1 Introduction

We propose development of an *adaptive, personalizable, information management tool*, which can be configured and trained by an individual biologist to most effectively exploit the particular KBs and document collections that are most useful for him or her.

[condense this: problem, new approach/opportunity

The proposed tool represents a novel approach for monitoring scientific progress in biology, which has become a formidable task. The most popular technical solution to this problem is creation of structured biological knowledge bases (KBs): currently, Entrez Pubmed contains over 700 of these. The large number of biological KBs is a function of the diversity of biological phenomena that are studied, but the proliferation of such knowledge bases has led to difficulties in integration *of information across many knowledge bases [ct: DILS 2004,2005,2006]. Integration of biological knowledge is a practically important problem, as many biologists' research seeks explicate some particular biological phenomenon (e.g., the role of a particular gene), and in pursuit of such a goal, biologists will often draw on techniques and research results from many subareas.*

Since each research biologist's interests are to some extent unique, one might argue that in an ideal world, every biologist would have their own personalized *biological KB, which would be automatically constructed and populated to suit his or her needs. Unfortunately, building a personalized KB with currently-used techniques is impractical: populating a traditional KB with high-quality information is time-consuming and expensive, regardless of whether the information is collected by extraction from published reports, or by integration of external knowledge sources. Moreover, populating a KB is only useful when one can accurately predict what the user will need to know—there is little value in storing a biological fact that nobody will ever bother to access. This is a further obstacle to developing personalized KBs, since it is difficult for a KB designer to* predict *the knowledge an individual biologist will need to access.*

Rather than populating a KB based on expected future information needs, as predicted by the KB designer, our system will learn from every episode in which a biologist seeks information. *By saving and generalizing from these episodes, the tool will converge over time to an approximation of a personalized KB. We believe that substantial improvements over current techniques can be obtained by building on and combining recent advances in a number of technical areas, notably database systems and machine learning.]*

We will accomplish the following specific aims.

1. We will implement a *configuration tool*, which allows a biologist to select a set of externally-built structured KBs, information extraction components, and document collections that are relevant to his or her work.

2. We will implement a *universal information store*, allows the biologist to access to all the selected external knowledge sources via *typed similarity queries*. To issue such a query, a user specifies a set of "query terms" (which may be ordinary keywords, terms from a KB, or some mixture) and the *type* of the desired output. The answer to a query is a ranked list of KB objects (documents, entities, or KB terms) that are "similar" to the query terms. Queries are thus *soft* (i.e., both matches and near-misses are returned to the user) and also *schema-free* (i.e., they can be formulated without any knowledge of the underlying database schemas).

3. We will implement techniques for *adapting the query system* based on past experience. In particular, the system will also *learn* a context- or biologist- specific definition of similarity, allowing further personalization of the results. This definition of similarity will be "seeded" by the choice of external structured sources: for instance, if the biologist has selected data sources describing ribosome assembly in eukaryotes, then proteins which are involved in similar stages of the assembly process will be considered highly similar by default.

To accomplish these goals, we propose a highly directed program of study. In the first year of the grant, we will field a functional tool that will be used by working biologists; we will then extend this tool and grow the user community over the next several years. The proposed tool would be a valuable tool for working biologists, and a potential aid for curators of community KBs. It would also further leverage community KBs, emerging standards for knowledge exchange, and the increasingly accessible biological literature.

2 Background and Significance

2.1 Biological KBs and their limitations

[more on problem, from intro.

The proposed tool represents a novel approach for monitoring scientific progress in biology, which has become a formidable task. The most popular technical solution to this problem is creation of structured biological knowledge bases (KBs): currently, Entrez Pubmed contains over 700 of these. The large number of biological KBs is a function of the diversity of biological phenomena that are studied, but the proliferation of such knowledge bases has led to difficulties in integration *of information across many knowledge bases [ct: DILS 2004,2005,2006]. Integration of biological knowledge is a practically important problem, as many biologists' research seeks explicate some particular biological phenomenon (e.g., the role of a particular gene), and in pursuit of such a goal, biologists will often draw on techniques and research results from many subareas.*

Since each research biologist's interests are to some extent unique, one might argue that in an ideal world, every biologist would have their own personalized *biological KB, which would be automatically constructed and populated to suit his or her needs. Unfortunately, building a personalized KB with currently-used techniques is impractical: populating a traditional KB with high-quality information is time-consuming and expensive, regardless of whether the information is collected by extraction from published reports, or by integration of external knowledge sources. Moreover, populating a KB is only useful when one can accurately predict what the user will need to know—there is little value in storing a biological fact that nobody will ever bother to access. This is a further obstacle to developing personalized KBs, since it is difficult for a KB designer to* predict *the knowledge an individual biologist will need to access.*

Rather than populating a KB based on expected future information needs, as predicted by the KB designer, our system will learn from every episode in which a biologist seeks information. *By saving and generalizing from these episodes, the tool will converge over time to an approximation of a personalized KB. We believe that substantial improvements over current techniques can be obtained by building on and combining recent advances in a number of technical areas, notably database systems and machine learning.]*

[pre-emptive integration/extraction vs need-driven

design-by-committee compromise: biologist must interact as a yeast biologist, then as a pathway biologist, then....]

[include this somewhere: A more subtle limitation of biological KBs in general is that highly-structured KBs are limited in the types of information that they can encode. As an example, consider the problem of representing the subcellular compartment to which a protein localizes. This can be formalized as simply associating one or more compartment names (e.g., mitochondrion, golgi apparatus, etc) with a protein—a representation well-suited to storage in a KB. However, localization can be affected by the state of the protein (e.g., whether it is phosphorylated) or the state of the cell, and localization can change over time (e.g., during apoptosis) [52]. Furthermore, when proteins are found in two or more compartments, the relative abundance of proteins in each compartment may be biologically meaningful. Capturing these effects requires more than a field or two in a KB; indeed, it has been proposed to represent subcellular localization with a generative probabilistic model relating localization to other factors [53].

This example of localization illustrates how a functional biological property becomes progressively more difficult to capture in structured form *as more aspects of function are modeled. We believe that this situation* *will become more common in the future, as molecular biology progresses from the study of easily-structured genotype information to analysis of harder-to-structure relationships between genes and function. This observation is an alternative motivation for developing "soft" similarity-based query systems for biologists: such systems form a useful bridge between structured and unstructured knowledge.]*

2.2 Information extraction and retrieval

Information extraction (IE) is the process of extracting structured data from text. IE has been widely investigated as an alternative to manual curation as a means for creating databases that organize and summarize scientific results [6, 66, 59, 71, 69]. One subtask of IE is *named entity recognition* (NER), i.e., locating in text the names of entities such as cell types [40, 61] and gene or protein names [40, 34, 61, 8]; techniques for extracting factual assertions from text have also been developed [26, 61, 40, 60, 68]. IE for biological text has special problems (e.g., the complexity of the language and the prevalence of semantically ambiguous entity references) as well as special opportunities (e.g., the existence of large lexical resources.)

Over the last several years, many researchers have turned to *machine learning* methods to develop IE systems (e.g., [60, 4, 23, 46]). One indication of the rising popularity of machine learning methods for IE is provided by a recent workshop [36], in which a number of systems were experimentally compared on the task of extracting protein names from abstracts of biomedical publications. Seven of the nine systems entered in this "competition" were based on machine learning methods.

Recently, encouraging results have been obtained on biological IE problems using a type of learning variously known as *bootstrap learning* or *learning from weakly labeled examples* [30, 51, 74, 29]. Normally IE systems are trained using carefully hand-labeled documents—for instance, an NER system for gene names would be trained from text in which every instance of a gene name is bracketed by an expert human annotator. In bootstrap learning, training data is instead produced automatically by matching text against entity names or assertions from a KB. This sort of training data is noisier, but much easier to produce.

2.3 Entity normalization and textual similarity measures

When strings corresponding to entity names are extracted from biological text, it is often necessary to map these names with identifiers from canonical databases: for instance, the strings "IL-6" and "interleukin 6" might both be mapped to the gene MGI:96559. Although many biological KBs contain string "aliases" for the entities they contain, lists of aliases are often incomplete, or fail to capture all possible lexical variants of a name. Therefore, in order to find KB objects that map to an extracted entity name, it is often necessary to use some sort of "soft" matching scheme for text—i.e., a *textual similarity measure*.

