OrganismTagger: detection, normalization and grounding of organism entities in biomedical documents

Nona Naderi1, Thomas Kappler2, Christopher J. O. Baker3 and René Witte1,∗

1Department of Computer Science and Software Engineering, Concordia University, Montréal, Québec, Canada, 2Swiss Institute of Bioinformatics, Geneva, Switzerland and 3Department of Computer Science and Applied Statistics, University of New Brunswick, Saint John, Canada

ABSTRACT

Finding organism mentions is the task of named entity (NE) detection. Simply tagging an entity as an organism is, however, not sufficient for more advanced text mining tasks. To account for variability within the same document, e.g. the use of both abbreviated and full forms, each mention of an organism must be additionally tagged with a canonical name through normalization. These unique names can then be used for downstream analysis tasks like co-reference resolution. To facilitate the disambiguation of other entities in a text, additional analysis, so-called grounding, is required in order to create a link between a textual reference and its entry in an external database. Combined with semantic technologies, such as ontologies and reasoners, further knowledge discovery approaches are made possible (Baker and Cheung, 2007).

Given the labile nature of scientific knowledge in the era of high-throughput biology, online resources are frequently updated and custom applications dependent on such resources must also be readily updated to avoid latency of processed content. Consequently, the ability to easily update the organism tagging system with respect to external taxonomic resources becomes an additional requirement. Further technical requirements include configurability

1 INTRODUCTION

Text mining solutions have become an integral part of biomedical research. An important class of entities to detect through natural language processing (NLP) techniques are organisms. Their detection facilitates taxonomy-aware text mining systems and provides users with the ability to find relevant subsets of papers based on species-specific queries. When textual mentions are further annotated with an external database identifier, they can provide additional benefits for disambiguation in the recognition of other named entities such as mutations, proteins or genes (Hakenberg et al., 2008; Hanisch et al., 2005; Wang and Matthews, 2008; Wang, 2007; Witte et al., 2007).

Primarily, organism mentions are based on established hierarchical nomenclature conventions defined in the 18th century (Linnaeus, 1767). However, the recognition of taxonomic groups in texts presents a number of ongoing challenges. Specifically, there is considerable ambiguity in the way taxonomic information is formulated in scientific documents. Abbreviations of species names are widespread and the use of common English names instead of Latin names further obscures the taxonomic identity of the organisms described in a text. The use of acronyms, which can be both species specific and species independent, also poses challenges for recognition tasks. Lastly, incorrect spellings have created yet more ambiguity.

Finding organism mentions is the task of named entity (NE) detection. Simply tagging an entity as an organism is, however, not sufficient for more advanced text mining tasks. To account for variability within the same document, e.g. the use of both abbreviated and full forms, each mention of an organism must be additionally tagged with a canonical name through normalization. These unique names can then be used for downstream analysis tasks like co-reference resolution. To facilitate the disambiguation of other entities in a text, additional analysis, so-called grounding, is required in order to create a link between a textual reference and its entry in an external database. Combined with semantic technologies, such as ontologies and reasoners, further knowledge discovery approaches are made possible (Baker and Cheung, 2007).

Given the labile nature of scientific knowledge in the era of high-throughput biology, online resources are frequently updated and custom applications dependent on such resources must also be readily updated to avoid latency of processed content. Consequently, the ability to easily update the organism tagging system with respect to external taxonomic resources becomes an additional requirement. Further technical requirements include configurability

© The Author 2011. Published by Oxford University Press. All rights reserved. For Permissions, please email: journals.permissions@oup.com
Web services are available for entity recognition and normalization, Section 3 describes existing taxonomy resources and Section 4 clients. A system architecture for running the system embedded, stand-alone, a machine learning-based classifier for strain detection; (v) flexible pipeline for organism detection, normalization and grounding; (iv) tools for automatically generating organism-specific resources and disambiguating strain information.

