Sharing annotations better: RESTful Open Annotation

Sampo Pyysalo[1] **Jorge Campos**[2] **Juan Miguel Cejuela**[2] **Filip Ginter**[1]
Kai Hakala[1] **Chen Li**[3] **Pontus Stenetorp**[4] **Lars Juhl Jensen**[5]
[1] University of Turku, Finland, [2] tagtog, Germany,
[3] Massachusetts Institute of Technology, United States,
[4] University of Tokyo, Japan, [5] University of Copenhagen, Denmark
sampo@pyysalo.net jorge@tagtog.net juanmi@tagtog.net
filip.ginter@utu.fi kai.hakala@utu.fi cli@csail.mit.edu
p.stenetorp@cs.ucl.ac.uk lars.juhl.jensen@cpr.ku.dk

Abstract

Annotations are increasingly created and shared online and connected with web resources such as databases of real-world entities. Recent collaborative efforts to provide interoperability between online annotation tools and resources have introduced the Open Annotation (OA) model, a general framework for representing annotations based on web standards. Building on the OA model, we propose to share annotations over a minimal web interface that conforms to the Representational State Transfer architectural style and uses the JSON for Linking Data representation (JSON-LD). We introduce tools supporting this approach and apply it to several existing annotation clients and servers, demonstrating direct interoperability between tools and resources that were previously unable to exchange information. The specification and tools are available from `http://restoa.github.io/`.

1 Introduction

Annotation is an important task in many fields ranging from historical and literary study to experimental sciences including biology. The value of annotations is closely associated with the ability to share them. The web has become instrumental to information sharing, and there has thus been much interest in web-based tools and repositories for the creation, collaborative editing and sharing of annotations. Unfortunately, design and implementation differences have resulted in poor interoperability, raising barriers to communication and introducing costs from the need to convert between data models, formats, and protocols to bridge different systems.

To fully interoperate, annotation tools and resources must agree at least on a way to name and refer to things, an abstract data model, a format capturing that model, and a communication protocol. Here, we suggest a web application programming interface (API) that resolves these questions by building upon web standards and best practices, namely Linked Data principles (Bizer et al., 2009), the Open Annotation data model (Bradshaw et al., 2013) and its serialization as JSON-LD (Sporny et al., 2014), and a minimal HTTP-based protocol adhering to the Representational State Transfer (REST) architectural style (Fielding and Taylor, 2002). By implementing support for the API in a variety of independently developed annotation tools and resources, we demonstrate that this approach enables interoperability and novel ways of combining previously isolated methods.

2 Design

We aim to define a minimal web API for sharing annotations that conforms closely to relevant standards and best practices. This should reduce implementation effort and ensure generality and compatibility with related efforts (Section 5). We next briefly discuss the components of our design.

Linked Data. We use representations based on the Resource Description Framework (RDF) standards for modeling data on the web, following the best practice of using HTTP uniform resource identifiers (URIs), which provide useful information when dereferenced (Bizer et al., 2009).

Open Annotation. We describe text annotations according to the OA draft W3C standard[1], which

[1]`http://www.openannotation.org/`

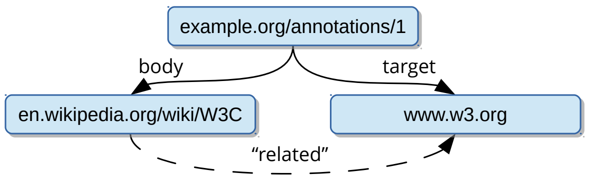


Figure 1: OA model example. The annotation expresses that the W3C Wikipedia article is related to the W3C homepage. The three resources are all in different domains, and the *"related"* relation is not represented explicitly.

is an RDF-based graph representation compatible with linguistic annotation formalisms such as LAF/GrAF (Ide and Suderman, 2007; Verspoor and Livingston, 2012). At its most basic level, the OA model differentiates between three key components: *annotation*, *body*, and *target*, where the annotation expresses that the body is related to the target of the annotation (Figure 1). The body can carry arbitrarily complex embedded data.

