CSE 6708 - Semantic Web

Assignment 2

Report on Paper Presentation

Paper Name Source Code Plagiarism Detection Method Using Protégé Built Ontologies

Samidhya Sarker
Student No. 1018052049
Group-2

September 21, 2019

Contents

List of Figures

1

Paper Summery

Extending the conclusion started in Assignment 1[1], I can say that the paper is an excellent scientific article if we think about how to present only the relevant information without any clutter.

Quoting from assignment 1[1], several tasks have been done:

1. Creating ontologies from source code manually by hand using protege.
2. Executing SPARQL queries on ontologies and compare the metrics.
3. Creating topic maps using Protege OntoGraf plugin.

The paper serves an excellent purpose of being a showcase of semantic web technologies. Also, It addresses an important issue in CS education. Then again, I can think of several shortcomings regarding this fine piece of scientific literature.

For instance the authors do not give a specific implementation, only a prototype which only performs manual owl ontology extraction and doing SPARQL query. But specific implementaions of source code parsing for source code navigation exists[3][5]. For example, github has already a *navigate*[2] feature which uses the open source sibrary *semantic*[3] to parse, analyze and compare source code. Incidentally, *semantic* is written in *Haskell* and I think functional programming languages with strong static typing is most appropriate for this task. Powerful transpilers (for example: Corrode [4]) have been written using *Haskell* and the main goal of using *Racket*[5] is to write domain specific languages.

[1] `https://semantic-web.netlify.com/report/report.pdf`
[2] `https://help.github.com/en/articles/navigating-code-on-github`
[3] `https://github.com/github/semantic`
[4] `https://github.com/jameysharp/corrode`
[5] `https://racket-lang.org/`

Another thing I can think of is that, the authors are only retrieving information from lexical analysis. Due to compiler optimization, we can say that, the source code programmer writes and the end result of a compiler toolchain has many leaps and bounds. So, a plagiarizer may escape scrutiny by changing unnecessary lexical things in source code or by subtle structural changes. On the contrary, there is a great chance of false-positives if we do not cross check the results of SPARQL query and topic maps with visual checking.

The main promising application of semantic web technologies is to create a knowledge base aiding information sciences. So, the main application of semantic technologies should be the analytics, navigation and information gathering from source codes. Although semantic indexing can aid traditional applications.

A by-product of semantic plagiarism detection will be creating a intelligent code search engine. There are a autocomplete plugin named *Kite*[6] which uses machine learning to search for code snippet or documentation. There are other tools like *Sourcegraph*[7] and *Bing C# Code Search*[8] which might use fuzzy matching and other graph search algorithms. Using semantic indexing will surely augment their search functions and provide multiplications of productivity.

In the case of plagiarization detections, Powerful utilites like **JPlag**[9], **MOSS**(Measure Of Software Similarity)[10] etc. already exists. As they are open source products, We may try to use these and add semantic elements in them rather than implementing a new system from scratch.

[6] `https://kite.com/`
[7] `https://about.sourcegraph.com/`
[8] `https://codesnippet.research.microsoft.com/`
[9] `https://jplag.ipd.kit.edu/`
[10] `http://theory.stanford.edu/~aiken/moss/`

2

Analysis of References present in Given Paper

2.1 List of References

1. M. K. Shenoy, K. C. Shet and U. D. Acharya. (2012, May). Semantic Plagiarism Detection System Using Ontology Mapping. Advanced Computing: An International Journal 3(3). [1]

2. The Protégé Ontology Editor and Knowledge Acquisition System. [2] (2013, July 1).

3. T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler and F. Yergeau. (2004, February 4). Extensible Markup Language (XML) 1.0. W3C Recommendation . Third Edition. [3]

4. F. Manola and E. Miller (2004, February 10). RDF Primer. W3C Recommendation. [4]

5. S. Harris, A. Seaborne. (2013, March 21). SPARQL 1.1 Query Language. W3C Recommendation. Available: [5]

6. J. Bao, D. Calvanese, B. C. Grau, et al. (2012, December 11). OWL 2 Web Ontology Language. W3C Recommendation. Second Edition. [6]

