Original article

Multi-source and ontology-based retrieval engine for maize mutant phenotypes

Jason M. Green[1], Jaturon Harnsomburana[1], Mary L. Schaeffer[2], Carolyn J. Lawrence[3] and Chi-Ren Shyu[1,4,*]

[1]Computer Science Department, University of Missouri, Columbia, MO, [2]USDA-ARS Plant Genetics Research Unit and Division of Plant Sciences, University of Missouri, Columbia, MO, [3]USDA-ARS Corn Insects and Crop Genetics Research Unit, Department of Genetics, Development and Cell Biology, Iowa State University, Ames, IA and [4]Informatics Institute, University of Missouri, Columbia, MO, USA

***Corresponding author:** Tel: +1 573 882 3884; Fax:+1 573 884 8709; Email: shyuc@missouri.edu

Submitted 29 September 2010; Revised 15 March 2011; Accepted 22 March 2011

Model Organism Databases, including the various plant genome databases, collect and enable access to massive amounts of heterogeneous information, including sequence data, gene product information, images of mutant phenotypes, etc, as well as textual descriptions of many of these entities. While a variety of basic browsing and search capabilities are available to allow researchers to query and peruse the names and attributes of phenotypic data, next-generation search mechanisms that allow querying and ranking of text descriptions are much less common. In addition, the plant community needs an innovative way to leverage the existing links in these databases to search groups of text descriptions simultaneously. Furthermore, though much time and effort have been afforded to the development of plant-related ontologies, the knowledge embedded in these ontologies remains largely unused in available plant search mechanisms. Addressing these issues, we have developed a unique search engine for mutant phenotypes from MaizeGDB. This advanced search mechanism integrates various text description sources in MaizeGDB to aid a user in retrieving desired mutant phenotype information. Currently, descriptions of mutant phenotypes, loci and gene products are utilized collectively for each search, though expansion of the search mechanism to include other sources is straightforward. The retrieval engine, to our knowledge, is the first engine to exploit the content and structure of available domain ontologies, currently the Plant and Gene Ontologies, to expand and enrich retrieval results in major plant genomic databases.

Database URL: http:www.PhenomicsWorld.org/QBTA.php

Background

Major plant genome databases like MaizeGDB (1), Gramene (2), TAIR (3), SGN (4), Soybase (5) and Oryzabase (6) have compiled and organized large quantities of data for their respective research communities. Though enormous amounts of new data are being generated and submitted, completion of the respective genomes is expected to increase the rate of data collection even more. Currently, each group provides browsing capabilities to allow individuals to sift through the available data as well as basic search mechanisms, mainly in the form of structure query language (SQL) queries, to aid users in locating specific information. While these search mechanisms meet basic needs of plant science researchers, they will become decreasingly useful because of several shortcomings.

One limitation that will cause problems in the future is a failure to rank search results by similarity to the query, which is a by-product of the Boolean style retrieval (7) used by most current plant search mechanisms. Instead, search results are sorted idiosyncratically by some field, which is sufficient when databases and result sets are relatively small. However, as database sizes increase causing result sets to become larger, manual perusal through a list of arbitrarily ordered or alphabetized results to find meaningful information will become tedious, inefficient and slow. To locate desired information more quickly, ranking of retrieved results according to their similarity to the query will be necessary.

A second issue is the limited utilization of free-text fields. A limited number of search mechanisms exist in the plant community that make use of free-text descriptions; most appear to rely on making selections from predefined lists of characteristics. Those mechanisms that do search free-text fields treat the description as a single string for character matching (description prefix, description suffix, description contains, etc), which is not sufficient due to variations in phenotype descriptions. More advanced information retrieval methods can be implemented to take better advantage of the wealth of information available in these fields.

A third shortcoming is the limited utilization of domain ontologies in current retrieval methods. Large amounts of time, money and energy have been put forth toward the development of several plant-related ontologies (8, 9). Though the rich knowledge embedded in these ontological structures is particularly well-suited to retrieval, especially for improving the efficiency and coverage of retrieved results, the domain ontologies remain underutilized. In the information retrieval community, ontologies have been used extensively for word sense disambiguation, which is the complex task of determining the correct meaning of a polysemous word from its context (10–12); thematic summarization and concept mapping, which involves identifying broad concepts in free text (13–15); and query expansion, which is a technique of enriching a query by adding additional relevant terms to it (16–18). State-of-the-art query expansion with ontologies generally uses hierarchical relationships, typically parents and children (18), and this is one technique that can be employed to improve the accuracy and coverage of plant phenotype searches.

As research in comparative and systems biology draws more attention, the information needs of researchers in these areas will become vastly more complex, and search mechanisms will be needed that can more fully utilize and integrate stored information to better accommodate these information needs. We have developed a flexible, advanced search mechanism that seeks to overcome the described shortcomings. This retrieval engine is designed to be multi-source, meaning it integrates multiple related free-text sources, to aid the user in retrieving desired information. The current version of the search engine combines three text sources that are associated with phenotypic variations in MaizeGDB. In total, the 2083 variations present in our local database are linked to 4103 phenotype image captions, 559 loci (a total of 1539 locus descriptions) and 32 gene products (a total of 32 descriptions).

Table 1. Statistics on ontology occurrence in the text sources

Text Source	Size	Number with PO terms (%)	Number with GO terms (%)
Phenotype description	4103	3009 (73.3)	615 (15.0)
Locus description	1539	584 (38.0)	598 (38.9)
Gene product description	32	4 (12.5)	9 (28.1)

The retrieval engine is also designed to utilize the knowledge and structure of existing domain ontologies for query expansion, with the expectation of improving the contextualization of the query. Given the above text sources, the Plant Ontology (PO) (9) and Gene Ontology (GO) (8) were natural choices for inclusion in the search engine, as they had frequent matches to terms in these text sources. As depicted in Table 1, PO terms occurred in nearly 73% of phenotype captions with PO terms and GO terms each appearing in roughly 39% of locus descriptions.

Implementation

Multi-source retrieval concept

In standard information retrieval systems, a query is submitted to a search engine, which then attempts to retrieve and rank the most relevant documents from the underlying corpus. However, in the case of multi-source retrieval, where a document is linked to other documents from other sources, the traditional retrieval model formulations cannot be directly applied, as the engine must be capable of retrieving and ranking groups of documents.

To make this more concrete, consider the set of three sources for the maize mutant phenotype search engine depicted in the top half of Figure 1. Each phenotype variation is linked to zero or more documents from each of the text sources. Thus, when a phenotype search is submitted, the retrieval engine will need to consider sets of related documents (the bottom half of Figure 1) from various sources, e.g. consisting of all related phenotype, locus and gene product descriptions. Collectively, these sets of documents, referred to as a folders, contain the information used for retrieval, and so retrieval and ranking are performed on folders rather than individual documents. For the remainder of this article, the term folder will be used to designate the set of documents from all sources related to a specific mutant phenotype.

The simplest solution to retrieving and ranking groups of documents is to merge the text from each document of a folder, referred to as a member or component document, into one mega-document. While this solution does have the advantage of allowing the use of the standard vector space retrieval model (19), it does not, however, reflect the data

