. the type of the math object, such as equation, graph, table, theorem, proof, lemma, bibliographic item, notation item, and so on;
2. the categorical importance of the member terms and other constituent (i.e., subset) objects;
3. the number and possible types of cross-references made to the object by other objects in the database (or even on the Web). The types of cross-references are taxonomized in two ways. In the first taxonomy, a cross-reference can be *local* (i.e., internal) or *global* (i.e., external):
   - A *local cross-reference* is one where the referring object and the referred-to object belong to one and the same division of information, such as one chapter or one manuscript or one Website. For example, suppose the object is a theorem T in an article A, and the database DB is a collection of math articles. The references (such as "see theorem T" or "as shown in T") made to theorem T inside article A are taken to be the local cross-references for A. Note that the manner in which such references are made (and recognized) is not of concern in this paper (see Section 5). In practice, if the cross-references are explicitly and formally marked-up in the contents using special XML elements/attributes, then recognizing and counting the cross-references is relatively straightforward, but if the cross-references are natural-language references (something like "see Theorem 3.17"), then elaborate heuristic algorithms will be needed to mine them, but that is outside the scope of this paper.
   - A *global cross-reference* is one where the referring object and the referred-to object belong to two different divisions of information. Using the example above, the cross-references to theorem T from all the articles in the DB (excluding the article A containing T) are taken to be the global references to T. In a Web-wide application, the global references are the URLs pointing to the object from outside the Website of that object.

   The above taxonomy is fairly natural, and has been successfully applied by Google in its search engine. The next taxonomy is math-specific, and premised on the common practice in math authoring, which follows the

definition-theorem-proof paradigm, and on the expectation that many math searchers will often look for definitions and for propositions.

Accordingly, in this taxonomy, cross-references can be *definitional cross-references* or *propositional cross-references*:

- A reference from object A to object B is definitional if both of the following conditions are met:
  - Object B defines some mathematical term/concept c (as for example the definition of "sin")
  - Object A refers explicitly to object B as the object that defines c; for example, A could use "sin" and refer readers to object B to see the definition of "sin".
- A reference from object A to object B is propositional if both of the following conditions are met:
  - Object B states and/or proves some proposition P (where the term "proposition" is used in the broadest sense, so it encompasses theorems, lemmas, corollaries, "inline" substantiated or stipulated claims, etc.). For instance, object B could be/contain/prove the De Moivre Theorem.
  - Object A refers explicitly to object B as the primary location of proposition P.

Clearly, although the taxonomies presented here are natural for math contents, they are not the only types of references. Different system designers may consider other taxonomies of references that best fit their purposes. The techniques presented here for using reference information to better capture relevance in math search can be easily extended to other taxonomies of references.

Based on the taxonomies presented above, the weight function $\mathbf{Weight}(mo)$, of a math object mo, can be defined as follows:

$$\mathbf{Weight}(mo) = \mathbf{Combine}\Big(\mathbf{Quantify}\big(\mathbf{Type}(mo)\big), \mathbf{TW}(mo), \mathbf{CR}(mo)\Big)\,,$$

where

- **Type** and **Quantify** are like those for terms except here the categories are those of math objects;
- $\mathbf{TW}(mo)$ captures the weight of the terms that make up the math object mo, such as

$$\mathbf{TW}(mo) = \sum_{t \in mo} Weight(t)\,,$$

  where $\mathbf{Weight}(t) = \mathbf{Quantify}(\mathbf{Type}(t))$, as defined above. For example, if mo is "`int sin^2 x dx`" and if we assume $\mathbf{Quantify}(\text{regular}) = 1, \mathbf{Quantify}(\text{operator}) = 2$, and $\mathbf{Quantify}(\text{special function}) = 4$, then $\mathbf{TW}(mo) = \mathbf{Weight}(int) + \mathbf{Weight}(sin) + \mathbf{Weight}(\hat{}) + \mathbf{Weight}(2) + \mathbf{Weight}(x) + \mathbf{Weight}(dx) = 2 + 4 + 2 + 1 + 1 + 1 = 11$
- $\mathbf{CR}(mo)$ captures and quantifies the statistics of cross-reference pointers pointing to mo. To illustrate, suppose mo is a math page that happens to be a

section S of a chapter (or article) C and that contains one definition D and one proposition P, and suppose the database DB is a collection of chapters or articles. Let