However, all these existing approaches have difficulties recognizing techniques for disambiguation, acronym resolution and filtering. A gazetteer (species dictionary) combined with post-processing performance tagging species names in biomedical texts using parsers. TaxonGrab (Koning et al., 2006) uses a lexicon of English words (combination of WordNet and SPECIALIST excluding the terms from NCBI Taxonomy, the Integrated Taxonomic Information System and the German Collection of Microorganisms and Cell Cultures) to compare against a text. Detected sets of two or three consecutive words that do not exist in the lexicon are further validated with regular expressions. The performance of their system is tested on Volume 1 of ‘The Birds of the Belgian Congo’ by James Paul Chapin with a precision of 96% and recall of 94%. FindIT (Leary et al., 2007), a Web service, tries to index the taxonomic names using pattern-matching expressions and a lexicon of English words, providing a confidence score for resultant names. Rule-based, word frequency and regular expression-based approaches manage to capture genus–species combinations with high levels of precision and recall. In some implementations, this is achieved without further grounding to database identifiers (Koning et al., 2006; Sautter et al., 2006). In some cases, grounding to database identifiers or nodes in a taxonomic tree is made; examples are TaxonFinder (http://code.google.com/p/taxon-finder) and uBioRSS (Leary et al., 2007). For such systems, common names still pose problems, whereas gazetteer-based approaches are able to handle common name issues. A number of gazetteer-based Web services are available for entity recognition and normalization, such as Whatizit (Rebholz-Schuhmann et al., 2008). Most recently, the Linnaeus system (Gerner et al., 2010) has illustrated good performance tagging species names in biomedical texts using a gazetteer (species dictionary) combined with post-processing techniques for disambiguation, acronym resolution and filtering. However, all these existing approaches have difficulties recognizing and disambiguating strain level information.

Our OrganismTagger addresses the above challenges with a number of novel contributions: (i) provision of semantic data models for organisms that can be re-used in other applications; (ii) tools for automatically generating organism-specific resources derived from the NCBI Taxonomy database; (iii) a text mining pipeline for organism detection, normalization and grounding; (iv) a machine learning-based classifier for strain detection; (v) flexible system architecture for running the system embedded, stand-alone, published as a Web service or integrated into a number of desktop clients.

We first present an overview of the OrganismTagger in Section 2. Section 3 describes existing taxonomy resources and Section 4 2722

Fig. 1. An overview of the main parts of the OrganismTagger system. Document processing is performed by an NLP pipeline running in GATE using resources automatically created from the NCBI Taxonomy database. their integration into our system. The text mining pipeline is then described in Section 5, followed by its evaluation in Section 6 and conclusions in Section 7.

2 INFRASTRUCTURE FOR ORGANISM TAGGING, NORMALIZATION AND GROUNDING

The life cycle of our OrganismTagger has two distinct parts: (i) the generation and initialization of NLP resources (like gazetteer lists and ontologies) from the NCBI database (described in Section 4); and (ii) the run-time processing of documents for semantic tagging, including normalization and grounding of detected entities (described in Section 5).

An overview of the OrganismTagger is shown in Figure 1. Our tagging and extraction methodology relies initially on external resources, namely pre-existing taxonomy databases, which are automatically translated for reuse in our platform, thereby providing users with the ability to update their installation when the database changes. Additionally, we created a custom-built organism ontology, which formally describes the linguistic structure of organism entities at different levels of the taxonomic hierarchy.

The run-time processing pipeline, implemented based on the General Architecture for Text Engineering (GATE) (Cunningham et al., 2011), consists of modules for: strain-specific text tokenization, a gazetteer for matching names or name fragments to the NCBI reference taxonomy, machine learning-based strain classification, grammar-based organism entity detection, normalization of abbreviations and other forms to their scientific names and a grounding step for assigning detected organisms an NCBI Taxonomy database identifier.

3 REUSE OF PRIMARY TAXONOMY RESOURCES

We use the Taxonomy database (Federhen, 2003) from NCBI (NCBI Taxonomy Homepage, http://www.ncbi.nlm.nih.gov/Taxonomy/) to initialize our gazetteering lists and ontologies. The Taxonomy database is ‘a curated set of names and classifications for all of the organisms that are represented in GenBank’ [see (Federhen, 2003) for a detailed description]. For the experiments described in this
We now describe the lexical and ontological resources used by our system in Table 1.

<table>
<thead>
<tr>
<th>name_txt</th>
<th>name_class</th>
</tr>
</thead>
<tbody>
<tr>
<td>'Bacterium coli commune' Escherich 1885</td>
<td>synonym</td>
</tr>
<tr>
<td>'Bacterium coli' (Migula 1895) Lehmann and Neumann 1896</td>
<td>synonym</td>
</tr>
<tr>
<td>Bacillus coli</td>
<td>synonym</td>
</tr>
<tr>
<td>Escherichia coli</td>
<td>misspelling</td>
</tr>
<tr>
<td>Escherichia coli (Migula 1895) Castellani and Chalmers 1919</td>
<td>scientific name</td>
</tr>
<tr>
<td>Escherichia coli retron Ec107</td>
<td>synonym</td>
</tr>
<tr>
<td>bacterium 10a</td>
<td>includes</td>
</tr>
<tr>
<td>bacterium E3</td>
<td>includes</td>
</tr>
</tbody>
</table>

