

Ausarbeitung zur Studienarbeit “Ranking Integrated Search Results in Columba”

Betreut durch: Silke Trissl, Prof. Ulf Leser



Philipp Hussels

hussels@informatik.hu-berlin.de

23. August 2007

Inhaltsverzeichnis

1	Vorwort	2
2	What's new? What's certain? - DILS 2007	4
A	Anhang - Implementierung	19
A.1	Quellen und Partitionen	19
A.2	Vorberechnung	20
A.3	Anfragen	21
B	Anhang - Quelltexte	23
B.1	Vorberechnung	23
B.2	Anfragen	29
	Selbstständigkeitserklärung	34

1 Vorwort

Im Bereich der molekularbiologischen Forschung existieren für verschiedene fachliche Domänen jeweils mehrere Datenquellen, die sich inhaltlich und strukturell überschneiden, aber in der Methode der Datengewinnung oder der Sicht auf die Daten differieren. Für praktische Fragestellungen ist es erforderlich Datenquellen unterschiedlicher Domänen anzufragen und Links zwischen diesen Quellen zu verfolgen. Je nach Auswahl der Quellen aus den einzelnen Domänen können die Anfrageergebnisse variieren. Durch Integration aller verfügbaren Datenquellen können Forschern vollständige Anfrageergebnisse zur Verfügung gestellt werden. Darüber hinaus können Überlappungen zwischen Datenquellen genutzt werden, um Anfrageergebnisse nach verschiedenen Kriterien zu ranken. So ist ein Anfrageergebnis beispielsweise besonders “vertrauenswürdig”, d.h. mit hoher Wahrscheinlichkeit korrekt, wenn es durch eine große Zahl von Datenquellen “bestätigt” wird. Im Rahmen dieser Studienarbeit wurden zwei verschiedene Scoring-Verfahren erarbeitet, die Anfrageergebnisse basierend auf statistischen Informationen über die Überlappung verschiedener Datenquellen bewerten. Die Ergebnisse der Studienarbeit wurden auf dem “Workshop on Data Integration in the Life Sciences (DILS) 2007” veröffentlicht. Das in Zusammenarbeit mit Silke Trissl und Prof. Ulf Leser verfasste Paper mit dem Titel “What’s new? What’s certain? – Scoring Search Results in the Presence of Overlapping Data Sources” bildet den Hauptteil dieser Ausarbeitung. Implementiert wurden beide Scoring-Verfahren für die integrierte Protein-Annotationsdatenbank Columba. Informationen zur Implementierung, ausgewählte SQL und PL/PGSQL-Quellen, sowie statistische Angaben zu den verwendeten Datenquellen sind im Anhang zu finden. Die vollständigen Quelltexte, alle verwendeten Daten und die Werkzeuge zur Integration dieser Daten (Parser, ETL-Skripte etc.) sind als ISO-CD beigelegt.

What’s new? What’s certain? – Scoring Search Results in the Presence of Overlapping Data Sources

Philipp Hussels, Silke Trißl, and Ulf Leser

Humboldt-Universität zu Berlin, Institute of Computer Sciences, D-10099 Berlin,
Germany

{hussels, trissl, leser}@informatik.hu-berlin.de

Abstract. Data integration projects in the life sciences often gather data on a particular subject from multiple sources. Some of these sources overlap to a certain degree. Therefore, integrated search results may be supported by one, few, or all data sources. To reflect these differences, results should be ranked according to the number of data sources that support them. How such a ranking should look like is not clear per se. Either, results supported by only few sources are ranked high because this information is potentially new, or such results are ranked low because the strength of evidence supporting them is limited.

We present two scoring schemes to rank search results in the integrated protein annotation database Columba. We define a *surprisingness* score, preferring results supported by few sources, and a *confidence* score, preferring frequently encountered information. Unlike many other scoring schemes our proposal is purely data-driven and does not require users to specify preferences among sources. Both scores take the concrete overlaps of data sources into account and do not presume statistical independence. We show how our schemes have been implemented efficiently using SQL.

1 Introduction

In research on molecular biology, very often knowledge from different domains is needed to answer practical questions. Imagine a researcher asking for the three-dimensional structure of a protein that participates in a certain metabolic pathway and is associated with a certain disease. This researcher has to query multiple data sources. For instance, she could access the Protein Data Bank (PDB) [1] for the protein structure, KEGG [7] for pathway information, and PubMed to find information about protein-disease associations. However, for the latter two aspects many other data sources could be used as well.

We call those different aspects of biomedical objects a *domain*. For a protein such domains are 3D structure, sequence, fold, functional classification, other proteins it interacts with, processes it is involved in, diseases it is associated with, etc. For many domains there exist multiple sources. For example, information about pathways can be found in KEGG [7], aMAZE [9], Reactome [6], and several other resources. These resources usually overlap extensionally, i.e.,

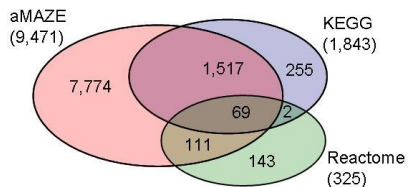


Fig. 1. The figures in brackets state the total number of enzyme-enzyme pairs that are connected through a chain of reaction \rightarrow substrate \rightarrow reaction. The figures in the different partitions show the number of pairs that occur in only one, two, or all three data sources.

store the same information (and not only the same kind of information) because they partially draw their content from the same data sets. For three sources on pathways the overlap can be seen in Figure 1. But the sources also contain different and potentially unique information, due to different methodology used to curate, integrate, select, or produce the data. Thus, when a researcher looks for the pathways a given protein is involved in, the results may vary considerably depending on the chosen source. We want to provide users with a ranking of search results depending on the particular set of data sources that support it.

1.1 Data Model

We assume that a user is interested in information about a particular class of biological entities, called the *primary domain* P . Objects in P are described by objects in other data sources. A group of data sources that contain information about the same type of entities or even the same entities is called *secondary domain* D_i . The content of secondary domains is comprised of data from various data sources S_{i1}, \dots, S_{im} and link sources R_{i1}, \dots, R_{il} , where i is the secondary domain. If the domain is clear i can be omitted.

The link sources R_1, \dots, R_l contain entries (s, p) with $s \in S_i$ and $p \in P$, i.e., they provide *links* between objects in data sources of a secondary domain and objects in the primary domain. Thus, every object in a data source of domain D_i is *linked* through link sources to one or more objects in P and vice versa. We also say that an object in P is *annotated* by objects in data sources of D_i . This situation is depicted in Figure 2.

A query selects entries from P by setting conditions on annotations in different domains. The result of a query, written as $res(q)$ is the set of objects in P that comply with these conditions through at least one data and one link source for every domain mentioned in q . For a single result $p \in res(q)$ we say that the result is *supported* by at least one qualified annotation in every secondary domain. As the data and link sources in a domain overlap, an annotation supporting a result may stem from different data sources and may be linked by different link sources. According to the degree of dependence between the data and link sources of a domain, certain combinations of sources frequently support

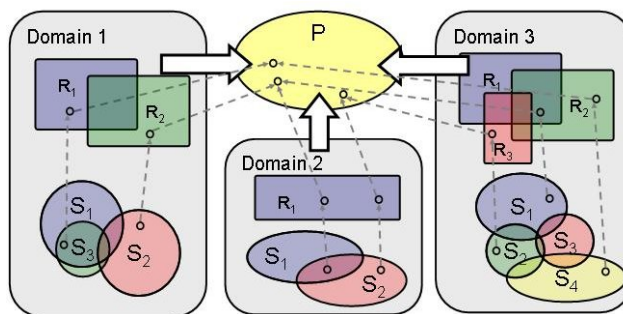


Fig. 2. The objects in the primary domain P are annotated by objects from three secondary domains ($D_1 \dots D_3$). Every domain contains several secondary data and link sources.

query results, while other combination of sources rarely do. We make use of this fact to assign query results scores for *confidence* and *surprisingness*.