A similar situation arises when databases records obtained from different sources are combined into a single database—in this case variations in representation can arise from systematic differences in the formats used to store data, or even typographical and OCR errors. The problem of identifying duplicate objects in heterogeneous data records has been studied by researchers in a number of areas, including statistics, databases, digital libraries, natural language processing, and data mining. (For recent surveys, see [5, 28]). The problem has been investigated under a number of names including record linkage, merge/purge, duplicate detection, database hardening, identity uncertainty, co-reference resolution, name-matching, and entity normalization. Here we will use the general term *information integration* for merging heterogeneous information derived from many sources, and the term *entity normalization* for the more specific task of associating a newly-extracted entity name with an existing entity.

For information integration tasks, a wide number of textual similarity metrics have been used. The similarity metrics most familiar to biologists are *edit-distance* methods (which are also used to compare genomic sequences in BLAST). However, in comparative experiments across multiple information integration tasks, metrics based on cosine similarity using *term-frequence/inverse document frequency* (TFIDF) weighting schemes are often more accurate [21]. TFIDF distance metrics were originally developed in the information retrieval community [62], and can be very efficient if appropriate indexing techniques are used.

[add summary of WHIRL?]

2.4 Schema-free querying

[should this go first?]

The "soft joins" used in WHIRL made it possible to integrate information spread across multiple databases without first establishing a set of common object identifiers. This eliminated some of the effort involved in data integration. However, it was still necessary to establish a concordance between the schemas used by the different databases. Likewise, because WHIRL's answers are always ranked lists, certain types of relaxations of a query are handled automatically (e.g., relaxing the phrase "scaffold protein" to the word "scaffold"). However, other relaxations of a query still must be formulated by the user (e.g., relaxing the strategies above to consider homology to scaffold proteins in other organisms).

Recent work on *schema-free keyword search* promises to lift these restrictions. Systems such as DBXplorer, DISCOVER, BANKS, XRANK, XKeyword, and others [1, 39, 3, 38, 41] allow users to submit keyword queries to a database. Answers to a query are tuples that (a) contain the queries submitted by the user, and (b) are the result of joining a small number of relations from the database. The advantage of such an interface is that the user need not formulate the join, and hence, need not understand the database schema.

In many schema-free query languages the answers are ranked, for instance by the complexity of the sequence of join operations needed to produce the tuple. Most of these query languages operate on relational databases or XML databases; however, in many cases, the underlying data model is simply a directed graph in which some nodes contain text. Hence these methods can be easily extended to operate on the sorts of KBs used in the biological community, including any KBs represented using the Open Biomedical Ontology (OBO) standard [56].

One important potential use of schema-free query languages is to query KBs that were formed by automatically combining the facts in many smaller KBs. Even if the smaller KBs are well-structured, it might be that schema of the larger combined KB is not completely understood by *any* single person—let alone the end user.

2.5 Significance

[significance of problem as stated:]

As noted above, each research biologist's information needs are unique, and these needs frequently encompass many diverse areas. The tool we propose will be freely distributed, and can be used as a personalized information collector by any working biologist. Beyond the obvious reduction in the time spent collecting information, our emphasis on search using *personalized, trained similarity metrics* that work across *multiple community KBs* increases the likelihood of serendipitous discovery of work in another research area that is synergistic with one's own.

Perhaps most importantly, our proposal will change the economics of KB creation. Soft, ranked-retrieval queries coupled with user-driven learning allows some of the cost of creating a structured KB to be shifted from the *producer* of the knowledge to the *consumer*. This flexibility will make it possible to explore new paradigms for exchange of structured biological information, beyond the well-studied approach of communally-created KBs.

[de-emphasize?] This project will develop methods for learning and information extraction that will benefit many applications beyond the one we propose. Algorithms and results will be disseminated through publications in the relevant computing conferences and journals, and data and software developed under the project will be shared with the research community via a project web site.

3 Preliminary Results

3.1 Information Extraction and Normalization

Cohen has worked extensively on machine learning methods in the area of information extraction, on both biological problems (e.g., [54, 23, 54, 46]) and other tasks (e.g., [12, 18, 22, 64, 19]). He is author of an open-source toolkit for information extraction called Minorthird [50] that is in wide use (with over 4,300 SourceForge downloads to date). He has developed state-of-the-art methods for a number of specific problems of biological interest, including gene-protein entity extraction [46] and structuring of biomedical paper captions [23]. Cohen also was the principal architect of the IE components of several larger-scale systems used in industry and academia: the WHIRL system, which collected data from several dozen subject-specific web sites [12, 13]; the WL^2 system, a commercial "wrapper induction" system used to extract job-related information from thousands of corporate sites [18]; and the SLIF system [55, 16, 23, 45, 54, 46], which processes on-line articles to extract information and images about the subcellular localization of proteins.

The current popularity of TFIDF-related metrics for information integration (e.g., [35, 65, 57]; for a recent tutorial, see [47]) largely derives from their successful use in Cohen's WHIRL system [14, 15], which is discussed below. In addition to prior published work on evaluating and developing textual similarity metrics, Cohen also authored SecondString [20], a widely-used open-source package of textual similarity metrics.

3.2 Similarity search for databases

[replace this glue: The argument above suggests that it is useful to search unstructured data, and also to combine and query structured data from multiple sources. One existing system that supports these sort of searches naturally is the WHIRL system [14, 15].]

As an example of a plausible biological search task, consider a biologist studying ribosome assembly in yeast, and imagine that previous experiments have suggested that an unknown "scaffold protein" p about 550 residues long is involved in this process.[1] The unknown protein p might well be a scaffold protein for other, non-ribosomal yeast proteins; or, it might be homologous to known scaffold proteins from some other organism. The widely-used Gene Ontology (GO) [25] includes a rich ontology of protein function; however, there is no GO concept corresponding to the (informal) term "scaffold protein". However, there are several GO concepts whose name (or definition) includes the phrase "scaffold protein". A simple text search also reveals many proteins that are referred to as "scaffold proteins" in the literature, including some (e.g., CDC53p, CDC11p) that are immediately recognizable as yeast proteins.

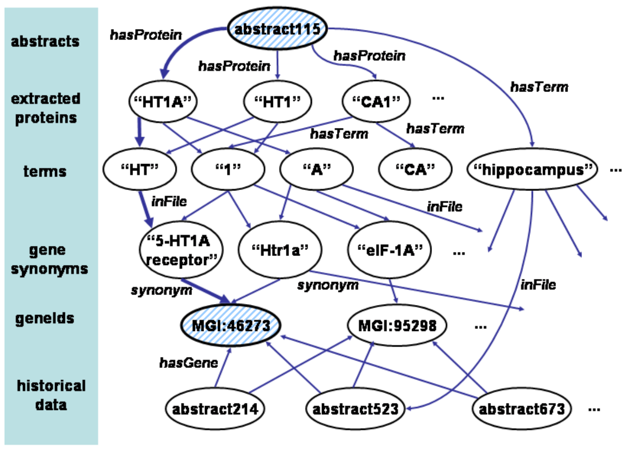
This suggests some strategies for finding candidates for p. One might look for strings s' that are (a) extracted by a gene-protein NER system from sentences containing the phrase "scaffold protein" such that (b) s' is similar to some (appropriately large) known yeast protein p'. If we replaced "similar to" with "equal to", this strategy might be implemented as follows in SQL:

```
SELECT g.* FROM extraction as ex, yeastGene as g
WHERE ex.containingSentence LIKE 'scaffold protein' AND ex.extractedString=g.name
```

(Here "LIKE" is the SQL "contains substring" operator, and we neglect the check on protein size, for brevity.) Alternatively, one might look for yeast proteins p' that are tagged with one of the GO terms whose name contains the phrase "scaffold protein". This strategy might be implemented in SQL as follows:

```
SELECT g.* FROM geneAnnotation as ann, yeastGene as g
WHERE ann.geneId=g.id AND ann.name LIKE 'scaffold protein'
```

[1] A scaffold protein acts as a substrate to which two or more other proteins bind, allowing them to interact.