- N_{ld} = the number of local definitional references to S, that is, the number of references made inside the chapter/article C to definition D (those references are made inside C but outside S). Assume for illustration that $N_{ld} = 5$ for this example.
- N_{lp} = the number of local propositional references to S, that is, the number of references made inside the chapter/article C to proposition P (those references are made inside C but outside S). Assume for illustration that $N_{lp} = 7$ for this example.
- N_{gd} = the number of global definitional references to S, that is, the number of references made outside the chapter/article C to definition D (those references are made inside DB but outside C). Assume for illustration that $N_{gd} = 20$ for this example.
- N_{gp} = the number of global propositional references to S, that is, the number of references made outside the chapter/article C to proposition P (those references are made inside DB but outside C). Assume for illustration that $N_{gp} = 100$ for this example.

Then, $\mathbf{CR}(mo)$ can be defined as a weighted sum of N_{ld}, N_{lp}, N_{gd}, and N_{gp}, where the weights reflect the relative importance of local vs. global and of definitional vs. propositional. For instance, if a designer decides that for his/her application and intended users the global references are twice as important to the notion of relevance as local ones, and definitions are three times as important to relevance as propositions, then

$$\mathbf{CR}(mo) = 3N_{ld} + N_{lp} + 2 * 3N_{gd} + 2N_{gp}\,.$$

For the concrete figures assumed in this example, $\mathbf{CR}(mo) = 3 * 5 + 7 + 2 * 3 * 20 + 2 * 100 = 342$.

- **Combine** combines **Quantify**(**Type**(mo)), $\mathbf{TW}(mo)$, and $\mathbf{CR}(mo)$ into either a scalar or a vector value, as explained next.

Combining several factors of various degrees of importance into a single ranking-metric can be done in two ways. The first way is to map the vector $V = (x_1, x_2, \ldots, x_n)$ of factors into a scalar value S, such as by adding or multiplying the components, where every component x_i is weighted by some scalar w_i to reflect its relative importance. That is, the scalar value formula can be $S = \prod_{i=1}^{n} x_i^{w_i}$ or $S = \sum_{i=1}^{n} w_i x_i$ (which are equivalent via the log isomorphism), among other modeling possibilities. With scalar metrics, the ranking is done by straightforward sorting of objects according to their scalar ranking metric.

The other way of combining factors is to map the vector of factors $V = (x_1, x_2, \ldots, x_n)$ into another, carefully ordered vector of factors $V' = (y_1, y_2, \ldots, y_m)$, resulting in a *vector ranking metric* of the same as or smaller dimensionality than that of the original vector V. The first component y_1 corresponds to the factor of highest weight, y_2 corresponds to the factor of the second

highest weight, and so on. The ranking of objects is then done by lexicographic sorting of the vector metric values of the objects.

Vector ranking metrics have several advantages. First, there is no need to concern oneself about how the weights of the various factors should be combined into a scalar metric formula. Second, vector metrics and lexicographic sorting strictly enforce the policy that a most important factor should not be overwhelmed by a combination of less important factors. To see this more clearly, assume that the metric is a two-component vector (x, y), where x is binary value ($x = 1$ if the hit is a definition of at least one of the keywords in the query, and $x = 0$ otherwise), and y is the tf∗idf value of the hit. The lexicographic order is simply defined as follows: $(x_1, y_1) < (x_2, y_2)$ if $x_1 < x_2$ or ($x_1 = x_2$ and $y_1 < y_2$). It can now be seen that all the hits that happen to be definitions of keywords in the query will have 1 in the first component of their relevance vector, whereas all the other hits will have 0 in the first component of the relevance vector; as a result, the first group of (defining) hits will always rank higher then the second group of (non-defining) hits no matter what the second components of the relevance vectors are. If this is what the users of a database want, then such an approach is perfect. If, on the other hand, the definitional nature of hits should not assume absolute primacy over all values of tf∗idf value, then the vector approach must not be used; instead, scalar metrics derived from a weighted sum of all the vector components would be preferable.

