Querying BLAST within a Data Federation

Barbara A. Eckman
IBM Life Sciences
1475 Paoli Pike
West Chester, PA 19380
baeckman@us.ibm.com

Arthur Kaufmann
IBM Silicon Valley Laboratory
555 Bailey Rd
San Jose, CA 95141
artkauf@us.ibm.com

Abstract

BLAST (Basic Local Alignment Search Tool) is one of the most widely used algorithms in bioinformatics and genomics. Frequently Life Science researchers wish to integrate the BLAST algorithm with other related data sources, either to supply BLAST query sequences or to provide additional annotations on sequences that are found to match. Another frequent need is to filter BLAST alignments based on match stringency or more complex criteria (e.g., the amino acid composition of the alignments). Further, in the context of data federation, wide-ranging multi-source queries involving BLAST searches often return unmanageably large result sets, requiring approaches that go beyond vanilla SQL to exclude extraneous data. Since 2001, IBM has provided the ability to access BLAST from within SQL queries, thus integrating BLAST results with relevant data from a wide variety of data sources, both local and remote, regardless of their format. In this paper we describe IBM® DB2® Information Integrator, IBM's federated database product, along with its BLAST wrapper and a suite of functions supporting complex non-relational queries over the composition of blast alignments.

1 Introduction

The BLAST (Basic Local Alignment Search Tool) [AGM+90] sequence comparison algorithm has a venerable history in bioinformatics and genomics, and is still likely the most widely used tool in these research communities. Frequently researchers wish to integrate BLAST with other related data sources, either to supply BLAST query sequences or to provide additional annotations on sequences that are found to match. While publicly funded organizations like the National Center for Biotechnology Information (www.ncbi.nlm.nih.gov) and the Swiss Institute for Bioinformatics (www.expasy.org) offer browser-based integration between BLAST and data sources such as GenBank [WCE+04] and Swiss-Prot/TrEMBL [BBFG04], they do not support integrating BLAST and related data sources via a declarative query language. Another frequent need among researchers is to filter BLAST alignments based on match stringency or more complex criteria (e.g., the amino acid composition of the alignments). Popular BLAST implementations themselves support only a limited set of filters over BLAST results (e.g., an E-value threshold). Since wide-ranging multi-source queries involving BLAST often return unmanageably large result sets, excluding extraneous data and thereby limiting the number of results that an expert scientist has to examine is a high priority. These filter criteria can be much more complex than a simple E-value threshold, often surpassing what can be expressed in vanilla SQL.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

Our solution to these critical needs is to provide access to BLAST within a federated [HM85] relational query engine, thus providing full SQL query capability over BLAST results and enabling joins between BLAST inputs and outputs and other data sources in the federation. In addition, special-purpose functions provide regular expression pattern-matching capability over the composition of BLAST alignments.

2 IBM DB2 Information Integrator

DB2 Information Integrator (DB2 II) is a federated RDMBS that traces its lineage from the Garlic Project at IBM's Almaden Research Center [CHS+95], IBM's DataJoiner® product (www.ibm.com/software/data/datajoiner), and our DiscoveryLink® solution [HSK+01]. A federated strategy is appropriate in the bioinformatics research community, since information is not contained solely in databases, but instead exists in a multiplicity of formats and is accessed by a wide variety of tools and algorithms. A typical question asked by a researcher may span many separate data stores, as illustrated in the following scenario.

Voltage-sensitive calcium channel proteins mediate the entry of calcium ions into cells, and are involved in such processes as neurotransmitter release. The discovery of a novel channel gene that codes for a protein mediating calcium or similar ions into cells would potentially be of great interest to pharmaceutical researchers. A popular method of novel gene discovery is to search EST (expressed) sequence databases for sequences similar to known genes or proteins.

To identify potential new human neurological drug targets, a researcher might wish to identify mouse genes annotated as channels that are expressed in central nervous system tissue from the Mouse Genome Database, retrieve the sequences of their protein products from SwissProt, BLAST the protein sequences against the human EST database, return only BLAST alignments that meet certain stringency criteria (e.g., > 60 % identical over > 100 base pairs), and finally retrieve sequence annotations on the hit sequences from GenBank.