	MAP	Δ(baseline)
BioCreAtIvE mouse-gene normalization		
baseline: NER + soft match	36.8%	–
graph-similarity ranking	64.0%	*+73.9%*
with extra links & learning	71.1%	*+93.2%*
also with weakly-trained NER	80.1%	*+117.7%*
email (CSpace) threading		
baseline: TFIDF similarity	36.2%	–
graph-similarity ranking	40.2%	*+11.0%*
with learning	50.3%	*+40.1%*
email (CSpace) person-name normalization		
baseline: soft match + nicknames	49.0%	–
graph-similarity ranking	66.3%	*+35.3%*
with learning	89.0%	*+81.6%*

Figure 1: Left, a simplified version of the graph used for associating gene identifiers with abstracts. Right, mean average precision (MAP) on three classes of problems: a set of 250 document-level gene extraction and normalization tasks from Pubmed abstracts, and two email-related tasks (see text).

If no plausible candidates were found, then these strategies might be relaxed, e.g., by considering GO terms with names that contain the shorter phrase "scaffold".

WHIRL extended conventional SQL by adding textual similarity as an additional operator: WHIRL's similarity operator "∼" could be used much as "=" operator would be used in a join (or as a "LIKE" operator would be used in a WHERE clause). WHIRL used a combination of artificial-intelligence search methods and information-retrieval (IR) indexing to compute these "soft joins" efficiently. The answer to a WHIRL query is a *ranked list* of database tuples, ordered by how well they satisfy the similarity tests in the WHERE clause. Thus a protein named by NER-extracted string s that was highly similar the corresponding protein name p would be ranked ahead of a string a' that was only weakly similar. Likewise, GO terms that contain only the term "scaffold" appear in the list, but would be ranked below terms that contain the phrase "scaffold protein".[2]

WHIRL's primary application was for querying relations extracted from multiple heterogeneous databases that contained no common object identifiers, but did contain similar informal textual "names" for objects. For databases of these type, conventional joins can be replaced with WHIRL "soft" joins in which equality tests on object identifiers were replaced by similarity constraints on "names", eliminating the necessity for creating common object identifiers to link the two databases.

3.3 Schema-free similarity queries

[do we want to give a pointer to the overall system architecture?]

3.3.1 Evaluating graph-similarity queries

A sample graph. To conduct a preliminary evaluation of similarity search for biological problems, we built a graph using the mouse-organism data from the BioCreAtIvE challenge [37], as shown in Figure 1. The graph contained 52,594 gene identifiers; 183,142 strings that are synonyms for these genes; 5000 Medline abstracts, each of which is associated with a set of related gene identifiers; and 250 test abstracts. The

[2]GO terms that contain the term "protein" but not scaffold would also contribute to the list, but those genes would have a very low score. Scores would be low because the word "protein" is not very informative in the context. In general, TFIDF assigns less weight to matches on more frequent words.

graph also contained strings extracted by two simple gene taggers that were trained on the YAPEX corpus [31], one tuned for recall and the other for precision. These taggers are fairly accurate on the YAPEX test corpus (with F1=73% for the better-performing tagger[3]) but quite inaccurate on the BioCreAtIvE mouse abstracts (see Table 1). One of the conclusions of our preliminary study is that surprisingly accurate query performance can be obtained with rather inaccurate NER components.

A set of target queries. We evaluated the accuracy of similarity queries in which the query was the node for a Medline abstract, and the desired output type was "gene identifier". We assume that the desired response to such a query would rank gene identifiers that appear in the abstract higher than ones that do not. Using labeled test data from BioCreAtIvE (which associates an abstract with identifiers for the genes that appear in it) we can evaluate how well the ranking achieves this goal. We evaluated performance with the widely-used IR measure of *mean average precision* (MAP).[4]

A similarity metric. One metric we believe to be promising is a certain generalization of the *heat diffusion kernels on graphs* of Kondor and Lafferty [44], variants of which have been widely used in the machine learning community for semi-supervised learning [70, 10, 67, 76, 75]. Heat diffusion kernels have in turn been shown to generalize Gaussian kernels, in the sense that the continuous limit of heat-diffusion kernels on a two-dimensional grid is a Gaussian kernel. Our metric can be best described as a random walk on a graph, analogous to that used by the PageRank algorithm [58]. More formally, similarity between nodes in a graph is defined by a *lazy random walk process*. To walk away from a node x, one first picks an edge label ℓ; then, given ℓ, one picks a node y connected to x via an edge labeled ℓ. At each step there is also some probability γ of simply and "emitting" the current node x. The similarity of a node z to x is simply the probability of emitting z in a lazy walk starting from x. Intuitively, this measures the similarity of two nodes by the weighted sum of all paths that connect the nodes, where shorter paths will be weighted exponentially higher than longer paths. One consequence of this measure is that information associated with paths like the one on the left-hand side of the graph—which represents a soft-match between a likely-protein and a synonym—can be reinforced by other types of paths, like the one on the right-hand side of the figure.

The graph-based similarity method has a MAP score of 64.0%. For comparison, we also implemented a baseline method which simply applies the better-performing library NER component to the abstract, and then maps each extracted name to the best soft match to the synonym dictionary. The baseline obtained a MAP score of only 36.8%.

3.3.2 Learning methods

No fixed similarity measure will be best for all purposes. Fortunately, a personal query system can be improved by *adaptation*—by *learning* how to best answer queries of its specific user. In addition to adapting to a specific user, learning might be be used to optimize performance at other levels of aggregation. For instance, most biologists will want to construct a "standing query" (a query to repeated over and over at regular intervals) that retrieves new articles in their specialty. Alternatively, if a particular class of queries is repeated, the performance on that class might be optimized by examining past data. So far, we have obtained preliminary results using two types of learning.

Learning to re-rank answers. In this approach, the top N outputs of a ranked list $x_1, \ldots, x_N$ are collected, and re-ordered in decreasing order according to some function $f(x)$. The function $f(x)$ is learned

[3]F1 defined as $\frac{2}{(1/p+1/r)}$, where p is precision (i.e., the number of correct positive predictions divided by all positive predictions) and r is recall (i.e., the the number of correct positive predictions divided by the number of positive examples).

[4]Briefly, consider a ranked list that has correct entries at exactly the ranks $K = k_1, \ldots, k_n$, and assume that the user will scan this list of answers and stop at some particular "target answer" k_i of interest. Define the *precision at rank* k_i, *prec*(k_i), to be the number of correct entries up to and including k_i, divided by k_i, and the *average precision* (AP) to be the average of *prec*(k) for each position $k_i \in K$. (If some correct entry does not appear in a ranking, then *prec* is defined to be zero for that entry.) MAP is the average of AP across all queries.

	Entity-level F1			
	mouse		drosophila	
	−repeats	+repeats	−repeats	+repeats
baseline methods				
soft match to KB-specific dictionary	57.6%	55.5%	40.1%	39.8%
trained on GENIA (human cell signalling)	35.7%	49.6%	18.9%	36.6%
trained on YAPEX (protein-protein interaction)	39.3%	56.9%	34.5%	45.8%
trained on 50 strong-labeled KB-relevant documents	38.6%	49.6%	15.3%	30.4%
with dictionary-based features	56.2%	57.5%	37.0%	40.4%
bootstrap learning				
trained on 1000 weak-labeled docs	50.6%	65.9%	52.4%	56.4%
with dictionary-based features	66.8%	71.1%	66.1%	63.0%
Δ(line above *vs* best baseline)	*+16.0%*	*+23.6%*	*+64.8%*	*+37.5%*

Table 1: Comparing NER components learned by conventional means on "mismatched" training sets; by conventional means on a small set of strong-labeled documents; and by bootstrapping using a dictionary and weakly labeled documents. F1 measures on strong-labeled documents are estimated by 10-fold cross-validation.

from data. One advantage of this approach is that f can be learned using constraints of the form "x_i should be ranked above x_j", which are relatively easy to collect from users.

We used the voted perceptron algorithm [33, 24] to learn the ordering function f. For these experiments we also extended the graph by explicitly linking extracted gene names to synonyms that are similar according to the "SoftTFIDF" metric [21], and also adding "short-cut" edges which link abstracts to gene identifiers found by a baseline method for the BioCreAtIvE task. With this graph structure, learning improves the MAP score on the BioCreAtIvE task to 71.1%. (We used here a very simple and general feature set; in brief, the features for x_i summarize the types of edges encountered on all paths from the query keywords to x_i.)