7. E Akin, Object Oriented Programming, Houston: Rice University Publishing House, 2001, pp. 33-34.

[1] http://airccse.org/journal/acij/papers/0512acij06.pdf
[2] http://protege.stanford.edu/
[3] http://www.w3.org/TR/2004/REC-xml20040204/
[4] http://www.w3.org/TR/2004/REC-rdfprimer-20040210/
[5] http://www.w3.org/TR/2013/RECsparql11-query-20130321/
[6] http://www.w3.org/TR/owl2-overview/

8. S. Kara, O. Alan and O. Sabuncu, "An ontology-based retrieval system using semantic indexing", Information Systems, vol. 37, no. 4, pp. 294–305, June 2012.

9. Pseudocode Standards, California Polytechnic State University Website. [7]

10. C. Liu, H. Wang, Y. Yu and L. Xu, "Towards Efficient SPARQL Query Processing on RDF Data", Tsinghua Science & Technology, vol. 15, no. 6, pp. 613–622, December 2010.

11. I. Ivan and C. Boja, Metode Statistice in analiza software. Bucharest: ASE Publishing House, 2004, pp. 218-224. Informatica Economică vol. 17, no. 3/2013

12. S. Russel and P. Norving, Artificial Intelligence: A Modern Approach (2nd edition). New Jersey: Pearson Education Inc., 2003, pp. 350-352.

13. P. Durusau, S. Newcomb and R. Barta (2007, November). Topic Maps Reference Model. International Organization for Standardization. [8]

14. D. Newman, T. Baldwin, L. Cavedon and E. Huang, "Visualizing search results and document collections using topic maps", Web Semantics: Science, Services and Agents on the World Wide Web, vol. 8, no. 2-3, pp 169–175, July 2010.

15. A. Hatzigaidas, A. Papastergiou, G. Tryfon and D. Maritsa, "Topic Map Existing Tools: A Brief Review", in Proc. The International Conference on Theory and Applications of Mathematics and Informatics, Thessaloniki, Greece, 2004, pp 185-201

2.2 Classification of Given Citation

- Reference 1 is a paper published in an international Journal.
- Reference 2 is the homepage of Protege editor.
- Reference 3 is World Wide Web Consortium (W3C) recommendation of XML.
- Reference 4 is World Wide Web Consortium (W3C) primer for RDF.
- Reference 5 is World Wide Web Consortium (W3C) recommendation for SPARQL.

[7] `http://users.csc.calpoly.edu/~jdalbey/SWE/pdl_std.html(2013,July1)`
[8] `http://www.isotopicmaps.org/TMRM/TMRM-7.0/tmrm7.pdf`

- Reference 6 is Wide Web Consortium (W3C) recommendation for OWL.
- Reference 7 is a book on OOP.
- Reference 8 is a Masters thesis from Middle East University (Turkey) on semantic indexing.
- Reference 9 is a dead link to a standard on pseudocode writing. Currently can be found on scribd [9].
- Reference 10 is an IEEE paper on SPARQL Query processing.
- Reference 11 is an book on Statistical methods in software analysis written in Romanian.
- Reference 12 is the most widespread book on AI.
- Reference 13 is the ISO reference model for topic maps.
- Reference 14 is a article from Elsevier on how to visualize data using topic maps.
- Reference 15 is an paper published in an international conference on tools required to implement topic maps.

2.2.1 Comment

As we can see that references 3-6, 13 are specifications from W3C, ISO. Reference 2 is a website. And References 11, 12 are excerpts from books. So, we shall analyze refences 1, 8, 10, 14, 15.

2.3 Reference 1: Semantic Plagiarism Detection System Using Ontology Mapping

In this paper, the authors discuss the applications of using Semantic web ontologies to detect plagiarism in text documents. The proposal is to train an ontology learner by feeding an ontology mapping algorithm (a) Main copies and (b) Probable plagiarizations.