JSON-LD was recently accepted as a standard RDF serialization format (Sporny et al., 2014) and is the recommended serialization of OA. Every JSON-LD document is both a JSON document and a representation of RDF data. Figure 2 shows an example of a simple annotation using the OA JSON-LD representation.[2]

```
{
   "@id":     "/annotations/1",
   "@type":   "oa:Annotation",
   "target":  "/documents/1#char=0,10",
   "body":    "Person"
}
```

Figure 2: Example annotation in JSON-LD format.

RESTful architecture We define a resource-oriented API that uses HTTP verbs to manipulate resources (Table 1). The API provides hypermedia controls in data using JSON-LD and established link relations, in conformance with best practices for RESTful APIs (Fielding and Taylor, 2002).

The API defines just two types of resources: an annotation and a collection of annotations. The former is defined according to the core OA specification. While there are no formal standards for the representation of collections in RESTful APIs, the basic collection pattern is well established. We specify a simple implementation, drawing on relevant draft standards such as Collection+JSON[3] and Hydra[4].

[2]The OA JSON-LD `@context` is understood to be active. Relative URLs are interpreted with respect to the HTTP request base.

Verb	Resource	Action
`GET`	Annotation	Read annotation
`GET`	Collection	Read all annotations
`PUT`	Annotation	Update annotation
`DELETE`	Annotation	Delete annotation
`POST`	Collection	Create annotation

Table 1: HTTP verbs, resources, and actions. Read-only services support only the two `GET` requests.

3 Reference Implementation

To support the development, testing and integration of RESTful OA API implementations, we have created a reference server and client as well as tools for format conversion and validation.

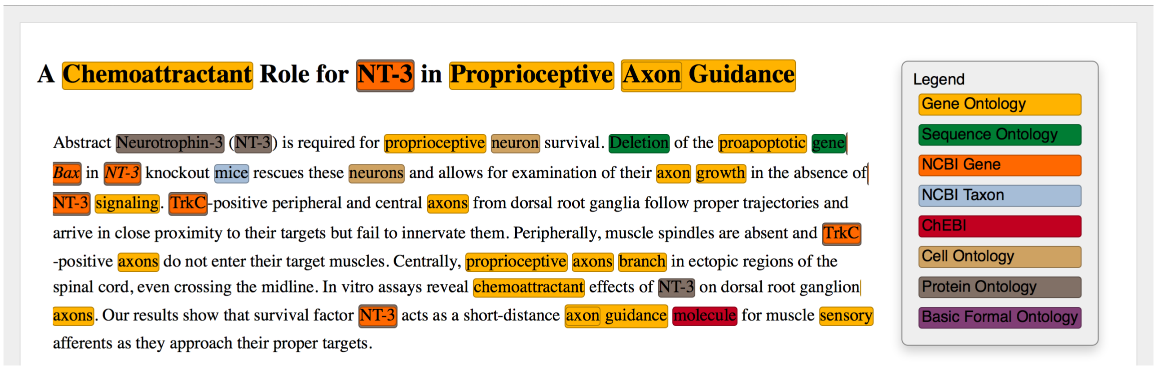
3.1 OA Store

The OA Store is a reference implementation of persistent, server-side annotation storage that allows clients to create, read, update and delete annotations using the API. The store uses MongoDB, which is well suited to the task as it is a document-oriented, schema-free database that natively supports JSON for communication. The API is implemented using the Python Eve framework, which is specifically oriented towards RESTful web APIs using JSON and is thus easily adapted to support OA JSON-LD.

3.2 OA Explorer

The OA Explorer is a reference client that provides an HTML interface for navigating and visualizing the contents of any compatible store (Figure 3). The service first prompts the user for a store URL and then provides the user with a dynamically generated view of the contents of the store, which it discovers using the API. OA Explorer is implemented in Python using the Flask microframework for web development.