Generality of the approach. We also evaluated several other types of queries in the domain of personal email [48]. Here the graph nodes are derived from email messages and an address book, and the the relationships are derived primarily from header information (e.g., sender and recipients information). We studied similarity queries for finding other messages from an email thread given one email from the thread as a query [48], and for finding attendees of a meeting [49]. In a third study, a similarity query consisted of two graph nodes—a token in an email message that corresponds to an ambiguous personal name (e.g., "John" or "Kumar"), and the email message that contained that name—and the goal was to find the identifier of the indicated person [48]. This problem is very like the gene-protein entity normalization problem studied in BioCreAtIvE.

In every one of these studies, substantial improvements were obtained by the graph-based approach over a plausible baseline, validating the generality of the approach. Figure 1 summarizes one of the email threading tasks and one of the person-name normalization tasks we considered. On these problems we used a variant of boosting [32] for the learning method.

3.4 Bootstrap learning of IE components.

Machine-learned IE components often perform poorly on text that is statistically different from the text on which they were trained [46]. Unfortunately, the data of most interest to the end user of QUERENDIPITY is not likely to be statistically identical to the data used to train the library of machine-learned NER components. This "data mismatch" problem is a fundamental obstacle to any personalized query system.

The preliminary results of Figure 1 show that our graph-similarity approach can provide good performance for answering certain types of queries, even when the underlying NER components are trained on mismatched data. (The YAPEX corpus on which they were trained is focused on protein-protein interactions, while the corpus on which they were tested focused on the mouse model organism.) However, it seems likely that performance can be improved further by adaptively customizing the library of IE components used in QUERENDIPITY.

As noted in the background section, NER components are usually trained on large amounts of manually annotated training examples, consisting of documents with the position of every named entity marked. Unfortunately, this training data is difficult to produce. In recent work [73] we have explored training an NER components with easier-to-obtain data: a synonym list (i.e., a list of entity identifiers, plus synonyms for each identifier) and *weakly labeled documents* (i.e., documents associated with identifiers of the entities that appear in the document.) Weakly labeled documents can be obtained automatically by analysis of curated KBs.

Our approach consists of four steps. First, we look up, for each abstract, its associated gene identifiers, and we label all possible locations of synonyms associated with these identifiers in that same abstract. Second, we train extractors on these weakly labeled abstracts, using word features such as string similarity to synonyms. We also investigated a pre-processing step of removing from the training set sentences not containing any weak labels, and a post-processing step that exploits intra-document repetition of names by soft-matching every instance of an extracted name against the document in which it occurs, and classifying every such soft-match as a new protein name. The post-processing step exploits the fact that when the same word or phrase is repeated several times in the same document, it tends to have the same meaning in each time: e.g., if "HTA1" can be identified as a protein name one place in a document, it can be assumed to be a protein name every time it occurs in a document.

In Table 1 we present these results. The soft-match against repeated names usually but not always improves performance, so we present results both with and without it, under the columns +repeats and −repeats respectively. As weak-labeled training data, we used 1000 abstracts from the BioCreAtIvE training set, and for test data, we hand-labeled 50 abstracts from the BioCreAtIvE devtest set in the conventional way. (We note that strong-labeling even 50 documents required several hours of an expert's time.) We trained a series of NER components using the same features and learning method,[5] but different training sets.

NER components trained on "mismatched" labeled corpora such as YAPEX [31] and GENIA (which focuses on signalling in human cells [43]) achieved F1 scores between 18% and 57%—only somewhat better than a hand-coded soft-matching scheme against the dictionary, or training on a small set of 50 hand-labeled documents. The best weak-labeled NER components achieved F1 of 71.4% for the mouse data, and 66.1% for the fly data. Importantly, replacing the original NER components (trained on the YAPEX corpora) with the NER components that were weakly trained on the BioCreAtIvE mouse data provides a substantial improvement in the performance of entity-normalization queries evaluated above: as Figure 1 shows, MAP rises from 71.1% to 80.1% when a more appropriate NER component is used. Average recall in the top 10 also rises from 85.8% to 89.9%.

As part of this project, we propose using similar bootstrap learning methods to develop new NER components that are more closely tuned to the interests of the biologist/user of QUERENDIPITY. Structured and semistructured KBs will be used to generate lists of entities, and later facts that relate multiple entities. These entity names and facts will be matched against unstructured text to form training data to bootstrap-train new NER components. The new NER components will then be added as additional annotators for unstructured text, which can be used to improve query performance for QUERENDIPITY.

3.5 Summary of preliminary results.

Table 2 summarizes the results for the gene-name normalization task. In addition to MAP scores, we give two additional measures. Participants in BioCreAtIvE were evaluated by pooling all predicted abstract-protein matches across the test set, and computing the F1 score of this pool. Unfortunately, the ranking

[5] Minorthird's implementation of voted-perceptron training for HMMs [24].

	baseline	graph-based ranking		
	(NER+softmatch)	−rerank	+rerank	+rerank
		−bootstrap	−bootstrap	+bootstrap
MAP	36.8%	64.0%	71.1%	80.1%
avg max F1	42.1%	69.5%	75.5%	83.7%
recall@10	35.3%	82.4%	85.8%	89.9%

Table 2: Summary of results for the BioCreAtIvE mouse-gene normalization task. The table shows MAP score, average maximum-obtainable F1, and recall considering only the top 10 answers. Results are given for the baseline, and three variants of the graph-based ranking system: with no learning, with learning-to-rerank only; and with learning-to-rerank over a graph populated with results of a bootstrap-learned NER component.

systems we have implemented can (and do) assign quite different absolute scores for each query, so there is no straightforward way to combine their predictions into a single pool. As a measure of relative performance, we computed the maximal F1 score (over any threshold) of each ranked list, and then averaged these measures over all queries. The averaged maximum F1-measure of the best graph-based system is 83.7%. This compares favorably to the F1 scores of BioCreatIvE participants: the median F1 score for this task was 73.8%, and the top score was 79.1%.

As another way to put these results in context, imagine that the biologist using QUERENDIPITY is interested in obtaining high recall with minimal effort–i.e., he or she wants to inspect a small number of answers and find almost all genes that occur in an abstract. The NER-based baseline system using library components returns very short lists (with 1 or 2 entries), but finds only a little more than 35% of the genes (and even the NER component tuned for high recall finds only slightly more than half of the genes). Using the graph-based method, the user can obtain an average recall of 82.4% by looking at only the top 10 results. This result is improved to almost 90% with learning.

4 Methods

Above we have reviewed methods that solve (imperfectly) many related tasks. Named entity recognition (NER) finds entity names in text, and entity normalization maps these names to entities from a KB. Information integration methods (such as WHIRL's soft joins) connect tuples in one KB to tuples in another. Finally, schema-free queries can be used to query the information in a KB with an unknown schema—for instance, a KB formed from the union of many smaller KBs. If these methods were sufficiently accurate, they could be used in concert to automatically combine and then query any collection of KBs and documents, thus allowing a biologist to easily create a personalized KB. Unfortunately, all of these methods are usually inaccurate, unless they are manually tuned by an expert on a particular problem.

We propose a novel solution: a system that will *reliably* answer end-user queries based on *imperfectly* extracted, normalized, and integrated information. To do this, the system will exploit: the *redundancy* of information in multiple sources (e.g., structured databases and text, or multiple documents); *similarity metrics* that robustly combine information from multiple KBs and text documents; and *learning methods* that tune the system's performance, in response to user feedback, after every episode in which the user seeks information.

In the remainder of this section we will present the architecture of the proposed system (henceforth QUERENDIPITY) and then present preliminary component-level evaluations we have conducted to date. These experiments strongly support the viability of this approach. We next discuss our longer-range research plans, and our plans for engineering a working system, obtaining practical experience with users, and conducting further systematic evaluations.