1.2 Scoring of Results

This paper is about ranking results in a setting described above. Integrating many sources instead of manually selecting some (the 'best' ones) comes at the risk of large result sets. Therefore, ranking of results becomes important. However, ranking is not a one-dimensional problem. Clearly, a user is most *confident* in results supported by all data sources. In the previous example a result that is supported by KEGG, aMAZE, and Reactome is one where users will be most confident that it is biologically true. However, such results are sort-of common place and thus potentially boring. Some researcher might be more interested in the contrary, i.e., *surprising* results supported by only few sources. For example, a result supported by Reactome but not by aMAZE is rather unexpected, because Reactome is much smaller. Thus, a-priori chances to find a result supported only by Reactome are small. If this occurs it makes a good starting point for a more thorough investigation with a higher chance to produce some new findings.

Both scores, confidence and surprisingness, are important. It depends on the concrete application which ranking scheme should be used for a search. In this paper we present a method to compute both scores for integrated search results over multiple domains where each domain is formed from multiple data and link sources. In contrast to much of the related work, our method does not require expert knowledge, but is merely based on the properties of the data sources themselves, i.e., the overlaps between them.

1.3 Paper outline

The paper is structured as follows. We discuss related work in the next section. The surprisingness score is defined in Section 3 and the confidence score in Section 4. In Section 5 we show how to expand both measures to multi-domain queries. Section 6 describes the application of the scores in the integrated database Columba. Section 7 shows how to implement the scoring scheme and also gives some experimental results. Section 8 concludes the paper.

2 Related work

One option to use information from different data sources is to provide the user only with information supported by all selected data sources, i.e., the information a user is most confident in. Marcotte et al. [10] proposed such a method for the reconstruction of metabolic pathways from protein-protein interaction data. Clearly, their results are highly trustworthy, but a biologically correct protein-protein interaction supported only by some data sources will not be considered. In contrast Yanai & DeLisi [17] used a union of different interaction data sources. This leads to good coverage, as all known interactions are listed, but possibly also many incorrect protein-protein interactions are included.

The problem of giving the user all possible information ordered according to some criteria is addressed by many projects. Internet search engines rank hits according to their expected usefulness for the query. The protein-protein database STRING [12] integrates information on protein-protein interactions from different data sources such as high-throughput experiments, literature search, or sequence comparison. A confidence score for every object is created. This score is either uniform within a data source, e.g., for an integrated source without further knowledge, or individual for every object, e.g., when text mining methods are used to extract protein-protein interactions from publications. Similar methods have been described in the area of functional analysis of microarray experiments [5]. A general framework for specifying and using such quality scores for query optimization and result ranking has been proposed in [13]. All these methods build on expert knowledge about the data sources. Such ratings are highly subjective and not easy to obtain.

In this paper we propose a method that ranks results without the need for expert knowledge. A similar idea was proposed by Florescu et al. in [4] for the purpose of query optimization. Given a query they want to optimize the ratio between the execution cost and the size of the result set. To answer the query they first estimate which sources will return most results and then choose k sources, based on the selectivity of the source and the overlap with other sources.

A different approach is described by Lacroix et al. in [8] for estimating the size of the result set. They assume a network of interlinked sources and data objects. A query poses conditions on a start source and returns results from a primary source by analyzing all paths from the start to the primary source. To estimate the size of the result set they pre-compute overlap statistics for different

paths using sampling. In Bleiholder et al. [2] these overlap statistics are used to optimize queries over multiple data sources to solve the Budgeted Maximum Coverage problem. In contrast to this work, we use a simpler model (primary and secondary sources) and focus on ranking of results in result sets, not on query optimization.

3 Surprisingness of Results

We now present a framework for measuring the surprisingness of a search result. Confidence will be defined in Section 4. We develop our model starting from a single domain with a single data and link source and then extend it to multiple data and link sources. The extension to multi-domain queries is given in Section 5.

We assume that the result set contains objects from the primary source. The user can restrict this set by setting conditions on objects in secondary domains. An object in the primary source is contained in the result set if it is supported by at least one qualified annotation in every queried domain.

3.1 Single Data Source

We start with the simple scenario of a single domain D , a single data source S , and a single link source R as shown in Figure 3(a). Without loss of generality we assume that every annotation $s \in S$ is linked to at least one object $p \in P$ through at least one link $r \in R$ (we can safely delete all other annotations and links since they can never select entries in P). A query selects objects in D and determines the set of objects in P that are linked by at least one link in R .

For a given query q we calculate the probability that a randomly chosen object $p \in P$ is part of the result set of q . We first derive the a-priori probability that a randomly chosen annotation $s \in S$ is linked to a randomly chosen object $p \in P$:

$$P((s, p) \in R) = \frac{|R|}{|P \times S|} = \frac{|R|}{|P| * |S|} \quad (1)$$

A randomly chosen $p \in P$ takes part in the query result if it is linked to at least one qualified annotation $s \in S$ by at least one link $r \in R$. If we assume that q selects k annotations and take into account that a single object in P can be selected by multiple annotations in S , then the probability that a concrete $p \in res(q)$ is selected is precisely the probability that not none of the k selected annotations is linked to p , which gives:

$$P(p \in res(q)) = 1 - \left(1 - \frac{|R|}{|P| * |S|}\right)^k \quad (2)$$

Clearly, we could also estimate the value of k a-priori using attribute selectivities. Note that this formula ranks all objects in a result set of a query equal.

This is expected, as we want to rank a result by the subset of sources that supports it in any domain. Therefore, differences in the computed score only appear when more than one source is present.

3.2 Multiple Sources in a Single Domain

We now extend our framework to the case of m data sources S_i and l link sources R_j , $1 \leq i \leq m$ and $1 \leq j \leq l$ for a single domain D as shown in Figure 3(b).

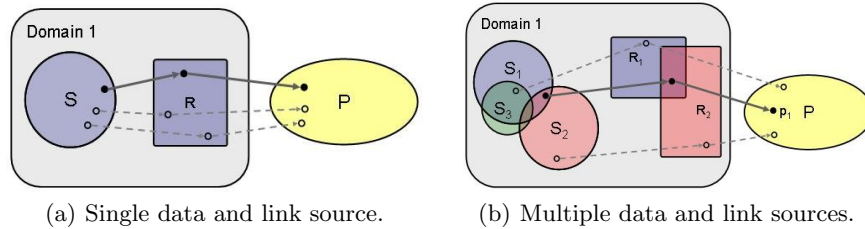


Fig. 3. An object $p \in P$ is supported by objects $s \in S_i$ in a domain.

An object $p \in P$ is in the result set of a given query q if it is supported by at least one qualified annotation s linked through at least one link r . However, s as well as r can be contained in various combination of sources. Consider the situation shown in Figure 3(b) with three overlapping data sources S_1 , S_2 , and S_3 and two overlapping link sources R_1 and R_2 . S_1 and S_3 strongly overlap, while S_2 mostly contains divergent data. In this situation it is likely that a query result is linked to a qualified annotation contained in $(S_1 \cap S_3) \setminus S_2$ or $S_2 \setminus (S_1 \cup S_3)$. Such query results shall be assigned a low score for *surprisingness*. We would rate a result more unlikely and therefore more surprising that is supported by a qualified annotation in $S_3 \setminus (S_1 \cup S_2)$. Clearly, to compute the score we also have to consider over which combination of link sources s is linked to p . Note that according to our understanding of surprisingness, a high score might also be assigned to results with incorrect annotations. This is in the line of our argument, since errors can be considered surprising and certainly require user attention.

The space of all annotations in D is partitioned into disjoint subsets according to the overlaps of data and link sources. Some of these subsets are represented by different colors in Figure 3(b). We call these partitions in data sources Z_1, \dots, Z_n . The assignment of annotations to partitions can be represented by a *domain-vector* v of size $m * l$ for a domain with m data and l link sources. If annotation $s \in S_i$ and $(s, p) \in R_j$ we set $v_{i,j} = 1$, and $v_{i,j} = 0$ otherwise. In Figure 3(b) an annotation contained in $S_1 \cap S_3 \setminus S_2$ that is linked over R_1 corresponds to the domain-vector $v_{i,1} = 101$ and $v_{i,2} = 000$. It follows that $2^{m * l}$ different domain vectors are possible. Now consider a single annotation s selected by q . Intuitively, a p linked to s is the more surprising, the smaller the partition Z_k is in which s lies.