The choice between the two approaches is left to the search system designer and/or the preferences of the specific users. Indeed, the choice between the two approaches, and the specific ordering or weights of the factors, can be made to vary dynamically based on the explicit choices of the user, or based on the automated deduction of the user's preferences through monitoring of his/her search behavior. Accordingly, the rest of the paper focuses primarily on the factors. Although the factors worth considering are listed in a vector form, the ordering of the components, or the weighted combining of the factors into a scalar value, will be left unspecified except when stated otherwise.

3.2. Dynamic weight information

Dynamic weight information relates to the weight of math object mo relative to the terms t of a query q. That information is incorporated into the function $\mathbf{Weight}(t, mo)$ or generally $\mathbf{Weight}(q, mo)$.

One possible definition of $\mathbf{Weight}(q, mo)$ is the same as $\mathbf{TW}(mo)$ except that the terms will be limited to those that are in the intersection of the object and the query. An elaboration on this definition would be to factor in the number $ND(q, mo)$ of the query keywords that are defined in the object mo. Therefore, assuming that the types of terms are $\{T_1, T_2, \ldots, T_k\}$,

$$\mathbf{Weight}(q, mo) = \big(ND(q, mo), N_1(q, mo), N_2(q, mo), \ldots, N_k(q, mo)\big)$$

where

$$N_i(q, mo) = |\{t \mid \mathbf{Type}(t) = T_i \text{ and } t \in mo \text{ and } t \text{ is a keyword of the query } q\}|\,.$$

3.3. Overall relevance metric

Based on the preceding analysis and discussions, the overall relevance metric is a vector made up of the components of the **Weight**(q, mo) vector and on the **Weight**(mo), ordered or weighted according to the system designer's assigned relative importance of each component.

One possible ordering, from the highest importance to the lowest importance, is:

- $ND(q, mo)$, which is the number of query keywords defined in the object mo,
- $N_i(q, mo)$ for the top one or two most important term types, where $N_i(q, mo)$ is the number of terms of type T_i that occur in both the query and the object,
- **CR**(mo), which captures the global/local definitional/propositional cross-reference statistics,
- the remaining $N_i(q, mo)$s
- **TW**(mo), which is the term-weight of the object mo, expressed either as a vector or a weighted sum,
- (optional) **Quantify**(**Type**(mo)), reflecting preferences for certain document/ object types over others,
- tf$*$idf(q, mo), as a final tie-breaker.

3.4. Speed performance evaluation of hit ranking

Hit ranking involves two major steps. The first is the computation of the relevance of each hit, and the second is the sorting of the hits according to their relevance values. The second step is rather straightforward and can be done using any fast sorting algorithm (such as mergesort or heapsort), which takes $O(n \log n)$ time, where n is the number of hits. The first step, on the other hand, depends on the specific relevance metric used. The relevance factors that are considered in this paper are computed at indexing time, that is, offline. Therefore, their computation time does not impact the query search time.

The relevance metrics discussed in this paper have been implemented and tested on the DLMF testbed, using a benchmark of queries that we have collected (some of those queries are shown in Table 1). Several queries with a range of numbers of hits were tested to measure the overhead of relevance ranking. A sample of the results is presented in Table 2. The table shows the queries, the number of hits per query, the search time for identifying the hits but without ranking, and the time to perform the relevance computation and relevance ranking of the hits. As can be seen, the relevance computation and ranking time is usually higher than the search time, and, naturally, it is higher for larger numbers of hits. Nevertheless, for a standalone database of the size range of DLMF (i.e., about 1000 pages of contents containing over ten thousand equations), the number of hits will usually be in the tens, hundreds, or at most in the thousands, and the relevance ranking overhead ranges from a tenth of second to at most a second, which is quite acceptable.

For Web search relevance ranking, where the number of hits could conceivably be in the hundreds of thousands or even millions, the relevance ranking time will be significantly higher. However, the overhead can be managed down to practical ranges. One possibility is to do a two-stage ranking. In the first stage, a coarse relevance metric is applied, which takes into account a carefully selected small subset of the relevance criteria when computing relevance, and instead of sorting all the hits, find the top 100 (or so) hits. In the second stage, a full-fledged relevance evaluation and sorting of those 100 hits is done and the hits are presented to the user in hit-pages, about 10 hits per page. Since the truly relevant hits are very likely to be in the top 100, and most users rarely search down beyond that level, this approach will often be sufficient. In the rare cases where a user wishes to see the hits below rank 100, the 2nd stage is repeated on the next 100 hits, and so on.