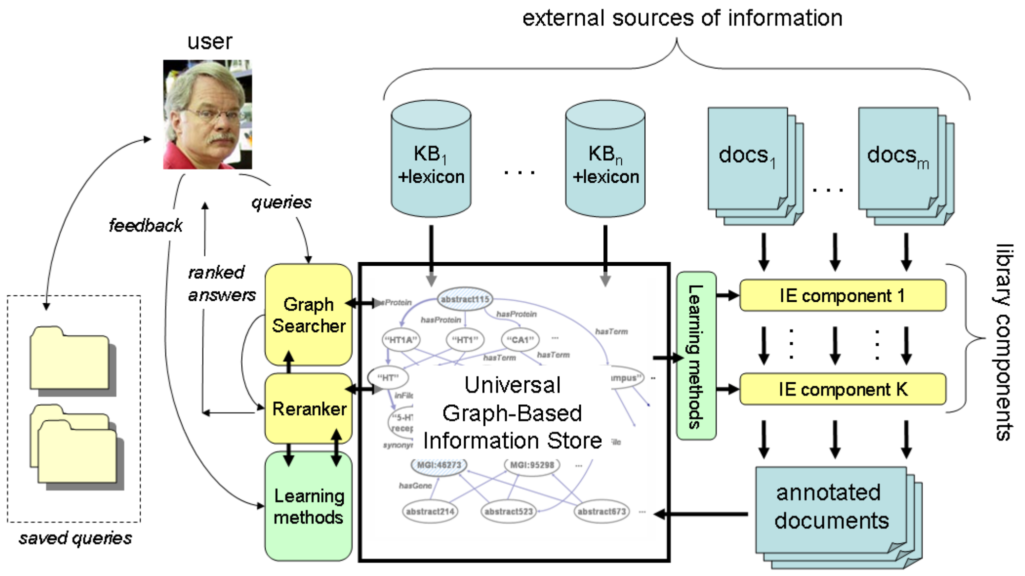


Figure 2: Overall system architecture. The user/biologist *configures* the system by selecting KBs, document collections, and IE components from a library. The IE-annotated text and KBs are loaded into a single data store: this is conceptually a directed graph, the nodes of which are KB entities, documents, or extracted strings; and the edges of which are KB relationships or textual similarity relationships. The user then *trains* the system: he submits a series of *similarity queries*, each of which is a set of keywords and a desired output type; the system answers these queries, with a ranked list of graph nodes of the desired type; and the user optionally provides feedback on the desirability of these answers.

4.1 Specific aim 1: End-to-end loosely integrated system

[explain why this sort of system integration is important]

Figure 2 shows the architecture of QUERENDIPITY. The user/biologist first *configures* the system by selecting KBs, document collections, and IE components from a library. A separate tool will be provided to aid in configuration. We assume that all KBs are "lexicalized"—i.e., that for every concept, there are one or more strings that indicate how that concept is normally written in biological publications. Document collections could also be streams of documents (e.g., from a journal, or a standing query to Pubmed) that are added to QUERENDIPITY as they become available; a similar mechanism could be used to periodically load updateed versions of, or updates of, an external KB.

After configuration, the selected text is annotated using the IE components, and this annotated text, together with the selected KBs, are loaded into a single data store. This *universal graph-based information store* contains a single large directed, labeled graph, that will be constructed as follows. Nodes are created for every concept in an imported KB; for every imported document; for every token in an imported document; and for every string that is either (a) the name of some KB concept or (b) an entity name extracted by an IE component for an imported document. Edges between nodes are created that correspond to KB relationships; textual containment (e.g., a token or extracted string will be connected to the document that contains it, and vice-versa); and textual similarity (e.g., strings can be linked to highly-similar strings). Nodes in the graph are also given a *type*, such as *string*, *token*, *document*, or *concept* (the last being a node from an imported KB). "Concept" nodes may also be given secondary types, derived from the KB (e.g., "gene" or "cellular compartment").

To use the information store, the user submits a query that is simply a desired output type, plus a set of keywords or KB entities. (For instance, the keywords "scaffold" and "ribosome" and the target type *SGD gene*.) The response to a query is a list of correctly-typed nodes from the graph, ordered by their "similarity" to the query. Clearly, the notion of "similarity" used here is crucial: the notion we propose, together with preliminary results on its utility, is discussed below.

After a query is answered, QUERENDIPITY optionally collects feedback from the user on its answers. The minimum information required for learning will be a desired partial order over the answers (i.e., specifying that some answers are better than others.) Over time QUERENDIPITY will learn how to produce more desirable answers using a combination of learning methods, as discussed below. (For instance, QUERENDIPITY might learn to give more weight to connections between a keyword and a protein that pass through a GO term definition.)

Another type of learning is suggested by the fact that some of the documents collected will contain text that supports some of the facts in the KB. This provides an opportunity to improve the library of IE components with bootstrap learning. Preliminary experiments (below) strongly suggest that both of these learning methods can be useful.

The interface will also allow a user to save and repeat a query (e.g., if more information has been added to the merged KB). It will also allow a user to group similar queries into folders; in this case, feedback for all queries in a folder will be used for training. Folders and queries can be shared among users, allowing members of a lab (for instance) to collaboratively train QUERENDIPITY.

4.2 Specific Aim 2: Benchmarks for Evaluation

We propose to conduct two types of evaluation of the system. First and most importantly, the system will be used by the labs of two working biologists (co-PIs Woolford and López). A second type of evaluation will be systematic, reproducible experiments on sets of saved queries.

The Woolford lab's main research area is the pathway of assembly of ribosomes in eukaryotes, using the yeast *s. cerevisiae* as the experimental organism. Ribosome assembly is a highly conserved process, involving many proteins in addition to the ones which form the ribosome itself. Many of the non-ribosomal proteins essential for ribosome assembly are also important for cell growth or proliferation, and most have homologs in humans and other metazoans. The Woolford lab uses a variety of experimental methods, including genomics, proteomics, and molecular biology, as well as classical genetic and biochemical approaches. The Woolford lab's emphasis on yeast makes them an ideal initial population of users, since structured databases for yeast are mature and well-integrated, and many text-mining tasks (e.g., NER) are simplified by the well-developed nomenclature used by yeast biologists.[6]

The López lab's main area of recent study is splicing of pre-RNA—in particular, study of a mechanism of splicing used for large introns called *recursive splicing*, in which an element called a *recursive splice site* first functions as a 3' splice site, and then regenerates a functional 5' splice site after ligation to the upstream exon. A number of mutations that cause human disease are known to work by disrupting splicing: examples of such diseases are breast cancer and spinal muscular atrophy. The López lab works primarily with Drosophila and cultured mammalian cells, using molecular genetic methods, as well as computational methods.

In later years of the grant we will collect sets of sample queries from members of the Woolford and López labs, along with the desired outputs for these queries. We will then use this data to perform systematic experiments with variants of QUERENDIPITY to evaluate and improve performance. Below we will report a series of preliminary experiments of this sort that we performed with data that is already publically available.

[6]For the Woolford lab version, QUERENDIPITY is an acronym for Query-based User-guided Exploration of Relations and ENtities in Data Integrated Probabilistically or Identified in Text about Yeast. An organism-independent acronym for QUERENDIPITY will be developed in later stages of the grant.

3 months	initial query system, based on a small subset of SGD; unit tests.
6 months	initial query system scaled to 50M words of text and 1-2M concepts from SGD learning methods for re-ranking
12 months	configuration tools; 20+ realistic, performance tests from Woolford lab; comparative learning experiments for re-ranking and parameter learning
18 months	initial version of the López-lab KB; alternative indexing and pruning methods; bootstrapping learning for NER
24 months	realistic performance tests from López lab; begin study of query efficiency; bootstrapping learning for IE of facts; learning for value functions
30 months	documented beta version available to public; more experimental studies
36 months	tools for sharing of trained, configured KBs; improvements as dictated by user requests
42 months	improvements as dictated by research opportunities and user requests
48 months	completion of grant

Table 3: Proposed development milestones, relative to start of grant.

We will describe the graph constructed, the similarity metric used, and our results, as well as additional results obtained in combination with learning methods.

4.3 Software Development Plans

[re-write as specific aims]

We will field a prototype within the first six months of the grant. To reduce complexity of the first prototype, we will initially allow only a small number of possible input formats—for instance, KBs encoded in the Open Biological Ontology (OBO) flat file format, comma-separated-value tables, and unicode document collections in a single flat directory—and will not allow incremental updates to the universal graph-based information store. All NER components will be selected from a fixed set of library components trained on standard annotated corpora. Also to reduce complexity, we will aim at a smaller-scale system focused on the model organism *s. cerevisiae* (yeast). We will also invest in a large-memory server machine, so that the first prototype can use a memory-based data store.[7]

Internally, all concept nodes and relations in the graph KB will be mapped to URIs, to facilitate a transition to RDF-based representations. We will use open-source components but non-GPL components as much as possible to make it possible to distribute the code as widely as possible. All code developed will be open-source. The query system will be server-based, with the user interface implemented in an appropriate AJAX toolkit. The graph data store will be implemented on top of of several candidate component packages, possibly JENA [9].