Answering this question manually is a daunting and error-prone task, as it entails visiting hundreds of web pages and manually collating and filtering the results. The question can also be answered using custom scripts, but this requires specialized skills and even a minor change in the question can require a substantial reprogramming effort. Another approach to answering the question might entail transferring or copying data from each data store into a single repository (e.g., a relational database) and performing the query from there. This introduces a number of problems, including data quality issues (currency and the potential for errors to be introduced during the transfer process) and an inability to use search tools that were not implemented to work with the repository (e.g., BLAST.)

DB2 II is specifically intended to address issues like these. In a federated system, diverse data stores and search tools (called *data sources*) remain outside the federator, avoiding data quality problems and loss of functionality. Through the use of software bridges called *wrappers* the data sources appear as if they are local to the federating software. The federator can then perform the actions that might otherwise be done manually or through custom programming.

In DB2 II, data sources are represented by a number of persistent objects, the most significant of which is a *nickname*. Nicknames have all the characteristics of relational tables and may be used in an SQL query anywhere a relational table may be used. Nicknames have the additional property that, rather than representing data stored in a relational table, they can represent data stored in many places and accessed through many tools. The sources and tools may be as diverse as relational DBMS's (e.g., Oracle® , Microsoft® SQL Server, Sybase® SQL Server), databases accessible via ODBC (e.g., Microsoft® Access, MySQL), web sites (e.g., Entrez Nucleotide and PubMed at www.ncbi.nlm.nih.gov), web services using SOAP/WSDL (e.g., XEMBL, www.ebi.ac.uk/xembl/XEMBL.wsdl, and KEGG, soap.genome.ad.jp/KEGG.wsdl), Microsoft® Excel spreadsheets, XML files, document repositories (e.g., IBM® Lotus® Extended Search), tabular text files (e.g., comma-separated value files), and search algorithms (e.g., BLAST, HMMER [Edd00].)

Thus, with DB2 II the question posed by the hypothetical researcher above may be answered by a single

SQL query:[1]

```
SELECT
    g.accnum, g.sequence
FROM
    genbank g, blast b, swissprot s, mgd m
WHERE
    m.exp = "CNS"                 /* expressed in central nervous system tissue*/
    AND m.defn LIKE  "%channel%"  /* genes annotated as channel genes */
    AND m.spid = s.id             /* join between MGD and SwissProt */
    AND s.seq = b.query           /* join between SwissProt and BLAST, to
                                     supply BLAST query sequence */
    AND b.hit = g.accnum          /* join between BLAST output and GenBank, to
                                     retrieve full-length sequence */
    AND b.percentid > 60          /* BLAST stringency filters */
    AND b.alignlen > 50
```

Besides ease of use, there are two significant advantages to the federated approach. First, the federator is able to apply sophisticated query optimization techniques to improve performance. The DB2 II Cost Based Optimizer interacts with the wrappers and their specific knowledge of each data source to produce an efficient plan to answer each query. The second advantage is one of compensation. Most non-relational data sources lack advanced filtering capabilities. The federator can compensate for this by applying filters itself. For instance, the *blastall* program cannot filter results by percent identity. In DB2 II, the federator can apply such a filter.

2.1 The Request-Reply-Compensate protocol

The DB2 II optimizer and wrapper cooperate during query planning to produce an optimal plan. This is done through the request-reply-compensate protocol [RS97][IBM03]. The process begins with the optimizer decomposing a query into parts that can be processed locally and parts that must be processed, at least in part, by wrappers and their remote data sources. In the case of a simple, single-table (or nickname) query, this decomposition will be simple. In the case of a query involving multiple tables and nicknames, there may be multiple possible decompositions.

Each decomposed part is called a query fragment. Fragments that reference nicknames for a particular wrapper are presented to that wrapper as a Request. Each Request consists of a list of projected values (head expressions), quantifiers (nicknames) and predicate expressions. The wrapper examines the Request and builds a corresponding Reply, consisting of the elements that the wrapper and data source can process. A predicate may represent a filter that the data source is unable to process. For example, a query fragment sent to the BLAST wrapper may contain a predicate on the length of the alignment. In this case, the predicate will not be added to the Reply.

In some cases, a Reply may be entirely empty. For instance, if the original query contains a join between two nicknames managed by the same wrapper, but that wrapper or data source does not process joins, the entire Request will be rejected. Another example involves data sources like BLAST and most web sites, which have required predicates (e.g., the BLAST input query sequence). If all required predicates are not included in the fragment, the wrapper will reject the Request.

When a Reply is received by the optimizer, it determines which elements from the Request have been left off, and adds local processing to compensate for those missing elements. Once a full set of replies for a single query has been received and local compensations determined, the optimizer uses costing information associated with the replies and the local processing to determine the best plan.