However, we need some more work to derive a suitable definition for surprisingness. We compute the surprisingness for each annotation selected by a query which might later be aggregated into a score for an object p linked to multiple annotations. Let Z_k be the partition in which an annotation s lies that is selected by a query q . We estimate the probability that p is verified by all sources that contain Z_k and no others, which depends on the a-priori overlaps of sources. That means, we want to know how likely it is that a result for a given query is verified by a certain combination of available sources. The less likely, the more surprising is the result.

To answer this we first estimate the probability that for a given query a result is verified by a particular data source S_x provided that it is verified by at least one source in D . This is different from Equation 2 because p can be selected by other sources than S_x . Let q_{S_x} denote the subset of $res(q)$ that is verified by S_x . Using Bayes's Theorem we get:

$$P(p \in q_{S_x} | p \in res(q)) = \frac{P(p \in res(q) | p \in q_{S_x}) * P(p \in q_{S_x})}{P(p \in res(q))} \quad (3)$$

Clearly, the probability that an object is verified by at least one data source provided that it is verified by a particular S_x , $P(p \in res(q) | p \in q_{S_x})$, is 1 because the first event logically implies the second one. The a-priori probability $P(p \in res(q))$ is given by Equation 2, where $|S|$ now denotes the set of all unique annotations in D . We only miss the a-priori probability $P(p \in q_{S_x})$. For this probability we must take into account that not every object in S is contained in S_x and not every link in R links annotations $s \in S_x$ to a $p \in P$. We therefore can identify a subset of R , denoted as R_x , that only contains links from $s \in S_x$. Analogously, we can distinguish a subset of P , called P_x that contains entries p that are supported by an annotation $s \in S_x$.

$$\begin{aligned} P(p \in q_{S_x}) &= 1 - \left(1 - \frac{|P_x|}{|P|} * \frac{|S_x|}{|S|} * \frac{|R_x|}{|P_x| * |S_x|} \right)^k \\ &= 1 - \left(1 - \frac{|R_x|}{|P| * |S|} \right)^k \end{aligned} \quad (4)$$

Thus, Equation 3 can be rewritten as:

$$P(p \in q_{S_x} | p \in res(q)) = \frac{1 - \left(1 - \frac{|R_x|}{|P| * |S|} \right)^k}{1 - \left(1 - \frac{|R|}{|P| * |S|} \right)^k} \quad (5)$$

We now determine the probability that a particular p is supported by qualified annotations in a partition Z_k . Here as well we denote the subset of $res(q)$ verified by annotations in Z_k as q_{Z_k} . This gives:

$$P(p \in q_{Z_k} | p \in res(q)) = \frac{1 - \left(1 - \frac{|\bigcap_{s_i \in Z_k} R_i \setminus \bigcup_{s_i \notin Z_k} R_i|}{|P| * |S|}\right)^k}{1 - \left(1 - \frac{|R|}{|P| * |S|}\right)^k} \quad (6)$$

3.3 Surprisingness Score of a Single Annotation

To value the surprisingness of a single annotation we use the measure of self-information as defined by Shannon. Consider a domain-vector v as a symbol in a message, the self-information of v , $I(v)$, depends on the probability of its occurrence and is defined as $I(v) = -\log_2(P(v))$. Accordingly, we calculate the surprisingness score for p that is contained in the result set of a given query by applying the probability that p is supported by an $s \in Z_k$ using Equation 7. Definition 1 formalizes this approach:

Definition 1 (Surprisingness for a single annotation) *Let q be a query selecting an annotation s , let p be linked to s , and let s lie in partition Z_k of D . The surprisingness $S(p, s)$ of p with respect to s is defined as:*

$$S(p, s) = -\log_2 P(p \in q_{Z_k} | p \in res(q)) \quad (7)$$

3.4 Surprisingness Score for a Single Domain

Equation 7 only gives the probability that a given p is linked to a given annotation s selected by a query q . But we want a score for p given all its linked annotations selected by q , as shown in Figure 4. Therefore, we will need to aggregate scores of multiple s .

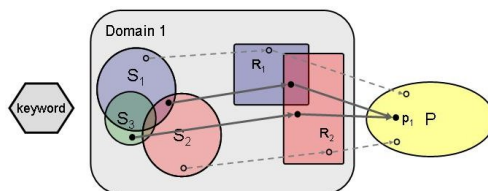


Fig. 4. An object linked to two qualified annotations.

Suppose, the primary domain contains protein structures and the secondary domain multiple data sources about scientific publications. Assume we query with a keyword and receive a structure p that is linked to multiple qualified publications. If all publications are contained in the same combination of data sources, intuitively the number of publications that verify p does not influence

its surprisingness. In this case p shall be assigned the same surprisingness score as assigned to a single publication. Now imagine p is linked to multiple qualified publications contained in different combinations of data sources as depicted in Figure 4. Clearly, if most selected publications linked to p are highly surprising we also want to assign p a high surprisingness score. We therefore define the surprisingness of p as the average of the surprisingness scores for every qualified publication that supports p .

Definition 2 (Surprisingness for multiple annotations) *Let q be a query and $p \in \text{res}(q)$ be linked to a set T of annotations selected by q . The surprisingness $S(p, T)$ of p is defined as:*

$$S(p, T) = \frac{1}{|T|} \sum_{s \in T} S(p, s) \quad (8)$$

4 Confidence of Results

As explained in Section 1, researches are not solely interested in highly surprising query results but also in trustworthy results. A researcher might want to rank those results high that are likely to be correct. Having multiple data sources in a domain, intuitively every data source that verifies a query result p increases the confidence in the correctness of p . Thus, a straightforward method to value the confidence of a query result would be to count the number of sources verifying the result. But here too we have to consider that the different data sources within a domain are not independent. If, for example, a query result p is verified by two data sources, the confidence in p being correct is the higher the lower the degree of dependence between those data sources is, because then it is more likely that information contained in both sources is the outcome of independent experiments rather than information stemming from the same resource.

Consider again the situation shown in Figure 3(b). We are most confident in annotations that are contained in $S_1 \cap S_2 \cap S_3$ and linked through both link sources R_1 and R_2 . If we consider annotations $s_1 \in (S_1 \cap S_3) \setminus S_2$ and $s_2 \in (S_1 \cap S_2) \setminus S_3$, both linked only over R_1 , we intuitively assign s_2 a higher confidence score because S_1 and S_3 strongly overlap, while S_1 and S_2 do not. More generally, for the confidence score we want to use the probability that an annotation is contained in a combination of sources given that the annotation is contained in at least one source.

Definition 3 (Confidence for a single annotation) *Let q be a query selecting an annotation s , let p be linked to s through r , and let s be contained in the partition Z_k . The confidence $C(p, s)$ of p with respect to s is defined as:*

$$C(p, s) = 1 - \log_2 \frac{\left| \bigcap_{S_i \supseteq Z_k} S_i \right|}{\left| \bigcup_{S_i \supseteq Z_k} S_i \right|} \quad (9)$$

Resulting from Definition 3 the score for a p linked to an annotation s that is contained in only one source is 1. Note, the confidence score for p_1 that is annotated by $s_1 \in Z_k$ is always lower or equal to the score for p_2 annotated by $s_2 \in Z_l$, with Z_l being the intersection between data sources in Z_k and an additional data source S_i .

So far, we considered the confidence for a result supported by only a single annotation. We shall now show how to aggregate confidence scores for multiple annotations. While the number of qualified annotations linked to a query result p does not influence its score for surprisingness, it clearly enhances the trust in the correctness of p . As we consider every single annotation as an evidence that p is an answer to a given query we sum up the confidence scores of all qualified annotations linked to p to calculate the confidence score of p .

Definition 4 (Confidence for multiple annotations) *Let q be a query and $p \in res(q)$ be linked to a set T of annotations selected by q . The confidence $C(p, T)$ of p is defined as:*

$$C(p, T) = \sum_{s \in T} C(p, s) \quad (10)$$

5 Multi-domain Query Results

In the last two sections we defined scores for surprisingness and confidence for single domain queries. In this section we explain how to use these values to compute a surprisingness and confidence score in a multi-domain setting.