The authors also experimented with a prototype TAO (Transitioning Applications to Ontologies) project but the implementation details are not given.

[9] `https://www.scribd.com/document/47856615/PSEUDOCODE-STANDARD`

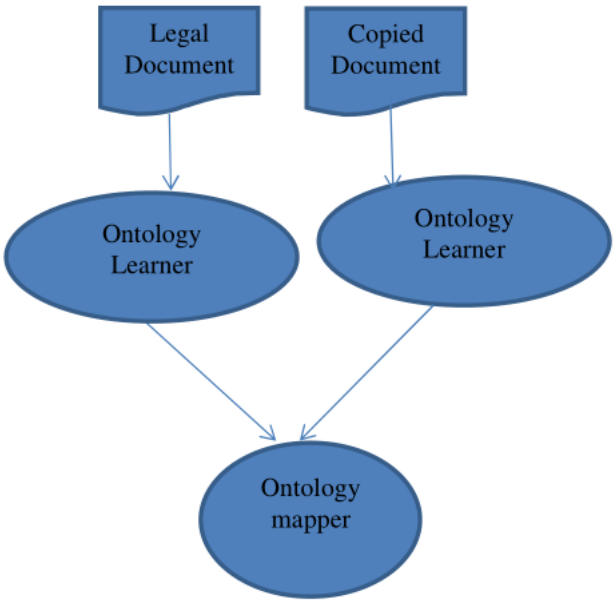

Figure 2.1: Architecture of the proposed detection system.

2.4 Reference 8: An ontology-based retrieval system using semantic indexing

This Masters thesis, can be divided into three parts. In chapter 2, Background information, the authors discuss general information science with an emphasis on information retrieval (IR). They also discuss optimizing the information retrieval or searching process implementing indexing, ranking etc. They also discusss evaluation metrices. Finally the concept of Semantic Web is introduced and the application to improve IR is discussed.

In chapter 3, various approaches including traditional, semantic approaches are discussed. The authors discuss semantic indexing and their proposed models.

In chapter 4, the authors conceptualize their own semantic retrieval process.

