Description of the POMELO System for the Task 2 of QALD-4

Thierry Hamon[1,2], Natalia Grabar[3], Fleur Mougin[4], and Frantz Thiessard[4]

[1] LIMSI-CNRS, Orsay, France
thierry.hamon@limsi.fr,
[2] Université Paris 13, Sorbonne Paris Cité, France
[3] STL UMR8163 CNRS, Université Lille 3, France
natalia.grabar@univ-lille3.fr,
[4] Université Bordeaux, ISPED, Centre INSERM U897, ERIAS, France
firstname.lastname@isped.u-bordeaux2.fr

Abstract. In this paper, we present the POMELO system developed for participating in the task 2 of the QALD-4 challenge. For translating natural language questions in SPARQL queries we exploit Natural Language Processing methods, semantic resources and RDF triples description. We designed a four-step method which pre-processes the question, performs an abstraction of the question, then builds a representation of the SPARQL query and finally generates the query. The system was ranked second out of three participating systems. It achieves good performance with 0.85 F-measure on the set of 25 test questions.

Keywords: Natural Language Processing, SPARQL, biomedical domain, semantic resources

1 Introduction

Biomedical knowledge is disseminated in knowledge bases which become increasingly available on the Web. These knowledge bases usually focus on a given type of information: chemical, pharmacological and target information on drugs in Drugbank [11], clinical studies in ClinicalTrials.gov[5], drugs and their side effects in Sider [8], etc. The connection of such life-science knowledge bases is crucial for obtaining more global and comprehensive view on the links that may exist between different biomedical components, factors and actors. Moreover, this allows inducing and producing new knowledge from the already available data. Particularly, the creation of fine-grained links between the existing knowledge bases related to drugs is a great challenge that is being addressed by the project Linked Open Drug Data (LODD) for instance[6]. In this project, the knowledge recorded in the knowledge bases and dataset interlinks is represented as RDF triples, on the basis of which the linked data can then be queried through a SPARQL endpoint. However, typical users of this knowledge, such as physicians, life-science

[5] http://clinicaltrials.gov/
[6] http://www.w3.org/wiki/HCLSIG/LODD

researchers or even patients, cannot manage the syntactic and semantic requirements of the SPARQL language neither can they manage the structure of various knowledge bases. This situation impedes the efficient use of knowledge bases and the retrieval of useful information. Therefore, it is important to design friendly interfaces that mediate the technical and semantic complexity of the task and provide simple approaches for querying the knowledge bases.

For instance, it has been shown that for querying the knowledge bases and the Semantic Web data, the use of full and standard sentences is preferred to the use of keywords, menus or graphs [6]. While this study is conducted on general knowledge data, we assume this observation is also relevant for the users of biomedical knowledge bases. Up to now the design of friendly user interface is mainly addressed for general knowledge bases [3], [7]. We can also mention another work which aim is to translate medical questions issued from a journal into SPARQL queries [1].

In relation with such research problems, the Question Answering over Linked Data (QALD-4) campaign proposes a task dedicated to the retrieval of precise biomedical information in linked knowledge bases according to questions in natural language. We present in this paper the methodology we propose to translate natural language questions in SPARQL queries and the system we developed for our participation to the challenge.

We start with the definitions of the main terms used in the proposed presentation (Sect. 2). Then, we describe the semantic resources available and developed for enriching the questions (Sect. 3). The methodology and the system are described in Sect. 4. The evaluation of the system on the QALD-4 queries is presented in Sect. 5.

2 Terminology

The main terms are used with the following meaning:

Question The questions are the natural language expressions uttered by human users in order to formulate their information need.

Query The queries respect the SPARQL syntax and semantics. They are created automatically on the basis of (natural language) questions.

Semantic type Semantic types are indicative of the word meaning. The semantic types are defined from URI prefixes such as *disease* for *rickets*, `drug` for `Cetuximab`, etc.

Frame Frames are defined as linguistic representations of RDF schema. Usually the frames contain one predicate and at least two elements with associated semantic types.

Linguistic annotation The linguistic annotations are obtained with the Natural Language Processing (NLP) tools and provide lemma, part-of-speech categories, and terms.

Semantic annotation Semantic annotations are obtained by tagging questions with semantic resources.

The main challenge of the proposed work is to design the optimal methodology for an easy and reproducible rewriting of (natural language) questions in SPARQL queries.