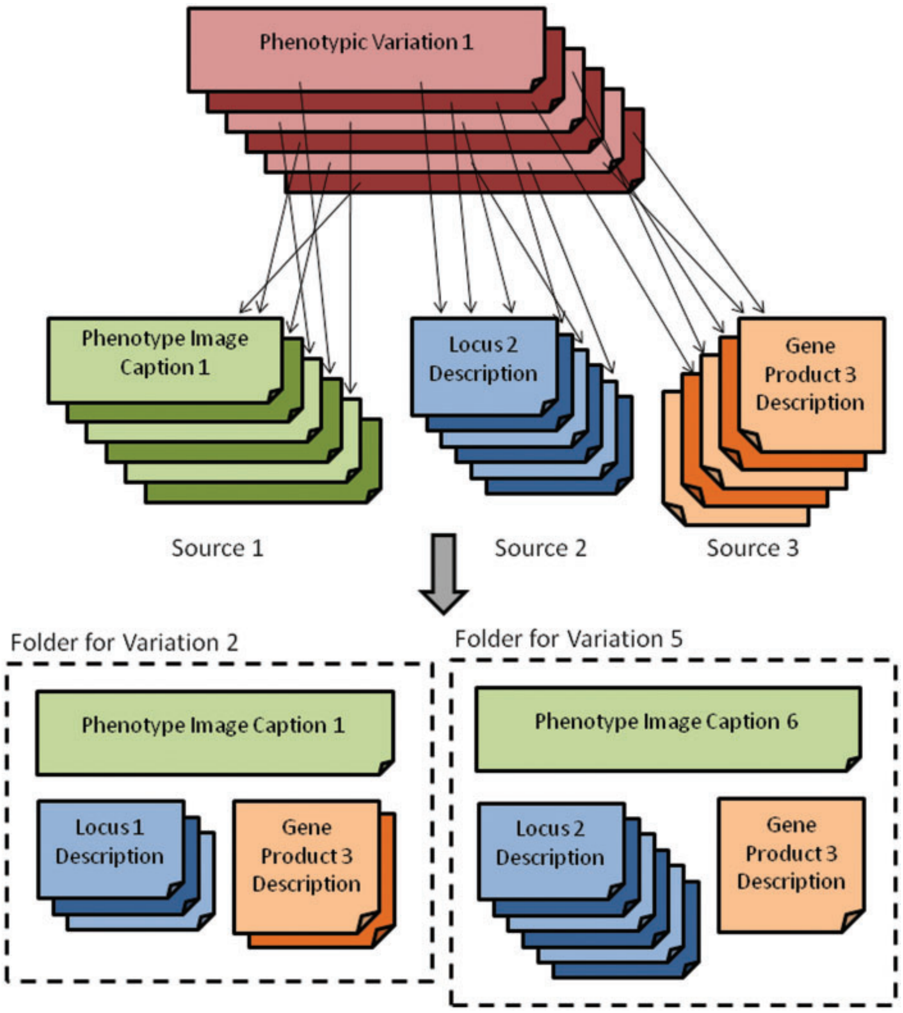

Phenotypic Variation 1
Phenotype Image Caption 1
Locus 2 Description
Gene Product 3 Description
Source 1
Source 2
Source 3
Folder for Variation 2
Phenotype Image Caption 1
Locus 1 Description
Gene Product 3 Description
Folder for Variation 5
Phenotype Image Caption 6
Locus 2 Description
Gene Product 3 Description

Figure 1. Depiction of the main grouping source and its relationship to the various text sources (top) as well as the folder view (bottom) where related documents for phenotypic variations are grouped.

making up these folders very well. This is because the phenotype descriptions, locus descriptions and gene product descriptions are describing very different entities in the maize domain. Thus, there is variability in the distribution of specific terms, both in terms of usage and rarity, between text sources. This can be illustrated concretely by examining the variability of ontology terms across these sources. Because the GO contains terms related to genes and gene products, we would expect these terms to appear more often in locus and gene product descriptions than in phenotype ones. Likewise, since the PO contains plant anatomy and morphology terms, we expect these to appear most often in the phenotype descriptions and, to a lesser extent, in the locus descriptions. By examining Table 2, this is precisely the situation we find in the MaizeGDB mutant phenotype collection. The trend shown is that PO terms appear less frequently as we move from phenotype to locus to gene product descriptions, with precisely the opposite pattern present with GO terms.

For these reasons, merging all the components of a folder into a single document does not make the most logical sense. The proposed approach is a transformation of this multi-source retrieval problem so that groups of related documents can be retrieved using the vector space model (7).

Approach

Representing folders. To be able to retrieve groups of documents, two issues must be decided. First, the representation of a folder, i.e. a group of related documents, must be addressed, and second, a novel similarity measure between folders and the query must be formulated.

Let K be the set of text sources relevant to a particular query. Let D^k be the set of all documents within source

Table 2. Ontology term breakdown per text source

Average terms/document	GO	Text source	PO	Average terms/document
0.18		Phenotype description		1.60
0.55	⇓	Locus description	⇑	1.02
0.34		Gene product description		0.16

k with D^1 denoting the base set, that is, the seed documents or identifiers that are used to build folders. The other document sets D^k, $k > 1$, are secondary sources with documents related to one or more folders. In this search engine, D^1 corresponds to the phenotype variation identifiers, with D^2, D^3 and D^4 being the phenotype image, locus and gene product descriptions, respectively. Let d_j^k denote document j within document set D^k, and $\vec{q}$ represent the vector associated with user's text query.

In order to formally define a folder, the document relationships between text sources must be modeled. Let f be the mapping relationship [Equation (1)] that, given a document j from D^1 and a text source k, returns the list of documents in source k that are related to d_j^1.

$$f(j,k) = \left\{ d_l^k \middle| d_l^k \sim d_j^1, 0 \le l < \left| D^k \right| \right\}, \tag{1}$$

where $a \sim b$ means that a is related to b, which in this search engine means one document is related to another through a series of database joins (i.e. common keys). Using this mapping relationship, we can define the folder corresponding to the base document d_j^1 as $F(j, K)$, which is the set of all documents mapped to d_j^1 for a given set of text sources K.

$$F(j, K) = \bigcup_{k \in K} f(j, k). \tag{2}$$

In the context of this search engine, $F(j, K)$ would correspond to a single phenotypic variation and would contain all the phenotype, locus and gene product descriptions related to that variation.

Constructing folder vectors. With the notion of folders formalized, considerations for folder representation and similarity can be addressed. Clearly, to use any variant of the vector space model, each folder must be represented as a vector of constituent term weights. Our approach is to treat the folder as a collection of $|K|$ documents, one for each of the text sources being searched. We form the representative vector for each source by applying some function $g(.)$ to the documents in $f(j, k)$. Two classes of functions that can be used for this purpose are merge and selection.

Merge: With this approach, all the documents in $f(j, k)$ are combined together and treated as a large document. Conventional vector space weighting schemes can be applied to the conglomerate document vector or a summarized version of the combined document. This option may be appropriate when the documents in $f(j, k)$ are more homogeneous, i.e. when each contains a textual description of the same entity. As an example, consider a mutant phenotype that is linked to a single genetic locus, but there may be multiple descriptions of that locus in the database. In this case, the amalgamation or summarization of the set of descriptions is more likely to represent the concept as a whole (e.g. the locus, in this example) than any one document in the set.

Selection: Instead of merging the documents in $f(j, k)$ together, one may alternatively choose to select a particular document from the set to represent the text source. This function class can be used in any situation, but is particularly appropriate when the documents in $f(j, k)$ are more heterogeneous. As an alternative example, consider a mutant phenotype that is linked to several different gene products in the database, each of which has a single description in the database. Each of these descriptions is describing a completely different entity and, therefore, merging all these descriptions does not make sense from a retrieval standpoint. However, if only one of the gene product descriptions represents a good match for a query, then the user may still want to see that mutant phenotype. This can be accomplished by selecting only the best match from $f(j, k)$, i.e. the document from this text source with the maximum similarity to the query. One could consider other schemas for selection that are based on the most average document or simply one selected at random.

The chosen function class is highly dependent on the text source. In the case of this maize mutant phenotype search engine, both classes of functions could be applied. The locus source would have a good candidate for the *merge* option. This is because in the current data set, each mutant phenotype is linked to a *single* locus, which may itself have several descriptions. On the other hand, the gene product source is more heterogeneous and is thus better suited for the *selection* option, as each phenotype description may be linked to several gene product descriptions, each describing a different gene product. In our search engine, the *selection* function class was chosen for all text sources and utilizes the function defined in Equation (3). Briefly, it selects the document from $f(j, k)$ that has the highest cosine similarity to the query.

$$g(.) = \mathrm{sel}(f(j,k)) = \left\{d_i^k | i = \max_l\left\{\mathrm{sim}\left(d_l^k,q\right)\right\}\right\}, \tag{3}$$

where q is the query vector and the cosine similarity is defined as

$$\mathrm{sim}\left(d_j^k,q\right) = \frac{\sum_{i=1}^t w_{j,i} \times w_{q,i}}{\sqrt{\left(\sum_{i=1}^t w_{j,i}^2\right)} \times \sqrt{\left(\sum_{i=1}^t w_{q,i}^2\right)}}. \tag{4}$$

Measuring folder similarity. In addition to choosing the function $g(.)$ for each text source, a decision must be made on how to use the $|K|$ vectors comprising the folder to define a similarity measure. Two alternatives (weighted average and folder vector) for the folder similarity are discussed.

Weighted average: the first method to combine the $|K|$ document vectors is to perform a weighted average as follows:

$$\mathrm{sim}(F(j,K),q) = \sum_{k\in K} w_k \mathrm{sim}(g(f(j,k)),q). \tag{5}$$

With this approach, a weight for each text source w_k can be provided to reflect the importance of text source k relative to the other sources. Also, the traditional cosine similarity measure can be applied between the query q and the formed vector (by *merge* or *selection*) for each text source.