4 TAXONOMY RESOURCE MANAGEMENT

4.1 Organism ontology

In order to provide an explicit description of the linguistic structure of organism entities and their relations, we developed, specifically for the OrganismTagger, a formal ontology in OWL-DL format (W3C Recommendation, OWL Web Ontology Language Overview, http://www.w3.org/TR/owl-features/). It is used both as a knowledge base during processing and for validating detected entities (Fig. 2). Where a document token is matched by several ontology tokens from classes in a hierarchical relationship, the most specific class is assigned. Thus, the recorded information is as specific as possible while the class entailment ensures that the token is also associated with the more general classes that it could be classified to. To gain additional expressiveness for plausibility checking, cardinality restrictions have been placed on the relations. For example, an organism can have only one genus or one species.

4.2 NCBI-database mapping

In the NCBI Taxonomy database, unranked nodes are allowed at any point in the classification and not all the taxa are assigned the Linnaean ranks (The NCBI Handbook, http://www.ncbi.nlm.nih.gov/books/shelf/fr.fcg?book=hansbook&part=ch4). To provide better accuracy, we decided to exclude mentions of type ‘no rank’ (comprising 10% of the Taxonomy database entries).

4.3 Gazetteer list generation

For organism tagging, we use the developed OWL ontology, which encodes the relationships between organism parts (taxa) as shown in Figure 2. In order to support named entity detection of organisms, the ontology must contain the taxonomical names so that they can be matched against words in a text using an onto-gazetteer NLP component. This information is extracted from the MySQL database with a number of Python scripts, including the names themselves and information like the hierarchical structure of taxa and organisms. Organism abbreviations, like P. fluorescens shown in Figure 3, do not appear in the NCBI Taxonomy database, so a list of abbreviated organisms is automatically generated from the database by adding entries with the first letter of the genus part followed by a period. Together with the taxonomical information, we store additional metadata, like the originating database and the ‘scientific name’, for each ontology instance. This becomes important when delivering provenance information to scientists working with the populated ontology.

Some organism names do not follow the binomial nomenclature rules, which is the genus followed by species. To handle these cases, a simple RDF schema (http://www.w3.org/RDF/) was developed that is limited to just one subclass hierarchy and associates the full name of the organism with the class Organism. The format of a triple is C1 rdf:s:subClassOf C2, where rdf:s:subClassOf is an instance of rdf:s:Property and states that C1, the entity, is an instance of rdf:s:Class and a subclass of C2, an instance of rdf:s:Class that is ‘Organism’. The rdf:s:domain and
To train our classifier, we extract several features for each strain candidate: (i) the ontology features of the two taxonomic units preceding the strain candidate (genus or species); (ii) the string of the strain candidate itself; and (iii) the string of the immediately following strain candidate (if any).

Finally, we provide for basic acronym detection by generating a gazetteer list of all acronyms recorded in the NCBI database. To reduce false positives, we further filter this list by removing known gene or protein names as recorded in the UniProtKB (http://ca.expasy.org/sprot/) (~2.6% of acronyms are removed in this step). More detail on acronym processing is provided in Section 5.4.

4.4 Training of the strain classifier

Not all strains that appear in the literature exist in the NCBI Taxonomy database and can be found through gazetteering. To be able to detect the remaining strains, we employ a machine learning approach using a binary classifier. Here, strain candidates are detected through a custom tokenization step and the classifier decides whether a candidate strain is actually a strain.

To train our classifier, we extract several features for each strain candidate: (i) the ontology features of the two taxonomic units preceding the strain candidate (genus or species); (ii) the string of the strain candidate itself; and (iii) the string of the immediately following strain candidate (if any).

Our classifier is based on a Support Vector Machine (SVM) model, for it is widely used in text classification tasks with unbalanced training and proved to perform better than other classifiers we also tested, including Naive Bayes and K Nearest Neighbor (KNN). Hence, in this article, we only report results of the SVM classifier. The SVM used is the implementation included with the SVMLight library.

5.RUN-TIME PROCESSING

When all the resources described above have been generated from the NCBI database, the OrganismTagger is ready for annotating documents. This is done through a pipeline of individual processing steps, where each step adds further information to a document in the form of annotations, which are then combined into a complete analysis pipeline.

5.1 Implementation

The implementation is based on the General Architecture for Text Engineering (GATE) (Bontcheva et al., 2004; Cunningham et al., 2011). GATE is a component-based architecture implemented in Java, where individual processing steps are performed by processing resources (PRs) that are then combined into a complete analysis pipeline.

By basing our implementation on GATE, our components can be easily embedded into more complex analysis pipelines and the results can be exported in various formats (XML, OWL, etc.) or accessed through a Web service. Detailed instructions on how to setup our components within an analysis pipeline are provided with the online documentation.