[3]`http://amundsen.com/media-types/collection/`
[4]`http://www.hydra-cg.com/spec/latest/core/`



A Chemoattractant Role for NT-3 in Proprioceptive Axon Guidance

Abstract Neurotrophin-3 (NT-3) is required for proprioceptive neuron survival. Deletion of the proapoptotic gene *Bax* in *NT-3* knockout mice rescues these neurons and allows for examination of their axon growth in the absence of NT-3 signaling. TrkC-positive peripheral and central axons from dorsal root ganglia follow proper trajectories and arrive in close proximity to their targets but fail to innervate them. Peripherally, muscle spindles are absent and TrkC -positive axons do not enter their target muscles. Centrally, proprioceptive axons branch in ectopic regions of the spinal cord, even crossing the midline. In vitro assays reveal chemoattractant effects of NT-3 on dorsal root ganglion axons. Our results show that survival factor NT-3 acts as a short-distance axon guidance molecule for muscle sensory afferents as they approach their proper targets.

Figure 3: OA Explorer shown visualizing annotations from the CRAFT corpus (Bada et al., 2012) converted to OA and served from the OA Store.

3.3 Format conversion

The OA Adapter is middleware that we created for sharing Open Annotation data. The software implements both the client and server sides of the API and a variety of conversions to and from different serializations of the OA model and related formats using the OA JSON-LD serialization as the pivot format. This allows the OA Adapter to operate transparently between a client and a server, providing on-the-fly conversions of client requests from representations favored by the client into ones favored by the server, and vice versa for server responses. Standard HTTP content negotiation is used to select the best supported representations. The adapter implements full support for all standard RDF serializations: JSON-LD, N-Triples and N-Quads, Notation3, RDF/XML, TriG, TriX, and Turtle. With the exception of named graphs for serializations that do not support them, conversion between these representations is guaranteed to preserve all information.

In addition to the general, reversible format translation services provided by the OA Adapter, we provide scripts for offline conversion of various annotation file formats into the OA JSON-LD format to allow existing datasets to be imported into OA stores. The following are currently supported: Penn Treebank format (including PTB II PAS) (Marcus et al., 1994), a number of variants of CoNLL formats, including CoNLL-U,[5] Knowtator XML (Ogren, 2006), and the standoff format used by the BRAT annotation tool (Stenetorp et al., 2012). We also provide supporting tools for importing files with OA JSON-LD data to a store and exporting to files over the RESTful OA API.

3.4 Validation

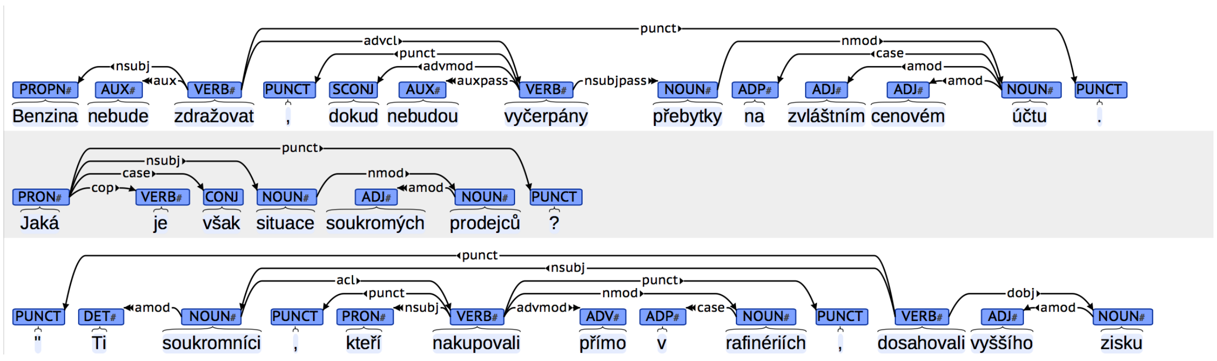
OA JSON-LD data can be validated on three levels: 1) whether the data is syntactically well-formed JSON, 2) whether it conforms to the JSON-LD specification, and 3) whether the abstract information content fulfills the OA data model. The first two can be accomplished using any one of the available libraries that implement the full JSON-LD syntax and API specifications.[6] To facilitate also validation of conformity to the OA data model, we define the core model of the OA standard using JSON Schema (Galiegue and Zyp, 2013). The JSON Schema community has provided tools in various programming languages for validating JSON against a JSON Schema. The schema we defined is capable of validating data for compliance against JSON-LD and OA Core at the same time. Complementing this support for data validation, we are also developing a standalone tool for testing web services for conformance to the RESTful OA API specification.

