BIOINFORMATICS

Vol. 20 no. 11 2004, pages 1718–1720
doi:10.1093/bioinformatics/bth151

Sight: automating genomic data-mining without programming skills

*Audrius Meskauskas, Frank Lehmann-Horn and Karin Jurkat-Rott**

Department of Applied Physiology, Ulm University, Albert-Einstein-Allee 11, D-89069 Ulm, Germany

Received on July 4, 2002; revised on August 25, 2003; accepted on January 27, 2004
Advance Access publication February 26, 2004

ABSTRACT

Summary: We created and tested Sight, a Java-based package that provides a user-friendly interface to generate and connect agents for automatic genomic data-mining for individual requirements without requiring programming skills from the user.

Availability: http://physiologie.uni-ulm.de//Seiten/Arbeitsgruppe/Jurkat-Rott/Jurkat-Rott.htm. The system does not require additional components and runs on IBM PCs under Windows (NT 4.0, 2000 and XP) or Linux (Phat 4.0 and Mandrake 9.0).

Contact: karin.jurkat-rott@medizin.uni-ulm.de

Simple Web agents like WebBlast (Ferlanti *et al*., 1999) or BioQuery (Brundege and Dubay, 2003) do not allow sequential connection of genome database servers. More complex data-mining systems provide an attractive alternative to sequential manual Internet submission by automating the most commonly used computations. However, these systems either follow only pre-determined workflows [Genotator (Harris, 2000), Pedant (Frishman *et al*., 2001), EDITto-TrEMBL (Moller *et al*., 1999)] or enable assembly of custom workflows using only fixed elements [Kleisli (Kolatkar *et al*., 1998), Tambis (Stevens *et al*., 2000) or GAIA (Bailey *et al*., 1998)]. Applications for tasks that need special workflows or that require integration of additional programs and new Web resources are realized by systems that require significant programming skills from the user. Typical members of this group are Jade (Stein *et al*., 1998), BioJava (Mangalam, 2002), BioPerl (Stajich *et al*., 2002) or Decaf (Graham *et al*., 2003). We have created Sight, a system incorporating features of both application types. However, Sight allows the assembly of an arbitrary tree-like workflow without requiring programming skills from the user. Additionally, Sight has generic data structures, can integrate new resources and can perform user-defined conversion from one structure to another—features generally found in large-scale programming-based packages, several of which are only commercially available, such as Ariadne (Knoblock *et al*., 2000).

*To whom correspondence should be addressed.

Sight (Fig. 1) contains a Web form analyzer that identifies data fields in the Web form and generates a test agent that extracts these data. The data are presented to the user, who can then define the data of interest upon which Sight finalizes the agent. In this procedure, four means of data extraction are employed: (i) a table-based analyzer, (ii) a text-based analyzer that identifies space-delimited tables, (iii) a modified Stalker algorithm (Muslea *et al*., 2001) that controls the marking of the data of interest in close vicinity to one another and (iv) a user-defined transformation for XML documents. The Web agents generated can also access DAS servers or connect to SSH servers by uploading parameters and capturing the output.

Sight agents are similar to the agents described by Sakiyama *et al*. (2000) and include the data structures for request and response and all task-specific codes required to submit the request, to retrieve the response and to test the agent. Generic data structures are used that consist of records for the request and record-sets for the response comparable with Java Beans (Boyle, 1998). A Java BeanInfo-like implementor has also been realized that provides information about methods and properties of agent data structures. Appropriate transforms can then be applied to map between models. Therefore, interfaces connecting any two Sight agents can be generated regardless of their original data structures. Sight agents report their response in a table format in which each retrieved record represents a line in the table.

A tree-like work flow is generated by the user, who simply connects the response data fields of one agent to the request data fields of another using the application generator (Fig. 1). During execution, a separate request for the subsequent agent is made for each record in the original response of the preceding (master) agent. Request fields may be initialized directly by the user if required. Conditional connection of agents can be achieved by integrated Weka text-based filtering and classification algorithms (Witten and Frank, 1999), which filter the input for subsequent agents according to user-defined criteria.

In order to minimize Internet connections for trivial tasks to be performed by primitive Sight Web agents, the underlying algorithms for several applications such as pattern searches, protein translator or simple sequence manipulations have been

Bioinformatics 20(11) © Oxford University Press 2004; all rights reserved.

Downloaded from https://academic.oup.com/bioinformatics/article-abstract/20/11/1718/300153
by Mount Sinai Icahn School of Medicine user
on 23 July 2018