We assume that different secondary domains are statistically independent. We can make that assumption as according to our model we group data sources that contain information about the same type of biological entities in one domain. To compute an overall surprisingness score we add up scores from all secondary domains given in a query. We do this as we consider a result surprising when it is surprising for at least one domain. In contrast, for the confidence, only those results of multi-domain queries shall be ranked high that have high confidence scores in many domains. To ensure this, we normalize single domain scores resulting from Equation 10 before multiplying all scores for the multi-domain confidence score.

6 Multi-domain Setting: Columba

In this section we introduce our real world example, where the scores presented in this paper are beneficial. We developed the integrated database Columba [14]. This database focuses on protein structures from the Protein Data Bank (PDB) [1] that are annotated by objects of different domains, such as fold, sequence, function, publication, metabolic pathway, or taxonomic classification.

We apply our scoring methods for ranking query results to parts of the Columba database. We use as primary domain the protein structures given by

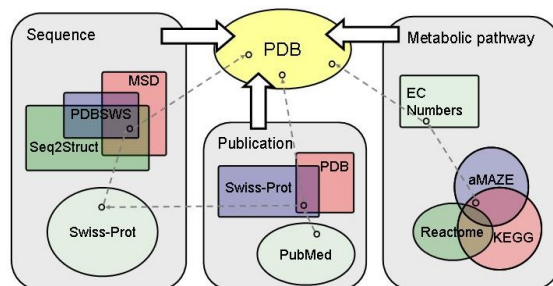


Fig. 5. The PDB annotated by secondary domains that contain multiple data sources. Note, the size and overlap of data and link sources does not necessarily reflect reality.

the PDB. Objects in the PDB are annotated by the secondary domains *sequence*, *publication*, and *metabolic pathway* as shown in Figure 5.

The domain *sequence* contains the data source Swiss-Prot [3] and has three different link sources linking entries from Swiss-Prot to entries in the PDB, namely PDBSWS [11], Seq2Struct [16], and MSD [15]. The overlap of link sources is given by identical entries in the sources.

Data sources for the domain *metabolic pathway* are KEGG [7], aMAZE [9], and Reactome [6]. To compare these heterogeneous data sources we extract information on the level of reactions. We store enzyme-enzyme pairs that are connected through a path enzyme \rightarrow reaction \rightarrow substrate \rightarrow reaction \rightarrow enzyme. The overlap of data sources is given by identical enzyme-enzyme pairs. If we consider enzymes as nodes and a pair of enzymes as edges in a graph we can compute paths between enzymes. We therefore can answer queries such as "Which PDB entries are less than 3 steps away from an enzyme with EC number 2.7.1.1 (Hexokinase)". The data sources are linked to the PDB through EC numbers given in the PDB as well as in the three data sources.

In the third domain *publications* we use PubMed as data source. The articles referenced in PubMed can be linked directly to the PDB using the references given in the PDB. But articles in PubMed can also be linked to the PDB via Swiss-Prot.

A multi-domain query for this setting is for example "Give me all protein structures that are up to 7 steps away from an enzyme with EC number 1.14.16.1 (Phenylalanine hydroxylase), linked to entries in Swiss-Prot that contain the keyword *Phenylalanine catabolism*, and linked to publications that mention the disease *Phenylketonuria*". This query returns in total 17 PDB chains. Using our scoring scheme we can rank the results according to their surprisingness and their confidence.

7 Implementation and Evaluation

In this section we show which values can be precomputed and how to implement this computation inside a relational database environment. We additionally show for some exemplary queries the impact of surprisingness and confidence scores.

7.1 Precomputation of Values

To compute the surprisingness of an object $S(p, T)$ we must first compute for every object in a domain the probability $P(p \in q_{Z_k} | p \in res(q))$ as given by Equation 6. We therefore require information about the size of the primary domain and the size of data and link sources in secondary domains. To gather $|P|$ we simply count the number of objects in the primary domain. To gather $|S|$ and $|R|$ we first have to integrate all objects of the data and link sources of D_i and then count the number of unique objects in both integrated sources. Knowing these values we can compute the value $c_1 = 1 - \frac{|R|}{|P|*|S|}$ for domain D_i . But we also require the size of different partitions of data and link sources. To gather these data we precompute and store the domain-vector v for every unique object in S of dimension D_i . We can determine the size for every partition Z_k by determining the frequency of different patterns in v , denoted as $freq(v)$. But to solve Equation 6 we require the value for the size of partitions in the link sources, denoted as $link_size(v)$. We can determine $link_size(v)$ by summing up for every $s \in Z_k$ the number of $(s, p) \in R$. Knowing this value we can compute the value $c_2 = 1 - \frac{link_size(v)}{|P|*|S|}$. The only value in Equation 6 that can not be precomputed is k , the number of qualified objects in D_i . But we can substitute variables in Equation 6 to gain the following equation for $S(p, s)$:

$$S(p, s) = \frac{1 - (c_2)^k}{1 - (c_1)^k} \quad (11)$$

To compute the confidence score for a result p we must compute $C(p, s)$ as given by Equation 9. Here we require the size of all unions and intersections of data sources S_i that contain s . Both values are independent of a particular query and therefore can be precomputed using $freq(v)$. For a given domain-vector v of length n the sum of $freq(v')$ with $v' : v' \wedge v = v$ is the size of the intersection and the sum of $freq(v'')$ with $v'' : v'' \wedge v \neq 0^n$ is the size of the union for a combination of sources. We can thus write Equation 9 as:

$$C(p, s) = 1 - \frac{\sum_{v' \wedge v = v} freq(v')}{\sum_{v'' \wedge v \neq 0^n} freq(v'')} \quad (12)$$

We can precompute the confidence score $C(p, s)$ for all possible intersections of sources in D_i , but we have to store 2^{m*l} confidence values for one domain with m data and l link sources. This means, we can precompute the size of partitions and unions only for a limited number of data and link sources. But we expect that in real world applications such as Columba this will not be a problem. If the problem arises, some heuristics for precomputation must be introduced.

7.2 Implementation

Precomputation The integrated database Columba is implemented on PostgreSQL 8.2. In Columba every data and every link source is stored in its own table. For every domain we store the domain-vectors v in a separate table. We precompute and store all sizes and frequencies mentioned in the last section in statistics tables. To compute $freq(v')$ and $freq(v'')$ we use the provided functions `bit_and()` and `bit_or()` of PostgreSQL.

Execution of Queries We now describe how to use the precomputed values to compile a ranked result set for a given query. The compilation of the result set with scores is done in four steps. For every domain given in the query we first select all annotations s fulfilling the conditions posed in the query and link them to entries in P . In this step we also return the precomputed values for every pair (p, s) , including $C(p, s)$. In the second step we determine k and calculate $S(p, s)$. In the third step we aggregate the surprisingness and confidence scores of a single domain for an object p . In the last step – if the query poses conditions on multiple domains – we aggregate the scores for an object p over all domains.

We will explain this by a simple example that selects chains of protein structures from the PDB that are supported by entries in Swiss-Prot, which contain the keyword *Phenylalanine catabolism*. Figure 6 shows the SQL query to find all combinations of PDB chains and qualified entries in Swiss-Prot. For every combination we return the values for c_1 and c_2 and the confidence score $C(p, s)$. In the next step we determine k by counting all unique Swiss-Prot ids and then compute $S(p, s)$. In the last step we aggregate the scores for every PDB chain by averaging over the surprisingness scores and sum over the confidence scores.

```
SELECT seq_int_links.pdb_chain, swissprot.id,
       stats.c1, stats.c2, stats.confidence_ps
FROM   swissprot, seq_int_data, seq_int_links, stats
WHERE  swissprot.keyword = 'Phenylalanine catabolism'
AND    swissprot.id = seq_int_data.swissprot_id
AND    seq_int_data.vector = stats.vector
AND    swissprot.id = seq_int_links.swissprot_id
```

Fig. 6. SQL query to return all PDB chain - Swiss-Prot id combinations given the keyword *Phenylalanine catabolism* and some constants.