3 Definition of the Semantic Resources

Some resources used are provided by the challenge organizers (Sect. 3.1), others are collected and built specifically for the challenge (Sect. 3.2) to support the method. We use these resources for rewriting the questions in queries.

3.1 Resources Provided by the QALD Challenge

Three datasets are provided by the QALD challenge:

- Drugbank[7] is dedicated to drugs [11]. It merges chemical, pharmacological and pharmaceutical information from other available knowledge bases. We exploited the documentation[8] of this resource to define rewriting rules and regular expressions in our named entity recognizer.
- Diseasome[9] is dedicated to diseases and genes linked among them by known disorder/gene associations [5]. It provides a single framework with all known phenotypes and disease gene associations, indicating the common genetic origin of many diseases. We exploited the RDF triples and the documentation of the resource to define the rewriting rules.
- Sider[10] is dedicated to adverse drug effects [8]. It contains information on marketed medicines and their recorded adverse drug reactions. The information is extracted from public documents and package inserts. The available information includes side effect frequency, drug and side effect classifications as well as links to other information, for example drug-target relations.

The content of each resource is provided in a specific format: RDF triples *subject predicate object*, so that they encode the useful and usable frame elements.

3.2 Resources Collected and Built for the QALD Challenge

On the basis of the RDF triples, we build frames from the RDF schema where the RDF predicate is the frame predicate, and subject and object of the RDF triples are the core frame elements. This also includes the OWL `sameAs` triples. Several types of frame entities are isolated:

- As indicated, subject, object and predicate become semantic entities. They may occur in questions: in this way, the frames are the main resource for rewriting questions in queries.

[7] http://www.drugbank.ca
[8] http://www.drugbank.ca/documentation
[9] http://diseasome.eu
[10] http://sideeffects.embl.de

– The vocabulary specific to questions is also built. It covers for instance aggregation operators, negation and types of questions.
– RDF literals, issued from named entity recognizer or term extractor, complete the resources. The RDF literals are detected with specifically designed automata that may rely on the source knowledge base documentation.

These entities are associated with the expected semantic type, which allows creating the queries and rewriting the RDF triples in the SPARQL queries. In that respect, we can consider IRI, strings, common datatype or regular expressions when literals are expected.

Most of the entities are considered and processed through their semantic type, although some ambiguous entities (e.g. interaction or class) are considered atomically. For these, the rewriting rules will be applied contextually to generate the semantic entities corresponding to the frames (see Sect. 4.2). When using the queries, the semantic types are variables and are used for connecting the edges of queries.

4 System Description

We design a four-step process based on NLP methods, semantic resources and RDF triple description (see Fig. 1):

1. We pre-process the questions in order to enrich them with linguistic and semantic information (Sect. 4.1).
2. We perform a question abstraction (Sect. 4.2).
3. We use the abstracted question to construct the corresponding SPARQL query representation (Sect. 4.3);
4. We generate the SPARQL query (Sect. 4.4).

The process is implemented as a module within the NLP platform Ogmios [4].

4.1 Pre-processing

The pre-processing of the questions is the same for the training and test sets. The annotation of the questions consists in recognition of numerical values (such as numbers and solubility values), word segmentation, part-of-speech tagging and lemmatization of the words with TreeTagger [9]. Then, we apply the TermTagger Perl module[11] for identifying semantic entities, i.e. terms with associated semantic types. TermTagger exploits the semantic resources (see Sect. 3) to recognize semantic entities such as disease names, side effects, etc.

However, as we realized during our preliminary experiments on the training set, the coverage of the terminological entities that appear in questions is not sufficient. We also apply the term extractor $Y_A T_E A$[12] [2] to improve the coverage of our method. The term extractor performs shallow parsing of the POS-tagged

[11] http://search.cpan.org/~thhamon/Alvis-TermTagger/
[12] http://search.cpan.org/~thhamon/Lingua-YaTeA/