4 Adaptation of Existing Tools

In addition to creating reference implementations, we have adapted two previously introduced web-based annotation tools to support the API. We further demonstrate the scope and scalability of the framework on several publicly available mass-scale datasets from the biomedical domain, showing how annotations on millions of documents can be transparently linked across well-established database services and to non-textual resources such as gene and protein databases.

[5]http://universaldependencies.github.io/docs/

[6]http://json-ld.org

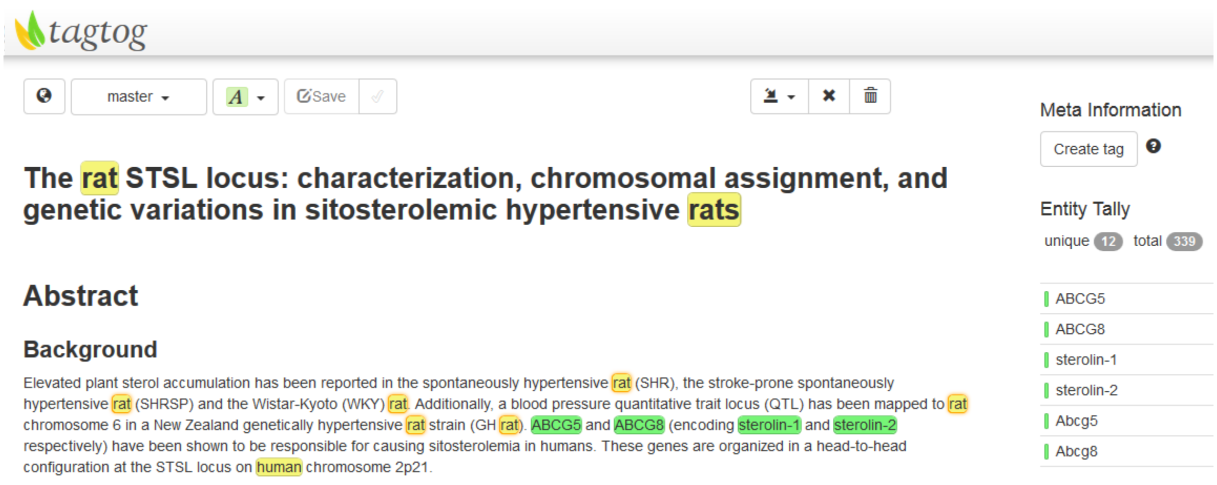

PROPN AUX VERB PUNCT SCONJ AUX VERB NOUN ADP ADJ ADJ NOUN PUNCT
nsubj aux advcl punct advmod auxpass nsubjpass punct nmod case amod amod
PRON VERB CONJ NOUN ADJ NOUN PUNCT
nsubj case cop punct nmod amod
PUNCT DET NOUN PUNCT PRON VERB ADV ADP NOUN PUNCT VERB ADJ NOUN
punct nsubj acl punct amod nsubj advmod nmod case punct dobj amod

Benzina nebude zdražovat , dokud nebudou vyčerpány přebytky na zvláštním cenovém účtu .