5.2 Preprocessing

First, documents are undergoing generic pre-processing steps including tokenization and sentence splitting, using the standard NLP components included in the GATE distribution.

5.3 Tokenization for strains

Since strains usually appear as character combinations, for instance, mixed upper and lower case or including digits, we developed a strain tokenizer to annotate all tokens that do not look like a normal word as a possible strain. For example, ‘C-125’, a strain of Bacillus halodurans can be annotated by the strain tokenizer as a possible strain. However, these possible strain words are not always strains and could be other abbreviations or address parts. For example, in Streptomyces thermoviolaceus Xyn1, ‘Xyn1’ is a protein abbreviation and not a strain. Thus, this tokenizer alone is not sufficient for determining strains. Further analysis is performed in a later step, where the trained Strain Classifier (Section 4.4) analyses the possible strains and makes the final decision whether a given possible strain is actually a strain.

5.4 Ontology-aware gazetteering

Each token in the text is now matched against the generated gazetteer lists (Section 4.3). Using an ontology-aware gazetteer, we incorporate mappings between the lists and ontology classes and assign the proper class in case of a term match. For example, the gazetteer will annotate the text segment Escherichia coli with two Lookup annotations, having their class feature set to ‘Genus’ for Escherichia and ‘Species’ for coli.

Text segments matching the acronyms are associated with Lookup annotations whose ‘class’ and ‘majorType’ features are set to ‘Organism’ and ‘acronym’, respectively. RDF gazetteering, which is used for the organisms not following the binomial nomenclature rules, is performed in a second step. This gazetteer creates annotations with the type Lookup and two features; ‘inst’, which contains the URI of the ontology instance, and ‘class’, which contains the URI of the ontology class that instance belongs to.

Note that these Lookup annotations do not yet represent a semantic organism annotation; they still need to be validated with specific grammar rules, described below.

This only needs to be done when the resources are updated.
5.5 Strain classification

Only a subset of all strains can be detected through gazetteering and rules. For detecting the remaining strains, we employ the trained machine learning model described in Section 4.4.

The classifier analyses the generated possible strains (Section 5.3) using the features described in Section 4.4. It then assigns a binary feature to every strain candidate, set to True if a strain was detected, otherwise the feature is set to False.

For example, the possible strain "endo-1.4-b" in Trichoderma reesei endo-1-4-b is detected as a false instance by our strain classifier and 'JM101' in E.coli JM101 is found as a true instance. All possible strains with the feature value of 'True' detected by the classifier are then further processed and added as a strain entity. Additional grammar rules in later steps can then consider these strains for inclusion in an organism entity.

5.6 Organism entity detection

In this step, we can now combine and verify the individual Token, Lookup and Strain annotations in order to create semantic Organism annotations. This is achieved through grammar rules written in the JAPE language (Cunningham et al., 2000), which are compiled into finite-state transducers. For example, the organism notation [genus species strain] can be encoded in JAPE as:

```
Rule: OrganismRule1
{ [Genus]:gen, [Species]:spe, [Strain]:str } -> \(\ldots\)
```

Five of these hand-written grammar rules are used within our system to detect organism entities. The result of this stage is a set of named organism entities, which are, however, not yet normalized or grounded.

5.7 Entity normalization and grounding

Detected entities now undergo a normalization and grounding step. First, we ensure that only valid organism names are extracted from texts. For example, we can reject a genus/species combination that might look like a valid name to a simple organism tagger, yet is not supported by the NCBI database. Second, by resolving abbreviations and acronyms, we can assign each detected organism a normalized name, which can subsequently be grounded to the taxonomy database by adding its database identifier.

5.7.1 Normalization. This step determines a canonical name for each organism entity by (i) resolving abbreviations and (ii) adding the scientific name as defined in the NCBI taxonomy database. This scientific name can be obtained through a simple lookup based on the lists generated above. In case of abbreviations, however, finding the canonical name usually involves an additional disambiguation step.

For example, if we encounter E.coli in a text, it is first recognized as an organism from the pattern 'species preceded by abbreviation'. Our NLP component can now query the internal lists for a genus instance with a name matching E* and a species named coli, and filter the results for valid genus–species combinations denoting an existing organism. Ideally, this would yield the single combination of genus Escherichia and species coli, forming the correct organism name. However, the above query returns in fact four entries. Two can be discarded because their names are classified by NCBI as misspellings of Escherichia coli, as shown by the identical TaxID (cf. Table 1). Yet the two remaining combinations, with the names Escherichia coli and Entamoeba coli, are both classified as 'scientific name'.

