Methods of Relevance Ranking and Hit-Content Generation in Math Search*,**

Abdou S. Youssef

Department of Computer Science
The George Washington University
Washington DC, 20052 USA

Abstract. To be effective and useful, math search systems must not only maximize precision and recall, but also present the query hits in a form that makes it easy for the user to identify quickly the truly relevant hits. To meet that 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, turned out to be inadequate in math search. Therefore, alternative relevance measures must be defined, which give more weight to certain types of information than to others and take into account cross-reference statistics. In this paper, new, multi-dimensional 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. Although the hit title accompanied by a few leading sentences from the target document is simple to produce, this often fails to convey to the user the document's relevant excerpts. This shifts the burden onto the user to pursue many of the hits, and read significant portions of their target documents, to finally locate the wanted documents. Clearly, this task is too time-consuming and should be largely automated. This paper presents query-relevant hit-summary generation methods, outlines implementation strategies, and presents performance evaluation results.

1 Introduction

Digital math libraries consist mostly of equations, graphs, tables, numerous embedded mathematical expressions, and text. Clearly, users will need specialized search systems to find and locate quickly the math information that is most relevant to their needs. A number of search systems have been built and are undergoing further enhancements, such as the NIST DLMF search system [13,17,18] and the Design Science's Mathdex [10].

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

** This work was supported in part by the National Science Foundation (NSF), USA, under Grant No. 0208818.

M. Kauers et al. (Eds.): MKM/Calculemus 2007, LNAI 4573, pp. 393–406, 2007.

For enhanced utility and user-satisfaction, math search systems must not only maximize precision and recall, but also present the query hits in a form that makes it easy for the user to identify quickly the truly relevant hits. To meet that 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, turned out to be inadequate in 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, multi-dimensional 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, 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. Although the hit title, possibly accompanied by a few leading sentences from the target document, forms a fast and simple way for hit packaging, it often fails to convey to the user the document's relevant excerpts. This shifts the burden onto the user to pursue many of the hits, and read significant portions of their target documents, to finally locate the wanted documents. Clearly, this task is too time-consuming, and should be done by the software on behalf of the user. 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 systems have received attention and/or have been built. The first is field-based search systems, which are now widely deployed in several mathematics databases and by many mathematical content providers, such as Zentralblatt's ZMATH and MathDi [19,9], the Jahrbuch Database [6], AMS's MathSCiNet [1], and various professional mathematical socities. Such systems are intended for conventional library search, and are outside the scope of this paper. The second is formal-math search, such as the search systems developed and researched by Guidi et al [4,5], MoWGLI of the Helm project [14], and MIZAR [2]. Formal-math search systems are highly specialized and usually intended for advanced mathematicians, and are thus outside the scope of the paper.

The third type of math-search is math-aware fine-grain search such as the DLMF search system [13,17,18], the Design Science's Mathdex Web search system [10,11], and Mathematica search system [12]. This type of math search is indended 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 [16,15]. 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 highly inadequate in math search. For example, if the hit targets are equations, a smaller-size equation is not necessarily more relevant or more important to the user. Also, in math, the frequency of occurrence of a term is much less important than the mathematical significance of that term. Finally, the importance of a term is context-dependent, especially in math, as for example in what part of a math structure the term lies, and what other terms the term co-occurs with.

The shortcomings of traditional relevance scores were recognized in Web search, especially by Google. It was realized 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 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 are primarily 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 better 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 of Design Science, 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 is somewhat more advanced in providing hit descriptions. Like Google, Mathematica 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 be sufficient 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 presnted 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 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:

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

where

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

and

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

($|d|$ is the number of terms in document d, and $|DB|$ is the number of documents in the database.)

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 in terms of 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 $\mathbf{Weight}(t,d)$, and $idf(t)$ will be replaced by a term weight function $\mathbf{Weight}(t)$, and a math object weight function $\mathbf{Weight}(mo)$ 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, it will be determined that a multidimensional relevance metric, which is then a *relevance vector*, is a more apt way of measuring relevance and thus of sorting the hits.

As argued earlier, mere frequency and size statistics do not fully capture the importance and relevance of documents. Rather, several other static (i.e., query-independent) and dynamic (i.e., query-dependent) aspects have to be taken into account when computing $\mathbf{Weight}(t,d)$, $\mathbf{Weight}(t)$ and $\mathbf{Weight}(d)$.

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 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*. Categorical importance is a primary determinant of the term-weight function $\mathbf{Weight}(t)$.

Accordingly, the term-weight function $\mathbf{Weight}(t)$ can be defined as follows:

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

where

- **Type** 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”, “special-function” and “regular” (for everything else).
- **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.

Much like terms, math objects (e.g., equations 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, 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 possibly 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* or *global*:
   - 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 Website;
   - a global cross-reference is one where the two objects belong to two different divisions of information.

   In the second 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
     - Object A refers explicitly to object B as the object that defines c.
   - 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 (where the term "proposition" is used in a broad sense, so it encompasses theorems, lemmas, corollaries, "inline" substantiated or stipulated claims, etc.)
     - Object A refers explicitly to object B as the primary location of proposition P.

