OGER: OntoGene's Entity Recogniser in the BeCalm TIPS Task

Lenz Furrer and Fabio Rinaldi

University of Zurich, Institute of Computational Linguistics
Andreasstr. 15, CH-8050 Zürich, Switzerland
{furrer,rinaldi}@cl.uzh.ch

Abstract. We present OGER, an annotation service built on top of OntoGene's biomedical entity recognition system, which participates in the TIPS task (technical interoperability and performance of annotation servers) of the BeCalm (biomedical annotation metaserver) challenge. The annotation server is a web application tailored to the needs of the task, using an existing biomedical entity recognition suite. The core annotation module uses a knowledge-based strategy for term matching and entity linking. The server's architecture allows parallel processing of annotation requests for an arbitrary number of documents from mixed sources. In the discussion, we show that network latency is responsible for significant overhead in the measurement of processing time. We compare the preliminary key performance indicators with an analysis drawn from the server's log messages. We conclude that our annotation server is ready for the upcoming phases of the TIPS task.

Key words: knowledge-based named entity recognition, biomedical entity linking, parallel processing

1 Introduction

The *technical interoperability and performance of annotation servers* (TIPS) task [4] is part of the *biomedical annotation metaserver* (BeCalm) challenge [3]. Participants are asked to build a web service for annotating biomedical entities in given documents on the fly. The goal of the task is to build a fast and reliable annotation server, which can act as a contractor in an inter-universitary biomedical annotation cloud.

2 System description and methods

In the following, we describe OGER, an annotation server built on top of the OntoGene entity recognition suite.

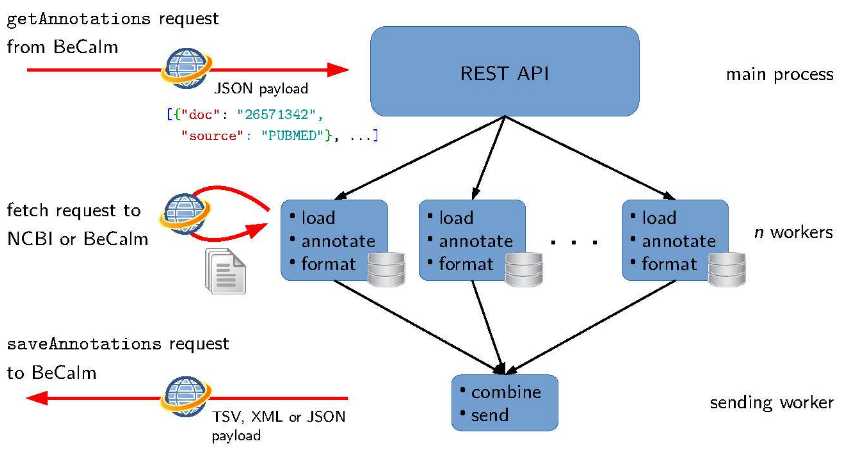

Fig. 1. System architecture of the annotation server.

2.1 Environment

The OGER annotation server is hosted on our institute's server infrastructure. It is run on a virtual machine (VM) dedicated to services. This VM has a high priority regarding the distribution of processing time among all VMs running on the same machine. It features 128 G of RAM (allocated exclusively to the service VM) and 16 CPUs (shared with other VMs). The operating system is the Debian-based Proxmox Virtual Environment. Incoming HTTPS requests are redirected to the annotation server by an Nginx reverse proxy server.

2.2 Annotation server architecture

The OGER annotation server is written in Python. Besides the standard library, we used two third-party libraries. The micro web-framework *Bottle*[1] is used to implement the REST API. XML parsing and serialising is performed through *lxml*.[2]

An overview of the system architecture is given in Figure 1. In order to allow a high processing capacity without blocking the API, the annotation server supports parallel processing. The main process runs the REST API, listening to incoming requests. Quick operations are handled directly, such as responding to the `getServerState` request or sending an acknowledgment response. A number[3] of annotation subprocesses (workers) are responsible for handling

[1] `http://bottlepy.org/`
[2] `http://lxml.de/`
[3] The number of annotation workers is fixed when the server is (re-)started. Throughout phase 1 of TIPS, we used 5 workers.

the `getAnnotations` requests. Each worker runs a separate instance of the term annotation pipeline (see Section 2.3). A separate subprocess (the sending worker) receives the extracted annotations from all annotation workers. It concatenates them (if necessary) and issues a `saveAnnotations` request to BeCalm's Metaserver. Finally, another subprocess (not shown in Fig. 1) collects processing errors and other incidents. These problem reports are sent by email to the person in charge of maintaining the server.