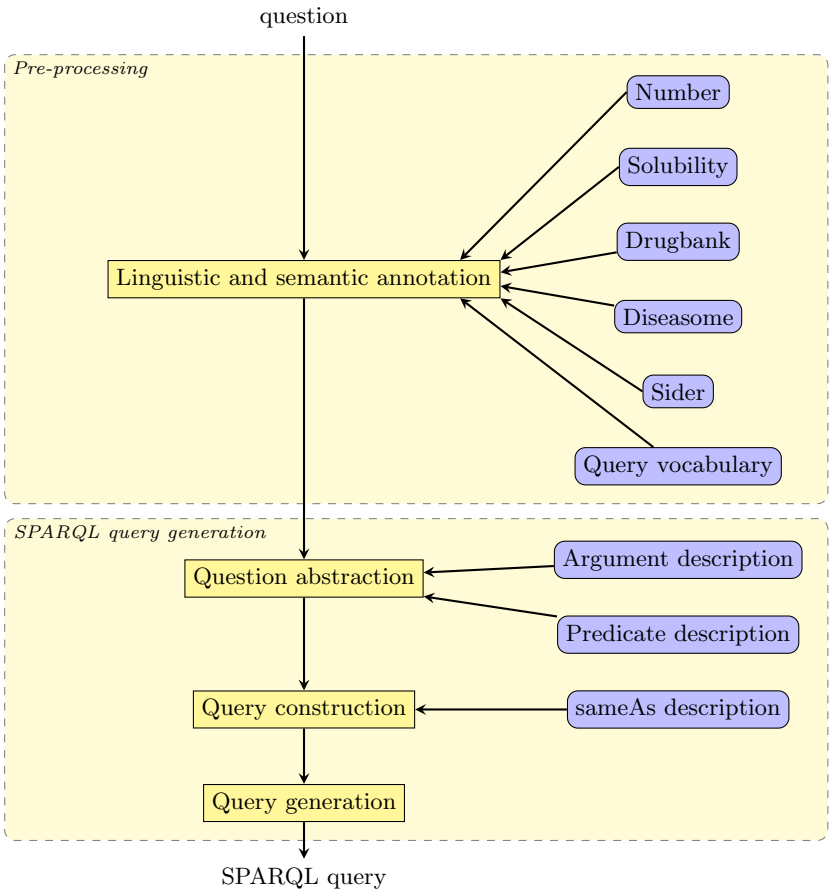

Fig. 1. Architecture of the POMELO system (the processing steps are in yellow, the resources in blue)

and lemmatized text by chunking it according to syntactic frontiers (pronouns, conjugated verbs, typographic marks, etc.) in order to identify noun phrases. Then, parsing patterns[13] that take into account the morpho-syntactic variation, are recursively applied and provide parsed terminological entities, usually noun phrases relevant for the targeted domain. Each term is represented as a syntactic tree, and sub-terms are also considered as terms in the current configuration. No semantic types are associated to the terms extracted by YaTeA. Figure 2 illustrates the linguistic and semantic annotation of a question.

On the training step, we observed that the lemmatizer and POS-tagger Genia Tagger [10] performs a better lemmatization. However, we did not use it on the test set because it is time consuming.

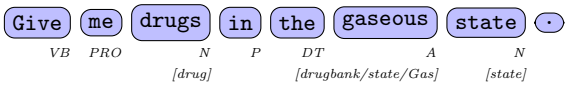

Fig. 2. Example of question pre-processing (question#22 of the test set). The blue boxes represent the word and semantic entites. The subscript texts are the Part-of-Speech tags and the bracketed subscript texts are the semantic types associated to the semantic entities.

4.2 Question Abstraction

This step aims at identifying the relevant elements within the questions and building the representation of these elements. At this step, we use the linguistic and semantic annotations associated to the question words in the previous step.

Before the identification of relevant elements, the disambiguation of the annotations is carried out with rewriting rules. Indeed, the annotated semantic entities may receive conflicting or concurrent semantic types, while the post-processing permits to select those entities and semantic types that may useful for the next steps. For instance, we keep larger terms which do not include other semantic entities.

Thus, we defined rewriting rules in order to modify or delete the semantic type associated with an entity according to the context. Other rules may also modify or delete the entity according to the context. For instance, the semantic entity *interaction* has to be rewritten in `interactionDrug1` if its context contains mention of drug, but it has to be rewritten in `foodInteraction` if its context contains a term with the semantic type *food*. On the whole, we defined 44 contextual rewriting rules based on the vocabulary used in the questions and

[13] We use the parsing patterns provided in the default configuration of the term extractor. These patterns have been previously manually defined and reflect basic syntactic dependency in terminological entities.

on the documentation of knowledge bases, mainly the one from Drugbank[14]. Besides the rewriting rules, additional disambiguation of the annotations is also performed during the *query construction* step when the arguments of the predicate are connected by selecting the correct semantic types.