Jaká je však situace soukromých prodejců ?

" Ti soukromníci , kteří nakupovali přímo v rafinériích , dosahovali vyššího zisku

Figure 4: BRAT showing Czech dependency annotations from the Universal Dependencies corpus (`http://universaldependencies.github.io/docs/`).



The rat STSL locus: characterization, chromosomal assignment, and genetic variations in sitosterolemic hypertensive rats

Abstract

Background

Elevated plant sterol accumulation has been reported in the spontaneously hypertensive rat (SHR), the stroke-prone spontaneously hypertensive rat (SHRSP) and the Wistar-Kyoto (WKY) rat. Additionally, a blood pressure quantitative trait locus (QTL) has been mapped to rat chromosome 6 in a New Zealand genetically hypertensive rat strain (GH rat). ABCG5 and ABCG8 (encoding sterolin-1 and sterolin-2 respectively) have been shown to be responsible for causing sitosterolemia in humans. These genes are organized in a head-to-head configuration at the STSL locus on human chromosome 2p21.

ABCG5
ABCG8
sterolin-1
sterolin-2
Abcg5
Abcg8

Figure 5: tagtog showing entity annotations for a full-text document from PubMed Central.

4.1 BRAT

The brat rapid annotation tool (BRAT) is an open-source web-based annotation tool that supports a wide range of text annotation tasks (Stenetorp et al., 2012). It provides intuitive visualization of text-bound and relational annotations and allows for annotations to be created and edited using a drag-and-drop interface (Figure 4). The server is a web service implemented in Python, whereas the client is a browser-based application written in JavaScript. For annotation storage, the server uses a file-based back-end with a stand-off file format[7].

The original client and server implement a custom communication protocol, leading to tight coupling between the two. We rewrote the client and server communication components to use OA JSON-LD and the RESTful API as an alternative to the native format and protocol, thus enabling both components to communicate also with other clients and servers.

[7]`http://brat.nlplab.org/standoff.html`

4.2 tagtog

The tagtog web-based annotation system is designed to combine manual and automatic annotations to accurately and efficiently mark up full-text articles (Cejuela et al., 2014). The system was originally developed with a focus on annotating biological entities and concepts such as genes and Gene Ontology terms. The web interface is implemented in JavaScript using the Play framework with Scala. The system is centered on the concept of user projects, each of which holds a corpus of annotated documents.

To make tagtog interoperable with other RESTful OA clients and servers, we made two major implementation changes. First, the server can now serve annotations in OA JSON-LD format, thus allowing them to be viewed by other clients. Second, the tagtog interface can visualize and edit OA JSON annotations from other OA stores, without a backing tagtog project. Figure 5 shows a sample document annotated in tagtog.

4.3 Biomedical entity recognition resources

We implemented the API for four large-scale databases of biomedical entity mentions. The COMPARTMENTS database integrates evidence on protein subcellular localization (Binder et al., 2014), and TISSUES and DISEASES similarly integrate evidence on tissue expression and disease-associations of human genes, respectively (Santos et al., 2015; Pletscher-Frankild et al., 2015). All three resources include a text mining component based on the highly efficient NER engine used also for detection of species names and names of other taxa in the ORGANISMS database (Pafilis et al., 2014). Together, these databases contain over 123M mentions of genes/proteins, cellular components, tissues and cell lines, disease terms and taxonomic identifiers. This dataset is regularly precomputed for the entire Medline corpus, which currently consists of more than 24M abstracts and 3B tokens.

To make this large collection of automatic annotations available as OA JSON-LD, we defined the annotations of each abstract to be a separate (sub)collection of a document resource, accessible using URL patterns of the form `http://.../document/{docid}/annotations/`. The web services were implemented as part of the Python framework common to all four databases. They query a PostgreSQL back-end for text and annotations, which are formatted as OA JSON-LD using the standard Python `json` module.