It is left to future work to address the important question of determining which subset of relevance criteria makes a good coarse-grained relevance metric to be used in the first stage of the 2-stage Web search relevance ranking process.

3.5. Outcome performance evaluation of hit ranking

Outcome performance evaluation of relevance ranking is extremely subjective. A thorough evaluation of this sort will be left to future work, where a statistically significant number of users and a benchmark of queries are identified and used, and a metric of user satisfaction is decided upon and utilized in the collection of user assessments of the search system, including the relevance ranking and the hit-description generation which is discussed in the next section.

For now, suffice it to say that based on the expectations that definitions will be sought after more often by more users, and based on the valuation scheme where the definitions/equations/plots that are cross-referenced more often are of more weight, the outcome is far superior to the default tf∗idf relevance ranking approach. Hundreds of queries were tested. In each and every case, definitions and notations of the query keywords ranked on top, and items of higher cross-reference values ranked higher. This is much better than ranking under the tf∗idf relevance model, where such hits were "buried" in the second, third, or fourth page of hits.

Before closing this section, though, it should be noted that the cross-reference information factors led to some curious and possibly confusing rankings. For example, suppose that a (novice) user is interested in the definition of sin and in the standard trigonometric equations involving sin. It is natural for a user to submit the query "sin". Now suppose that a document D contains the equation

$$sin(\nu\pi)H_\nu^{(2)}(ze^{m\pi i}) = e^{\nu\pi i}\sin(m\nu\pi)H_\nu^{(1)}(z) + \sin\big((m+1)\nu\pi\big)H_\nu^{(2)}(z)\,,$$

and assume that equation is heavily cross-referenced in the database. Then, document D will not only match (because it has sin), but, due to the contributions of the cross-reference data to the relevance metric, will probably rank higher than the documents that the user wants. If, furthermore, the database contains many documents that have very advanced equations involving sin and that are highly cross-referenced, then the wanted documents will rank so low that the user may

TABLE 1. A sample of queries in the test benchmark.

Queries	Comments
`Ai^2`	
`int sin`	
`eulerBeta`	
`sin^2`	
`jacobisn OR Si`	
`eulerGamma`	
`cos`	
`sin`	
`Ai^2+Bi^2`	
`Ai'=`	the derivative of Airy function Ai
`sin^2+cos^2`	
`besselJ_nu`	
`int_0^x e^(t^2)`	
`int_0^(x or z) e^(?^2)`	the "?" is a wildcard standing for any arbitrary single character
`sqrt pi`	
`(d w)/(d z)`	for $\frac{dw}{dz}$
`int (d z)/z`	
`n choose k`	
`/Gamma`	expressions where Γ is in the denominator
`Gamma'/Gamma`	searching for $\frac{\Gamma'}{\Gamma}$
`function^2+function^2`	searching for a pattern of the sum of the squares of two functions
`(function^2 + function^2)^(1/2)`	
`function^2 (+ or -) function^2`	
`trigonometric^2+trigonometric^2`	
`jacobi^2+jacobi^2`	searching for a pattern of the sum of the squares of two Jacobi functions
`kelvin^2+kelvin^2`	
`function^3 +4 function`	
`_2 F_1`	hypergeometric function ${}_2F_1$
`_? F_?`	hypergeometric function ${}_pF_q$ for whatever p and q
`lim ?-> 0 (sin/?)`	$\lim_{x->0} \frac{sinx}{x}$, where x is an arbitrary symbol
`besselI =eulergamma`	to find equations where BesselI is expressed in terms of Euler Gamma

continued on next page

TABLE 1. A sample of queries in the test benchmark.

Queries	Comments
`pi =`	for the value of π
`e=`	
`gamma=`	for the value of the Euler constant γ
`Gamma(1/2)=`	for the value of $\Gamma(\frac{1}{2})$
`e^(i pi)=`	for the value of $e^{i\pi}$
`e^z= 1! 2!`	for the series expansion of e^z

TABLE 2. Speed performance of search and hit ranking. (All time measurements are in milliseconds.)