A disambiguation step now has to determine which one is the correct normalized form for Escherichia coli. Here, we apply a search heuristic based on the closest non-abbreviated form appearing in the document that matches the genus (Witte et al., 2007). This heuristic relies on the convention that each abbreviated form is usually introduced by the full form at least once within a text.

In case the non-abbreviated form does not appear in the document, a second normalization heuristic retrieves all genus mentions that match the abbreviated form. These are considered as candidates and the heuristic attempts to find a matching organism. For example, in pmc1891629, none of the abbreviated forms of C.sheathia, C.homonota or C.pipiens appear with their full form, but "Culex" does appear separately and we can successfully resolve the abbreviations using this genus.

If the abbreviated organisms still cannot be resolved, all possible matching entries are returned and added as annotations. Although less precise, this list is still valuable for the disambiguation of other entities in a text, like proteins (Witte et al., 2007).

Detected acronyms are also normalized to their scientific name as defined in the NCBI taxonomy database, for example, FMDV is normalized to foot-and-mouth disease virus.

5.7.2 Grounding. Once the normalized name has been determined, we can uniquely ground it in the NCBI database by adding the corresponding ID. Since the database record can now be unambiguously looked up, the entity is grounded with respect to an external source. The end result of this step is a semantic annotation of the named entities as they appear in a text, which includes the detected information from normalization and grounding, as shown in Figure 3.

Some organisms with strains or subspecies are associated with several different NCBI IDs in the taxonomy database. For these organisms, both IDs for the full name of the organism with strain (or subspecies) and also for the organism without the strain (or subspecies) are provided by our tagger. For example, as shown in Figure 3, Pseudomonas fluorescens subsp. cellulosa is annotated with two NCBI Taxonomy IDs (294, 155077), and two scientific names (Pseudomonas fluorescens, Cellulohirao japonicus), one for the taxa level that is Pseudomonas fluorescens and the other with the subspecies cellulosa. This provides a user of our tagger with the most comprehensive information for further processing.

Some common names also cause ambiguity (e.g. mice, rats), which need to be disambiguated and grounded. For example, Mice can refer to Mus with NCBI Taxonomy ID ‘10088’ as ‘genus’, Mus sp. with NCBI Taxonomy ID ‘10090’ as ‘species’ and Mus musculus with NCBI Taxonomy ID ‘10090’ as ‘species’. We normalize mice based on the document context. When there are mentions of mouse, transgenic mice or nude mice, it is grounded to the NCBI Taxonomy ID ‘10090’. When it appears alone, we report both NCBI Taxonomy IDs 10090 and 10095, discarding the NCBI Taxonomy ID ‘10088’, as we do not report IDs for genus parts. Rats can be also grounded to Rattus sp. with NCBI Taxonomy ID ‘10118’ as ‘species’, Rattus with NCBI Taxonomy ID ‘10114’ as ‘genus’ and Rattus norvegicus with NCBI Taxonomy ID ‘10116’ as ‘species’. If there are any mentions of rat, laboratory rat or Sprague-Dawley rat, rats is grounded to the NCBI Taxonomy ID 10116. Otherwise, two NCBI Taxonomy IDs, 10118, 10116’, are reported and ‘10114’ is rejected.

6 EVALUATION

In this section, we provide a detailed performance evaluation of our system. We first discuss the manually annotated corpora (gold standard) in Section 6.1, followed by the metrics used for evaluation in Section 6.2, the results in Section 6.3, a comparison with other systems in Section 6.4 and an analysis of strain recognition in Section 6.5.

6.1 Corpora

Two manually annotated corpora, containing full-text articles, are used for evaluation: (i) a corpus of 41 documents on protein engineering and fungi (incl. the following named OT corpus) and (ii) a corpus of open access biomedical documents from the PMC (here named Linnuus-100).

6.1.1 Corpus preparation and manual annotation. Documents were converted from their original format [e.g. HTML, PDF] into XML format.
The different versions of the Linnaeus corpora used for evaluating the OrganismTagger

<table>
<thead>
<tr>
<th>Corpus</th>
<th>L-100A</th>
<th>L-100B</th>
<th>L-100C</th>
<th>L-100D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tax-Names</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Non-Tax-Names</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CommonNames</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Acronyms</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>AuthorDefinedAbbreviations</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>IncorrectNameUsage</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>AddedMissingOrganisms</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>AddedMissingStrains</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Different entity types are included in different versions to provide a detailed performance analysis.

The reference sections often contain organism names, but are not processed by all external systems. Considering them would lead to inconsistencies when performing cross-system evaluations. Hence, we decided to target only the abstract and main parts of each document and removed the reference sections.