[1]Slightly simplified, with intra-source join predicates omitted for clarity.

3 The DB2 II BLAST wrapper

The BLAST wrapper is one of many wrappers for non-relational Life Science data sources written by IBM and provided with DB2 II. It is designed to work with NCBI *blastall* as well as other similar BLAST engines such as TurboBLAST® (www.turboworx.com). The only requirement for compatibility is that the BLAST search tool support the same command line arguments and return results with the same XML schema as NCBI *blastall*. In wrapping the native BLAST search engines we are able to take advantage of their speed, while incurring minimal overhead.

Each BLAST nickname represents a potential search using a particular algorithm (blastN, blastP, blastX, TblastN, or TblastX) against a particular BLAST-able database (e.g., SwissProt, EMBL [KAA+04], GenBank nr) Inputs to and outputs from the BLAST program are represented by columns of the nickname.

Inputs to the BLAST program are applied through predicates on designated columns. For instance, the sequence to be used in a search is provided through a predicate on the BlastSeq column:

... WHERE BlastSeq = 'ACTCGATC'

Outputs from the BLAST program may be projected out of the query, used in selection predicates, or joined with local tables or other data sources. The BLAST wrapper provides the ability to parse the definition line (*defline*) associated with each entry in a BLAST-able database. This allows the user to extract values such as GI number, accession number, and organismal taxonomy. For example, a scientist may join BLAST output with GenBank Nucleotide on accession number, using the DB2 II Entrez wrapper, to retrieve sequence annotation associated with the BLAST hits, or she may filter BLAST output on organismal taxonomy, returning only hits from mammalian species.

3.1 Improving Performance: Materialized Query Tables

Whether performed in the context of a federated system or on their own, BLAST queries tend to be very expensive, in terms of CPU utilization and elapsed time. Materialized Query Tables (MQTs), a feature of a soon-to-be-released version of DB2 II, enable the reuse of query results, thus yielding cost savings.

MQTs must be explicitly defined by a user or database administrator. An MQT definition consists of a table specification and a query. At some point in time (this varies, depending on the MQT definition), DB2 II executes the MQT query and populates a local table with the results. Then, when DB2 II receives another query that can be partially, or completely, satisfied by the data in the MQT, the DB2 II optimizer produces a plan that references the local table rather than the data source. When the query to the data source is expensive, this results in a savings of time and resources. The ability quickly to re-execute queries over previously generated BLAST results is especially important in cases where the initial queries' predicates were too stringent, resulting in no results returned.

There are caveats with the use of MQTs. The first is data currency. With MQTs that involve nicknames, currently DB2 II must be explicitly instructed to refresh the information in the local table (non-nickname MQTs may be updated automatically in some situations.) The second issue relates to how the MQT is defined: if the definition is not sufficiently broad, the MQT will not be reused and a net loss of resources will result (the time to populate the local table and the storage the local table occupies.)

4 How Scientists use BLAST Within DB2 Information Integrator

Scientists often want to see more annotations on BLAST hit sequences than the BLAST application itself can return, since it is limited to displaying the annotations encoded in the sequences' definition lines in the FASTA input file. They also often want to filter BLAST output based on the content of the definition line, which is not standardized but varies with each individual BLAST-able sequence database. For example, customers at the Center for Medical Genomics and Research and Academic Computing at Indiana University were interested in the following: Given a query sequence, BLAST the GenBank nucleotide database (NT) and return only hits against sequences that were not associated with a cloning vector. For each hit, retrieve the Cluster ID and Title from Unigene (a relational version in DB2), in addition to the GenBank accession number, description and E-Value. Display only the top five hits (i.e., the five lowest E-values). The following SQL query accomplishes this:

```
SELECT
   nt.GB_ACC_NUM, nt.DESCRIPTION, nt.E_VALUE, useq.CLUSTER_ID, ugen.TITLE
FROM
   ncbi.BLASTN_NT nt, unigene.SEQUENCE useq, unigene.GENERAL ugen
WHERE
    BLASTSEQ = 'GGCCGGGCGCGGTGGCTCACGCCTGTAATCCCAGCACTTTGGGAGGC
                CGAGGCGGGCGGATCACGAGGTCAGGAGATCGAGACCATCCTGGCTA
                ACACGGTGAAACCCCGTCTCTACTAAAAATACAAAAATTAGCCGGGC
                GTGGTGGCGGGCGCCTGTAGTCCCAGCTACTC'
    AND  nt.DESCRIPTION  NOT LIKE  '%cloning vector%'
    AND nt.GB_ACC_NUM = useq.ACC
    AND useq.CLUSTER_ID = ugen.CLUSTER_ID
ORDER BY E_VALUE
FETCH FIRST 5 ROWS ONLY
```