Accordingly, the math-object weight function **Weight**(mo) can be defined as follows:

$$\mathbf{Weight}(mo) = \mathbf{Combine}(\mathbf{Quantify}(\mathbf{Type}(mo)), \mathbf{T}W(mo), \mathbf{C}R(mo)),$$

where

- **Type** and **Quantify** are like those for terms except here the categories are those of math objects;
- $\mathbf{T}W(mo)$ captures the weight of the terms that make up the math object mo;
- $\mathbf{C}R(mo)$ captures and quantifies the statistics of cross-reference pointers pointing to mo;

– **Combine** combines the various aspects (i.e, object type, wieght of the constituent terms of the object, and cross-reference information) 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 magnified by some weight 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 $S = \sum_{i=1}^{n} w_i x_i$, among many possibilities of combining weighted factors. 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 lexigraphic 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 quantified and factored into metric formula. Second, and more importantly, vector metrics and lexicographic sorting stricly enforce the policy that a most important factor should not be overwhelmed by a comination of less important factors. For example, if an object A has the highest y_1 value among a set of objects, it will rank ahead of all the other object regardless of the values of the other y_is. In particular, if definitional types of objects are desired to rank at the tops of hits, the system can have the first component of V' correspond to object type, and give the largest value to definition types (compared to other object types such as propositions, graphs, etc.).

In this paper, the vector ranking metric approach is adopted. To be precise, the **combine** function used employs a hybrid of scalarization and vectorization as seen next.

The $\mathbf{T}W(mo)$ function can have scalar or vector values. Specifically, assume that the types of terms are $\{T_1, T_2, \ldots, T_k\}$, as for example {regular, operator, special-function}. Then,

$$\mathbf{TW}(mo) = \sum_{i=1}^{k} \mathbf{Quantify}(T_i) \times N_i(mo)$$

or

$$\mathbf{TW}(mo) = (N_1(mo), N_2(mo), \ldots, N_k(mo))$$

where

$N_i(mo)$=number of terms of type T_i in the object mo.

The $\mathbf{C}R(mo)$ function maps the cross-reference information into a vector that reflect the number of cross-references of the four possible types identified earlier:

$$\mathbf{C}R(mo) = (GD(mo), GP(mo), LD(mo), LP(mo))$$

where

– $GD(mo)$ = number of global, definitional cross-references to object mo
– $GP(mo)$ = number of global, propositional cross-references to object mo
– $LD(mo)$ = number of local, definitional cross-references to object mo
– $LP(mo)$ = number of local, propositional cross-references to object mo.

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 $\mathbf{T}W(mo)$ 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 $\{T_1, T_2, \ldots, T_k\}$,

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

where

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

Overall Relevance Vector Metric

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

It is the author's judgement, and for the sake of performance evaluation presented later, that a good relevance metric be a vector where the relevance components, ordered from the highest importance to the lowest importance, are:

– $ND(q, mo)$, 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,
– $\mathbf{CR}(mo) = (GD(mo), GP(mo), LD(mo), LP(mo))$. It captures the global and local definitional/propositional cross-reference statistics,
– the remaining $N_i(q, mo)$s
– $\mathbf{TW}(mo)$, which is the term-weight of the object mo, expressed either as a vector or a weighted sum,
– (optional) $\mathbf{Quantify}(\mathbf{Type}(mo))$, reflecting preferences for certain document/object types over others,
– tf-idf(q, mo), as a final tie-breaker.

3.1 Speed Performance Evaluation of Hit Ranking

The relevance ranking scheme discussed in this paper has been implemented and tested on the DLMF testbed. 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 1. 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 thousands equations), the number of hits will usually be in the tens, hundreds, or at most in the thousands, 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 conceivable 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.

Table 1. 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

3.2 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 utlized 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 and 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. Under the tf-idf relevance model, such hits were"buried" in the second, third, or fourth page of hits.

We predict that future evaluation of user satisfaction will confirm the hypothesis that the new relevance metrics are far superior to the tf-idf metric. Of course, further refinements will be suggested by the future subjective evaluation.

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 doument 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 dtetermine whether or not a hit is worth pursuing. Furthermore, of those parts are determined carefully, they may often be all that the user needs from a document, thus saving him/her from 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 as they make the best representation of hits. 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, 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.

Index-Time Tasks for Hit Generation

1. Fragment 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 graph, (4) a fragment of a table (in the case

where tables are large), (5) a title of a chapter/section/subsection, (6) a notational item, and so on. This fragmentation will take place when the documents are indexed.
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 vector of the mini-document at search time.
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.

Search-Time Tasks for Hit 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. The following outlines the tasks to be performed:

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 vectors described in the previous section. Note that the relevance vector values 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 descriptive summary that is presented along with the hit title in the hits page.

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 conputation 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 that 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 disvantage.