For performing the abstraction of questions, we identify information related to the query structure:

1. Definition of the Result form: the question is scanned for identifying words expressing the negation, e.g. *no*, and its scope, the aggregation operation on the results, e.g. *number* for `count`, *mean* for `avg` or *higher* for `max`, and specific result form such as boolean queries (`ASK`). The information concerning the presence of the negation and aggregation operators or of specific result form is recorded in data structures to be used at the end of the *query construction* step or during the *query generation* step.
2. Identification of the Question topic: we consider the first semantic entity with a given expected semantic type to be the question topic. The expected semantic types are those provided by the RDF subjects and objects in the Drugbank, Diseasome and Sider. This information will be used during the *query construction* step.
3. Identification of Predicate and Argument: according to our internal frame-based representation of the three resources, the potential predicates, subjects and objects are identified among the semantic entities and described into a symbol table.[15] At this step, the subjects and objects are fully described in the symbol table. Concerning the predicates, only the semantic types of their arguments are instantiated in the symbol table within the RDF schema. The subjects and objects can be URI, RDF typed literals (numerical values or strings) and extracted terms (these are considered as elements of regular expressions).

Figure 3 presents the abstraction of the question 22 of the test set.

4.3 Query Construction

The objective of the *query construction* step is to connect previously identified elements and to build the SPARQL graph pattern (introduced by the keyword `WHERE`). Thus, the symbols of the predicate arguments are instantiated by either URI associated with the subjects and objects, variables, numerical values or strings.

For each question, we perform several connections:

[14] http://www.drugbank.ca/documentation

[15] Similarly to a compiler, RDF schema of the predicates and semantic types of their arguments are considered as entries of the symbol table. Information associated with the symbols (inflected and lemmatized form of the word or term, corresponding SPARQL type of the symbol, semantic types of the arguments, indicators of the use of the symbol as object or subject of a predicate, etc.) are recorded in a data structure.

Agregation operator:	
Question topic:	drug (`sider/drugs`)

Predicates:	Frame		
	`drugbank/drugs` state `STRING`		
Arguments:	Semantic type	Word	SPARQL type
	`drugbank/state/Gas`	gaseous	STRING/Gas

Fig. 3. Example of question abstraction (question#22 of the test set)

1. The question topic is connected to the predicate(s). A variable is associated with the question topic and the predicate arguments that matched the semantic type of the question topic. Note that at the end of this stage, the question topic may remain non-associated with any predicate;
2. The predicate arguments are connected to the subjects and objects identified during the question abstraction: they concern elements referring to URI;
3. The predicates are connected between them through their subjects and objects. The connection between two predicates is then represented by a variable;
4. The predicates from different datasets are connected. We use the `sameAs` description to identify URI referring to the same element. New variables are defined to connect two predicates;
5. The remaining question topic is connected to arguments of the `sameAs` predicate;
6. The arguments are connected to the `string` type to extracted terms annotated in the question. We assume these arguments will be related to the string matching operator `REGEX`. Thus, terms are considered as string expressions.

At this point, the predicate arguments which remain unassociated are replaced by new variables in order to avoid empty literals. Finally, the negation operator is processed: the predicates are marked as negated and the arguments within the negation scope are included in a new predicate `rdf:type` if required.

At this stage, the question is fully translated into data structures representing the SPARQL query. Figure 4 illustrates the construction of the query corresponding to the question 22 of the test set.

Agregation operator:			
Question topic:	?v0		
Predicates:	Frame		
	?v0 state `STRING/Gas`		
Arguments:	Semantic type	Word	SPARQL type
	`drugbank/state/Gas`	gaseous	STRING/Gas

Fig. 4. Example of query construction (question#22 of the test set)

4.4 Query Generation

This final step aims at generating the SPARQL query string based on the data structures built during the *query construction* step. The output of this step is the string corresponding to the SPARQL query. It is composed of two parts:

1. The generation of the result form which takes into account the expected type of the result form (`ASK` or `SELECT`), the presence of aggregation operators and the variable associated to the question topic;
2. The generation of the graph pattern. Basically, the part of the query generation consists in generating the strings representing each RDF triple and filter if predicates are negated. But when aggregation operators are used, we also need to recursively generate sub-queries computing the subsets of expressions, before their aggregation.