In second half of first year, will work closely with members of Woolford's lab (who work primarily with yeast) to evaluate the utility of the system. One advantage of this focus is that named entity recognition and normalization for yeast proteins is relatively straightforward, so we can focus on query methods, user issues, and entity-similarity metrics, without duplicating other researchers' work on gene-protein NER. In parallel, we will work on scaling the system to larger collections.

In the second year, we will extend the system to support the López lab, as a first test of generality. The López lab works with Drosophila and cultured mammalian cells, and NER systems for these organisms are less mature currently, although progress in this area is rapid. Key technical issues will include scaling the system to the larger collections of potentially relevant documents and KBs. We also plan to present demonstrations at key biology and bioformatics conferences, in order to recruit further "beta" users. In year

[7]We estimate our initial graph KB will be about 50 million words (approximately 60,000 yeast-related Pubmed abstracts, plus additional user-selected text) and 1-2 million additional nodes (primarily from the SGD database.)

two we will also collect a series of sample queries (with user feedback, and KB snapshots on which the feedback was provided) to use in systematic evaluation of the system.

In years three and four, we will gradually increase the user population and explore further scientific issues as suggested by user experience. We will also gradually increase the size of the sets of user queries that we have available as benchmarks. These will be used as for systematic evaluation of the system according to standard metrics for ranked retrieval query systems (e.g., mean average precision). After the completion of the grant, we expect that QUERENDIPITY will have an active user community, and can continue to function as a collaborative open-source resource.

Specific milestones are listed in Table 3.

4.4 Component-level specific aims

These experiments above strongly support the claim that a well-engineered adaptive query system would be a use to working biologists—even one that used only the technology developed to date. However, there are many ways that our preliminary results potentially could be improved.

4.4.1 Learning parameters of the random ralk.

[soumen's netlearn] Recall that similarity in the graph is defined by a random walk process, which is defined in part by $\Pr(\ell|x)$, the probability of picking an edge with label ℓ at a node x. Prior researchers have explored schemes for adjusting $\Pr(\ell|x)$ for other tasks: methods used include gradient descent (e.g., [27]) or expectation-maximization (e.g., [72]).

We plan to implement these methods and evaluate them as an alternative to, or complement to, reranking. One advantage of learning-to-rerank is that one can use a wider range of features (for instance, the arbitrary features of the final item x_i can be used). One advantage of learning-to-walk is that it can also improve the efficiency of the similarity computation.

4.4.2 Reinforcement learning.

A more novel learning scheme for this task is suggested by analogy to the problem of value function estimation in the area reinforcement learning (RL) [42]. To clarify, let q be a query, and let $R_q(x)$ be the value placed by the user on node x as an answer to a query, which we will assume to be either 0 or 1. Let $\mathcal{A}$ be a set of graph-walk "actions" which includes every edge label ℓ and a special action `STOP`, which indicates that the walk will stop and emit the current node. Let $R(x, \texttt{STOP}) = (1-\gamma)R_q(x)$, and for $a \neq \texttt{STOP}$, define $T(x, a, x')$ to be $\Pr(x'|x, a)\Pr(a|x)$.

Finally define $V_q(x)$ to be the probability that a random walk that starts from the node x emits some node x' such that $R_q(y) = 1$. It is easy to show that

$$V_q(x) = \sum_{a \in A} \left(R(x, a) + \gamma \sum_{x'} T(x, a, x') V_q(x') \right) \tag{1}$$

There is a close similarity between this recurrence and the recurrence relations that define the *value function* of a policy, as studied in the field of *reinforcement learning*. Compare Equation 1 to the equation for the optimal value function $V^*(s)$ for a state s, as given by Equation (1) in Kaelbling *et al*:

$$V^*(s) = \max_{a \in A} \left(R(s, a) + \gamma \sum_{s'} T(s, a, s') V^*(s') \right)$$

Note that the two equations are identical except for the replacement of the summation with a "$\max$" operator.

This analogy suggests a number of new approaches to the problem of learning to walk a graph. We believe that many of the standard RL approaches for learning a value function from data could be adapted to learn a function defined by Equation 1. (In fact, a "summation" variant of Bellman backup has already been analyzed, and used for the purpose of Monte Carlo matrix multiplication [11].) Nodes encountered in a random walk could then be re-ranked by their estimated "value".

4.4.3 Reshaping the graph.

In the experiments of Figure 1, we observed that performance was improved by adding "short-cut" edges that linked abstracts to gene identifiers found by a baseline method. To understand how these extra edges might improve performance, consider a single path through the graph, such as

$$x \overset{hasProtein}{\longrightarrow} \text{ISU-1} \overset{similarTo}{\longrightarrow} \text{ISU1p} \overset{aliasOf}{\longrightarrow} y$$

Adding a new edge directly from x to y makes it easier to walk from x to y, and thus increases their similarity—while at the same time decreasing the similarity of x to other nodes in the graph.

There are a number of ways such short-cut links might be discovered automatically. One approach would be to examine a number of high-scoring paths from x to y, where x is some query node and y some correct answer for x, and look for sequences of edges that are frequently followed. Such sequences could be added to the graph as a shortcut with a low initial probility of being used; then, a parameter-learning approach could be used to adjust the weight for the new edge.

Distances in the graph can also be changed by replacing a pair of nodes x_1, x_2 with a single node x_{12}, which inherits all the ingoing and outgoing edges associated with either x_1 or x_2. Intuitively, this would be desirable if x_1 and x_2 are duplicates—e.g., they were derived from concepts in different heterogeneous KBs that referred to the same entity. If a is similar to x_1 and b is similar to x_2, then merging x_1 and x_2 will increase the similarity of a to b.

A natural scheme for finding candidates for this sort of node-merging operation would be to use some existing clustering method on the nodes. (One possible feature representation for a node x would be a set of nodes y that are similar to x according to the graph-walk, weighted by similarity; this approximates the representation of x in the Hilbert space associated with the heat diffusion kernel of the graph.) Again, the graph could be modified by adding a new node x_{12} and its incident edges, giving the new edges low initial weight, and then using parameter-learning to reweight the new edges.

4.4.4 Bootstrap learning.

We believe that existing bootstrap training methods can be extended in a number of ways. We plan to explore bootstraping using many partially independent sources of information—an approach inspired by co-training [7]. One source of information is "context" information: certain contexts indicate an entity name of a certain type. (E.g., the word XYZZY1 is almost certainly a gene in the phrase "The XYZZY1-expressing strain…"). Another source is "content" information: certain strings almost always refer to a known entity. (E.g., "*s. cerevisiae*" is always a model organism). The approach of semi-Markov information extraction [22, 64] is a clean way of separating the features used for content and context for co-training purposes, and newly-derived algorithm for semi-Markov sequence segmentation [63] now make this feasible to do quite efficiently. A third information source is intra-document repeats; we believe that combining semi-Markov information extraction with another recent technique called *stacked sequential learning* [17] can incorporate this additional source of information cleanly.

4.4.5 Scaling up.

There are number of ways that the current prototype count be made more efficient—many of which are simply a matter of more careful engineering of the components used, and more effective use of main memory. Some techniques for improving performance, however, are research topics in themselves.

For instance, a well-estimated value function could also be used to increase the efficiency of a walk: e.g., nodes with low values could be discarded from the search. We believe that effective pruning techniques could be developed using the A^* search formalism used in WHIRL: for this purpose, the novel problem of learning an *upper bound* on the value function would need to be solved.

Another scalability issue is that the similarity metric we use is expensive to compute exactly. (It is $O(n^3)$, where n is the number of nodes in the graph.) In our experiments with the BioCreatIvE data, we used an approximation to this metric inspired by *particle filtering* [2]. The walk proceeds in levels. To compute weight of nodes in level n, some number of the nodes in level $n-1$ are *sampled* according to the probability of encountering that node at the $(n-1)$-th step of a random walk from the query node, and then one-step walks are conducted from every sampled node. The zero-th level contains only the query nodes.

This approach allows one to easily trade off computation time for accuracy—for instance, in our experiments, we restricted every query to sample a total of 5000 nodes. We propose to exploit this property by using an "iterative-deepening" approach to similarity computation, where one initially generates a set of nodes using a very small series of samples, and then repeats the process with larger and larger samples. After each repetition, the ranking is modified to reflect the more accurate computation, and at any point, the user can request the most recently-produced ranking. This allows the user to get results immediately without sacrificing accuracy. The iterative-deepening approach also allows one to estimate a *query-specific* value function, which could be used for other sorts of performance improvements, as discussed above. Value functions could also be used to replace the sampling step with *importance sampling*, where the "importance" is the probability of ultimately reaching a useful node.