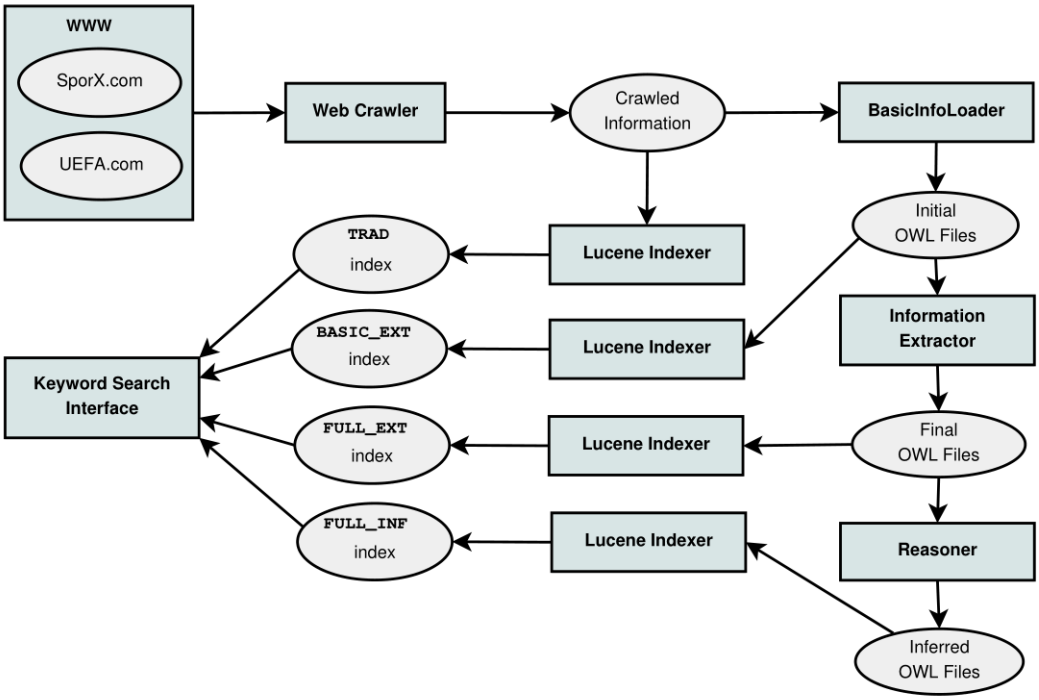

Figure 2.2: System design of the proposed semantic retrieval system system.

So, in this system, information is gathered from `www.uefa.com` and `www.sporx.com` which are not semantically readable. These information are then fed into Lucene Indexer and owl ontologies are formed from the gathered xml. In this way, both a indexed search engine and semantically vaiable Knowledge base is created which are interlinked. Semantic ruleset is created and a reasoner is used. So, information can be looked up efficently and accurately.

2.5 Reference 10: Towards Efficient SPARQL Query Processing on RDF Data

In this original research paper, the authors talk about SPARQL query optimiztion. The main claims of the authors are, in some RDF data storage systems (eg. APACHE Jena, SOR), SPARQL queries are often translated into SQL statements. But as SPARQL triples does not translate well enough into SQL statements, many self-joins and table joins are required which causes slow-downs.

To properly index RDF/SQL The authors propose:

1. Give each RDF row their own ID. More like whone PostgreSQL stores JSON data like document objects.

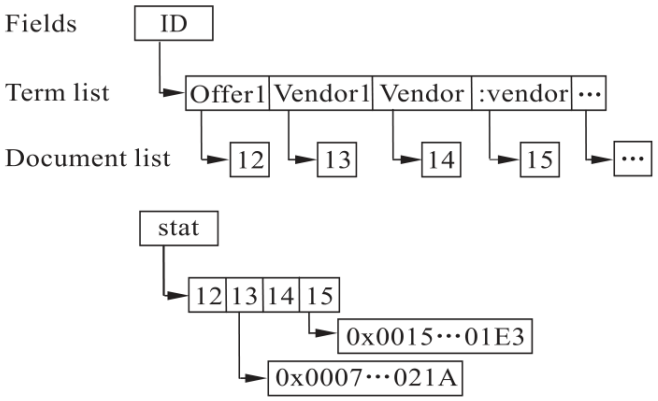

Figure 2.3: Assigning ID to SPARQL Triples.

2. Index RDF triple. The authors give some complex schemes.

3. Optimize query execution and convert SPARQL RDFs into a Tree.

2.6 Reference 14: Visualizing search results and document collections using topic maps

In this descriptive journal paper, the authors discuss how topic maps can be of use on visualizing information gathered by conducting search.

Also, the authors discuss how to use topic models to mine data for unsupervised learning. Latent Dirichlet Allocations ie. Topic Maps can create probabilistic models for creating classification of text documents.

2.7 Reference 15: Topic Map Existing Tools: A Brief Review

This is also a descriptive article regarding topic maps. Topics include:

1. What is a topic map? How are different entities interconnected?
2. What software to use? How can they be compared?
    - XML Topic Maps [1]
    - TOLOG
    - Topic Map Engines:
        (a) GooseWorks
        (b) Omnigator by Ontopia
        (c) SemanText
        (d) empolis k42
        (e) Tm4Jscript
        (f) tmproc
        (g) tinyTIM
        (h) TM4J
        (i) xSiteable
    - Topic Map Navigator - Visualizers:
        (a) HyperGraph
        (b) Panckoucke
        (c) The 'V' Topic Map Browser
        (d) ThinkGraph
        (e) TM3D
        (f) TM4Web/Velocity
        (g) UNIVIT
        (h) TMNAV
    - Topic Map Editors:
        (a) AsTMa
        (b) LTM
        (c) mapalizer
        (d) Simple Topic Maps Management
        (e) Topic Map Designer
        (f) TMTab

3

Analyis of Citations of Given Paper

Using google scholar[1] I have been able to find two scientific articles that cite the given paper as a reference.