7.3 Evaluation

Overlap of Sources in Columba The three data sources that link Swiss-Prot entries to chains in the PDB have an overlap of 51,051, i.e., most of the links of MSD (total 69,785) and PDBSWS (total 69,303) are contained in that overlap (data not shown). Seq2Struct contains in total 216,539 links, i.e., most links between the PDB and Swiss-Prot are only contained in that source. The overlap

for the data sources of metabolic networks is given in Figure 1. aMAZE contains the highest number of enzyme-enzyme combinations, mainly due to the fact that reactions in aMAZE are always bi-directional. In the publications domain we have 73,945 links from PDB chains directly to PubMed, most of which are contained in the 223,156 links over Swiss-Prot to PubMed.

Queries on Columba To evaluate our approach we queried the Columba database using keywords on a single domain, e.g., "Give me all PDB chains annotated by Swiss-Prot entries that contain the keyword *Phenylalanine catabolism*". We used all distinct keywords from Swiss-Prot (in total 881) to query the sequence domain and 1,000 randomly chosen MeSH terms to query the publication domain. For evaluation we excluded empty result sets and result sets in which all entries had the same confidence or surprisingness score. This results in 727 result sets for the sequence domain and 695 for the publication domain.

For every result set we normalized the confidence and surprisingness scores to gain values between 0 and 1. We sorted entries in the result set into 11 buckets ($[0, 0.1)$, $[0.1, 0.2)$, ..., and an own bucket for $[1]$) according to their confidence or surprisingness scores. Figures 7(a)-(d) show the average frequencies of entries in a bucket for the result sets of the sequence and publication domain.

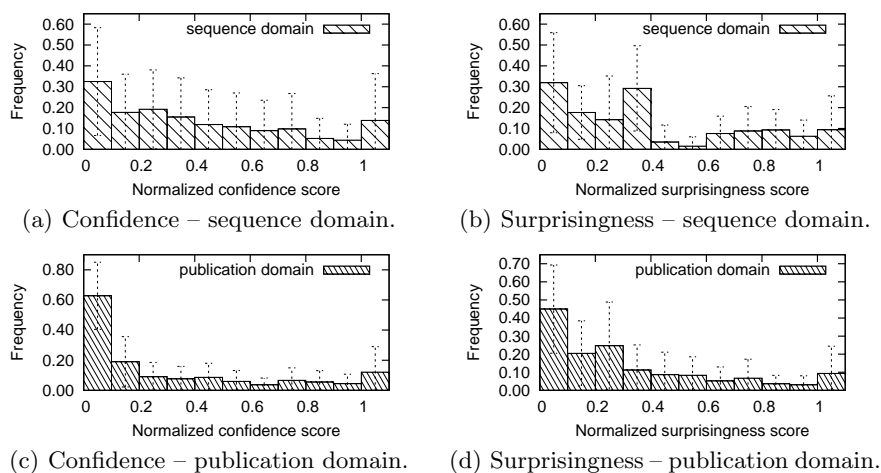


Fig. 7. Average frequency and standard deviation of normalized confidence and surprisingness scores for queries on the sequence and the publication domain.

Figures 7(a) and (c) show that on average only 14 % of the entries in a result set for the sequence and 12 % for the publication domain have a normalized confidence value of 1. Most entries in the result set (50 % for sequence and 82 % for publication domain) have normalized confidence values between 0 and 0.2. Figures 7(b) and (d) show the data for the surprisingness score. In both domains

on average about 9 % of entries in result sets have a normalized surprisingness score of 1. Here as well the largest bucket is the bucket that contains entries with scores between 0 and 0.1. The high standard deviation for all buckets can be explained by varying distributions of scores within the result sets. Consider a result set in which the entries only have two different scores, which is typical for small result sets. Clearly, a subset of entries will be in the bucket with value 1, while the other subset is in one of the remaining 10 buckets. This subset can contain one entry or all but one entry of the result set. The figures for the metabolic pathway domain are not displayed. Concluding, the figures indicate that both scores will nicely rank entries in the result set for the given domains.

We now present an exemplary query on multiple domains and parts of its result set. The query "Give me all PDB chains that are annotated by Swiss-Prot entries with the keyword *Glycolysis*, that are linked to PubMed articles containing the word *Glycolysis*, and that are at most three steps away from the protein with EC number 2.7.1.1 (Hexokinase)" returns 109 chains from the PDB. Table 1(a) and 1(b) show the top 5 results sorted either by confidence or surprisingness.

Table 1. The top 5 query results for different sorting.

(a) Sorted by Confidence				(b) Sorted by Surprisingness			
PDB id	chain	Confidence	Surprisingness	PDB id	chain	Confidence	Surprisingness
1dqr	A	1.0	20.6	1pky	C	0.1	22.8
1dqr	B	1.0	20.6	2pgi	-	0.5	22.2
1g98	A	0.6	18.3	1c7q	A	0.5	22.2
1g98	B	0.6	18.3	1c7r	A	0.5	22.2
1xtb	A	0.6	18.3	1i33	D	0.1	22.2

The most confident results are structures for the protein phosphoglucose isomerase. This is expected as the protein is only one reaction away from the hexokinase in the glycolysis pathway. Note, the top 5 most surprising results contain completely different chains in the PDB, including a pyruvate kinase (1pky) that is also in the glycolysis pathway, but further away from hexokinase than phosphoglucose isomerase.

8 Conclusion

In this paper we defined the surprisingness and the confidence score for an object in a result set that is annotated by multiple, possibly overlapping data sources. Both scores can be used to rank objects in a result set. We showed its applicability to biological data using parts of the integrated database Columba. In the future we plan to integrate both scoring schemes in the Columba web interface. In addition we will further investigate the possibility to extend both score

definitions to also account for the distribution of source combinations within a single result set.

References

1. Berman, H., Westbrook, J., Feng, Z., Gilliland, G., et al. The Protein Data Bank. *Nucleic Acids Research*, 28(1):235–242, Jan 2000.
2. Bleiholder, J., Khuller, S., Naumann, F., Raschid, L., and Wu, Y. Query Planning in the Presence of Overlapping Sources. In *Proceedings of the EDBT*, pages 811–828, 2006. Springer.
3. Boeckmann, B., Bairoch, A., Apweiler, R., Blatter M.-C., et al. The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003. *Nucleic Acids Research*, 31(1):365–370, Jan 2003.
4. Florescu, D., Koller, D., and Levy, A. Using Probabilistic Information in Data Integration. In *Proceedings of the VLDB*, pages 216–225, 1997. Morgan Kaufmann.
5. Huttenhower C., Hibbs, M., Myers, C., and Troyanskaya, O. A scalable method for integration and functional analysis of multiple microarray datasets. *Bioinformatics*, 22(23):2890–2897, Dec 2006.
6. Joshi-Tope, G., Gillespie, M., Vastrik, I., D’Eustachio, P., et al. Reactome: a knowledgebase of biological pathways. *Nucleic Acids Research*, 33:D428–D432, Jan 2005.
7. Kanehisa, M., Goto, S., Kawashima, S., and Nakaya, A. The KEGG databases at GenomeNet. *Nucleic Acids Research*, 30:42–46, Jan 2002.
8. Lacroix, Z., Murthy, H., Naumann F., and Raschid, L. Links and Paths through Life Sciences Data Sources. In *Proceeding of the DILS in Lecture Notes in Computer Science*, volume 2994, pages 203–211, 2004. Springer.
9. Lemer, C., Antezana, E., Couche, F. and Fays, F., et al. The aMAZE LightBench: a web interface to a relational database of cellular processes. *Nucleic Acids Research*, 32:D443–D448, Jan 2004.
10. Marcotte, E., Pellegrini, M., Ng, H., Rice, D., et al. Detecting protein function and protein-protein interactions from genome sequences. *Science*, 285(5428):751–753, Jul 1999.
11. Martin A. C. Mapping PDB chains to UniProtKB entries. *Bioinformatics*, 21(23):4297–4301, Dec 2005.
12. von Mering, C., Jensen, L., Snel, B., Hooper, S., et al. STRING: known and predicted protein-protein associations, integrated and transferred across organisms. *Nucleic Acids Research*, 33:D433–D437, Jan 2005.
13. Naumann, F., Leser, U., Freytag, J.-C. Quality-driven Integration of Heterogenous Information Systems. In *Proceedings of the VLDB*, pages 447–458, 1999. Morgan Kaufmann.
14. Rother, K., Müller, H., Trissl, S., Koch, I., et al. Columba: Multidimensional Data Integration of Protein Annotations. In *Proceeding of the DILS in Lecture Notes in Computer Science*, volume 2994, pages 156–171, 2004. Springer.
15. Velankar, S., Mcneil, P., Mittard-Runte, V., Suarez, A., et al. E-MSD: an integrated data resource for bioinformatics. *Nucleic Acids Research*, 33:D262+, Jan 2005.
16. Via, A., Zanzoni, A., and Helmer-Citterich, M. Seq2Struct: a resource for establishing sequence-structure links. *Bioinformatics*, 21(4):551–553, Sep 2004.
17. Yanai, I. and DeLisi, C. The society of genes: networks of functional links between genes from comparative genomics. *Genome Biology*, 3(11):research0064, Oct 2002.