Folder vector: alternatively, one could form a folder vector by concatenating the $|K|$ component document vectors [see Equation (6)].

$$\vec{F}(j,K) = \left(\vec{d}^{k_1}_{g(f(j,k_1))}, \ldots, \vec{d}^{k_n}_{g(f(j,k_n))}\right). \tag{6}$$

The similarity between the query and a folder can then be expressed as a reformulation of the cosine similarity measure in terms of text sources and component documents. The similarity formula is provided for both the *merge* and *selection* approaches.

For the *merge* case, the similarity measure is given by Equation (7). In keeping with the idea behind *merge*, each component document from each source is involved in the calculations of the cross product (numerator) and the folder normalization factor (left term in the denominator).

$$\mathrm{sim}_{\mathrm{merge}}(F(j,K),q) = \frac{\sum_{k\in K}\left(\sum_{i=1}^t\left(\left(\sum_{d=f(j,k)} w_{d,i}^k\right) \times w_{q,i}^k\right)\right)}{\sqrt{\sum_{k\in K}\left(\sum_{i=1}^t\left(\sum_{d=f(j,K)} w_{d,i}^k\right)^2\right)} \times \sqrt{\sum_{k\in K}\left(\sum_{i=1}^t w_{q,i}^{k^2}\right)}}. \tag{7}$$

where q is the query vector, the weight of term i in document d of text source k, $w_{d,i}^k$, is expressed by

$$w_{d,i}^k = \mathrm{TF}_{d,i}^k \times \mathrm{IDF}_i^k \tag{8}$$

and the analogous query term weights are given by

$$w_{q,i}^k = \mathrm{TF}_{q,i}^k \times \mathrm{IDF}_i^k. \tag{9}$$

Note that term frequency (TF) and inverse document frequency (IDF) are calculated using standard definitions, as in ref. (7). These quantities give higher weight to terms that appear frequently within a single document and to those that are rare across all documents, respectively. It should be noted that while many documents in this data set are rather short, the use of a TF*IDF weighting scheme is rationalized by the fact that there is a sizable number of terms that appear more than once in a single document (9343 out of 85 193, or roughly 11%), and a few words that appear up to 25 times in a single document. This variability makes TF a worthwhile factor for those terms. For the rest of the terms that appear once per document, the IDF is still a very useful parameter to determine term importance, as the more rare terms in the entire text source become the higher weighted and hence more important terms.

The similarity measure when using the selection function may be computed as follows:

$$\mathrm{sim}_{\mathrm{selection}}(F(j,K),q) = \frac{\sum_{k\in K}\left(\sum_{i=1}^t\left(w_{g(f(j,k)),i}^k \times w_{q,i}^k\right)\right)}{\sqrt{\sum_{k\in K}\left(\sum_{i=1}^t w_{g(f(j,k)),i}^{k^2}\right)} \times \sqrt{\sum_{k\in K}\left(\sum_{i=1}^t w_{q,i}^{k^2}\right)}}. \tag{10}$$

Recall that with selection, only one document from each text source, chosen using the function $g(.)$, is used in the similarity calculation of the document case to the query.

System details

Offline and online processing. Using the derived similarity measure, a multi-source retrieval engine can then be implemented using the vector space model. The construction of the search mechanism can be partitioned into two main steps: (i) offline preprocessing of the entire document corpus and (ii) online query processing and retrieval. For offline preprocessing, each document from each source within MaizeGDB proceeds as in Figure 2. Documents are parsed into words and phrases to facilitate matching to ontology concepts. Phrases that do not match ontology terms and synonyms are removed, as are stop words. TFs are calculated for the remaining words and phrases in each document and stored in a MySQL database. Once all the documents have been processed, IDFs are calculated for each word and phrase from each text source and also stored. IDFs are calculated separately for each text source to maintain the differences in term discrimination between sources as shown in Table 1. A PHP script is

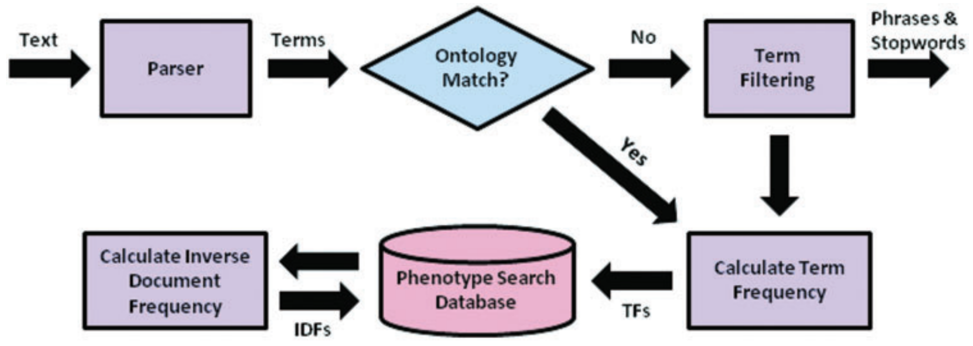


Figure 2. Preprocessing flow chart for the system, including parsing of text descriptions into individual term, matching terms to ontological concepts, calculation of needed quantities for determining term weights, and insertion of terms and quantities into the MySQL database.

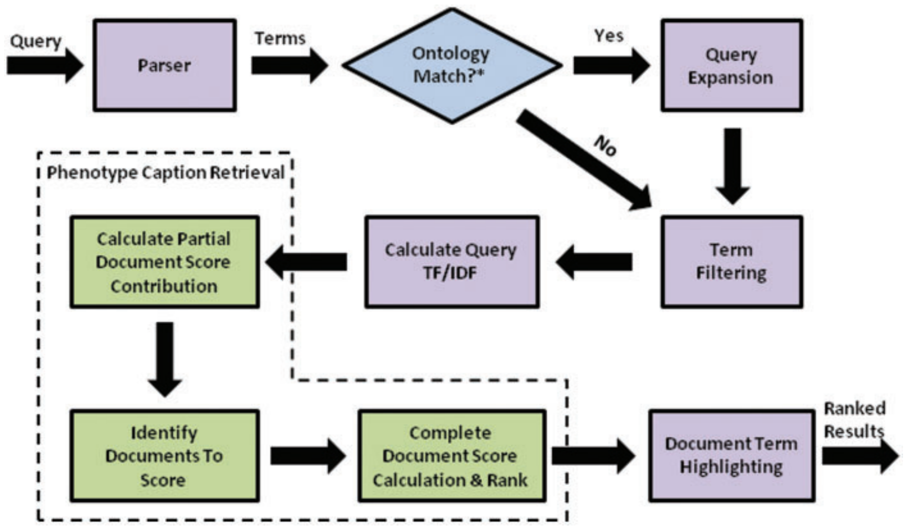

Query
Parser
Terms
Ontology Match?*
Yes
Query Expansion
No
Phenotype Caption Retrieval
Calculate Partial Document Score Contribution
Calculate Query TF/IDF
Term Filtering
Identify Documents To Score
Complete Document Score Calculation & Rank
Document Term Highlighting
Ranked Results

Figure 3. Flow chart for search engine retrieval, from query processing, to similarity calculations for individual text descriptions and folders, to ranking and highlighting of the search results.

used to automate the offline processing of all text sources and insert the appropriate information into the MySQL database.

Once the above offline procedure has been completed, the retrieval engine is functional. A submitted query can proceed as in Figure 3 to retrieve appropriate results for the user. The query follows a similar procedure as a document in offline processing. It is parsed into words and phrases that are matched to the available ontologies. Matched concepts undergo query expansion, in which term synonyms, parents and children may be added to the query. Unmatched phrases and stop words are removed, and TFs are calculated for the remaining words and phrases. Query term weights are then calculated using Equation (11).

$$w_{q,i} = \mathrm{TF}_{q,i} \times \mathrm{IDF}_{i,k} \times w_{\mathrm{source}} \times w_{\mathrm{ont}} \times w_{\mathrm{rel}}, \tag{11}$$

where $\mathrm{TF}_{q,i}$ is the query TF defined in the standard manner by Equation (12), IDF_i is the IDF of term *i* in source *k*, w_{source} is the weight of the text source, w_{ont} is the weight of the matched ontology, w_{rel} is the weight of the expansion relationship type (term, synonym, parent, child).

$$\mathrm{TF}_{q,i} = \left(0.5 + \frac{0.5\ \mathrm{freq}_{q,i}}{\max_l\{\mathrm{freq}_{q,l}\}}\right). \tag{12}$$

The default weights for the various sources, ontologies, and relationships are given in Table 3, though each parameter can be manually adjusted by the user before each search. Default weights for the query expansion terms were based on experimental results from ref. (18).