Accessing BLAST alignments via SQL also permits filtering BLAST results in ways that BLAST itself does not support, such as by percent identity or alignment length. In addition, DB2 II also permits complex queries over the composition of alignments using IBM's regular expression pattern-matching user-defined functions (UDFs). For example, scientists sometimes want to filter alignments based on motifs found in the target sequence. In the paper introducing PHI-BLAST [ZSM+98], CED4, the C elegans regulator of cell death, is used as the query sequence vs. nr, the non-redundant NCBI protein database. The scientist wishes to return only alignments in which the target sequence includes the P-loop ATPase domain, specified by the PROSITE [HSS+04] motif '[GA]-x(3)-G-K[ST]'. The following query expresses this economically:

```
SELECT
    accnum, definition
FROM
    Blastp b, GBseq gs, gbfeat gf, gbqual gq
WHERE
    gs.primary_accession = 'X69016'          /* CED4 nucleotide sequence */
    AND gs.sequencekey = gf.sequencekey
    AND gf.featurejoinkey = gq.featurejoinkey
    AND gf.FeatureKey = 'CDS'                /* get its coding sequence */
    AND gq.QualifierName = 'translation'     /* translate it to protein */
    AND gq.QualifierValue = b.BlastSeq       /* use protein as blast query */
    AND db2ls.LSPatternMatch(                /* filter by PROSITE motif */
            HSP_H_Seq,
            db2ls.LSPrositePattern('[GA]-x(3)-G-K[ST]')
        ) > 0;
```

This query calls two DB2 Life Science-specific built-in functions: LSPatternMatch(*string*, *Perl regular expression*) takes as input a string and a pattern in Perl regular expression syntax, and returns the number of maximal matches in the string. LSPrositePattern(*string*) takes a string expressing a pattern in PROSITE's motif syntax and returns the equivalent Perl regular expression.

In another example from our prior work with the OPM/TINet federated system [EKL01], when BLASTing with a proline-rich query sequence, a scientist may want to return only alignments that contain $< 25\%$ prolines among their perfect matches. To address this query, we have created the following elegantly-named DB2 function, which takes as input three strings and three patterns, finds the location(s) in the three strings where the three patterns are simultaneously satisfied, and outputs the offsets and the maximal matching substrings in each of the three input strings:

```
    CREATE FUNCTION LSMultiMatch3(
        string1     VARCHAR,
        pattern1    VARCHAR,
        string2     VARCHAR
        pattern2    VARCHAR,
        string3     VARCHAR,
        pattern3    VARCHAR)
    RETURNS TABLE (
        position        INTEGER,
        string1_match   VARCHAR,
        string2_match   VARCHAR,
        string3_match   VARCHAR).
```

To exclude overly proline-rich alignments from the BLAST output, the scientist may simply use the following query. Here *p* aliases the number of prolines that match perfectly in the alignment, and *m* aliases the total number of perfect matches. The LSBarCode( ) function reformats the midline string of the alignment, replacing all perfect matches with the symbol "—" to facilitate pattern matching.

```
SELECT
    b.* , float(p)/ float(m) AS percent_prolines
FROM
    BlastOutput b,
    table(SELECT COUNT(*) AS p FROM table(
                db2ls.LSMultiMatch3(
                    b.HSP_Q_Seq, 'P',
                    db2ls.LSBarCode(b.HSP_Midline), '\|',
                b.HSP_H_Seq, 'P')
                 ) AS f
            ) AS y,
    table(SELECT COUNT(*) AS m FROM table(
                db2ls.LSMultiMatch3(
                    b.HSP_Q_Seq, '.',
                    db2ls.LSBarCode(b.HSP_Midline), '\|',
                    b.HSP_H_Seq, '.')
                 ) AS f
            ) AS z
WHERE
    BLASTSEQ =
        "MFETEADEKREMALEEGKGPGAEDSPPSKEPSPGQELPPGQDLPPNKDSPSGQEPAPSQE
        PLSSKDSATSEGSPPGPDAPPSKDVPPCQEPPPAQDLSPCQDLPAGQEPLPHQDPLLTKD
        LPAIQESPTRDLPPCQDLPPSQVSLPAKALTEDTMSSGDLLAATGDPPAAPRPAFVIPEV
        RLDSTYSQKAGAEQGCSGDEEDAEEAEEVEEGEEGEEDEDEDTSDDNYGERSEAKRSSMI"
    AND float(p) / float(m) < 0.25;
```