A Anhang - Implementierung

A.1 Quellen und Partitionen

MSD	PDBSWS	Seq2Struct
78075	83744	747243

Tabelle 1: Kardinalitäten der integrierten Link-Quellen der Dimension Sequence

PDB	MSD	PDBSWS	Seq2Struct
126446	65935	67564	195349

Tabelle 2: Kardinalitäten der Link-Quellen der Dimension Publications

aMAZE	KEGG	Reactome
9471	1843	325

Tabelle 3: Kardinalitäten der Daten-Quellen der Dimension Pathways

domain-vector	MSD	PDBSWS	Seq2Struct	Kardinalität
001	-	-	+	678295
010	-	+	-	6036
011	-	+	+	7423
100	+	-	-	5379
101	+	-	+	2411
110	+	+	-	11171
111	+	+	+	59114

Tabelle 4: Partitionen der integrierten Link-Quelle der Dimension Sequence

domain-vector	PDB	MSD	PDBSWS	Seq2Struct	Kardinalität
0001	-	-	-	+	135232
0010	-	-	+	-	2197
0011	-	-	+	+	4965
0100	-	+	-	-	3575
0101	-	+	-	+	2045
0110	-	+	+	-	6241
0111	-	+	+	+	39075
1000	+	-	-	-	110482
1001	+	-	-	+	335
1010	+	-	+	-	110
1011	+	-	+	+	520
1100	+	+	-	-	221
1101	+	+	-	+	322
1110	+	+	+	-	1601
1111	+	+	+	+	12855

Tabelle 5: Partitionen der integrierten Link-Quelle der Dimension Publications

domain-vector	aMAZE	KEGG	Reactome	Kardinalität
001	-	-	+	143
010	-	+	-	255
011	-	+	+	2
100	+	-	-	7774
101	+	-	+	111
110	+	+	-	1517
111	+	+	+	69

Tabelle 6: Partitionen der integrierten Daten-Quelle der Dimension Pathways

A.2 Vorberechnung

Wie in Abschnitt 7.1 des Papers beschrieben, werden die verschiedenen Daten- und Link-Quellen einer Dimension zunächst physisch integriert. Jedem Link wird dabei ein Bit-Vektor, der sogenannte Domain-Vektor zugeordnet, der angibt welche Link-Quellen diesen Link enthalten und in welchen Datenquellen die entsprechende Annotation enthalten ist. In Columba vereinfacht sich diese Zuordnung dadurch, dass überlappende Quellen entweder nur für die Links (Dimensionen Sequence und Publications) oder nur für die Annotationen (Dimension Pathways) zur Verfügung stehen. Die Integration überlappender Quellen wird nun am Beispiel der Dimension Sequence erläutert.

Für die Dimension Sequence stehen in Columba drei verschiedene Link-Quellen zur Verfügung. Diese Quellen verlinken PDB-Chains mit SwissProt-Sequenzen. Ein Link ist ein Tupel (*pdb_id*, *pdb_chain_id*, *swissprot_accession*). Jedes dieser

	Sequence	Publications	Pathways
Integration	61	40	110
Statistiken	41	8	4

Tabelle 7: Für die Vorberechnungen benötigte Zeit gerundet auf Sekunden

Tupel wird um einen Bit-Vektor der Länge drei erweitert. Das 1-Bit im Vektor gibt an aus welcher Datenquelle ein Tupel stammt. Die Integration erfolgt mit einem einzelnen SQL-Statement (Listing 1), das alle Links vereint, nach *pdb_id*, *pdb_chain_id*, *swissprot_accession* gruppiert und durch bitwise-or über die Vektoren aggregiert. Das Ergebnis dieser Aggregation sind die gesuchten Domain-Vektoren. Für die Dimension Publications wird analog vorgegangen. Hierbei können die Domain-Vektoren der Dimension Sequence als Teilergebnis genutzt werden (Listing 2). Für die Dimension Pathways muss dagegen die Partitionierung der Link-Quelle aus der Verteilung der Annotationen über die verschiedenen Datenquellen berechnet werden (Listing 3).

Die Häufigkeiten der Domain-Vektoren können nun durch einfaches Gruppieren und Zählen aus der integrierten Datenquelle ermittelt werden. Aus diesen Häufigkeiten werden wie in Abschnitt 7.1 des Papers ausführlich erläutert die Confidence einzelner Links und die dort definierten Größen c_1 und c_2 vorberechnet. Die entsprechenden SQL-Skripte sind im zweiten Teil des Anhangs zu finden (Listing 4 - 6). Die für die Integration und die Vorberechnung der Statistiken benötigte Zeit liegt im Bereich weniger Minuten. Die auf dem Testsystem ermittelten Zeiten sind in Tabelle 7 aufgeführt.

A.3 Anfragen

Während die Confidence einzelner Links vollständig vorberechnet werden kann, muss die Surprisingness zur Anfragezeit ermittelt werden. Dazu werden die aus der integrierten Datenquelle einer Dimension selektierten Tupel über den Domain-Vektor mit den in einer Statistiktabelle gespeicherten vorberechneten Werten gejoint und in einer temporären Tabelle zwischengespeichert. Aus dieser temporären Tabelle wird die Anzahl der selektierten Annotationen k ermittelt und entsprechend Gleichung 11 (Paper) die Surprisingness der einzelnen Links berechnet. Für jede Dimension wurde zu diesem Zweck eine PL/PGSQL-Funktion implementiert (Listing 7 - 9). In einer zweistufigen Aggregation werden schließlich aus der Confidence und Surprisingness einzelner Links die entsprechenden Werte für multidimensionale Anfrageergebnisse berechnet. Diese Aggregation führt die in Listing 10 angegebene PL/PGSQL-Funktion für beliebige Kombinationen von Dimensionen durch. Diese Funktion berechnet zunächst Surprisingness und Confidence einzelner Anotationen entsprechend Gleichung 8 bzw. 10 (Paper) und aggregiert diese anschließend wie in Abschnitt 5 des Papers beschrieben. Die für 3-dimensionale Anfragen benötigte Zeit lag bei manuellen

Tests zwischen 3 und 7 Sekunden. Ein randomisiertes Testen ist nicht möglich, da zufällige mehrdimensionale Anfragen i.A. leere Ergebnismengen haben. Es ist aber davon auszugehen, dass die Laufzeiten der Testanfragen repräsentativ sind.