Query search interface. The query interface for the search engine is depicted in Figure 4 and contains the standard text box for the user's free-text query as well as all the weighting options for each type of term, located in the 'search options' box underneath these controls. Each column of controls represents a related set of options.

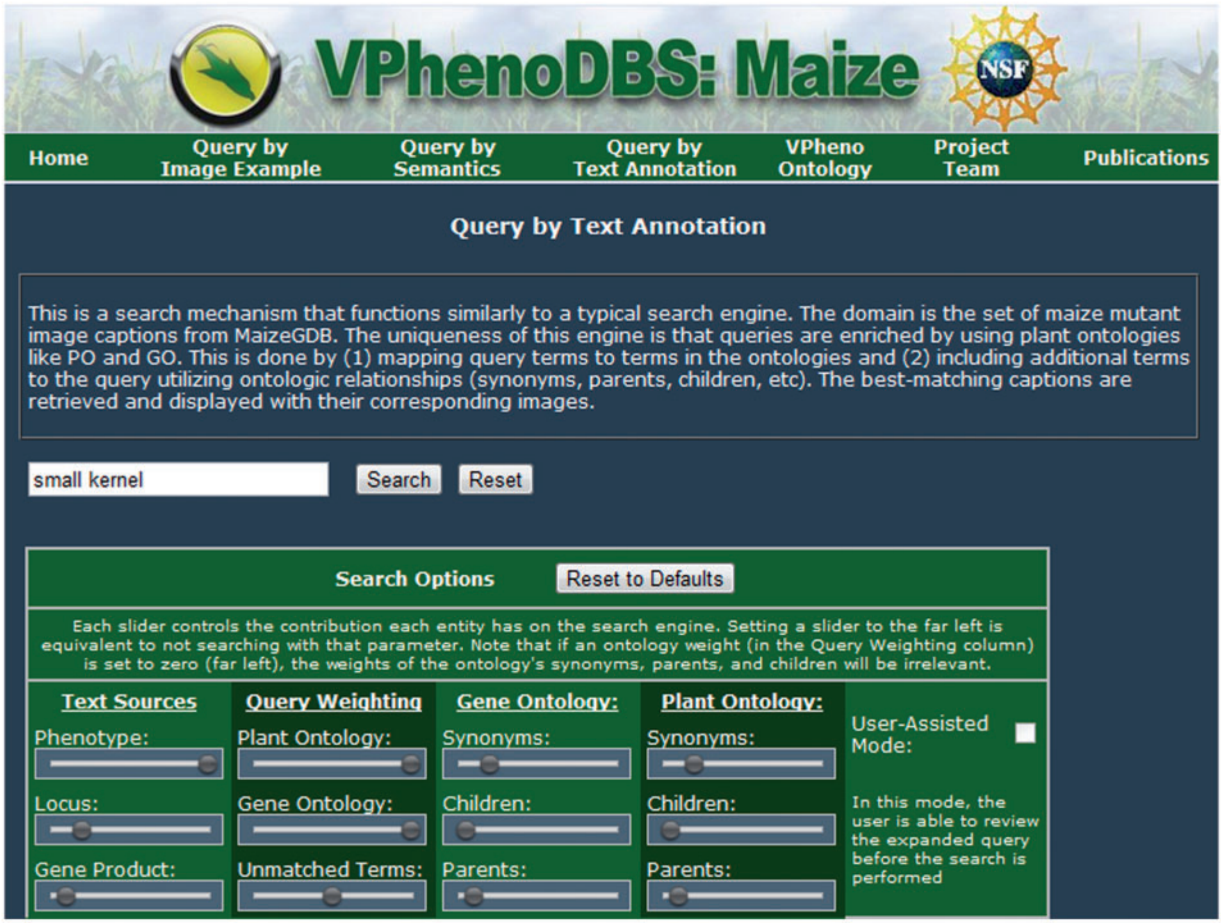
Table 3. Default weights of text sources, ontologies and relationships

Symbol	Weight description	Default weight
$w_{\mathrm{phenotype}}$	Phenotype source weight	1.00
w_{locus}	Locus source weight	0.20
$w_{\mathrm{gene\_product}}$	Gene product source weight	0.10
w_{GO}	Gene ontology weight	1.00
w_{PO}	Plant ontology weight	1.00
$w_{\mathrm{unmatched}}$	Nonontology terms	0.50
w_{synonym}	Synonym expansion weight	0.20
w_{parent}	Parent expansion weight	0.10
w_{child}	Child expansion weight	0.05

The first column regards the available searchable text sources. The position of the slider bars can be used to select how much emphasis to place on each source relative to the other sources, ranging from 0, which corresponds to no weight (slider at the far left), to 1, which corresponds to the greatest amount of emphasis (slider at the far right). The default values for the text sources are given in Table 3.

Similarly, the second column controls the emphasis of different types of terms present in the query. The top two sliders control the importance of terms matching to GO and PO, respectively; the bottom slider in this column controls the weight for query terms that did not match an ontology. This gives the user the flexibility to really stress specific types of terms.

The third and fourth columns control the emphasis of terms added through query expansion from the GO and PO, respectively. These sliders determine what kinds of ontology terms are used to expand the query and how much weight is given to each type of term (again, relative to the other term types). Initially, the expansion terms are



Query by Text Annotation

This is a search mechanism that functions similarly to a typical search engine. The domain is the set of maize mutant image captions from MaizeGDB. The uniqueness of this engine is that queries are enriched by using plant ontologies like PO and GO. This is done by (1) mapping query terms to terms in the ontologies and (2) including additional terms to the query utilizing ontologic relationships (synonyms, parents, children, etc). The best-matching captions are retrieved and displayed with their corresponding images.

small kernel | Search | Reset

Search Options | Reset to Defaults

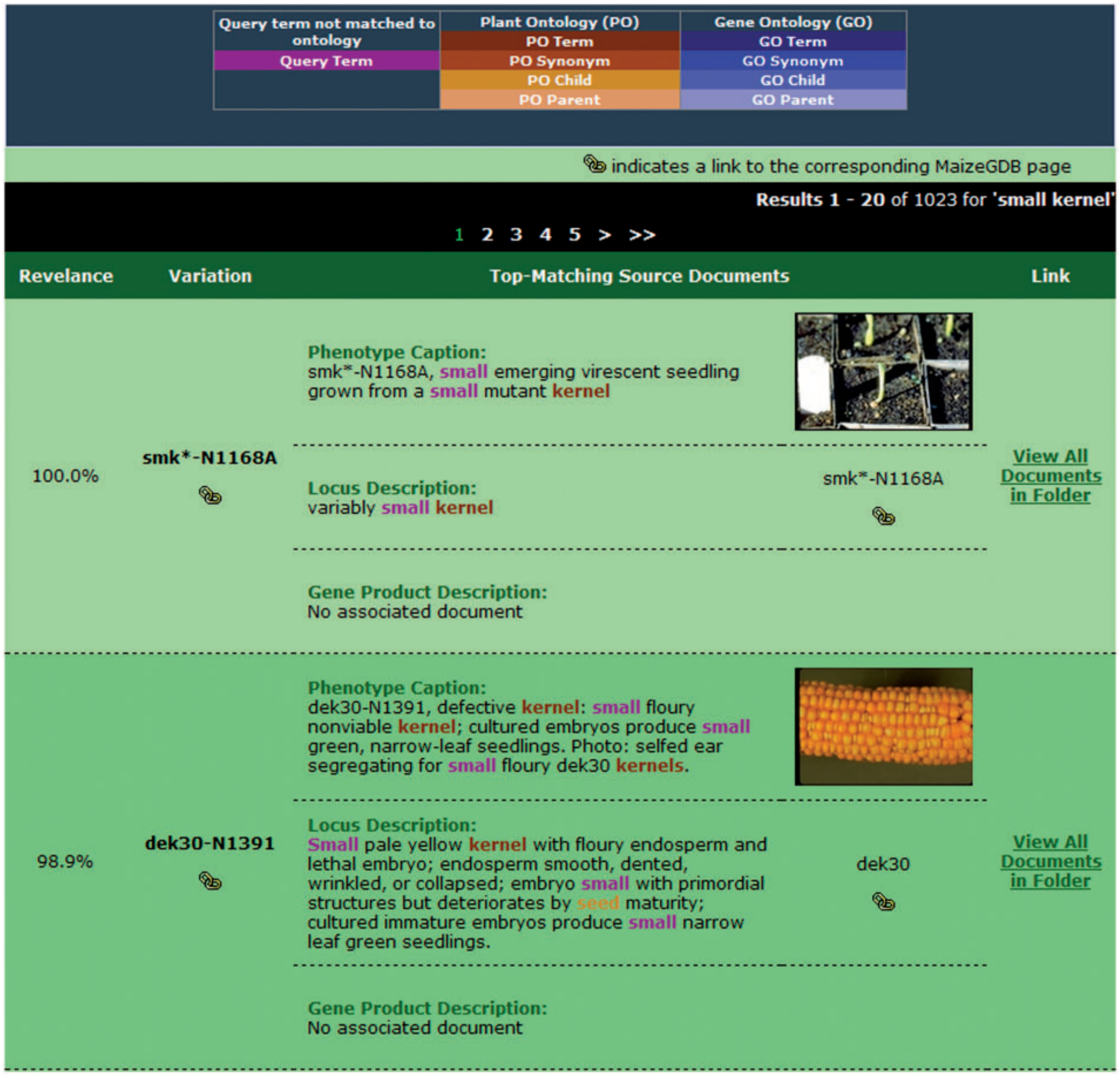
Each slider controls the contribution each entity has on the search engine. Setting a slider to the far left is equivalent to not searching with that parameter. Note that if an ontology weight (in the Query Weighting column) is set to zero (far left), the weights of the ontology's synonyms, parents, and children will be irrelevant.