3.1 List of Citations

1. Tim Breedveld (July 10, 2018): Finding Similar Software with Ontology Matching Techniques Master Thesis, Utrecht University

2. Diana Kocanjer, Nikola Kadoić (April, 18 - 22 2016, GV-Global Virtual Conference): Raising students awareness about ethical behavior.

3. Oscar Karnalim (December 2017): A Low-Level Structure-based Approach for Detecting Source Code Plagiarism. IAENG International Journal of Computer Science

4. Claudiu Epure, Adrian Iftene (2016): Analysis of Source Code in Object A Case Study for C#. Romanian Journal of Human-Computer Interaction

5. Feixiang Xu, Xinhui Liu and Chen Zhou (Published: 4 May 2018): Developing an Ontology-Based Rollover Monitoring and Decision Support System for Engineering Vehicles.

6. Vilarino Ribeiro, L. (2015). Gestión de Conocimiento en el Diseño e Implementación de Modelos de Capacidades en Ciencias de la Empresa en escenario EEES.

[1] `scholar.google.com`

7. Xu, Fei-xiang, et al. "An Ontology and AHP Based Quality Evaluation Approach for Reuse Parts of End-of-Life Construction Machinery." Mathematical Problems in Engineering 2018 (2018).

8. Wang, Yujun, et al. "Developing an ontology-based cold chain logistics monitoring and decision system." Journal of Sensors 2015 (2015).

3.1.1 Comment

Citations 6 is written in Spanish and 5, 7-8 are the application of semantic web technologies on mechanical engineering related topics. We shall discuss citations other than these.

3.2 Citation 1: Finding Similar Software with Ontology Matching Techniques

In this masters thesis, the author tried to implement a prototype for using semantic web technologies for software testing. In the end the author created ontologies from source code and use semantic matchers to find similiarities between them. The plan is to use these similiarities to generate automated unit tests. The project is essentially a plagiarism checker for Java.

The project works in several layers. The specific matching framework the author is using is AgreementMakerLight (AML).

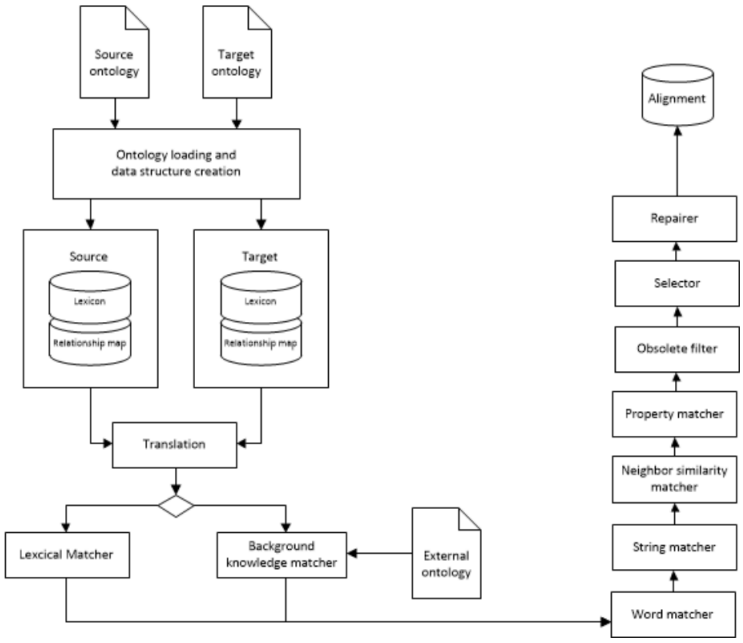

Figure 3.1: AML alignment process.