The annotation workers operate on a batch basis. Every batch consists of a list of document IDs and a source specifier, referring to one of PubMed, PubMed Central, BeCalm's abstract server, or BeCalm's patent server. The requested documents are obtained from their respective remote source, using either NCBI's *efetch* or BeCalm's REST API. Both interfaces allow an unlimited number of documents to be requested at once, which means that the entire batch can be retrieved through a single request. Upon retrieval, the documents are converted to a unified internal representation and passed to the entity recogniser. The extracted annotations of the whole batch are accumulated and serialised into the required output format (BeCalm JSON, BeCalm TSV, or BioC XML).[4] The formatted annotations are then pushed to the sending worker.

The annotation workers can only handle documents from a single source per batch. Therefore, if an annotation request asks for documents from different sources,[5] the main process groups the documents by source and creates multiple batches. For example, for a 10-document request, it would initiate a batch of 5 documents to be fetched from PubMed, and another batch of 5 abstracts to be requested from BeCalm's patent server. The batches are then processed in parallel by separate workers. This parallelisation results in a speed benefit even for small batches, since the network-related waiting times do not add up (as they would in a fully sequential approach).

The sending worker, finally, ensures that there is exactly one `saveAnnotations` request for each `getAnnotations` request. If an incoming request was split up into multiple batches, it waits for the formatted annotations from each batch to be completed and merges them into a single structure before sending them to the Metaserver. The concatenation of multiple batches needs some care, in that certain structural elements must not be repeated: In TSV format, the headers may only occur once. In JSON, the top-level array must span the entire collection using a single pair of brackets. Similarly, in XML, there can only be one root node, and BioC's collection-level metadata may not be repeated.

[4] Currently, the output format is specified through the server configuration and cannot be changed without restarting the server. However, the API could easily be changed to accept the output format as a parameter.

[5] This was never the case in phase 1 of TIPS, as all requests asked for a single document only. However, we successfully tested this functionality with private requests through BeCalm's web interface.

2.3 OntoGene term annotation pipeline

The OntoGene term annotation pipeline is a knowledge-based concept recognition system for biomedical entities. Designed for information extraction systems targeting scientific literature, it has been successfully applied to a range of entity types (genes/proteins, chemicals, diseases, among others [7, 8, 5, 6]). It has been recently reimplemented in Python as an integral processing suite [2], replacing the former amalgamation of modules written in various programming languages, communicating through a multitude of intermediate files written on disk. While the new pipeline provides a command-line interface with a lot of flexibility, we used it as a Python library for the annotation server.

As a knowledge-based system, the core recognition procedure relies on a list of target terms, which are connected to entity identifiers. The coverage of matching term variants is raised through a series of preprocessing steps with a normalising effect, such as an aggressive, lossy tokenisation strategy which collapses spelling alternations like e. g. "SRC1"/"SRC 1"/"SRC-1" into the same representation. A more detailed description of the annotation process can be found in [1].

For the present work, we used the following terminology resources (with their respective entity types):

- Cellosaurus (cell lines)[6]
- Comparative Toxicogenomics Database (CTD) (chemicals, diseases)[7]
- Gene Ontology (cellular components only, labelled "subcellular structure" in TIPS)[8]
- NCBI Taxonomy (organisms)[9]

These resources were aggregated and converted to a unified format using the Bio Term Hub.[10] Due to the highly flexible design of the system, the range of supported entity types can be extended very easily. By simply including additional terminology resources, more target entities can be covered.

3 Discussion

At the time of writing, only the key performance indicators are available. Also, global results have not been released, meaning that each participating group only sees results for their own system(s).

3.1 Performance

According to the statistics in the participant area of BeCalm's web interface, the key performance indicators for our system are as follows:

[6] `http://web.expasy.org/cellosaurus/`
[7] `http://ctdbase.org/`
[8] `http://geneontology.org/`
[9] `https://www.ncbi.nlm.nih.gov/taxonomy`
[10] `http://pub.cl.uzh.ch/purl/biodb/`

MAD 14.7923
MPDV 0.0119
MTDV 0.00086 s
MTSA 0.07227 s
ART 1.06943 s

The first two indicators (MAD and MPDV) hint at the sensitivity of the annotation server, as they represent the number of annotations per document and per Byte of a document, respectively. Without evaluating the correctness of the annotations, however, it is not clear what conclusions can be drawn from these figures. MAD might help put the other performance indicators in context (MTSA, in particular).

The other three indicators (MTDV, MTSA, ART) are very similar, in that they represent the average time needed to process one Byte of a document, one annotation, and one document, respectively. In our analysis, we will focus on ART, the time needed to process a document.

Based on previous experience with our annotation pipeline, an average processing time of more than a second per document seems exceptionally high. Especially when processing abstracts, we expect the annotation process to be faster by two or three orders of magnitude. It is thus important to understand how the processing time is measured.