Text Sources	Query Weighting	Gene Ontology:	Plant Ontology:
Phenotype:	Plant Ontology:	Synonyms:	Synonyms:
Locus:	Gene Ontology:	Children:	Children:
Gene Product:	Unmatched Terms:	Parents:	Parents:

User-Assisted Mode:

In this mode, the user is able to review the expanded query before the search is performed

Figure 4. Query interface for the multi-source ontology-based retrieval engine for maize mutant phenotypes.



Query term not matched to ontology	Plant Ontology (PO)	Gene Ontology (GO)
	PO Term	GO Term
Query Term	PO Synonym	GO Synonym
	PO Child	GO Child
	PO Parent	GO Parent

indicates a link to the corresponding MaizeGDB page

Results 1 - 20 of 1023 for 'small kernel'

1 2 3 4 5 > >>

Revelance	Variation	Top-Matching Source Documents		Link
100.0%	smk*-N1168A	Phenotype Caption: smk*-N1168A, small emerging virescent seedling grown from a small mutant kernel		View All Documents in Folder
		Locus Description: variably small kernel	smk*-N1168A	
		Gene Product Description: No associated document		
98.9%	dek30-N1391	Phenotype Caption: dek30-N1391, defective kernel: small floury nonviable kernel; cultured embryos produce small green, narrow-leaf seedlings. Photo: selfed ear segregating for small floury dek30 kernels.		View All Documents in Folder
		Locus Description: Small pale yellow kernel with floury endosperm and lethal embryo; endosperm smooth, dented, wrinkled, or collapsed; embryo small with primordial structures but deteriorates by seed maturity; cultured immature embryos produce small narrow leaf green seedlings.	dek30	
		Gene Product Description: No associated document		

Figure 5. Sample search results for our retrieval engine using the query ‘small kernels’. Highlighted terms indicate matches to the query (including original terms and terms included through query expansion), with the color providing information about the term.

given fairly low weights to guide the search primarily towards the query terms. Of the expansion terms, synonyms are defaulted with the most weight, followed by parent terms and then children, in accordance with the results in ref. (18).

Ranked results interface. After the query is submitted, the search engine retrieves and ranks folders in decreasing order of similarity to the query. For the sample query ‘small kernel’, the results page is shown in Figure 5. At the top of the page, the user’s query and the current system weights are redisplayed in the search interface. This interface is repeated so that the user can make adjustments to the query if desired and resubmit.

Below the search interface is the list of ranked folders. With each folder, the variation name, along with a link to the corresponding MaizeGDB page, is provided to the user as well as a phenotype image, the top-ranking descriptions from each text source, and the folder’s relevance. Relevance is calculated using Equation (13), which consists of two factors. The first is a normalization term for the folder’s cosine similarity value, and the second penalizes a folder for each query term that is not matched by the folder.

$$\mathrm{rel}_i = \frac{\mathrm{sim}(F(i,K),q)}{\max_j\{\mathrm{sim}(F(j,K),q)\}} \times \left(1 + \frac{1 - \left|\left\{q_i \middle| \exists k,j \text{ such that } w_{j,i}^k > 0\right\}\right|}{-3 \times \left|\{q_i\}\right|}\right). \tag{13}$$

Only the best match from each text source is displayed in the ranked results because these are the documents that were used to calculate the folder similarity measure. To view all the documents associated with the phenotypic variation, including the ones that were not used to calculate the folder score, the user can click on the 'view all documents in folder' link in the last column (Figure 5).

When performing query expansion with synonyms, parents and/or children, it is possible that a query will be enriched with many terms from the available ontologies. As such, results containing few words from the original query may appear in the top-ranked results if the expansion weights are set sufficiently high. To aid the user in determining which terms were added and how they relate to the query, color highlighting of terms is provided. Terms matched to or added from a specific ontology are given the same general color. Different shades of that color will distinguish the relationship as matched term, synonym, parent, or child and are defined in the legend. For example, the term 'seed' appears in the second result in Figure 5 in an orange color, indicating a child relationship to the PO synonym 'kernel'.

Results and discussion

The developed multi-source ontology-based maize mutant phenotype search engine utilizes interconnected free-text fields from MaizeGDB, specifically descriptions of phenotypes, loci and gene products, as well as two domain ontologies, the GO and PO. Users are given the ability to search with any combination of text sources and ontologies and can adjust the weights of these entities as desired to customize retrieval of specific queries. This is the first retrieval tool in the plant community that utilizes sophisticated information retrieval techniques to search free-text fields and provide ranking of results according to similarity to the query and that utilizes domain ontologies in this manner for query expansion.

The search engine was evaluated according to retrieval performance, in terms of speed and accuracy, as well as scalability. The setup and results of the corresponding experiments are described below. All these experiments were conducted on a standalone development server with Intel X5570 dual 2.93 GHz quad core CPU and 72 GB of RAM.

Table 4. List of experimental queries

ID	Query
1	Lysine
2	Gibberellins
3	Aleurone
4	Pericarp
5	Mottled appearance
6	Tassel
7	Purple aleurone
8	Endosperm
9	Tassel branch
10	Andromonoecious plant
11	Leaf
12	Leaf blade
13	Necrotic tissue
14	Seedling
15	Narrow leaf
16	Kernel
17	Yellow leaf
18	Ear
19	Broad leaf
20	Segregating ears
21	Collapsed kernels
22	Floury kernel
23	Immature ear
24	Small kernel
25	Broad green leaf
26	Opaque dented kernel
27	Small floury kernel
28	Small yellow kernel
29	Leaf blade with white stripes

Retrieval performance

Experiments were conducted to measure both the speed and accuracy of the developed search engine. For all experiments, the list of 29 test queries in Table 4 was used. These test queries included a set of anatomical terms, most of which were linked to phenotypic variations in MaizeGDB, anatomical terms plus one or more modifiers and other miscellaneous queries.

To determine the overall speed of the retrieval system, each query was executed 20 times against one, two and all three text sources. The overall average retrieval time for all the test queries in all situations was measured at 2.16 s/query. The fastest query was 'lysine', which completed in 0.28 s but only had seven matching documents in the database. The slowest query was 'leaf blade with white stripes', which without query expansion partially matched 452 phenotypic variations, and finished in 5.67 s.