The SPARQL queries have been submitted without retrieving the answers. We let this task to the evaluation tool. Figure 5 presents the generated query which corresponds to the question 22 of the test set.

```
SELECT DISTINCT ?v0
WHERE {
?v0 <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/state> "Gas".
}
```

Fig. 5. Example of query generation (question#22 of the test set)

5 Results

5.1 Evaluation Metrics

The automatically generated SPARQL queries are evaluated with the online evaluation tool[16]. The answer of each query q is compared with the gold standard. The evaluation measures (F-measure, precision and recall) are computed. The system results are evaluated with macro-measures. The challenge provides 25 questions for the training and 25 questions for the test.

5.2 Global Results

In Tab. 1, we present the overall results on the training and test sets. Our system was ranked 2nd out of 3 submissions. We can observe that our results are similar for these two sets of queries (training and test sets).

We can observe that in both cases, for 19 questions, the system exactly provides the expected answers. For 4 questions from the training set and 3 questions from the test set, we obtain partial answers. In the training set, 2 questions receive no answer while in the test set, we have 3 such questions.

[16] http://greententacle.techfak.uni-bielefeld.de/~cunger/qald/index.php?x=evaltool&q=4

Table 1. Results on the training set and the test set

query set	Recall	Precision	F-measure
Training	0.87	0.83	0.85
Test	0.87	0.82	0.85

Error Analysis To our opinion, the reference SPARQL query for the question 19 of the training set is not correct: the expected result of the question is a list of drugs, while the reference SPARQL query returns a list of diseases. On the test set, we can propose two observations on the limitations of our system:

- In question 1,[17] the contextual rewriting rules cannot be correctly applied because the semantic entity (`gene ... associated`) is discontinued;
- In question 18,[18] the system correctly detects the semantic entities and the predicates, including the `sameAs` predicate. The remaining problem is that the system assumes that the `sameAs` predicate is reflexive while in the resources provided, the instances of this predicate do not encode the reflexivity of the relation.

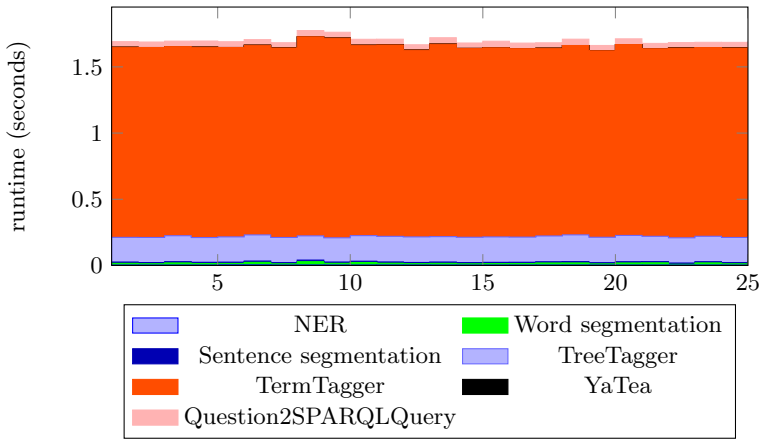

Fig. 6. performance per sub-step for each question of the test set.

5.3 System Performance

We analyzed the system performance on a standard computer (2.7GHz dual-core CPU and 4 Gb of memory). Figure 6 presents the running time for each query ac-

[17] *Which genes are associated with Endothelin receptor type B?*

[18] *List the number of distinct side-effects of drugs which target genes whose general function involves cell division.*

cording to the pre-processing sub-steps (named entity recognition, word and sentence segmentation, POS tagging, semantic entity tagging and term extraction) and the question translation into SPARQL queries (Question2SPARQLQuery). Most of the processing time is dedicated to the TermTagger which aims at recognizing the semantic entities. With the internal Ogmios processing (i.e. mainly the input/output), each question is processed in 1.94 seconds on the average on the training set and 1.97 seconds on the average on the test set.

Figure 7 shows the overall system performance according to the number of questions to process. The time needed for processing all the questions is higher on the test set. The variation of running time between processing one question and the whole set of questions is less than one second on the training set and a little more than one second on the test set. We assume the difference is due to the higher complexity of the questions in the test set.