For the OT corpus, manual annotation was performed by us, using an XML-annotation schema, in the GATE Developer GUI environment (Cunningham et al., 2011). The annotations are saved, together with the original content, in the XML file using stand-off markup similar to the Tipster standard (Grishman, 1997). The markup schema allows for annotating the organism names (document names), scientific names, individual taxa (genus, species, strain) and the NCBI ID. Organisms with no existing NCBI taxonomy ID are annotated, but no IDs are assigned to them, hence indicating a valid organism entity that cannot be grounded to the taxonomy database. While we currently cannot distribute the original documents with their annotations due to copyright restrictions, we provide our manual annotations in a separate file in tab-delimited format, similar to other publicly available annotations.

The Linnaeus-100 corpus has been annotated by Germer et al. (2010) and is also freely available online. However, for the detailed evaluation performed here, we needed to map their tab-delimited annotations to the full text in order to compare the resulting annotations. For this, we obtained the documents from the authors of the Linnaeus system and converted them into the same stand-off markup as our corpus. In their original annotations, organisms that do not appear in the taxonomy database received an NCBI ID of 0. To keep the data consistent for this evaluation, these NCBI IDs were removed and no IDs are assigned to these organisms, since an ID in our system indicates successful grounding to an existing NCBI entry.

6.1.2 Corpus variations. To be able to analyze the system performance depending on the type of the organism mention (full form, common name, abbreviation, etc.), we prepared different versions of both the OT and the Linnaeus corpus. The Linnaeus corpus is already distributed with two sets of tags: one set includes only the NCBI and non-NCBI taxonomy names tags and the other set additionally includes non-taxonomy names like ‘participant’, ‘children’ and ‘patient’ (L-100-A in Table 2).

An analysis of the original version of the Linnaeus evaluation data revealed that 45% of strain mentions were missing. This suggests that the performance of our system can be better reflected by our proposed revised data, where we added the missing mentions. Moreover, we corrected some erroneous or missing annotations in the Linnaeus-100 corpus: for example, in the documents: pmoA47033 and pmoA2265096, the organismis transgenic mice and Escherichia coli strain BL21 were not manually annotated. We added these mentions as they would otherwise be flagged as false positives in the evaluation. All these changes are documented in the same format as the original Linnaeus-100 corpus and distributed with the OrganismTagger system.

6.2 Evaluation metrics.

We use the standard metrics precision, recall and F-measure, as well as accuracy, to evaluate the performance of our system (Witten and Baker, 2007). Here, the number of correctly identified items as a percentage of the total number of correct items is defined as recall (R). Conversely, the number of correctly identified items as a percentage of the number of items identified is specified as precision (P). The F-measure (F) is used as a weighted (geometric) average of precision and recall.
The performance of the OrganismTagger (v.1.1) is evaluated in three different
tables. The performance results are computed according to different criteria:
Strict (S) and Lenient (L). In 'Strict', we measure all partially correct
responses as incorrect: e.g. cases where the strains or subspecies are not found
or only partially found are considered incorrect. In 'Lenient', all partially
correct responses are measured as correct.
The accuracy (A) for the correctly retrieved NCBI IDs is separately
calculated based on the number of correctly identified NCBI IDs over
the number of correctly normalized organisms. If an abbreviated form
of an organism mention could not be uniquely resolved in a document,
the OrganismTagger retrieves all the possible matching organisms. In this
evaluation, these cases are all reported as false positives, even if the result
does include the correct NCBI taxonomy ID.

| Table 5. Comparative evaluation of the OrganismTagger and Linnaeus systems on the different corpora |
|---------------------------------------------------|---|---|---|---|---|---|---|
| OT-A | OT-B | L-100-A | L-100-B | L-100-C | L-100-D |
| S (%) | L (%) | S (%) | L (%) | S (%) | L (%) | S (%) | L (%) |
| OrganismTagger |
| Detected entities | P | 94 | 99 | 94 | 99 | 92 | 95 | 96 | 97 | 96 | 97 | 97 | 98 |
| R | 95 | 99 | 94 | 99 | 61 | 63 | 98 | 99 | 97 | 98 | 98 | 98 |
| E | 95 | 99 | 94 | 99 | 76 | 76 | 97 | 98 | 97 | 98 | 97 | 98 |
| Normalized | P | 95 | 99 | 95 | 100 | 93 | 96 | 97 | 97 | 98 | 99 | 99 | 100 |
| R | 94 | 98 | 93 | 98 | 61 | 63 | 98 | 98 | 96 | 97 | 97 | 98 |
| F | 94 | 98 | 94 | 99 | 74 | 76 | 97 | 98 | 97 | 98 | 98 | 99 |
| Grounded | A | 97.5 | 99.3 | 96.7 | 97.7 | 97.5 | 97.4 |
| Linnaeus |
| Normalized | P | 71 | 83 | 66 | 78 | 97 | 98 | 96 | 98 | 94 | 98 | 96 | 100 |
| R | 81 | 94 | 78 | 93 | 93 | 94 | 90 | 92 | 83 | 87 | 83 | 86 |
| E | 76 | 88 | 72 | 85 | 95 | 96 | 93 | 95 | 89 | 92 | 89 | 92 |
| Grounded | A | 94 | 95.2 | 96.5 | 94.7 | 94.35 | 95.0 |