So, to align two ontologies, we first extract the ontology from source using Apache Jena or Spoon. We can extract AST from java classes using these and onwards OWL. After that, we insert these both ontologies into AML and it will give a alignment metric.

3.3 Citation 2: Raising students awareness about ethical behavior

This paper is mainly philosophic in nature as it consults with morality and ethics regarding cheating. Despite conducting surveys the conclusion of the paper is rather subjective as the authors link cheating with the culture and economy of their home country of Croatia.

3.4 Citation 3: A Low-Level Structure-based Approach for Detecting Source Code Plagiarism

In this paper, the authors is addressing an important problem of our paper. In our paper, plagiarism detection is attribute or text based. The author is proposing a Structure-based approach. Although the author is calling it 'Low-Level', but actually he is doing a java bytecode level comparison already developed by himself in another paper[4]

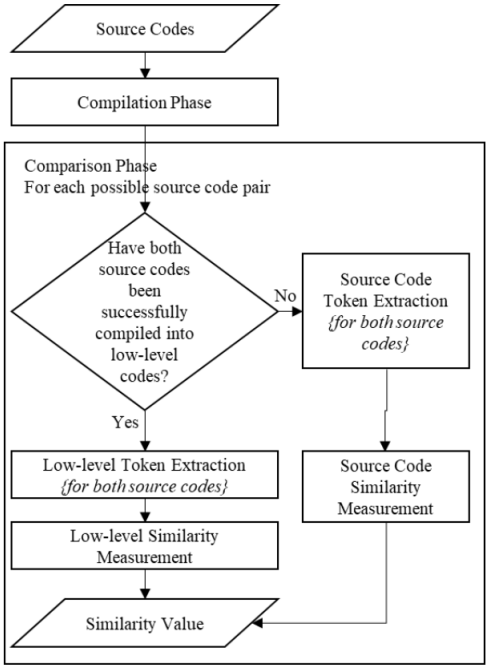

Figure 3.2: Flowchart for "low level" plagiarism detection.

From the flow-chart, we can see that there is an extra step of token extraction and token extraction phase. We also check similiarities between tokens. So it's more like both high-level and low-level plagiarism detection.

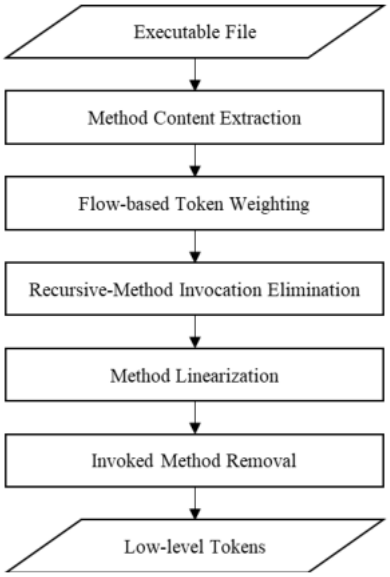

Figure 3.3: Low level token exraction.

Also if we break down the low level token extraction phase, we can see there are several key elements:

Flow-based Token Weighting There is a flow-based token weighting which depends on basic branch prediction. So that unnecessary branching can be ignored.

Method Content Extraction It is extracting low level method information from java bytecode.

Recursive-Method Invocation Elimination All method calls are converted to directed graoh and we find out strongly connected components and remove them.

The problem is that, not all programming languages have JIT compilers. A low level language that compiles directly to machine code or assembly will not be available to be used. High level languages that uses interpreters will also not be able to use these. LLVM based languages or Languages may be able to avail of this offer.

3.5 Citation 4: Semantic Analysis of Source Code in Object A Case Study for C#

This paper is in a unique position as It is similiar to our paper but the authors used an existing antology (SCRO)[2] and actually implemented the parser[2].