B Anhang - Quelltexte

B.1 Vorbereitung

Listing 1: Integration von Datenquellen der Dimension Sequence

```
— integrate link sources
DROP TABLE lap.preagg_pdbswiss;

SELECT  pdb_id, chain_id, accession,
        bit_or(bit_vector) AS bit_vector
INTO lap.preagg_pdbswiss
FROM(
    SELECT  pdb_id, chain_id, accession,
            B'001' AS bit_vector
    FROM lap.pdbswiss_seq2struct
    UNION
    SELECT  pdb_id, chain_id, accession,
            B'010' AS bit_vector
    FROM lap.pdbswiss_pdbsws
    UNION
    SELECT  pdb_id, chain_id, accession,
            B'100' AS bit_vector
    FROM lap.pdbswiss_msd
) AS agg_helper
GROUP BY pdb_id, chain_id, accession;

ALTER TABLE lap.preagg_pdbswiss
    ADD PRIMARY KEY(pdb_id, chain_id, accession);
```

Listing 2: Integration von Datenquellen der Dimension Publications

```
— integrate link sources
DROP TABLE lap.preagg-pdbpubmed;

SELECT  pdb_id, chain_id, pubmed_id,
        bit_or(bit_vector) AS bit_vector
INTO lap.preagg-pdbpubmed
FROM(
    SELECT  pdb_id, chain_id, pubmed_id,
            B'0' || bit_vector AS bit_vector
    FROM lap.preagg-pdbswiss,
         lap.pubmed.swissprot2pubmed
    WHERE accession = swissprot_accession
    UNION
    SELECT  pdb_id, chain_id, pubmed_id,
            B'1000' AS bit_vector
    FROM lap.pubmed.pdb2pubmed
) AS agg_helper
GROUP BY pdb_id, chain_id, pubmed_id;

ALTER TABLE lap.preagg-pdbpubmed
    ADD PRIMARY KEY(pdb_id, chain_id, pubmed_id);
```

Listing 3: Integration von Datenquellen der Dimension Pathways

```

— integrate data sources
DROP TABLE lap.preagg_path;

SELECT from_ec, to_ec, distance,
       bit_or(bit_vector) AS bit_vector
INTO lap.preagg_path
FROM(
  SELECT from_ec, to_ec, distance,
         B'001' AS bit_vector
  FROM lap.path_tc_ec_min_reactome
  UNION
  SELECT from_ec, to_ec, distance,
         B'010' AS bit_vector
  FROM lap.path_tc_ec_min_kegg
  UNION
  SELECT from_ec, to_ec, distance,
         B'100' AS bit_vector
  FROM lap.path_tc_ec_min_amaze
) AS agg_helper
GROUP BY from_ec, to_ec, distance;

UPDATE lap.preagg_path SET distance = distance/2;

ALTER TABLE lap.preagg_path
  ADD PRIMARY KEY (from_ec, to_ec, distance);

— partitionate link source
SELECT * INTO lap.path_temp FROM lap.preagg_path
WHERE distance = 1;

DROP TABLE lap.preagg_pdbpath;

SELECT pdb_id, chain_id, ec,
       bit_or(bit_vector) AS bit_vector
INTO lap.preagg_pdbpath
FROM (
  SELECT to_ec, bit_vector
  FROM lap.path_temp
  GROUP BY to_ec, bit_vector) AS t1,
  lap.pdb2ec AS t2
WHERE ec = to_ec
GROUP BY pdb_id, chain_id, ec;

ALTER TABLE lap.preagg_pdbpath
  ADD PRIMARY KEY (pdb_id, chain_id, ec);

DROP TABLE lap.path_temp;

```

Listing 4: Vorberechnung statistischer Werte für Dimension Sequence

```

— link statistics
DROP TABLE lap.pdbswiss_stats;
CREATE TABLE lap.pdbswiss_stats(
    bit_vector bit(3) PRIMARY KEY,
    frequency int8,
    confidence float8,
    c_2 float8,
    c_1 float8
);

INSERT INTO lap.pdbswiss_stats(bit_vector, frequency)
    SELECT bit_vector, count(*) FROM lap.preagg_pdbswiss
    GROUP BY bit_vector ORDER BY bit_vector;

— partially precompute surprisingness
UPDATE lap.pdbswiss_stats SET c_1 =
    (SELECT count(DISTINCT accession) FROM lap.preagg_pdbswiss)*
    (SELECT count(*) FROM
        (SELECT DISTINCT pdb_id, chain_id
         FROM lap.preagg_pdbswiss) as h);

UPDATE lap.pdbswiss_stats SET c_2 =
    frequency::float8 / c_1;
UPDATE lap.pdbswiss_stats SET c_1 =
    (SELECT count(*) FROM lap.preagg_pdbswiss)/c_1;

— precompute confidence
UPDATE lap.pdbswiss_stats t1 SET confidence = 1-log((
    SELECT sum(frequency)::float8 FROM lap.swiss_stats t2
    WHERE t1.bit_vector & t2.bit_vector = t1.bit_vector
)::FLOAT8 / (
    SELECT sum(frequency)::float8 FROM lap.swiss_stats t3
    WHERE t1.bit_vector & t3.bit_vector != B'000'
)::FLOAT8)/log(2);

```

Listing 5: Vorberechnung statistischer Werte für Dimension Publications

```

— link statistics
DROP TABLE lap.pdbpubmed_stats;
CREATE TABLE lap.pdbpubmed_stats(
    bit_vector bit(4) PRIMARY KEY,
    frequency INTEGER,
    confidence float8 ,
    c_2 float8 ,
    c_1 float8
);

INSERT INTO lap.pdbpubmed_stats(bit_vector , frequency)
    SELECT bit_vector , count(*) FROM lap.preagg_pdbpubmed
    GROUP BY bit_vector ORDER BY bit_vector;

— partially precompute surprisingness
UPDATE lap.pdbpubmed_stats SET c_1 =
    (SELECT count(DISTINCT pubmed_id) FROM lap.preagg_pdbpubmed)*
    (SELECT count(*) FROM (
        SELECT DISTINCT pdb_id , chain_id
        FROM lap.preagg_pdbpubmed) AS h);

UPDATE lap.pdbpubmed_stats SET c_2 =
    frequency::float8 / c_1;
UPDATE lap.pdbpubmed_stats SET c_1 =
    (SELECT count(*) FROM lap.preagg_pdbpubmed)/c_1;

— precompute confidence
UPDATE lap.pdbpubmed_stats t1 SET confidence = 1-log((
    SELECT sum(frequency)::float8 FROM lap.pubmed_stats t2
    WHERE t1.bit_vector & t2.bit_vector = t1.bit_vector
)::FLOAT8 / (
    SELECT sum(frequency)::float8 FROM lap.pubmed_stats t3
    WHERE t1.bit_vector & t3.bit_vector != B'0000'
)::FLOAT8)/log(2);

```

Listing 6: Vorbereitung statistischer Werte für Dimension Pathways

```

— link statistics
DROP TABLE lap.pdbpath_stats;
CREATE TABLE lap.pdbpath_stats(
    bit_vector bit(3) PRIMARY KEY,
    frequency int8,
    confidence float8,
    c_2 float8,
    c_1 float8
);

INSERT INTO lap.pdbpath_stats(bit_vector, frequency)
SELECT bit_vector, count(*) FROM lap.preagg_pdbpath
GROUP BY bit_vector ORDER BY bit_vector;

— partially precompute surprisingness
UPDATE lap.pdbpath_stats SET c_1 =
    (SELECT count(DISTINCT ec) FROM lap.preagg_pdbpath)*
    ((SELECT count(*) FROM (
        SELECT DISTINCT pdb_id, chain_id
        FROM lap.preagg_pdbpath) AS h));

UPDATE lap.pdbpath_stats SET c_2 =
    frequency::float8 / c_1;
UPDATE lap.pdbpath_stats SET c_1 =
    ((SELECT count(*) FROM lap.preagg_pdbpath) / c_1);

— precompute confidence
UPDATE lap.pdbpath_stats t1 SET confidence = 1-log((
    SELECT sum(frequency)::float8 FROM lap.path_stats t2
    WHERE t1.bit_vector & t2.bit_vector = t1.bit_vector
)::FLOAT8 / (
    SELECT sum(frequency)::float8 FROM lap.path_stats t3
    WHERE t1.bit_vector & t3.bit_vector != B'000'
)::FLOAT8) / log(2);