Query	Number of Hits	Search Time	Hit-ranking Time
Ai^2	7	16	15
$\int \sin$	19	15	16
eulerBeta	28	15	32
$\sin^2$	80	15	78
jacobisn OR Si	94	31	63
eulerGamma	653	31	344
cos	666	16	297
sin	707	15	329
z	2499	16	828

give up looking for them in the hit list. Therefore, the cross-reference aspect of ranking needs further research and refinements (such as obtaining preference information from the user like definitions, elementary equalities, and so on), which will be the subject of future work.

4. Hit-description generation in math search

As mentioned earlier, the rather simple way of putting together the hit-title and a few leading sentences of the hit-target fails to convey to the user why a document matched and whether the matching parts are indeed relevant. It will be much better to the user if those parts are extracted and provided with the hit so the user can quickly determine whether or not a hit is worth pursuing. Furthermore, if those parts are determined carefully, they may often be all that the user needs from a document, thus saving him/her extra efforts. This section will provide new methods for determining query-relevant excerpts from math documents. Before starting, it must be noted that if the hit targets are small math objects (e.g., equations or graphs), then such objects should be displayed directly with the hits, achieving maximum immediacy at the same screen space size as typical hit descriptions. Therefore, for the rest of this section, it will be assumed that the hit

targets are relatively sizable objects that cannot be conveniently displayed along with hits, such as sections, chapters, articles, Web pages, and so on.

The approach to hit-description generation consists of several tasks. Some tasks must be carried out at indexing time, while other tasks must be at search time. One major goal is to minimize the computations that must be done at search time so that query turn-around time is short enough for users.

4.1. Index-time tasks for hit-description generation

The following are the tasks to be performed at indexing time to implement and speed up hit-description generation:

1. Fragment, at indexing time, each document in the database into very small units of information, where a unit can be (1) an equation, (2) a sentence, which may contain inline math expressions, (3) a phrase or a non-syntactic portion of a sentence, (4) a graph, (5) a fragment of a table (in the case where tables are large), (6) a title of a chapter/section/subsection, (7) a notational item, (8) a bibliographic entry, and so on.
2. Each fragment is then turned into a mini-document with its own ID. The mini-document contains, besides its contents, several fields of information that will facilitate and speed up the hit-description generation at search time. One field is the ID of the document of which the mini-document is a fragment. Other fields contain static information that will be used to measure the relevance of the mini-document at search time, such as the number and weights of the terms in the fragment, and the numbers of cross-references to that fragment.
3. Index the fragments (i.e., mini-documents) of all the documents, and store the index information in a separate index structure, termed the *fragment index*. That index is different from the index for the documents. Note that fragment contents and the fields in the fragment are stored verbatim in the fragment index. The reason for this will be explained below.

4.2. Search-time tasks for hit-description generation

At search time, when the IDs of the hits that match a query have been determined, the hits are presented one page at a time (typically 10 hits per page). For each page of hits, the descriptions of those hits are generated as follows:

1. For each hit in a hit-page, identify the ID of the target document, and formulate a *derivative query* made up of the conjunction of the original query and the ID of the target document.
2. Submit the query for search against the *fragment index*. This results in several "sub-hits", each of which is a fragment of the hit target document.
3. If no sub-hits are returned, relax the derivative query so that the keywords in the original query are combined into a disjunctive query (i.e., an OR-query of the keywords), and repeat step 2, resulting this time with one or more sub-hits.

4. The sub-hits are then relevance-ranked using the relevance metrics described in the previous section. Note that the relevance score of the fragments can be computed fast because much of the weight information (i.e., the static weight information) is stored in the fragment index, and thus need not be computed from scratch.
5. A few top-scoring sub-hits (i.e., fragments) are selected, retrieved from the fragment index, and combined (in document-order) into a description/summary that is presented along with the hit title in the hits page.