In addition, the authors go several steps ahead of our paper. One is, instead of manually writing SPARQL Query, they implement a NLP (Natural Language Processing) toolkit which processes human readable questions into SPARQL Query.

Another is, instead of storing source code information in OWL or XML/RDF format, the authors have created a knowledge base center to host its own triple store for easy information retrieval.

And to wrap things up, they have implemented a IIS server with API endpoints to perform query.

[2] `https://github.com/mdesalvo/RDFSharp`

4

Future Work

Extending the future work section discussed in Assignment 1[1], we can see that the authors have specified several further probable developments themselves:

1. Create a parser generator so that we can automatically extract ontologies from various sources.

   - The authors did not specify any single toolsets for this purpose. I have tried to use:
     (a) ANTLR for C and Javascript.
     (b) Flex/Bison for C.
     (c) CodeOntology[2] [3]. of Java.

     Although I now think a programming language with static typing like *Haskell* or softly typed like *Typed Racket* would have been better suited.

2. Create a automated web crawler. It will automatically parse the code and extract the OWL ontology.

   - Although the authors do no specify anything, the most appropriate place of a crawler would be online Disctributed version controlled services. A crawler would run in the servers gather semantic information about the source hosted in the server and there should be an API endpoints for conducting semantic searches. Semantic information should augment traditional indexing based search methods. VCS behemoths like Github[4] and Atlassian Bit-

[1] `https://semantic-web.netlify.com/report/report.pdf`
[2] `http://codeontology.org/`
[3] `https://github.com/codeontology/parser`
[4] `https://help.github.com/en/articles/navigating-code-on-github`

bucket [5] has already started to use semantic technologies to augment the search function.

- Also there are self hosted open source VCSs like Cgit[6] and Gitea[7] where we can implement our own crawlers.

3. Defined metric generator. The program user may define a set of metrics that will be used to measure plagiarism. This set of metrics can be created automatically by using applying Machine Learning or other software development techniques.
   This constitutes two things.

   (a) Augmenting the parser generator and web crawler so that, it knows which semantic entity to search for when parsing source code

   (b) Dynamically generate SPARQL query.

We can extend these goals.

- The authors only showed the creation of ontologies from C and Javascript. We can create ontologies from different lanugages from different paradigms.
- The use of **prolog** to make a better reasoner than reasoners provided by Protege can be done. The application of logic programming will extend the Plagiarism detectors capabilities. Ontopia omnigator provides a language called tolog [8] for this specific purpose.
- Usage of JSON-LDs. Using JSON datatypes rather than XML/RDF can will greatly extend the capability of our application's interface with modern API paradigms like GraphQL.

Doing the above things will allow us to create a automated plagiarism checker which can instantanously tell us while we are writing code the related source files. So, it shall not only be used for plagiarism detection but also for intelligent and dynamic code search.

[5] `https://bitbucket.org/blog/introducing-code-aware-search-for-bitbucket-cloud`
[6] `https://git.zx2c4.com/cgit/`
[7] `https://gitea.io/en-us/`
[8] `https://ontopia.net/omnigator/docs/query/tutorial.html`

Bibliography

[1] XML Topic Maps (xtm) 1.0. `http://www.topicmaps.org/xtm/`, 2001. Accessed: 2019-09-21.

[2] A Alnusair and T Zhao. Source code representation ontology (scro). *Design Pattern Ontologies, and Framework Ontologies*, 2012.

[3] Mattia Atzeni and Maurizio Atzori. Codeontology: Rdf-ization of source code. In *International Semantic Web Conference*, pages 20–28. Springer, 2017.

[4] Oscar Karnalim. Detecting source code plagiarism on introductory programming course assignments using a bytecode approach. In *2016 International Conference on Information & Communication Technology and Systems (ICTS)*, pages 63–68. IEEE, 2016.

[5] Christian Zimmer and Axel Rauschmayer. Tuna: ontology-based source code navigation and annotation. In *Workshop Ontologies as Software Engineering Artifacts in OOSPLA*, 2004.