```

B.2 Anfragen

Listing 7: Confidence und Surprisingness für Links der Dimension Sequence

```
CREATE OR REPLACE FUNCTION lap.query_pdbswiss_keyword(keyword text)
RETURNS SETOF lap.query_pdbswiss AS $$
DECLARE
    row RECORD;
    result_size INTEGER;
BEGIN
    — create temporary table for intermediate results
    EXECUTE 'CREATE TEMPORARY TABLE query_pdbswiss_temp '
        || '(LIKE lap.query_pdbswiss) ON COMMIT DROP';

    — get precomputed results and statistics
    SET enable_mergejoin TO OFF;
    EXECUTE 'INSERT INTO query_pdbswiss_temp ('
        || 'SELECT DISTINCT preagg.pdb_id, preagg.chain_id, '
        || 'preagg.accession, preagg.bit_vector, '
        || 'stats.c_2, stats.c_1, stats.confidence, '
        || '0.0 AS surprisingness '
        || 'FROM lap.preagg_pdbswiss preagg, '
        || 'lap.pdbswiss_stats stats, '
        || 'lap.swissprot_keywords kwords '
        || 'WHERE stats.bit_vector = preagg.bit_vector '
        || 'AND preagg.accession = kwords.accession '
        || 'AND kwords.keyword = ''' || keyword || ''')';
    SET enable_mergejoin TO ON;

    — compute result size
    EXECUTE 'SELECT count(DISTINCT accession) '
        || 'FROM query_pdbswiss_temp' INTO result_size;

    — compute surprisingness
    — using precomputed constants and result size
    EXECUTE 'UPDATE query_pdbswiss_temp SET surprisingness = '
        || '(1-(1-c_2)^(1/result_size)) / '
        || '(1-(1-c_1)^(1/result_size))';

    EXECUTE 'UPDATE query_pdbswiss_temp SET surprisingness = '
        || '-log(surprisingness) / log(2.0::float8)';

    — return results
    FOR row IN EXECUTE 'SELECT * FROM query_pdbswiss_temp' LOOP
        RETURN NEXT row;
    END LOOP;
END;
$$ LANGUAGE 'plpgsql';
```

Listing 8: Confidence und Surprisingness für Links der Dimension Publications

```

CREATE OR REPLACE FUNCTION lap.query_pubmed_term(term text)
RETURNS SETOF lap.query_pubmed AS $$
DECLARE
    row RECORD;
    result_size INTEGER;
BEGIN
    — create temporary table for intermediate results
    EXECUTE 'CREATE TEMPORARY TABLE query_pubmed_temp '
        || '(LIKE lap.query_pubmed) ON COMMIT DROP';

    — get precomputed results and statistics
    SET enable_mergejoin TO OFF;
    EXECUTE 'INSERT INTO query_pubmed_temp ('
        || 'SELECT DISTINCT preagg.pdb_id, preagg.chain_id, '
        || 'preagg.pubmed_id, preagg.bit_vector, '
        || 'stats.c_2, stats.c_1, stats.confidence, '
        || '0.0 AS surprisingness '
        || 'FROM lap.preagg_pdbpubmed preagg, '
        || 'lap.pdbpubmed_stats stats, '
        || 'columba.text_pubmed fulltext '
        || 'WHERE stats.bit_vector = preagg.bit_vector '
        || 'AND preagg.pubmed_id = fulltext.pubmed_id '
        || 'AND (public.to_tsquery('' ' || term || ''') @@ idxfti))';
    SET enable_mergejoin TO ON;

    — compute result size
    EXECUTE 'SELECT count(DISTINCT pubmed_id) '
        || 'FROM query_pubmed_temp' INTO result_size;

    — compute surprisingness
    — using precomputed constants and result size
    EXECUTE 'UPDATE query_pubmed_temp SET surprisingness = '
        || '(1-(1-c_2)^(result_size || ')) / '
        || '(1-(1-c_1)^(result_size || '))';

    EXECUTE 'UPDATE query_pubmed_temp SET surprisingness = '
        || '-log(surprisingness) / log(2.0::float8)';

    — return results
    FOR row IN EXECUTE 'SELECT * FROM query_pubmed_temp'
    LOOP
        RETURN NEXT row;
    END LOOP;
END;
$$ LANGUAGE 'plpgsql';

```

Listing 9: Confidence und Surprisingness für Links der Dimension Pathways

```

CREATE OR REPLACE FUNCTION lap.query_path_ec(ec text, distance int)
RETURNS SETOF lap.query_path AS $$
DECLARE
    row RECORD;
    result_size INTEGER;
BEGIN
    EXECUTE 'CREATE TEMPORARY TABLE query_path_temp '
        || '(LIKE lap.query_path) ON COMMIT DROP';

    SET enable_mergejoin TO OFF;
    EXECUTE 'INSERT INTO query_path_temp ('
        || 'SELECT DISTINCT pdb2ec.pdb_id, pdb2ec.chain_id, '
        || 'pdb2ec.ec, preagg.bit_vector, '
        || 'stats.c_2, stats.c_1, stats.confidence, '
        || '0.0 AS surprisingness '
        || 'FROM lap.preagg_path preagg, '
        || 'lap.pdbpath_stats stats, '
        || 'lap.path_pdb2ec pdb2ec '
        || 'WHERE stats.bit_vector = preagg.bit_vector '
        || 'AND preagg.from_ec = ''' || ec || ''' '
        || 'AND distance < ' || distance
        || ' AND pdb2ec.ec = preagg.to_ec)';
    SET enable_mergejoin TO ON;

    EXECUTE 'SELECT count(DISTINCT ec_number) AS count_all '
        || 'FROM query_path_temp' INTO result_size;

    EXECUTE 'UPDATE query_path_temp SET surprisingness = '
        || '(1-(1-c_2)^( ' || result_size || ')) / '
        || '(1-(1-c_1)^( ' || result_size || '))';

    EXECUTE 'UPDATE query_path_temp SET surprisingness = '
        || '-log(surprisingness) / log(2.0::float8)';

    FOR row IN EXECUTE 'SELECT * FROM query_path_temp'
    LOOP
        RETURN NEXT row;
    END LOOP;
END;
$$ LANGUAGE 'plpgsql';

```

Listing 10: Confidence und Surprisingess für multidimensionale Anfrageergebnisse

```

CREATE OR REPLACE FUNCTION
lap.query_multiple_dimensions
(keyword text, term text, ec text, distance int)
RETURNS SETOF lap.query_multiple AS $$
DECLARE
    row RECORD;
    query VARCHAR;
    dimensions BIT(3);
BEGIN
    query = 'SELECT DISTINCT pdb_id::CHAR(4), chain_id::CHAR(1), '
    || 'prod(confidence)::FLOAT8 AS confidence, '
    || 'sum(surprisingness)::FLOAT8 AS surprisingness FROM(';

    dimensions = B'000';

    IF NOT keyword IS NULL THEN
        dimensions = dimensions | B'100';
        query = query ||
        'SELECT pdb_id, chain_id, B''100'' AS bit_vector, '
        || 'sum(confidence) AS confidence, '
        || 'avg(surprisingness) AS surprisingness FROM '
        || 'lap.query_pdbswiss_keyword(''' || keyword || ''')'
        || 'GROUP BY pdb_id, chain_id';
    END IF;

    IF NOT term IS NULL THEN
        dimensions = dimensions | B'010';
        IF NOT keyword IS NULL THEN
            query = query || ' UNION ';
        END IF;
        query = query ||
        'SELECT pdb_id, chain_id, B''010'' AS bit_vector, '
        || 'sum(confidence) AS confidence, '
        || ' avg(surprisingness) AS surprisingness '
        || 'FROM lap.query_pubmed_term(''' || term || ''')'
        || 'GROUP BY pdb_id, chain_id';
    END IF;

    IF NOT ec IS NULL THEN
        dimensions = dimensions | B'001';
        IF NOT keyword IS NULL OR NOT term IS NULL THEN
            query = query || ' UNION ';
        END IF;
        query = query ||
        'SELECT pdb_id, chain_id, B''001'' AS bit_vector, '
        || 'sum(confidence) AS confidence, '
        || 'avg(surprisingness) AS surprisingness FROM '
        || 'lap.query_path_ec(''' || ec || ''', ' || distance || ''')'
        || 'GROUP BY pdb_id, chain_id';

```

```

END IF;

query = query || ') AS agg_helper GROUP BY pdb.id, '
      || 'chain_id HAVING prod(confidence) > 0 '
      || 'AND bit_or(bit_vector)::INTEGER = '
      || dimensions::INTEGER;

FOR row IN EXECUTE query
LOOP
    RETURN NEXT row;
END LOOP;
END;
$$ LANGUAGE 'plpgsql';

```

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die die vorliegende Studienarbeit selbstständig und nur unter Zuhilfenahme der angegebenen Quellen angefertigt habe.

Berlin, den 23. August 2007

Philipp Hussels