5 Related Work

The OPM/TINet and K2/Kleisli [DCB+01] federated multidatabase query systems both include wrappers/drivers for BLAST. Instead of the relational data model of DB2 II, they are based on an object-relational data model and a functional programming language model, respectively. These richer data models have advantages in terms of expressivity, but make query optimization more difficult. OPM/TINet caches BLAST results with automatic expiration and refresh, but does not cache on the view level (e.g., a view joining BLAST output with GenBank annotations), as DB2 MQTs do.

Oracle's approach to integrating BLAST in their Oracle® Database10g release (www.oracle.com) requires users to maintain a local copy of sequence data in an Oracle data warehouse. The parsing and loading steps required to maintain such a warehouse constitute a potential point of failure, may introduce errors in the data, and may significantly delay scientists' access to the sequences. Oracle reimplements BLAST as a series of functions that are executed within the database against pairs of sequences; consequently, BLAST output statistics like E-value that are based on the size and composition of the BLAST-able database will differ from the "gold standard" of NCBI BLAST. In addition, any enhancements that NCBI makes to its BLAST implementation must be reimplemented in order to be available to Oracle's BLAST users. Performance will generally be slower than the native BLAST, since the native BLAST optimizations will not be available in the reimplemented version. On the other hand, since BLAST-able datasets are defined dynamically by a SQL query rather than created ahead of time with *formatdb*, if the dynamically defined dataset is much smaller than the closest *formatdb*ed database, there may be a net performance gain, especially with expensive searches like TBlastN. Finally, Oracle's federated capability is limited in terms of optimization strategies and the number and range of Life Science-specific data sources that are currently federated.

PHI-Blast and Bla [TK94] both enable users to filter BLAST output based on motifs found in the target sequences. PHI-Blast uses the motif of interest to restrict the similarity search space, yielding the advantages that in general it can be expected to be faster than a normal BLAST search, and its statistical analysis is tailored to this approach. On the other hand, Bla is similar to our approach in post-processing BLAST results to find motif matches. Neither of these approaches, however, is as flexible as ours, since they allow only limited filters on BLAST output while we provide the full power of SQL in specifying filters, in addition to full Perl regular-expression pattern-matching. Instead of generating a new BLAST executable, we apply our federated query capability to the output of standard BLAST–a more maintainable and scalable strategy.

6 Challenges and Future Directions

As noted, the DB2 II BLAST wrapper is currently certified on NCBI BLAST and TurboBLAST®. Since the BLAST wrapper architecture is designed to support any BLAST engine that implements the NCBI interface, certifying it on additional BLAST engines in the bioinformatics community is an obvious next step.

Beyond the BLAST wrapper itself, there are plenty of areas in which further research is needed. For the query engine, a key topic is the exploitation of parallelism to enhance performance. There is also a need for additional tools and facilities that enhance the basic DB2 II offering. We have done some preliminary work on a system for data annotation that provides a rich model of annotations, while exploiting the DB2 II engine to allow querying of both annotations and data separately and in conjunction. We are also building a tool to help users create mappings between source data and a target, integrated schema [MHH00] to ease the burden of view definition and reconciliation of schemas and data that plagues today's system administrators.

7 Conclusion

The BLAST algorithm is used nearly universally in the bioinformatics community to identify sequence similarity in support of a wide variety of research aims, e.g., gene function characterization, pharmaceutical target identification, protein structure prediction, vaccine development, personalized approaches to healthcare, and bioterrorism detection. In all these activities, integrating BLAST with related data sources, regardless of their location and format, is a critical need. IBM's DB2 Information Integrator is an effective means to efficiently and accurately integrate BLAST into a scientific data source federation, while safeguarding the scientific integrity of the results.

References

[AGM+90] S. Altschul, W. Gish, W. Miller, E. Myers, and D. Lipman. Basic Local Alignment Search Tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.

[BBFG04] A. Bairoch, B. Boeckmann, S. Ferro, and E. Gasteiger. Swiss-Prot: Juggling between Evolution and Stability. *Brief Bioinform*, 5(1):39–55, 2004.