In phase 1 of TIPS, all requests were concerned with a single document only. This means that the processing overhead (handling network connections, communication between subprocesses) per document is maximal. Indeed, the proportion of overhead in the total processing time is substantial: Using the BeCalm web interface, we triggered private requests with either one or ten documents to be annotated. In this (non-representative) test, the observed time difference between the two request sizes was negligible; there were even counter-intuitive examples, where a 10-document request was processed in less time than a single-document request.

Figure 2 shows all major processing steps that contribute to the total time of completing one request (which equals the response time for one document in phase 1 of TIPS). While we do not know the exact details of how the task organisers define the processing time, the depicted interpretation of start and end point is our best guess based on the documentation and on email conversations with the BeCalm team. The figure shows clearly that each request involves three separate network connections (hatched boxes). Compared to annotating an abstract as short as several hundred characters, we estimate the time contribution of the network latency to be inordinate. It is therefore difficult to draw meaningful conclusions from ART.

As another concern, the choice of measuring mean time has a strong bias for statistical outliers on the positive side. For example, if a single request (for whatever reason) takes 100 times longer to complete than the typical case, this has a much larger impact on the mean time than the inverse would have an unbelievably fast request that is 100 times faster than the typical case. Measuring

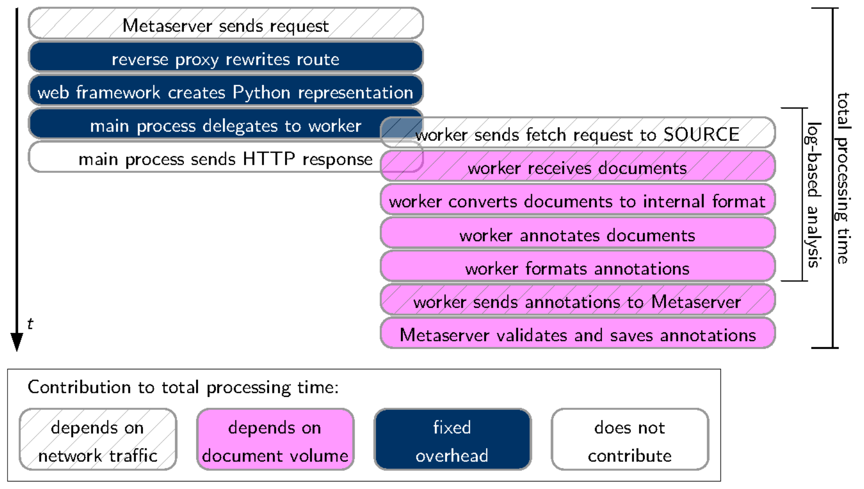
Metaserver sends request
reverse proxy rewrites route
web framework creates Python representation
main process delegates to worker
main process sends HTTP response
worker sends fetch request to SOURCE
worker receives documents
worker converts documents to internal format
worker annotates documents
worker formats annotations
worker sends annotations to Metaserver
Metaserver validates and saves annotations
t
log-based analysis
total processing time
Contribution to total processing time:
depends on network traffic
depends on document volume
fixed overhead
does not contribute

Fig. 2. Individual processing steps and their contribution to the total processing time.

median time, however, is much more robust in this scenario, as it gives more of an impression of the *typical* processing time.

Moreover, statistical outliers can have a negative effect on the expressiveness of the ART measure. If a few requests take many times longer than usual to complete, this is most likely due to a technical incident, such as a network problem or a server component being temporarily unavailable, causing a delay before the request can be processed or responded. Thus, we argue that outliers reflect aspects of the server's *reliability*, rather than its *performance* under normal conditions. Therefore we think that they should be captured by the reliability metrics rather than ART/MTSA/MTDV.

For these reasons, we carried out an alternative analysis of processing time based on the logs of our annotation server. The analysis covers the period starting on February 20, 2017, 11:51 CET until the end of phase 1 on March 31. We excluded all private requests from the analysis as well as all requests received on March 6, since on that day electrical power cuts caused BeCalm's infrastructure to malfunction. To our understanding, these data are also excluded from the official evaluation. Based on the total number of requests given in BeCalm's web interface (314 539 requests), there is a substantial overlap between the data sets.

Table 1 shows the results of our log-based analysis. Instead of measuring all steps involved in an annotation request, we restricted the analysis to the steps at the core of the annotation process. The measured period starts with the first log entry for a request, which is written immediately after the web-framework has preprocessed an incoming request, and ends with the last log message which is written just before sending back the formatted annotations (see the span labelled "log-based analysis" in Figure 2). Thus, the analysed periods span only

Table 1. Processing time analysis based on the server logs. All time measurements are in seconds.

	requests	min.	max.	average	median	std. dev.
PubMed	100 332	0.424	78.154	0.540	0.526	0.440
Abstract server	130 206	0.079	167.037	0.170	0.099	1.854
Patent server	36 945	0.079	13.671	0.103	0.096	0.144
all	267 483	0.079	167.037	0.300	0.111	1.336

one of the three network connections involved in a complete cycle. The start and end point of each request were obtained by parsing the log messages, which are printed with millisecond precision.