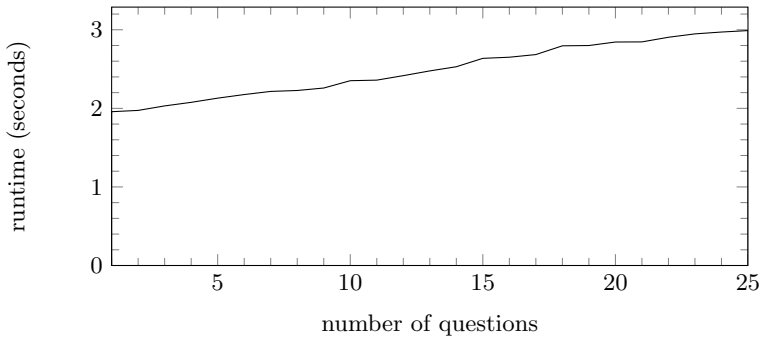

Fig. 7. System performance for an increasing number of questions issued from the test set.

6 Conclusion

We proposed a four-step process based on Natural Language Processing methods, semantic resources and RDF triples description. The system achieves good performance with 0.85 F-measure on the test set of 25 questions. It is ranked 2nd out of 3 submissions. Further work includes optimization of the running time for the question processing and the extension of the SPARQL syntax by taking into account the operators on collections of sets. We also plan to investigate the integration of other biomedical resources such as Dailymed or RxNorm, and the use of the developped system for the text mining applications.

Acknowledgments

This work was partly funded through the project POMELO (*PathOlogies, MEdicaments, aLimentatiOn*) funded by the MESHS (Maison européenne des sciences de l'homme et de la société) under the framework *Projets Émergents*.

References

1. Abacha, A.B., Zweigenbaum, P.: Medical question answering: Translating medical questions into sparql queries. In: ACM SIGHIT International Health Informatics Symposium (IHI 2012) (2012)
2. Aubin, S., Hamon, T.: Improving term extraction with terminological resources. In: Salakoski, T., Ginter, F., Pyysalo, S., Pahikkala, T. (eds.) Advances in Natural Language Processing (5th International Conference on NLP, FinTAL 2006). pp. 380–387. No. 4139 in LNAI, Springer (August 2006)
3. Damljanovic, D., Agatonovic, M., Cunningham, H.: Natural language interfaces to ontologies: Combining syntactic analysis and ontology-based lookup through the user interaction. In: Proceedings of the 7th International Conference on The Semantic Web: Research and Applications - Volume Part I. pp. 106–120. ESWC'10, Springer-Verlag, Berlin, Heidelberg (2010)
4. Hamon, T., Nazarenko, A., Poibeau, T., Aubin, S., Derivire, J.: A robust linguistic platform for efficient and domain specific web content analysis. In: Proceedings of RIAO 2007. Pittsburgh, USA (2007), http://riao.free.fr/papers/64.pdf, 15 pages
5. Janji, V., Prulj, N.: The core diseasome. Mol Biosyst 8(10), 2614–2625 (Aug 2012), http://dx.doi.org/10.1039/c2mb25230a
6. Kaufmann, E., Bernstein, A.: How useful are natural language interfaces to the semantic web for casual end-users? In: Proceedings of the Forth European Semantic Web Conference (ESWC 2007). Innsbruck, Austria (June 2007)
7. Kuchmann-Beauger, N., Aufaure, M.A.: Natural language interfaces for datawarehouses. In: 8mes journes francophones sur les Entrepts de Donnes et l'Analyse en ligne (EDA 2012), Bordeaux. RNTI, vol. B-8, pp. 83–92. Hermann, Paris (Juin 2012)
8. Kuhn, M., Campillos, M., Letunic, I., Jensen, L.J., Bork, P.: A side effect resource to capture phenotypic effects of drugs. Molecular Systems Biology 6(1) (2010), http://msb.embopress.org/content/6/1/343
9. Schmid, H.: Probabilistic part-of-speech tagging using decision trees. In: Jones, D., Somers, H. (eds.) New Methods in Language Processing Studies in Computational Linguistics (1997)
10. Tsuruoka, Y., Tateishi, Y., Kim, J.D., Ohta, T., McNaught, J., Ananiadou, S., Tsujii, J.: Developing a robust part-of-speech tagger for biomedical text. In: Proceedings of Advances in Informatics - 10th Panhellenic Conference on Informatics. pp. 382–392. LNCS 3746 (2005)
11. Wishart, D.S., Knox, C., Guo, A.C., Shrivastava, S., Hassanali, M., Stothard, P., Chang, Z., Woolsey, J.: Drugbank: a comprehensive resource for in silico drug discovery and exploration. Nucleic Acids Research 34, D668D672 (2006), database issue