4.4.6 Sharing configured, trained KBs.

In later years of the grant, we will implement mechanisms for exchanging configured, trained QUERENDIPITY KBs between different biologists. This would be useful for biologists who want to distribute queryable structured data quickly, without spending effort on data cleaning and database development. We will also investigate mechanisms for exporting a subset of a local KB (e.g., retaining some not-yet-published data locally), and for merging data from external QUERENDIPITY KBs into a local KB.

The long-term goal of this effort would be a web-like network of structured knowledge bases, where it is as easy to import externally-generated structured knowledge into a local KB as it is to add an external link to web page. Although this is not likely to be fully realized in the period of this grant, any reduction of the costs of distributing data would be very valuable to biologists, who are both consumers and producers of data.

4.4.7 Use for KB curation.

One particularly interesting class of users for QUERENDIPITY are those biologists that work as curators of community KBs (e.g., SGD, MBI, Flybase, Wormbase, etc). These biologists have more predictable information needs than a typical working biologist; thus there is great potential benefit from a system that adaptively learns to collect information for possible curation. If curators trained a copy of QUERENDIPITY to produce a queue of likely-to-be-curateable material, this work queue could also be shared with working biologists, who might then search it for information of particular relevance to them.

References

[1] Sanjay Agrawal, Surajit Chaudhuri, and Gautam Das. Dbxplorer: A system for keyword-based search over relational databases. In *ICDE '02: Proceedings of the 18th International Conference on Data Engineering*, page 5, Washington, DC, USA, 2002. IEEE Computer Society.

[2] C. Andrieu, A. Doucet, S.S. Singh, and V.B Tadic. Particle methods for change detection, system identification, and control. *Proceedings of the IEEE*, 92(3), 2004.

[3] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan. Keyword searching and browsing in databases using BANKS. In *Proceedings of the 18th International Conference on Data Engineering, 2002*, pages 431–440, San Jose, CA, USA, February 2002.

[4] D. M. Bikel, R. L. Schwartz, and R. M. Weischedel. An algorithm that learns what's in a name. *Machine Learning*, 34:211–231, 1999.

[5] Mikhail Bilenko, William W. Cohen, Stephen Fienberg, Raymond J. Mooney, , and Pradeep Ravikumar. Adaptive name-matching in information integration. *IEEE Intelligent Systems*, 18(5):16–23, September/October 2003.

[6] Christian Blaschke, Miguel A. Andrade, Christos Ouzounis, and Alfonso Valencia. Automatic extraction of biological information from scientific text: Protein-protein interactions. In *Proceedings of the 1999 International Conference on Intelligent Systems for Molecular Biology (ISMB-99)*, pages 60–67, 1999.

[7] Avrin Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the 1998 Conference on Computational Learning Theory*, pages 92–100, Madison, WI, 1998.

[8] Razvan Bunescu, Ruifang Ge, Rohit J. Kate, Edward M. Marcotte, Raymond J. Mooney, Arun Kumar Ramani, and Yuk Wah Wong. Learning to extract proteins and their interactions from medline abstracts. Available from http://www.cs.utexas.edu/users/ml/publication/ie.html, 2002.

[9] Jeremy J. Carroll, Ian Dickinson, Chris Dollin, Dave Reynolds, Andy Seaborne, and Kevin Wilkinson. Jena: implementing the semantic web recommendations. In *WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 74–83, New York, NY, USA, 2004. ACM Press.

[10] O. Chapelle, J. Weston, and B. Sch olkopf. Cluster kernels for semisupervised learning. In *Advances in Neural Information Processing Systems, 15*. MIT Press, 2002.

[11] Cohen and Lewis. Approximating matrix multiplication for pattern recognition tasks. In *SODA: ACM-SIAM Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Analysis of Discrete Algorithms)*, 1997.

[12] William W. Cohen. A Web-based information system that reasons with structured collections of text. In *Proceedings of Autonomous Agents-98*, St. Paul, MN, 1998.

[13] William W. Cohen. Reasoning about textual similarity in information access. *Autonomous Agents and Multi-Agent Systems*, pages 65–86, 1999.

[14] William W. Cohen. Data integration using similarity joins and a word-based information representation language. *ACM Transactions on Information Systems*, 18(3):288–321, July 2000.

[15] William W. Cohen. WHIRL: A word-based information representation language. *Artificial Intelligence*, 118:163–196, 2000.

[16] William W. Cohen. Infrastructure components for large-scale information extraction systems. In *Proceedings of The Fifteenth Annual Conference on Innovative Applications of Artificial Intelligence (IAAI-2003)*, Acapulco, Mexico, 2003.

[17] William W. Cohen and Vitor Carvalho. Stacked sequential learning. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pages 671–676, Edinburgh, 2005.

[18] William W. Cohen and Wei Fan. Web-collaborative filtering: Recommending music by crawling the web. In *Proceedings of The Ninth International World Wide Web Conference (WWW-2000)*, Amsterdam, 2000.

[19] William W. Cohen, Einat Minkov, and Anthony Tomasic. Learning to understand web site update requests. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pages 1028–1041, Edinburgh, 2005.

[20] William W. Cohen and Pradeep Ravikumar. SecondString: An open-source Java toolkit of approximate string-matching techniques. Project web page, http://secondstring.sourceforge.net, 2003.

[21] William W. Cohen, Pradeep Ravikumar, and Stephen E. Fienberg. A comparison of string distance metrics for name-matching tasks. In *Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web (IIWeb-03)*, 2003.

[22] William W. Cohen and Sunita Sarawagi. Exploiting dictionaries in named entity extraction: combining semi-markov extraction processes and data integration methods. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 89–98, New York, NY, USA, 2004. ACM Press.

[23] William W. Cohen, Richard Wang, and Robert F. Murphy. Understanding captions in biomedical publications. In *Proceedings of The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2003)*, Washington, DC, 2003.

[24] Michael Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2002.

[25] The Gene Ontology Consortium. Gene ontology: tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000.

[26] Mark Craven and Johan Kumlien. Constructing biological knowledge bases by extracting information from text sources. In *Proceedings of the 7th International Conference on Intelligent Systems for Molecular Biology (ISMB-99)*, pages 77–86. AAAI Press, 1999.

[27] Michelangelo Diligenti, Marco Gori, and Marco Maggini. Learning web page scores by error backpropagation. In *IJCAI*, 2005.

[28] A. Elmagarmid, P. Ipeirotis, , and V. Verykios. Duplicate record detection. Technical Report CeDER Working Papers Series CeDER-06-05, Stern School of Business, 2005. Available at http://hdl.handle.net/2451/14760.

[29] Eleazar Eskin and Eugene Agichtein. Combining text mining and sequence analysis to discover protein functional regions. In *Pacific Symposium on Biocomputing (PSB)*, 2004.

[30] Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu Tal Shaked, Stephen Soderland, Daniel S.Weld, and Alexander Yates. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 165:91–134, 2005.

[31] Kristofer Franzén, Gunnar Eriksson, Fredrik Olsson, Lars Asker Per Lidén, and Joakim Coster. Protein names and how to find them. *International Journal of Medical Informatics*, 67(1-3):49–61, 2002.

[32] Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. In *Machine Learning: Proceedings of the Fifteenth International Conference (ICML '98)*, Madison, WI, 1998. Morgan Kaufmann.

[33] Yoav Freund and Robert E. Schapire. Large margin classification using the perceptron algorithm. In *Computational Learning Theory*, pages 209–217, 1998.

[34] K. Fukuda, T. Tsunoda, A. Tamura, and T. Takagi. Toward information extraction: Identifying protein names from biological papers. In *Proceedings of 1998 the Pacific Symposium on Biocomputing (PSB-1998)*, pages 707–718, 1998.

[35] Luis Gravano, Panagiotis G. Ipeirotis, Nick Koudas, and Divesh Srivastava. Text joins for data cleansing and integration in an RDBMS. In Umeshwar Dayal, Krithi Ramamritham, and T. M. Vijayaraman, editors, *Proceedings of the 19th International Conference on Data Engineering (ICDE)*, pages 729–731. IEEE Computer Society, 2003.

[36] L. Hirschman, A. Yeh, C. Blaschke, and A. Valencia. Overview of BioCreAtIvE: critical assessment of information extraction for biology. *BMC Bioinformatics*, 6((Suppl 1):S1), May 2005.