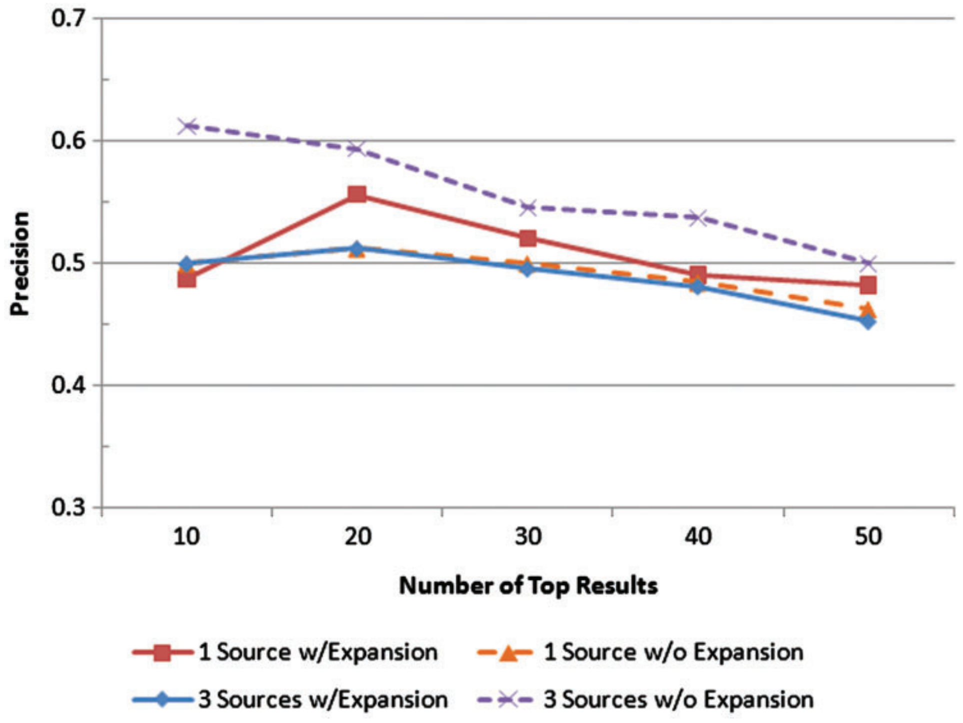


Figure 6. Precision was measured at the top 10, 20, 30, 40 and 50 results for four scenarios: one text source with query expansion, one text source without query expansion, all three text sources with query expansion, and all three text sources without query expansion. The precision values at each of those levels for each scenario are shown.

The quality and accuracy of the search mechanism were also measured. We designed an experiment that utilized eight body part terms ('kernel', 'ear', 'leaf', 'tassel', 'aleurone', 'pericarp', 'endosperm' and 'seedling') to query the system, with the expectation of retrieving variations whose affected body part matches the query. The queries were executed in four environments: with one source (phenotype captions) or all three sources and with or without query expansion. In each situation, precision was measured at 10% recall intervals and was also measured for the top 10, 20, 30, 40 and 50 results. The affected body parts, which have been manually curated by humans, in MaizeGDB for each variation were used as the standard to assess the relevance of each ranked result. The standard equations for precision and recall (2) were used and are shown below.

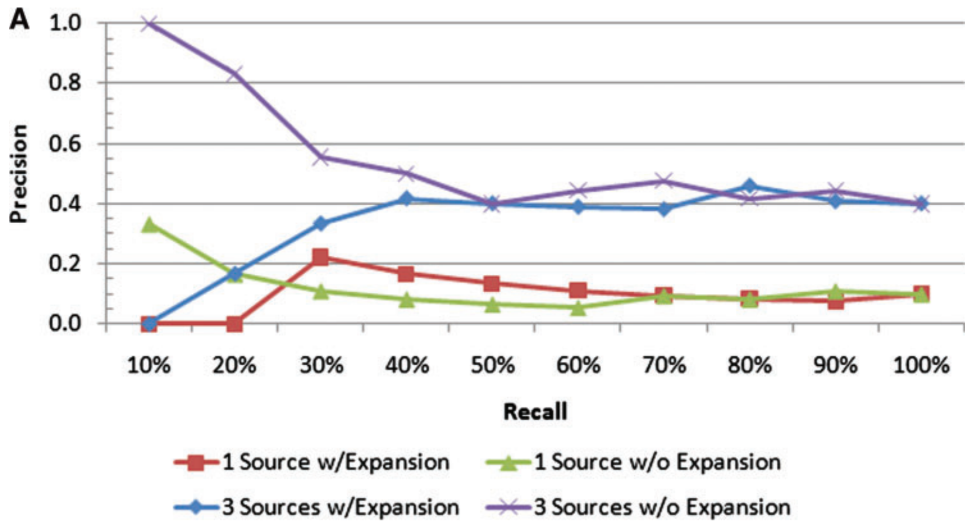
$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|} \tag{14}$$

and

$$\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}. \tag{15}$$

The precision values at the top 10 through 50 results for the four scenarios are shown in Figure 6. It is noteworthy to mention that the scenario with the best performance is the one that uses all three text sources but does not perform query expansion. This result helps to establish the usefulness of one of the major components of this search engine—the integration of multiple text sources to improve search—as demonstrated in this situation by the increased precision when all the text sources are utilized. The scenario with the second best performance occurs when there is the phenotype caption source is used alone in conjunction with query expansion, which gives credence to the utilization of ontologies, particularly with respect to query expansion, to help improve search results.

These ideas are reinforced further by looking at the some of the precision–recall plots for the individual queries. Consider Figure 7, which shows the results for the 'endosperm' and 'ear' queries. While all the scenarios for 'endosperm' perform comparably at the early recall levels, it is the scenario with all sources and with query expansion that maintains the highest precision for the higher recall levels. In addition, by utilizing these features, the result set is able to pick up more than 90% of all the variations whose affected body part is 'endosperm' versus roughly 45% for the single source search without expansion. The 'ear' query is a quite clear example of how extra text sources and query expansion can improve the search results, as the top 10 results all retrieve correct variations, and the precision



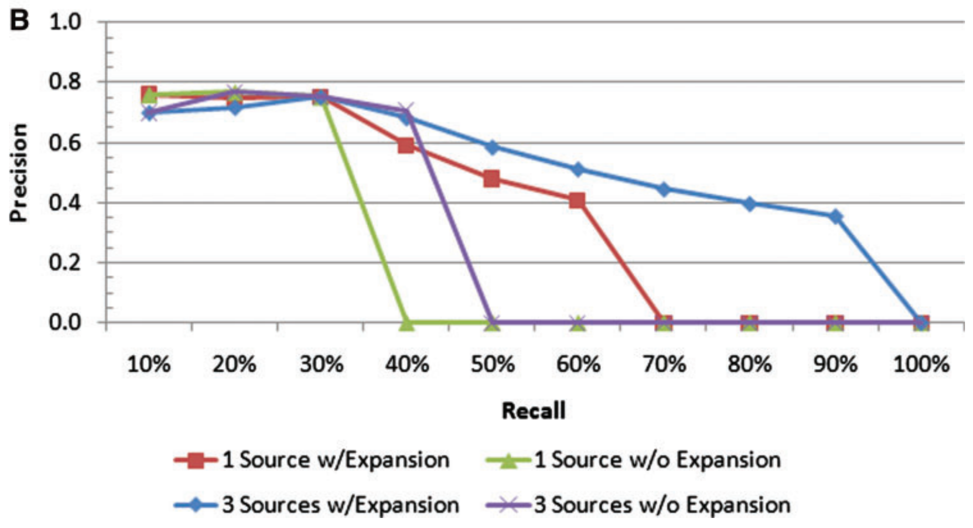


Figure 7. Precision–recall plots for two individual queries: (**A**) 'ear' and (**B**) 'endosperm'. For the 'endosperm query, all four scenarios have similar precision values at low recall levels; however, the scenario with all the text sources and query expansion includes many more of the relevant documents in the result sets than the others, as indicated by the higher precision at the higher recall levels. For the 'ear' query, the best performance is achieved by all the text sources with no query expansion.

stays above all the other scenarios for all but one recall level.

System scalability

The scalability of the system was assessed using three separate experiments. First, we designed an experiment to examine the effect of query length on retrieval time. Retrieval speed was measured for each of the test queries, which were composed of one, two or three terms. The average query speeds were then used to illustrate the trend in retrieval time as the length of the query increases. Figure 8 shows the average query time for each query using only the caption text source. The figure is organized by query length with the single-word queries on the left, the two-word queries in the middle and the three-term queries on the right. While the average times did increase as query length increased (0.43 s between query lengths of one and two terms, and 1.55 s between query lengths of two and three words), there were overlaps in the retrieval times of individual queries across the sets. It was noted that the more dominant factor in retrieval time was not the number of query terms, but rather the number of descriptions matched by the query terms.