To illustrate the workings of this hit-packaging method, consider the query "`int sin^2`", and assume that one hit (say hit No. 9) is a page titled "Integrals of Trigonometric Functions" that contains both some explanatory text (with no math in them) and many integrals involving trigonometric functions. In particular, assume that the only fragments containing "sin" in the hit are the following four equations, in that order, but not necessarily consecutive:

1. $\int(\sin^2 x + \cos x)dx = \sin x + \frac{x}{2} - \frac{1}{4}\sin 2x$
2. $\int(\sin^2 x + \cos x - \sin x)dx = \sin x + \cos x + \frac{x}{2} - \frac{1}{4}\sin 2x$
3. $\int_0^{\pi} \sin^2 x dx = \frac{\pi}{2}$
4. $\int \sin^2 x dx = \frac{x}{2} - \frac{1}{4}\sin 2x$

The first step of the method identifies the unique ID of that hit (say the ID is *paper*. 217). The second step forms the query "`int sin^2` AND paper. 217" and submits it to the search engine to search against fragments; the search returns 4 hits (referred to as sub-hits because they are fragments of the document). The 4 sub-hits are the 4 equations listed above. Since Step 2 resulted in sub-hits, Step 3 does not execute. Step 4 ranks the 4 sub-hits, say in the following order: 4, 1, 2, 3. Assuming that the top 3 ranking fragments are to be picked, Step 5 will take equations 4, 1 and 2, and will then re-order them in document order, that is, (1, 2, 4), and finally present the hit as:

9- Integrals of Trigonometric Functions
$\ldots \int(\sin^2 x + \cos x)dx = \sin x + \frac{x}{2} - \frac{1}{4}\sin 2x$
$\ldots \int(\sin^2 x + \cos x - \sin x)dx = \sin x + \cos x + \frac{x}{2} - \frac{1}{4}\sin 2x$
$\ldots \int \sin^2 x dx = \frac{x}{2} - \frac{1}{4}\sin 2x.$

Several remarks are in order. First, this hit-description method requires no file IO since all the fragment contents are stored in the fragment index, which is a file that remains open as long as the search system is running. This greatly speeds up the hit-description generation process. Second, the identification of the relevant excerpts (i.e., fragments) is rather fast and straightforward: it is a search-within-search process. Third, the relevance ranking of the matching fragments is also fast since the static weight statistics are computed and stored at indexing time, thus reducing the computation time for obtaining the relevance vectors of the fragments. Last, hit-description generation requires considerably more disk space to store the fragment index, which is much larger than the document index because the actual fragments are stored in the fragment index. However, since disk space is very inexpensive, the cost overhead is not a serious disadvantage.

TABLE 3. Speed performance of hit-description generation. (All time measurements are in milliseconds, and a page has 10 hits).

Query	**Hit-packaging Time per Hit**
Ai^2	26
$\int \sin$	10.11
eulerBeta	7
$\sin^2$	19.42
jacobisn OR Si	4.97
eulerGamma	6.33
cos	4.28
sin	4.16
z	5.32

4.3. Speed performance evaluation of hit-description generation

The same performance evaluation was done for hit-description generation as for relevance ranking. A sample of the results is shown in Table 3. The table shows the queries and the time it takes to derive the description of a single hit; the time to derive the descriptions for a 10-hit page takes 10 times as much. As can be seen, the time for generating the descriptions for the hits in one 10-hit page ranges from a few to less than 200 milliseconds.

It is important to note that based on the two Tables 2 and 3, the total wait time for a query to be processed and searched, plus the time to relevance-rank all the hits, plus the time to generate the hit-descriptions per 10-hit page, is about one second or less, making quite feasible the whole approach of math searching, relevance ranking, and hit packaging.

4.4. Outcome performance evaluation of hit-description generation

The outcome performance is subjective to some extent. Nevertheless, extensive testing was done on the DLMF testbed on over 100 queries, and the hit-descriptions were examined closely. For each hit, up to 3 top-ranking fragments were identified and presented as the description, and were examined manually, and compared to the rest of the document. In each case, the 3 top-ranking fragments were found to be truly the most query-relevant and representative of the hit-document. For example, for the query "sin", Figure 2 shows the 3 top-ranking hits for that query, along with their descriptions. Observe that the top-ranking hit is the one where $\sin z$ is defined.

Of course, a thorough subjective evaluation involving a large number of users and a carefully selected benchmarks of queries will have to be conducted in the future.