[37] Lynette Hirschman, Alexander Yeh, Christian Blaschke, and Alfonso Valencia. Overview of BioCreAtIvE: critical assessment of information extraction for biology. *BMC Bioinformatics*, 6(S1), 2005.

[38] Vagelis Hristidis, Luis Gravano, and Yannis Papakonstantinou. Efficient ir-style keyword search over relational databases. In *VLDB 2003, Proceedings of 29th International Conference on Very Large Data Bases*. Morgan Kaufmann, 2003.

[39] Vagelis Hristidis and Yannis Papakonstantinou. DISCOVER: Keyword search in relational databases. In *VLDB 2002, Proceedings of 28th International Conference on Very Large Data Bases*, pages 670–681. Morgan Kaufmann, 2002.

[40] K. Humphreys, G. Demetriou, and R. Gaizauskas. Two applications of information extraction to biological science journal articles: Enzyme interactions and protein structures. In *Proceedings of 2000 the Pacific Symposium on Biocomputing (PSB-2000)*, pages 502–513, 2000.

[41] Varun Kacholia, Shashank Pandit, Soumen Chakrabarti, and S. Sudarshan. Bidirectional expansion for keyword search on graph databases. In *Proceedings of the 31st International Conference on Very Large Data Bases*. ACM, 2005.

[42] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *J. Artif. Intell. Res. (JAIR)*, 4:237–285, 1996.

[43] Jin-Dong Kim, Tomoko Ohta, Yoshimasa Tsuruoka, Yuka Tateisi, and Nigel Collier. Introduction to the bio-entity recognition task at jnlpba. In *Proceedings of the International Workshop on Natural Language Processing in Biomedicine and its Applications (JNLPBA-04)*, pages 70–75, 2004.

[44] R. I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete structures. In *Proceedings of the ICML*, 2002.

[45] Zhenzhen Kou, William W. Cohen, and Robert F. Murphy. Extracting information from text and images for location proteomics. In *Proceedings of the 3nd ACM SIGKDD Workshop on Data Mining in Bioinformatics (BIOKDD 2003)*, pages 2–9, 2003.

[46] Zhenzhen Kou, William W. Cohen, and Robert F. Murphy. High-recall protein entity recognition using a dictionary. *Bioinformatics*, 21(Supplement 1):i266–i273, June 2005.

[47] Nick Koudas and Divesh Srivastava. Approximate joins: Concepts and techniques (tutorial). In Klemens Böhm, Christian S. Jensen, Laura M. Haas, Martin L. Kersten, Per-Åke Larson, and Beng Chin Ooi, editors, *Proceedings of the 31st International Conference on Very Large Data Bases*, page 1365. ACM, 2005.

[48] Einat Minkov, William Cohen, and Andrew Ng. A graph framework for contextual search and name disambiguation in email. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on research and development in information retrieval*, August 2006. To appear.

[49] Einat Minkovp and William W. Cohen. An email and meeting assistant using graph walks. In *Proceedings of the Conference on Email and Anti-Spam*, Mountain View, California, 2006.

[50] Minorthird: Methods for identifying names and ontological relations in text using heuristics for inducing regularities from data. http://minorthird.sourceforge.net, 2004.

[51] Alex Morgan, Lynette Hirschman, Alexander Yeh, and Marc Colosimo. Gene name extraction using FlyBase resources. In Sophia Ananiadou and Jun'ichi Tsujii, editors, *Proceedings of the ACL 2003 Workshop on Natural Language Processing in Biomedicine*, pages 1–8, 2003.

[52] R.F. Murphy. Cytomics and location proteomics: Automated interpretation of subcellular patterns in fluorescence microscope images. *Cytometry*, 67A:1–3, 2005.

[53] R.F. Murphy. Location proteomics: A systems approach to subcellular location. *Biochemical Society Transactions*, 33:535–538, 2005.

[54] Robert F. Murphy, Zhenzhen Kou, Juchang Hua, Matthew Joffe, and William W. Cohen. Extracting and structuring subcellular location information from on-line journal articles: The subcellular location image finder. In *Proceedings of the IASTED International Conference on Knowledge Sharing and Collaborative Engineering*, pages 109–114, St. Thomas, US Virgin Islands, 2004.

[55] Robert F. Murphy, Meel Velliste, and Gregory Porreca. Searching online journals for fluorescence microscope images depicting protein subcellular location patterns. In *Proceedings of the 2nd IEEE International Symposium on Bio-informatics and Biomedical Engineering (BIBE-2001)*, pages 119–128, December 2001.

[56] OBO: Open biomedical ontologies (web site). http://obo.sourceforge.net/, 2006.

[57] Fatma Ozcan, editor. *SPIDER: flexible matching in databases*. ACM, 2005.

[58] Larry Page, Sergey Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. In *Technical Report, Computer Science department, Stanford University*, 1998.

[59] J. Pustejovsky, J. Castaño, J. Zhang, M. Kotecki, and B. Cochran. Robust relational parsing over biomedical literature: Extracting inhibit relations. In *Proceedings of 2002 the Pacific Symposium on Biocomputing (PSB-2002)*, pages 362–373, 2002.

[60] S. Ray and M. Craven. Representing sentence structure in hidden markov models for information extraction. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pages 584–589, Seattle, WA, 2001.

[61] T.C. Rindflesch, Lorraine Tanabe, John N. Weinstein, and L. Hunter. Edgar: Extraction of drugs, genes and relations from the biomedical literature. In *Proceedings of 2000 the Pacific Symposium on Biocomputing (PSB-2000)*, pages 514–525, 2000.

[62] Gerard Salton, editor. *Automatic Text Processing*. Addison Welsley, Reading, Massachusetts, 1989.

[63] Sunita Sarawagi. Efficient inference on sequence segmentation models. In *Proceedings of the 23th International Conference on Machine Learning (ICML)*, 2006.

[64] Sunita Sarawagi and William W. Cohen. Semi-markov conditional random fields for information extraction. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1185–1192. MIT Press, Cambridge, MA, 2005.

[65] Sunita Sarawagi and Alok Kirpal. Efficient set joins on similarity predicates. In *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 743–754, New York, NY, USA, 2004. ACM Press.

[66] T. Sekimizu, H. Park, and J. Tsujii. Identifying the interaction between genes and gene products based on frequently seen verbs in Medline abstracts. In *Genome Informatics*, pages 62–71. Universal Academy Press, Inc, 1998.

[67] A. Smola and R. Kondor. Kernels and regularization on graphs. In *Proceedings of the Sixteenth Annual Conference on Computational Learning Theory and Kernel Workshop*. Springer-Verlag, 2003.

[68] B.J. Stapley, L.A. Kelley, and M.J.E. Sternberg. Predicting the sub-cellular location of proteins from text using support vector machines. In *Proceedings of the 2002 Pacific Symposium on Biocomputing*, pages 374–385, 2002.

[69] M. Stephens, M. Palakal, S. Mukhopadhyay, R. Raje, and J. Mostafa. Detecting gene relations from medline abstracts. In *Pacific Symposium on Biocomputing*, pages 483–496, 2001.

[70] M. Szummer and T. Jaakkola. Partially labeled classification with Markov random walks. In *Advances in Neural Information Processing Systems, 14*, 2001.

[71] J. Thomas, D. Milward, C. Ouzounis, S. Pulman, and M. Carroll. Automatic extraction of protein interactions from scientific abstracts. In *Proceedings of 2000 the Pacific Symposium on Biocomputing (PSB-2000)*, pages 538–549, 2000.

[72] Kristina Toutanova, Christopher D. Manning, and Andrew Y. Ng. Learning random walk models for inducing word dependency distributions. In *ICML*, 2004.

[73] Richard Wang, William Cohen, Robert Frederking, and Anthony Tomasic. Learning to extract gene-protein names from weakly-labelled text. In preparation, 2006.

[74] Hong Yu and Eugene Agichtein. Extracting synonymous gene and protein terms from biological literature. In *Proceedings of the 11th International Conference on Intelligent Systems for Molecular Biology (ISMB)*, 2003.

[75] X. Zhu. Semi-supervised learning with graphs. Technical Report CMU-LTI-05-192, Carnegie Mellon University, 2005.

[76] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of ICML-03, the 20th International Conference on Machine Learning*, 2003.