Second, in order to determine the effect of the number of text sources on query performance, each of the test

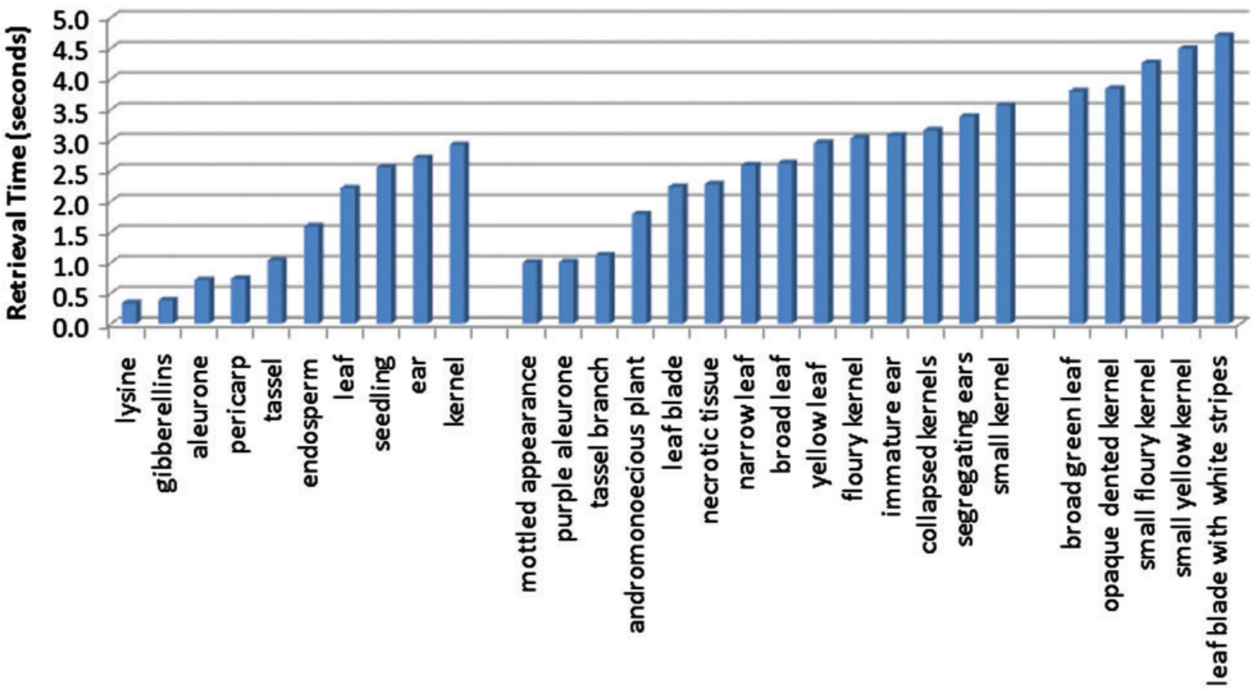


Figure 8. Average retrieval speeds for queries, ordered by query length. The leftmost group corresponds to the single-term queries, and these have the quickest execution times. The middle group contains queries consisting of two terms, and the query speeds for these are on average slightly slower than the single-term queries. The rightmost group of queries all contain three terms, and these are the most time consuming of the queries tested.

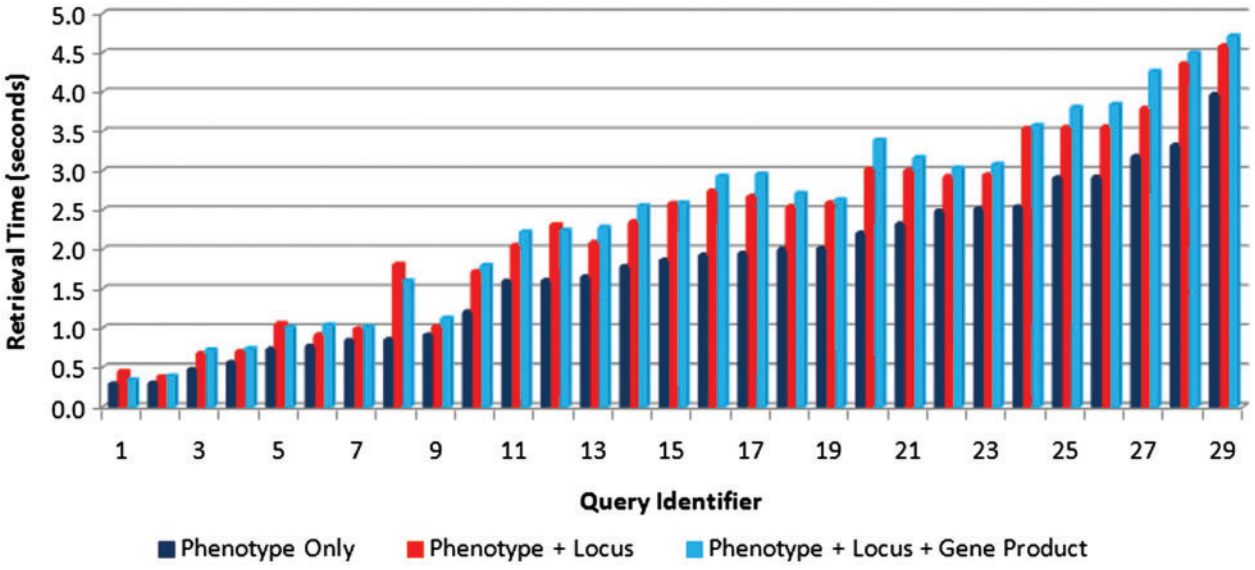


Figure 9. Comparison of retrieval speeds for each of the test queries (see Table 4 for the link between query identifier and query text) executed on one text source (phenotype captions only) (right bars), two sources (phenotype caption + locus descriptions) (middle bars) and three sources (left bars). There is a slight increase in execution as the number of sources increases.

queries was executed using just one text source (phenotype captions), two sources (captions and locus descriptions) and all three text sources. For each scenario, each query was executed 20 times with the average retrieval speeds measured and shown in Figure 9. This figure shows slight increases in retrieval time with the inclusion of additional text sources, but not a significant increase. The average increase in retrieval time for all the test queries in adding the second and third text sources was found to be 0.49 and 0.12 s, respectively.

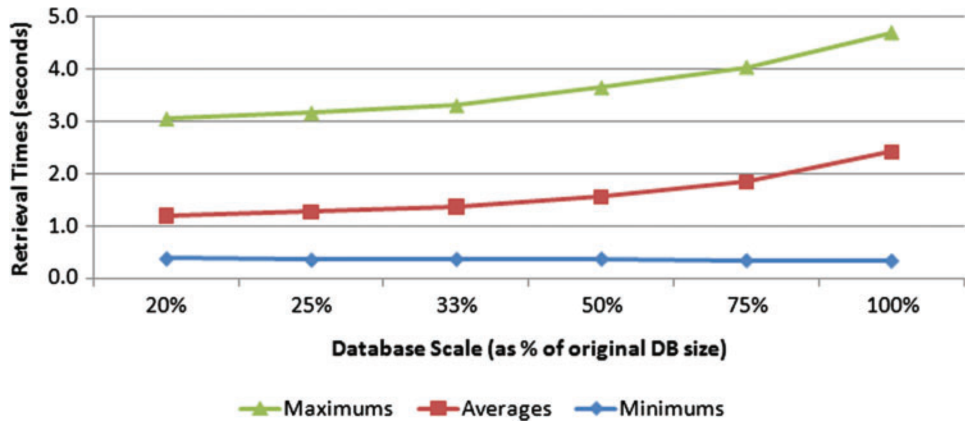


Figure 10. Effect of database size on retrieval speeds. Six different-sized test databases were constructed from subsets of the original data set. All the test queries were executed against each of these test databases, with the minimum, average and maximum query times measured. The trend suggests a nonlinear complexity for the search task.

Finally, we designed an experiment to examine how the size of the database, in terms of the number of documents, affects retrieval time. To perform this experiment, smaller databases were generated from the current data set by decreasing the size of the phenotype caption table. Six versions of each phenotype caption table size, which included 20, 25, 33, 50 and 75% of the original table size, were constructed. Each query was executed five times on each of the constructed databases. Figure 10 shows the trend in query performance as the number of documents in the database increased. The 5-fold increase in database size depicted shows an increase in retrieval time from 1.19 s to only 2.42 s. Should the size of the database eventually cause retrieval times to become unacceptable, various strategies could be employed to speed up the search. Performing the search on each text source in parallel and then merging the results of each of those threads could make significant inroads in decreasing retrieval speed. In addition, because the search engine is currently implemented via a PHP application that communicates with a MySQL database, porting portions of the search procedure to a higher performance language, like C++, could also allow for substantial gains in query speeds.