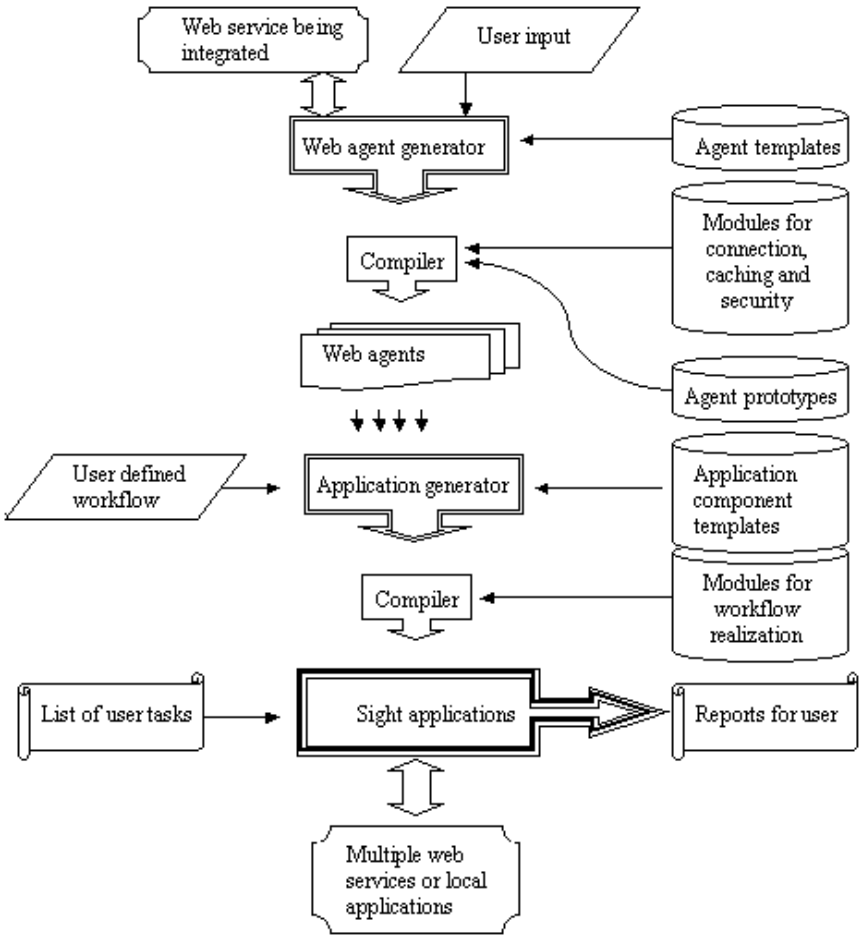

Fig. 1. Sight components. The user-interactive agent generator produces agents that can be connected for sequential tasks using the application generator. Compiler modules and agent templates are available at our Website.

included in the system for local use. Additionally, non-Java applications can be integrated by a wrapper generator that produces a skeleton agent with compatible request and response data structures. The output from both types of local applications can then be analyzed just like responses from typical Sight agents. Vice versa, Sight agents can be imported into user-written programs due to the easy accessibility of the Java code and data structures of request and response.

The user can observe all opened Internet connections by a pictogram-based real-time viewer. A security system suspends request submission if the same agent has not yet received a response to the previous request. Several retries are performed before an error result is generated. If the final report of the master contains incomplete sections, i.e. caused by subsequent agents who have generated error reports due to server breakdowns, the whole program can be restarted. This results in resubmission only of the requests without cached values, i.e. only of the previously failed requests. The expiration time of the cache is set during agent generation but may be modified during agent assembly. In the cache, the complete request at its specific position in an agent work flow is saved; alternatively, for other integrated applications, the cache is computed using hash codes from all request fields.

To test the usability of Sight for non-programmers, we instructed 16 molecular biologists to use it during the Channelomics Summer School Ulm in Fall 2002. Fourteen participants were able to develop their individual agents using Sight after a 3 h course. To test system stability, we generated a Sight-based genome walker, a typical application with long runtimes and potentially facing numerous server and network errors. To test the generated Web agents in specialized Java applications, we created a splicing signals analyzer. To test co-operability with non-Java programs, we integrated a locally installed BLAST (tblastx) version into a Sight-based gene finder. The results and figures of these and additional applications are at our Website. In brief, they are the following.

Genome walker: Sight generated agents performed the following steps. A sequence retriever agent loaded the sequences from the Internet. Then, the GenScan module predicted genes.

Downloaded from https://academic.oup.com/bioinformatics/article-abstract/20/11/1718/300153
by Mount Sinai Icahn School of Medicine user
on 23 July 2018

After getting the protein sequence, the following set of agents started their work in parallel: BLAST similarity search (at NCBI), prediction of the transmembrane helixes (TmPred), prediction of the rapid degradation signals (PSORT) and integrated classification (InterPro or PROSITE). The predicted RNA sequence was submitted to the BLAST similarity search using the complete expressed sequence tag (EST) database. The list of hits returned by this search contained hyperlinks to the corresponding Internet pages. These links were visited to obtain RNA expression patterns. The system detected which NCBI contig was assembled using a clone with a given accession number and included the hyperlink to this contig in the reports. To test this system, we scanned human chromosome 8, thereby classifying 1305 genes.

Splicing signals analyzer: Sight was used to look at the average distribution of nucleotide frequencies at the intron/exon boundaries and around the intronic splicing branch point of genomic DNA. Our system consisted of a sequence retriever, a GenScan module and specialized analyzers. We plotted the occurrence, relative frequencies and levels of confidence. Apart from the known pyrimidine-rich region downstream of the branch point, we found another pyrimidine-rich region upstream of the branch point and confirmed the known pyrimdine-rich region upstream of the splicing acceptor site. In the coding regions, the system detected a hitherto unknown decrease in T in the first position of each coding triplet—presumably to avoid stop codons, which all commence with T.