4.4 EVEX

The EVEX database is a collection of events from the molecular biology domain obtained by processing the entire collection of PubMed articles and PubMed Central Open Access full-text articles (Van Landeghem et al., 2013), together constituting a corpus of nearly 6B tokens. In total, EVEX contains 40M individual events among 77M entity mentions. The events are of 24 different types (e.g. POSITIVE REGULATION, PHOSPHORYLATION) and the participants are primarily genes and proteins. Where possible, the entity mentions are grounded to their corresponding Entrez Gene database identifiers.

The event structures consist of entity mentions, *trigger phrases* expressing events, and typed relations identifying the roles that the entities play in the events. All of this data is accessible through a newly implemented EVEX API compliant with the OA JSON-LD format. Every document is defined as a separate annotation collection following the approach described in Section 4.3. The EVEX web service is written in Python using the Django web framework. Data are stored in a MySQL database and the OA JSON-LD interface uses the standard Python `json` module for formatting.

5 Related work

Our approach builds directly on the OA data model (Bradshaw et al., 2013), which harmonizes the earlier Open Annotation Collaboration (Haslhofer et al., 2011) and Annotation Ontology Initiative (Ciccarese et al., 2011) efforts and is currently developed further under the auspices of the W3C Web Annotation WG.[8] Approaches building on RESTful architectures and JSON-LD are also being pursued by the Linguistic Data Consortium (Wright, 2014) and the Language Application Grid (Ide et al., 2014), among others. A number of annotation stores following similar protocols have also been released recently, including Lorestore (Hunter and Gerber, 2012), PubAnnotation (Kim and Wang, 2012), the Annotator.js store[9], and NYU annotations[10].

6 Conclusions and future work

We have proposed to share annotations using a minimal RESTful interface for Open Annotation data in JSON-LD. We introduced reference implementations of a server, client, validation and conversion tools, and demonstrated the integration of several independently developed annotation tools and resources using the API. In future work, we will continue to develop the API specification further in collaboration with the relevant standardization efforts and interested parties using a fully open process. We will focus in particular on modular extensions to the specification for supporting search, tagging, and bulk modifications. We will also continue to develop and extend the tools, with emphasis on reversible conversions between OA JSON-LD and major related formats. Except for tagtog, a commercial tool, all of the tools and resources introduced in this study are available under open licenses from `http://restoa.github.io/`.

[8] `http://www.w3.org/annotation/`
[9] `http://annotateit.org/`
[10] `http://annotations.dlib.nyu.edu/home/`

Acknowledgments

This work was in part funded by the Novo Nordisk Foundation Center for Protein Research [NNF14CC0001], by the National Institutes of Health [U54 CA189205-01], by JSPS KAKENHI [13F03041], and by Quanta Computer Inc.

References

Michael Bada, Miriam Eckert, Donald Evans, Kristin Garcia, Krista Shipley, Dmitry Sitnikov, William A Baumgartner, K Bretonnel Cohen, Karin Verspoor, Judith A Blake, et al. 2012. Concept annotation in the craft corpus. *BMC bioinformatics*, 13(1):161.

Janos X Binder, Sune Pletscher-Frankild, Kalliopi Tsafou, Christian Stolte, Sean I O'Donoghue, Reinhard Schneider, and Lars Juhl Jensen. 2014. COMPARTMENTS: unification and visualization of protein subcellular localization evidence. *Database*, 2014:bau012.

Christian Bizer, Tom Heath, and Tim BernersLee. 2009. Linked Data the story so far. *International Journal on Semantic Web & Information Systems*.

Shannon Bradshaw, Dan Brickley, Leyla Jael Garca Castro, Timothy Clark, Timothy Cole, Phil Desenne, Anna Gerber, Antoine Isaac, Jacob Jett, Thomas Habing, et al. 2013. Open annotation data model (community draft).