[CHS+95] M. J. Carey, L. M. Haas, P. M. Schwarz, M. Arya, W. F. Cody, R. Fagin, M. Flickner, A. W. Luniewski, W. Niblack, D. Petkovic, J. Thomas, J. H. Williams, and E. L. Wimmers. Towards Heterogeneous Multimedia Information Systems: The Garlic Approach. In *Proceedings of the 5th International Workshop on Research Issues in Data Engineering-Distributed Object Management (RIDE-DOM'95)*, page 124. IEEE Computer Society, 1995.

[DCB+01] S.B. Davidson, J. Crabtree, B.P. Brunk, J. Schug, G.C. Overton, and Jr. C.J. Stoeckert. K2/Kleisli and GUS: Experiments in Integrated Access to Genomic Data Sources. *IBM Systems Journal*, 40(2):512–531, 2001.

[Edd00] S. Eddy. HMMER: Profile Hidden Markov Models for Sequence Analysis. http://hmmer.wustl.edu. 2000.

[EKL01] B. A. Eckman, A. S. Kosky, and L. A. Laroco. Extending Traditional Query-Based Integration Approaches for Functional Characterization of Post-genomic Data. *Bioinformatics*, 17(7):587–601, 2001.

[HM85] D. Heimbigner and D. McLeod. A Federated Architecture for Information Management. *ACM Trans Office Inf Systems*, 3:253–278, 1985.

[HSK+01] L. M. Haas, P. M. Schwarz, P. Kodali, E. Kotlar, J. E. Rice, and W. C. Swope. DiscoveryLink: A System for Integrated Access to Life Sciences Data Sources. *IBM Systems Journal*, 40:489–511, 2001.

[HSS+04] N. Hulo, C.J. Sigrist, S. Le Saux, P.S. Langendijk-Genevaux, L. Bordoli, A. Gattiker, E. De Castro, P. Bucher, and A. Bairoch. Recent Improvements to the PROSITE Database. *Nucleic Acids Res*, 32:D134–D137, 2004.

[IBM03] IBM. DB2 Information Integrator V8.1 Developer's Guide. http://publibfp.boulder.ibm.com/epubs/pdf/c1873590.pdf, 2003.

[KAA+04] T. Kulikova, P. Aldebert, N. Althorpe, W. Baker, K. Bates, P. Browne, A. van den Broek, G. Cochrane, K. Duggan, R. Eberhardt, N. Faruque, M. Garcia-Pastor, N. Harte, C. Kanz, R. Leinonen, Q. Lin, V. Lombard, R. Lopez, R. Mancuso, M. McHale, F. Nardone, V. Silventoinen, P. Stoehr, G. Stoesser, M.A. Tuli, K. Tzouvara, D. Vaughan, R.and Wu, W. Zhu, and R. Apweiler. The EMBL Nucleotide Sequence Database. *Nucleic Acids Res*, 32:D27–D30, 2004.

[MHH00] Renee J. Miller, Laura M. Haas, and Mauricio Hernandez. Schema Mapping as Query Discovery. *Proc. of the Conf. on Very Large Data Bases (VLDB), Cairo, Egypt*, pages 77–88, 2000.

[RS97] M. Tork Roth and P Schwarz. Don't Scrap It, Wrap It! A Wrapper Architecture for Legacy Data Sources. *Proceedings of the 23rd VLDB Conference, Athens, Greece*, 1997.

[TK94] R. L. Tatusov and E. V. Koonin. A Simple Tool to Search for Sequence Motifs that are Conserved in BLAST Outputs. *Computer Applications in the BioSciences (CABIOS)*, 10(4):457–459, 1994.

[WCE+04] D.L. Wheeler, D.M. Church, R. Edgar, S. Federhen, W. Helmberg, T.L. Madden, J.U. Pontius, G. D. Schuler, L.M. Schriml, E. Sequeira, T.O. Suzek, T. A. Tatusova, and L. Wagner. Database resources of the National Center for Biotechnology Information: update. *Nucleic Acids Res*, 32:D35–D40, 2004.

[ZSM+98] Z. Zhang, A. A. Schaffer, W. Miller, T. L. Madden, D. J. Lipman, E. V. Koonin, and S. F. Altschul. Protein Sequence Similarity Searches Using Patterns as Seeds. *Nucl. Acids Res.*, 26(17):3896–990, 1998.