Gene finder: A database was generated by selecting all proteins containing known ion channel signatures from the NCBI non-redundant protein database. Corresponding DNA sequences were then submitted to a NCBI BLAST search against the NCBI non-redundant nucleic acid database. This search returned hits containing ESTs, annotated RNA sequences and genome regions. Next, the hit with the highest similarity level points to a sequence from non-human organisms was added to the report. We evaluated ion channel genes on chromosome 8, and the program reported a set of fragments that were not mentioned in the corresponding NCBI contig. It was located in clones AP000075.1 and AP000074.1 of contig NT_008251.5 and was most similar to the mouse potassium large-conductance pH-sensitive channel (gi:6680542), suggesting an as yet unannotated ion channel to exist on human chromosome 8.

ACKNOWLEDGEMENTS

This work was supported by the Interdisciplinary Clinical Research Center (iZKF) of Ulm University and funded by the Federal Ministry of Research (BMBF) and the GRK 460 Graduate College of the German Research Foundation (DFG).

REFERENCES

Bailey,L.C., Fischer,S., Schug,J., Crabtree,J., Gibson,M. and Overton,G.C. (1998) GAIA: framework annotation of genomic sequence. *Genome Res.*, **8**, 234–250.

Boyle,J. (1998) A visual environment for the manipulation and integration of JAVA beans. *Bioinformatics*, **14**, 739–748.

Brundege,J.M. and Dubay,C. (2003) BioQuery: an object framework for building queries to biomedical databases. *Bioinformatics*, **19**, 901–902.

Ferlanti,E.S., Ryan,J.F., Makalowska,I. and Baxevanis,A.D. (1999) WebBLAST 2.0: an integrated solution for organizing and analyzing sequence data. *Bioinformatics*, **15**, 422–423.

Frishman,D., Albermann,K., Hani,J., Heumann,K., Metanomski,A., Zollner,A. and Mewes,H.W. (2001) Functional and structural genomics using PEDANT. *Bioinformatics*, **17**, 44–57.

Graham,J., Decker,K.S. and Mersic,M. (2003) Decaf—a flexible multi-agent system architecture. *J. Aut. Agents Multi-Agent Syst.*, **7**, 7–27.

Harris,N.L. (2000) Annotating sequence data using Genotator. *Mol. Biotechnol.*, **16**, 221–232.

Knoblock,C.A, Minton,S., Ambite,J.L., Ashish,N., Muslea,I., Philpot,A. and Tejada,S. (2000) The Ariadne approach to web-based information integration. *Int. J. Coop. Inf. Syst.*, **10**, 145–169.

Kolatkar,P.R., Sakharkar,M.K., Tse,C.R., Kiong,B.K., Wong,L., Tan,T.W. and Subbiah,S. (1998) Development of software tools at BioInformatics Centre (BIC) at the National University of Singapore (NUS). *Pac. Symp. Biocomput.*, 735–746.

Mangalam,H. (2002) The Bio* toolkits—a brief overview. *Brief. Bioinform.*, **3**, 296–302.

Moller,S., Lesser,U., Fleischmann,W. and Apweiler,R. (1999) EDITtoTrEMBL: a distributed approach to high-quality automated protein sequence annotation. *Bioinformatics*, **15**, 219–227.

Muslea,I., Minton,S. and Knoblock,C.A. (2001) Hierarchical wrapper induction for semistructured information sources. *J. Aut. Agents Multi-Agent Syst.*, **4**, 93–114.

Sakiyama,T., Takami,H., Ogasawara,N., Kuhara,S., Kozuki,T., Doga,K., Ohyama,A. and Horikoshi,K. (2000) An automated system for genome analysis to support microbial whole-genome shotgun sequencing. *Biosci. Biotechnol. Biochem.*, **64**, 670–673.

Stajich,J.E., Block,D., Boulez,K., Brenner,S.E., Chervitz,S.A., Dagdigian,C., Fuellen,G., Gilbert,J.G., Korf,I., Lapp,H. *et al.* (2002) The Bioperl toolkit: Perl modules for the life sciences. *Genome Res.*, **12**, 1611–1618.

Stein,L.D., Cartinhour,S., Thierry-Mieg,D. and Thierry-Mieg,J. (1998) JADE: an approach for interconnecting bioinformatics databases. *Gene*, **209**, 39–43.

Stevens,R., Baker,P., Bechhofer,S., Ng,G., Jacoby,A., Paton,N.W., Goble,C.A. and Brass,A. (2000) TAMBIS: transparent access to multiple bioinformatics information sources. *Bioinformatics*, **16**, 184–185.

Witten,I.H. and Frank,E. (1999) *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann Publishers, San Francisco, 416 p.

Downloaded from https://academic.oup.com/bioinformatics/article-abstract/20/11/1718/300153
by Mount Sinai Icahn School of Medicine user
on 23 July 2018