4.1 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 2. The table shows the queries, the time it takes to derive the description of a single hit, and the time to derive the descriptions of 10 hits that make up a hit-page. As can be seen, the time for generating the descriptions for the hits in one page ranges from a few to less than 300 milliseconds.

It is important to note that based on the two Tables 1 and 2, 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.

Table 2. Speed Performance of Hit-Description Generation (All time measurements are in milliseconds, and a page has 10 hits)

Query	Hit-packaging Time per Hit	Hit-packaging Time per Page
Ai^2	26	260
$\int \sin$	10.11	101.1
eulerBeta	7	70
$\sin^2$	19.42	194.2
jacobiSN OR Si	4.97	49.7
eulerGamma	6.33	63.3
cos	4.28	42.8
sin	4.16	41.6
z	5.32	53.2

4.2 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, the 5 top-ranking fragments that were identified and presented as the description were found to be truly the most query-relevant and representative of the hit-document. For example, for the query "sin", the top-ranking hit was the one where $\sin z$ is defined, and the description of that hit is:

1. Definitions and Periodicity (in Elementary Functions Chapter)
   $\sin z = \frac{e^{iz}-e^{-iz}}{2i}$, ... $e^{\pm iz} = \cos z \pm i \sin z$, ... $\tan z = \frac{\sin z}{\cos z}$, ...
   $\csc z = \frac{1}{\sin z}$, ... $\cot z = \frac{\cos z}{\sin z} = \frac{1}{\tan z}$

The hit-document contains other contents involving sin, such as $\sin(z + 2k\pi) = \sin z$, but because the number of fragments per hit-description was limited to 5, some fragments had to be left out. If the description size is set to larger numbers of fragments, more will be included per description. The ideal hit-description size in math search is an aspect that requires further research.

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.

5 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 far superior to the conventional tf-idf metric, and the new hit-descriptions are more query-relevant and representative of the hit targets than convential 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. Also, incorporating highlighting into the hit-descriptions, and turning each fragment in a hit-description into a hot link that would lead the user to the right location in the hit target, are subjects for further research and implementation.

References

1. MathSciNet. American Mathematical Society (AMS). `http://www.ams.org/mathscinet`
2. Bancerek, G.: The 5th International Conference on Mathematical Knowledge Management, Wokingham, UK, pp. 266–279 (August 11-12, 2006)
3. Einwohner, T.H., Fateman, R.: Searching techniques for integral tables. International symposium on Symbolic and algebraic computation, ACM (1995) `http://torte.cs.berkeley.edu:8010/tilu`
4. Guidi, F.: Searching and Retrieving in Content-based Repositories of Formal Mathematical Knowledge. Ph.D. Thesis in Computer Science, University of Bologna, Technical report UBLCS 2003-06 (March 2003)

5. Guidi, F., Schena, I.: A Query Language for a Metadata Framework about Mathematical Resources. In: The 2nd International Conf. Mathematical Knowledge Management, Bertinoro, Italy (February 2003)
6. Jahrbuch Database. http://www.emis.de/MATH/JFM/JFM.html
7. Lozier, D.W.: The DLMF Project: A New Initiative in Classical Special Functions. In: International Workshop on Special Functions - Asymptotics, Harmonic Analysis and Mathematical Physics. Hong Kong (June 21-25, 1999)
8. Lozier, D.W., Miller, B.R., Saunders, B.V.: Design of a Digital Mathematical Library for Science, Technology and Education. In: Proceedings of the IEEE Forum on Research and Technology Advances in Digital Libraries; IEEE ADL '99, Baltimore, Maryland (May 1999)
9. MathDi (Mathematics Didactics Database). http://www.emis.de/MATH/DI.html
10. Mathdex search tool: http://www.mathdex.com:8080/mathfind/search
11. Mathdex description: http://www.ima.umn.edu/2006-2007/SW12.8-9.06/activities/Miner-Robert/index.html
12. Mathematica: http://www.mathematica.com
13. Miller, B., Youssef, A.: Technical Aspects of the Digital Library of Mathematical Functions. Annals of Mathematics and Artificial Intelligence 38, 121–136 (2003)
14. MoWGLI: Mathematics on the Web: Get It by Logics and Interfaces. http://mowgli.cs.unibo.it/
15. Salton, G., McGill, M.J.: Introduction to Modern Information Retrieval. McGraw Hill, New York (1993)
16. Baeza-Yates, R., Ribeiro-Neto, B.: Modern information retrieval. Addison-Wesley, London (1999)
17. Youssef, A.: Information Search And Retrieval of Mathematical Contents: Issues And Methods. In: The proceedings of the ISCA 14th International Conference on Intelligent and Adaptive Systems and Software Engineering (IASSE-2005), July 20-22, Toronto, Canada (2005)
18. Youssef, A.: Roles of Math Search in Mathematics. In: The 5th International Conference on Mathematical Knowledge Management, Wokingham, UK, pp. 2–16 (August 11-12, 2006)
19. Zentralblatt MATH database at European Mathematical Information Service (EMIS). http://www.emis.de/ZMATH/