Limitations

As with any system, there are some limitations to this search mechanism. The first limitation is the requirement for exact matches when pairing terms to ontology concepts. This, at times, prevents the matching of concepts in documents to ontologies and thus prevents the ability to perform query expansion on those terms. For example, the term 'aleurone' on its own is not in the PO; however, 'aleurone layer' is a concept. Due to the large number of concepts that can be returned by performing a partial match to an ontology concept, it was decided that missing an occasional ontology match was a better alternative than falsely identifying many ontology pairings. Though stemming is performed on words in the text documents, ontologies and queries in an effort to reduce each term to its base word, variations in terms do exist that stemming does not account for. The result of this is the inability to match some variations of words (e.g. 'necrosis' and 'necrotic' map to 'necrosi' and 'necrot', respectively). A third limitation of this approach is the potential steep learning curves for users. Because of the built in flexibility, it may take some time for users to get accustomed to the available weighting options and it may take some exploration of the capabilities of the system to learn how to best utilize it.

Future work

There are several places where this project could be expanded through future work. First, we could investigate the use of linked grammars in this type of search engine, which may allow us to weight terms better by considering their word classes (e.g. noun, verb or adjective). Query expansion is performed automatically in our search engine; however, we also in the future plan to incorporate user-assisted query expansion, which allows the user to see the list of candidate expansions to the query and remove any unwanted terms from that list before the search is performed. We are also investigating the possibility of giving the user the flexibility to weigh individual terms in the query (including candidate expansions) rather than one weight for an entire class of terms. In addition, we would like to investigate using relevance feedback techniques to automatically determine user-driven weights for the various parameters. Currently only free-text sources are searchable through this mechanism; however, future development to integrate other kinds of sources into this search engine is planned. First, attribute fields, like those searched in current plant retrieval tools, will be incorporated. This would allow a phenotype description search to be limited to, for example, only those phenotypes linked to a specific trait or anatomical body part or to gene products of certain types. Leveraging attribute searches in conjunction with free-text information is promising. We also hope to explore the possibility of including nontext sources, like images and sequences, into the search mechanism as well. Phenotype searches using these modalities have already been developed; MaizeGDB has a sequence search mechanism called POPcorn available, and we have already explored in a previous work (20) how to represent phenotype images as feature vectors and perform image searches. In both cases, however, an approach to intelligently combine these various types of sources remains unstudied. Nevertheless, such a hybrid search mechanism that merges varied information sources would have great potential.

Finally, the MaizeGDB resource itself is in the early stages of a full redesign. One component of the redesign will be the inclusion of the system described here as MaizeGDB's phenotype search tool. Full deployment of the new resource including the VPhenoDBS: Maize search is anticipated for March of 2013.

Conclusions

We have developed a novel retrieval engine for maize mutant phenotypes. The main contribution is the development of multi-source retrieval, in which several interconnected text sources are utilized for searching and folders are retrieved and ranked for the user. Folder representation was defined, and a similarity measure was constructed so that traditional information retrieval techniques could be employed. The developed search engine also utilized domain ontologies for query expansion to help improve the accuracy and coverage of search results. This is the first such retrieval tool in the plant community. As plant genome databases continue to increase in size, next-generation search mechanisms will be needed to meet the needs of plant science researchers in a timely fashion.

Acknowledgements

The authors would like to acknowledge MaizeGDB for contributing the data for the retrieval engine as well as the MaizeGDB team itself for fruitful discussions.The multi-source retrieval was developed and implemented by J.G., who also prepared the manuscript. J.H. contributed to the development of multi-source retrieval and also was involved with manuscript preparation. M.S. provided maize expertise, ideas for making the retrieval engine most useful to the plant community, and provided guidance during manuscript preparation. C.J.L. provided insight regarding MaizeGDB, outlined unmet needs of maize researchers that were addressed by this work, and provided guidance during manuscript preparation. C.R.S. directed the project and provided essential guidance during development of the system as well as during preparation of the manuscript.

Funding

National Science Foundation (DBI-0447794) and the National Library of Medicine (NLM) Bioinformatics and Health Informatics Training (BHIRT) (2T15LM007089-17 to J.G.). Funding for open access charge: Shumaker Endowment for Bioinformatics.

Conflict of interest. None declared.

References

1. Lawrence,C., Harper,L., Schaeffer,M. *et al*. (2008) MaizeGDB: The maize model organism database for basic, translational, and applied research. *Int. J. Plant Genomics*, **2008**, Article ID 496957.
2. Jaiswal,P., Ni,J., Yap,I. *et al*. (2006) Gramene: a bird's eye view of cereal genomes. *Nucleic Acids Res.*, **34**, D717–D723.
3. Swarbreck,D., Wilks,C., Lamesch,P. *et al*. (2008) The arabidopsis information resource (TAIR): gene structure and function annotation. *Nucleic Acids Res.*, **36**, D1009–D1014.
4. Mueller,L., Solow,T., Taylor,N. *et al*. (2005) The SOL genomics network: a comparative resource for solanaceae biology and beyond. *Plant Physiol.*, **138**, 1310–1317.
5. SoyBase and the Soybean Breeder's Toolbox. http://soybase.org/index.php (7 April 2011, date last accessed).
6. Kurata,N. and Yamazaki,Y. (2006) Oryzabase: an integrated biological and genome information database for rice. *Plant Physiol.*, **140**, 12–17.
7. Baeza-Yates,R. and Ribeiro-Nero,B. (1999) *Modern Information Retrieval*. Addison Wesley, Reading, MA.
8. The Gene Ontology Consortium. (2000) Gene Ontology: tool for the unification of biology. *Nature Genet.*, **25**, 25–29.
9. Jaiswal,P., Avraham,S., Ilic,K. *et al*. (2005) Plant ontology: a controlled vocabulary of plant structures and growth stages. *Comp. Funct. Genomics*, **6**, 388–397.
10. Banerjee,S. and Pedersen,T. (2002) An adapted lesk algorithm for word sense disambiguation using WordNet. In: *Computational Linguistics and Intelligent Text Processing*, Vol. 2276. Springer, Berlin, Heidelberg, pp. 117–171.
11. Li,X., Szpakowicz,S. and Matwin,S. (1995) A WordNet-based algorithm for word sense disambiguation. In: *Proceedings for the IJCAI'95*. Montreal, Canada, pp. 1368–1374.
12. Widdows,D., Peters,S., Cederberg,S. *et al*. (2003) Unsupervised monolingual and bilingual word-sense disambiguation of medical documents using UMLS. In: *Proceedings of ACL 2003 Workshop on NLP*. Stroudsburg, PA.
13. Aronson,A. (2001) Effective Mapping of Biomedical Text to the UMLS Metathesaurus: The MetaMap Program. *Proc. AMIA Symp.*, 17–21.
14. Harabagiu,S. and Lacatusu,F. (2005) Topic themes for multi-document summarization. *Proceedings of the 28th Annual*

International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, New York, NY, pp. 202–209.

15. Lee,C.-S., Jian,Z.-W. and Huang,L.-K. (2005) A fuzzy ontology and its application to news summarization. *IEEE Trans. Systems, Man, Cybernetics*, **35**, 859–880.
16. Fu,G., Jones,C. and Abdelmoty,A. (2005) Ontology-based spatial query expansion in information retrieval. *Lecture Notes in Computer Science*, **3761**, 1466–1482.
17. Navigli,R. and Velardi,P. (2003) An analysis of ontology-based query expansion strategies. In: *Proceedings of the Internationl Workshop & Tutorial on Adaptive Text Extraction and Mining*. CavtatDubrovnik, Croatia.
18. Wollersheim,D. and Rahayu,W. (2005) Ontology based query expansion framework for use in medical information systems. *J. Web Inf. Systems*, **1**, 101–115.
19. Salton,G., Wong,A. and Yang,C. (1975) A vector space model for automatic indexing. *Information Retrieval and Language Processing*, **18**, 613–620.
20. Shyu,C.-R., Harnsomburana,J., Green,J. *et al*. (2007) Searching and mining visually-observed phenotypes of maize mutants. *J. Bioinformatics Comput. Biol.*, **5**, 1193–1213.