The most evident finding is that disregarding the initial and final network connections substantially reduces the measured time span: The average processing time of all requests (last row) is 300 ms, rather than 1069 ms (ART). The median processing time is again much lower (111 ms). It is also interesting to see the differences by origin: Fetching and processing a PubMed abstracts takes considerably longer (540 ms on average) than to BeCalm's abstract (170) and patent server (103). A possible explanation for this discrepancy is the fact that we obtain PubMed abstracts in XML format, while BeCalm provides the abstracts in a flat JSON structure, which is much more lightweight. Another observation is that PubMed abstracts tend to be larger in terms of file size (if only for the additional markup), which might have an impact on transmission time. It might also be that the NCBI servers are simply busier than BeCalm's in terms of network traffic.

Another conclusion that can be drawn from these statistics is that parallel processing did not pay off in phase 1 of TIPS, which is no surprise. In busy times, the Metaserver issued a request every 2 to 10 seconds. Thus, most of the time, when a new request arrived, the previous one had long been completed, meaning that the "parallel" annotation workers almost never worked in parallel effectively.

3.2 Conclusion

In the current evaluation, where each request asked for a single document only, it is hard to measure the speed of the annotation process. The task's protocol with three separate network connections for each annotation request entails significant overhead. In future phases of TIPS, where multiple documents per request, larger documents (full-text) and simultaneous requests will be required, our annotation server will be able to better show its strengths. The OGER annotation server is ready now for phases 2 through 4!

References

1. Basaldella, M., Furrer, L., Colic, N., Ellendorff, T.R., Tasso, C., Rinaldi, F.: Using a hybrid approach for entity recognition in the biomedical domain. In: Neves, M.,

Rinaldi, F., Nenadic, G., Rebholz-Schuhmann, D. (eds.) Proceedings of the 7th International Symposium on Semantic Mining in Biomedicine. pp. 11–19. Potsdam, Germany (2016), http://dx.doi.org/10.5167/uzh-125712

2. Colic, N.: Dependency Parsing for Relation Extraction in Biomedical Literature. Master's thesis, University of Zurich, Switzerland (2016), http://www.cl.uzh.ch/dam/jcr:609b8c4a-5d9f-4e4b-99a9-69355027509d/Master_Thesis_Nicola_Colic.pdf
3. Krallinger, M., Pérez, M.P., Rabal, O., Pérez Rodríguez, G., Vazquez, M., Fdez-Riverola, F., Oyarzabal, J., Lourenco, A., Valencia, A.: The BioCreative V.5/BeCalm evaluation workshop: tasks, organization, sessions and topics. In: Proceedings of the BioCreative V.5 Challenge Evaluation Workshop. pp. 1–2 (2017)
4. Pérez-Pérez, M., Pérez-Rodríguez, G., Blanco-Míguez, A., Fdez-Riverola, F., Valencia, A., Krallinger, M., Lourenco, A.: Benchmarking biomedical text mining web servers at BioCreative V.5: the technical interoperability and performance of annotation servers – TIPS track. In: Proceedings of the BioCreative V.5 Challenge Evaluation Workshop. pp. 12–21 (2017)
5. Rinaldi, F., Clematide, S., Hafner, S.: Ranking of CTD articles and interactions using the OntoGene pipeline. In: Proceedings of the 2012 BioCreative workshop. Washington D.C. (April 2012), https://doi.org/10.5167/uzh-62066
6. Rinaldi, F., Clematide, S., Marques, H., Ellendorff, T., Rodriguez-Esteban, R., Romacker, M.: OntoGene web services for biomedical text mining. BMC Bioinformatics 15(14) (2014), http://dx.doi.org/10.1186/1471-2105-15-S14-S6
7. Rinaldi, F., Kappeler, T., Kaljurand, K., Schneider, G., Klenner, M., Clematide, S., Hess, M., von Allmen, J.M., Parisot, P., Romacker, M., Vachon, T.: OntoGene in BioCreative II. Genome Biology 9(2), S13 (2008), http://dx.doi.org/10.1186/gb-2008-9-s2-s13
8. Rinaldi, F., Schneider, G., Kaljurand, K., Clematide, S., Vachon, T., Romacker, M.: OntoGene in BioCreative II.5. IEEE/ACM Transactions on Computational Biology and Bioinformatics 7(3), 472–480 (Juli 2010), http://dx.doi.org/10.5167/uzh-46282