1–10 of 212 matching pages for query "sin":

1. 4.14. Definitions and Periodicity
$\ldots \sin z = \frac{e^{iz}-e^{-iz}}{2i}$,
$\ldots e^{\pm iz} = \cos z \pm i \sin z$,
$\ldots \tan z = \frac{\sin z}{\cos z}$,

2. 17.3. q-Elementary and q-Special Functions
$\ldots \sin_q(x) = \frac{1}{2i}(e_q(ix) - e_q(-ix)) = \sum_{n=0}^{\infty} \frac{(1-q)^{2n+1}(-1)^n x^{2n+1}}{(q;q)_{2n+1}}$,

3. 4.42. Solution of Triangles
$\ldots \frac{a}{\sin A} = \frac{b}{\sin B} = \frac{c}{\sin C}$
$\ldots \frac{\sin A}{\sin a} = \frac{\sin B}{\sin b} = \frac{\sin C}{\sin c}$
$\ldots \sin a \cos B = \cos b \sin c - \sin b \cos c \cos A$

FIGURE 2. The first three hits and their summaries, for query "sin".

5. Relation to Mathdex, MathWebSearch, and DLMF search

Much of what was presented in this paper relates directly to the three math search systems that have been recently developed: MathDex, MathWebSearch, and DLMF search. This section will discuss the state of the art of these systems with respect to ranking and hit summarization, and address the applicability of the presented techniques to these three systems.

In relevance ranking, MathDex, MathWebSearch, and the earlier experimental version of DLMF search, all used the standard tf∗idf relevance metric. The designers of all the three systems have recognized the inadequacy of that metric for math search. The work presented in this paper is part of solving the math relevance ranking problem in general, with specific implementations tailored for the DLMF in particular. No published work has appeared on addressing relevance ranking in MathDex or MathWebSearch, but the designers of the latter system plan to study this matter in the context of math-on-the-Web formula search.

The weight-based approach presented here applies to both MathDex and MathWebSearch. The major challenge in applying it, however, is in gathering the necessary statistics from Web-published manuscripts. In a highly controlled database like the DLMF, the designers have write-access to the contents, and can thus add, algorithmically or manually, necessary markup to facilitate the gathering of data that would factor into computing relevance metrics. But for general Web pages of math contents, no such facility exists, and the designers of search engines have to develop heuristics to mine the contents for relevance information. While cross-reference information can be easily identified (because of the markup nature of references in XML, XHTML, and even HTML), term weights and object weights, which depend on knowing something about the semantics of terms and

objects, will be highly challenging to compute, and will require significant natural language processing and mathematical writing analysis algorithms, before the relevance ranking approach of this paper can be fully utilized.

With regard to hit summarization, the MathWebSearch is for formula search, so the hits are presented as the matching formulas themselves, and thus no hit-summarization is needed. The MathDex has a hit summarization sub-system, which displays the first sentence (or two) of the hit document, followed by one equation that caused the document to match. This was a step in the right direction, especially the inclusion of a matching equation. Nevertheless, the leading sentences are not always relevant to the query; the approach presented in this paper addresses the situation by capturing only relevant fragments to make up the summary.

As in the case of relevance ranking, the presented hit-summarization approach works especially well in controlled database environments like the DLMF. Fortunately, the fragmentation component and the indexing of fragments are easily applicable to Web documents since the various types of fragments (e.g., sentences, equations, graphs, and tables or table fragments) are quite recognizable in nearly all math authoring formats in use, such as LaTeX, XHTML/MathML, HTML, and even MS Word. The search-time assembling of matching fragments into summaries is also easily applicable in MathDex (and similar math Web search systems). The only challenge is in using weight-based ranking of the fragments, as explained above. But for identifying relevant fragments, the tf∗idf relevance metric is likely to give better summaries than no ranking at all, and better than using leading sentences of documents as summaries.

6. Conclusions

In this paper, new relevance ranking metrics and hit-description generation techniques were presented and analyzed, and their performance was evaluated. It was found that the new relevance metrics are superior to the conventional tf∗idf metric, and the new hit-descriptions are more query-relevant and representative of the hit targets than conventional methods of providing the title and some leading sentences of the target document. Furthermore, it was determined that the system response time was about one second or less, which attests to the feasibility of the new approaches working collectively as a system.

Future research will focus on subjective evaluation of the new techniques, with a cross-section of users, using standard testbeds and query benchmarks that the research community will hopefully generate and agree upon. Refinements and extensions of the techniques will undoubtedly have to be carried out as a result of the subjective evaluation and the users' feedback.