For the OrganismTagger, we show the performance of the named entity recognition (Detected Entities), assigning a canonical name (Normalization) and adding the ID for the NCBI
Taxonomy database (Grounding). Data are represented as percentage.

6.3 Results
The performance of the OrganismTagger (v.1.1) is evaluated in three different
steps (Table 5): first, the mentions of the organisms are evaluated against the
gold standard without taking the NCBI IDs into consideration (Detected Entities). In this step, the focus is on the entity recognition
performance of the OrganismTagger.
The performance is further evaluated based on the normalized mentions of
the organisms. 'Normalized' If the organism is not successfully normalized, it is removed from the list of retrieved organisms.

And finally, the computed NCBI IDs are compared against the manual
annotations, specified as 'Grounded'.

Linnaeus (Gerner et al., 2010), a species name identification system, uses
deterministic finite-state automatons (DFAs) to capture species mentions and
then assign them their NCBI taxonomy IDs. However, when the full form
appears but no NCBI IDs exist for the species, all the abbreviated forms of
this species are associated with other possible NCBI IDs. To compare our
results with the results of the Linnaeus system (v.1.5), we applied it in the
same way as the OrganismTagger on the manually annotated corpora. The results for the Linnaeus system are also shown in Table 5.

6.4 Discussion
False negatives of the OrganismTagger in the Linnaeus-100 corpora are mainly due to misspellings. Some acronyms resembling gene and protein
names are filtered for minimizing error propagation. These acronym
mentions are ignored by the OrganismTagger. Also, we do not cover the occurrence of species ranked as ''no rank'' in the Taxonomy database, e.g.
Salmonella typhimurium. The Linnaeus system uses additional synonyms to
capture terms like 'patient' and 'children'. This is reflected by the recall
section of the L-100-A corpus. For better comparison of the Linnaeus
performance with that of our system, these additional synonyms of the
Linnaeus system are ignored in the L-100-B, L-100-C and L-100-D corpora.
False positive mentions in the Linnaeus-100 corpora arise from common
names like small white for Pieris rapae, NCBI ID: 64459 and white
underwing for Catocala relicta, NCBI ID: 423327. Some acronyms, namely
PAR and ATP, captured by the OrganismTagger refer to non-organism
mentions. We expect that in some cases, problematic acronyms can be filtered out by more accurate protein and gene name lists. However,
the elimination of all false positives will negatively impact on recall. False negatives of the Linnaeus system are due to its inability to handle
ligatures and the limited strain recognition, which is analyzed in more
detail below. Moreover, some organisms are renamed and the old names
usually appear following the new names in parentheses, like in Emericella
(Aspergillus) nidulans; these cases are also ignored by the Linnaeus system.

While many abbreviated organisms can be resolved to their non-
abbreviated form after locating the full form in the document, a few
still remain ambiguous. Using our additional heuristic, some of these
ambiguous organisms can be resolved to their non-abbreviated format. In
particular, L-100-D contains 69 abbreviated organism mentions without the
corresponding full form in OT-A and OT-B. However, when applying the
second normalization heuristic, the OrganismTagger successfully resolved
50 and 21 in L-100-D and OT-B, respectively.

We also performed comparisons with other systems, but do not report
them in detail here as their performance is generally below that of the
Linnaeus system. Using a combination of a lexicon with non-taxonomic
words and rules, TaxonGrab (Koning et al., 2006) finds the longest match
without grounding the entity. NaCTeM Species-Word Detector and NaCTeM
Species Disambiguator (Wang and Grover, 2008; Wang et al., 2010) mostly
annotate the species level. For example, Pseudomonas fluorescens
Table 6. OrganismTagger versus Linnaeus system in the strain recognition task