Juan Miguel Cejuela, Peter McQuilton, Laura Ponting, Steven J Marygold, Raymund Stefancsik, Gillian H Millburn, Burkhard Rost, et al. 2014. tagtog: interactive and text-mining-assisted annotation of gene mentions in PLOS full-text articles. *Database*, 2014:bau033.

Paolo Ciccarese, Marco Ocana, Leyla Jael Garcia-Castro, Sudeshna Das, and Tim Clark. 2011. An open annotation ontology for science on web 3.0. *J. Biomedical Semantics*, 2(S-2):S4.

Roy T Fielding and Richard N Taylor. 2002. Principled design of the modern web architecture. *ACM Transactions on Internet Technology (TOIT)*, 2(2):115–150.

Francis Galiegue and Kris Zyp. 2013. JSON Schema: Core definitions and terminology. *Internet Engineering Task Force (IETF)*.

Bernhard Haslhofer, Rainer Simon, Robert Sanderson, and Herbert Van de Sompel. 2011. The open annotation collaboration (oac) model. In *Proc. MMWeb'11*, pages 5–9.

Jane Hunter and Anna Gerber. 2012. Towards annotopiaenabling the semantic interoperability of web-based annotations. *Future Internet*, 4(3):788–806.

Nancy Ide and Keith Suderman. 2007. Graf: A graph-based format for linguistic annotations. In *Proc. LAW'07*, pages 1–8.

Nancy Ide, James Pustejovsky, Christopher Cieri, Eric Nyberg, Denise DiPersio, Chunqi Shi, Keith Suderman, Marc Verhagen, Di Wang, and Jonathan Wright. 2014. The language application grid. *Proc. LREC'14*.

Jin-Dong Kim and Yue Wang. 2012. Pubannotation: a persistent and sharable corpus and annotation repository. In *Proc. BioNLP'12*, pages 202–205.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The penn treebank: annotating predicate argument structure. In *Proc. HLT*, pages 114–119.

Philip V Ogren. 2006. Knowtator: a protégé plug-in for annotated corpus construction. In *Proc. HLT-NAACL'06 demos*, pages 273–275.

Evangelos Pafilis, Sune Pletscher-Frankild, Lucia Fanini, Sarah Faulwetter, Christina Pavloudi, Aikaterini Vasileiadou, Christos Arvanitidis, and Lars Juhl Jensen. 2014. The SPECIES and ORGANISMS resources for fast and accurate identification of taxonomic names in text. *PLoS ONE*, 8:e65390.

Sune Pletscher-Frankild, Albert Palleja, Kalliopi Tsafou, Janos X Binder, and Lars Juhl Jensen. 2015. DISEASES: Text mining and data integration of disease-gene associations. *Methods*, 74:83–89.

Alberto Santos, Kalliopi Tsafou, Christian Stolte, Sune Pletscher-Frankild, Sean I O'Donoghue, and Lars Juhl Jensen. 2015. Comprehensive comparison of large-scale tissue expression datasets. *PeerJ*.

Manu Sporny, Dave Longley, Gregg Kellogg, Markus Lanthaler, and Niklas Lindström. 2014. JSON-LD 1.0: A JSON-based serialization for linked data.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. Brat: a web-based tool for nlp-assisted text annotation. In *Proc. ACL'12 demos*, pages 102–107.

Sofie Van Landeghem, Jari Björne, Chih-Hsuan Wei, Kai Hakala, Sampo Pyysalo, Sophia Ananiadou, Hung-Yu Kao, Zhiyong Lu, Tapio Salakoski, Yves Van de Peer, et al. 2013. Large-scale event extraction from literature with multi-level gene normalization. *PLoS ONE*, 8(4):e55814.

Karin Verspoor and Kevin Livingston. 2012. Towards adaptation of linguistic annotations to scholarly annotation formalisms on the semantic web. In *Proc. LAW'12*, pages 75–84.

Jonathan Wright. 2014. Restful annotation and efficient collaboration. In *Proc. LREC'14*.