The most challenging and arguably most useful future direction in this area of research is to develop mining techniques for classifying math terms and math objects, as a first step toward assessing their weights, which would be incorporated

into the relevance metrics. This is likely to be a sub-part of a larger effort of *math data mining* research that would heuristically determine the semantics (along with the disambiguation that will be needed) of mathematical writing that involves little or no markup, which is the case in all the legacy math literature.

Acknowledgements

Many thanks to Bruce Miller, Daniel Lozier, and Ronald Boisvert for valuable discussions and input into this work.

References

[1] R. A. Baeza-Yates and B. Riberto-Neto, Eds. (1999) *Modern Information Retrieval*. Addison Wesley.

[2] A. Bookstein, "Relevance", Journal of the American Society of Information Science, 30(5), 1979, pp. 269–273.

[3] C. Botev and J. Shanmugasundaram, "Context-Sensitive Keyword Search and Ranking for XML", WebDB, 2005.

[4] T. H. Einwohner and R. Fateman, "Searching Techniques for Integral Tables", International symposium on Symbolic and algebraic computation, ACM, 1995. (http://torte.cs.berkeley.edu:8010/tilu).

[5] F. Guidi, *Searching and Retrieving in Content-based Repositories of Formal Mathematical Knowledge*. Ph.D. Thesis in Computer Science, University of Bologna, March 2003. Technical report UBLCS 2003-06.

[6] F. Guidi and I. Schena, "A Query Language for a Metadata Framework about Mathematical Resources", The 2nd International Conf. Mathematical Knowledge Management, Bertinoro, Italy, Feb. 2003.

[7] Jahrbuch Database. http://www.emis.de/MATH/JFM/JFM.html.

[8] M. Kohlhase and I. Sucan, "A Search Engine for Mathematical Formulae", 8th International Conference of Artificial Intelligence and Symbolic Computation, pp. 241–253, China, September 2006.

[9] D. W. Lozier, "The DLMF Project: A New Initiative in Classical Special Functions", International Workshop on Special Functions – Asymptotics, Harmonic Analysis and Mathematical Physics. Hong Kong, June 21–25, 1999.

[10] D. W. Lozier, B. R. Miller, and B. V. Saunders, "Design of a Digital Mathematical Library for Science, Technology and Education", Proceedings of the IEEE Forum on Research and Technology Advances in Digital Libraries; IEEE ADL '99, Baltimore, Maryland, May 1999.

[11] MathDi, http://www.emis.de/MATH/DI/, Mathematics Didactics Database.

[12] Mathematica, http://www.mathematica.com.

[13] MathSciNet, http://www.ams.org/mathscinet, American Mathematical Society (AMS).

[14] R. Miner and R. Munavalli, "An Approach to Mathematical Search Through Query Formulation and Data Normalization", Calculemus/MKM 2007, p. 342–355, Hagenberg, Austria, June 27–30, 2007.

[15] B. Miller and A. Youssef, "Technical Aspects of the Digital Library of Mathematical Functions", Annals of Mathematics and Artificial Intelligence, **Vol. 38** (2003) 121–136.

[16] MoWGLI, Mathematics on the Web: Get It by Logics and Interfaces. http://mowgli.cs.unibo.it/.

[17] G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*. McGraw Hill, New York (1993).

[18] A. Youssef, "Information Search And Retrieval of Mathematical Contents: Issues And Methods", The proceedings of the ISCA 14th International Conference on Intelligent and Adaptive Systems and Software Engineering (IASSE-2005), July 20–22, 2005, Toronto, Canada.

[19] A. Youssef, "Roles of Math Search in Mathematics. The 5th International Conference on Mathematical Knowledge Management", Wokingham, UK, August 11–12, 2006, pages 2–16.

[20] Zentralblatt MATH database at European Mathematical Information Service (EMIS). http://www.emis.de/ZMATH/.

Abdou S. Youssef
Department of Computer Science
The George Washington University
Washington DC, 20052
USA
e-mail: ayoussef@gwu.edu

Received: December 15, 2007.
Revised: March 2, 2008.
Accepted: June 12, 2008.