<table>
<thead>
<tr>
<th>Corpus</th>
<th>No. of strains</th>
<th>Strains in NCBI</th>
<th>OrganismTagger</th>
<th>Linnaeus</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No. (%)</td>
<td>Gazetteer No. (%)</td>
<td>Rule No. (%)</td>
<td>ML No. (%)</td>
</tr>
<tr>
<td>OT-A</td>
<td>307 (23.4)</td>
<td>33 (10.7)</td>
<td>58 (18.8)</td>
<td>158 (51.4)</td>
</tr>
<tr>
<td>L-100-D</td>
<td>85 (64.9)</td>
<td>42 (44.9)</td>
<td>37 (43.5)</td>
<td>24 (28.2)</td>
</tr>
</tbody>
</table>

Number of strains appearing in the evaluation corpora, strains recorded in the NCBI Taxonomy database, strain recognition by method (gazetteering, rule-based, ML), and overall strain recognition performance (Total columns).

strain naming convention. Moreover, not all strains are recorded in the NCBI taxonomy database, which impedes gazetteering-based approaches.

We further analyzed the organism mentions that include strains. Table 6 shows the number of strains appearing in the evaluation corpora: only about 24%–49% of these strains are recorded in the NCBI Taxonomy database. The table also shows how many of these strains have been correctly recognized (81.1%–94% for the OrganismTagger versus 20.1%–37.6% for the Linnaeus system). For our system, we also analyzed which method contributed to the strain recognition (gazetteering, rule-based or machine learning). If a strain was recognized both through machine learning and another method, we only counted it once for gazetteering or rule based, as we were interested in the number of strains that cannot be captured with either of these two approaches.

Overall, mentions of strains are usually ignored by other existing systems or limited to strains that exist in the NCBI Taxonomy database or can be captured by rules. In contrast, the OrganismTagger is explicitly designed to capture strains, and if possible, provide the NCBI database IDs both for the taxon level and the three-part taxonomic designation. In these cases, the Linnaeus system takes the approach of choosing only the longest match, for example, *Pseudomonas fluorescens subsp. cellulosa* is grounded with the NCBI Taxonomy ID ‘155077’.

7 CONCLUSION

In this article, we described the OrganismTagger system for semantically annotating organism mentions in documents.

First, we emphasized the importance of system updateability with respect to external resources. Our approach is to provide tools that allow users to automatically transform the well-known NCBI Taxonomy database into data structures suitable for text mining tasks.

Second, we provide ontologies for the lexical description of organism mentions using the standardized OWL and RDF formats, which can also be reused in other semantic systems.

Third, we have implemented a comprehensive, open-source text mining system for detecting organism mentions, including normalization and grounding. In particular, it includes a novel strain recognition part, which is capable of annotating strains that do not appear in the taxonomy database.

Fourth, we evaluated the system on multiple corpora. In this process, we reproduced the results reported for the Linnaeus system and created additional manual annotations for the biomedical literature. We provided detailed evaluations that break down the challenges in detecting organism mentions and demonstrated significant improvements in strain recognition and normalization.

Our evaluation shows a number of directions for future improvements. Finding the remaining missing entities is challenging due to their diverse nature: in some cases, misspellings, either introduced by the authors or through format conversions, prevent organism recognition. This could be partially addressed by introducing automatic spelling corrections. Author-defined abbreviations and acronyms are another source of false negatives; this can be partially addressed by integrating a database like Allie (http://allie.dbcls.jp/). Further improvements in this area require the development of new co-reference resolution strategies, which is also important for many other entity types, such as proteins or mutations. The detection of common names introduces a number of ambiguities, in particular for the normalization and grounding tasks. We believe this will ultimately need to be addressed by the end user, by tailoring the system’s configuration to their specific subdomain and application scenario. Strain recognition is one of the most challenging parts of organism detection, as strain designations do not follow any naming conventions. While our approach already demonstrates higher performance than existing systems, additional improvements in recall, without sacrificing precision, will be a continuing challenge.

ACKNOWLEDGMENTS

Marie-Jean Meurs and Jonas B. Laurila are acknowledged for critical reading of the article and valuable suggestions. We also thank Martin Gerner for providing us with the Linnaeus corpus.

**Funding**: Natural Sciences and Engineering Research Council of Canada (NSERC) discovery grant, in part; Quebec/New Brunswick University Co-operation in Advanced Education grant, in part.

Conflict of Interest: none declared.

REFERENCES


Cunningham, H. et al. (2000) IAPE: a Java Annotation Patterns Engine, 2nd edn. Research Memorandum CS-00-10, Department of Computer Science, University of Sheffield, Sheffield, UK.
Cunningham, H. et al. (2011) Text Processing with GATE (Version 6). Department of Computer Science, University of Sheffield, Sheffield, UK.
Hakenberg, J. et al. (2008) Inter-species normalization of gene mentions with GNAT. Bioinformatics, 24, i426.