Analyzing and Expanding NPDS Biomedical Record Databases via a Semantic Search Engine

Shiladitya Dutta and Carl Taswell MD, PhD

***Abstract*—CoVaSEA (Concept-Validating Search Engine Agent) is an automated semantic search engine that is interoperable with the Nexus-PORTAL-DOORS System(NPDS). Interoperable with NPDS, CoVaSEA serves to apply SPARQL-based search to external lexical databases by converting relevant biomedical resources to a linked data format which can be searched by the query engine. CoVaSEA consists of 3 components: the federated query engine searching, the web-crawler expanding, and the quadstore storing. Acting in concert, these can provide a composite open-web SPARQL search utility. For researchers, this opens the possibility to search external biomedical databases via a semantic web approach and is a solution for situations in which SPARQL queries need to be applied to non-linked data resources.**

***Index Terms*—Nexus-PORTAL-DOORS System, CoVaSEA, SPARQL, web-crawler, semantic web**

Introduction

The Nexus-PORTAL-DOORS System (NPDS) is a software system that provides the capability to publish both semantic and lexical resources regarding specific target areas. NPDS has inbuilt REST API services via the Scribe Registrar along with a standardized messaging protocol, enabling exchange among client applications and servers. The structure of the Nexus PORTAL-DOORS System is sub-divided according to the Hierarchically Distributed Mobile Metadata architectural style[1] into 3 primary parts: Nexus, PORTAL, and DOORS. The PORTAL registry controls entities that have unique labels and their associated lexical meta-data. The DOORS directory contains the semantic meta-data which is primarily comes in the form of RDF descriptions. The Nexus diristry (an aggregation of the terms DIRectory and regISTRY) is a single server that serves as a combination of the PORTAL registry and DOORS directory.[2] Semantic descriptions are a prerequisite for many of the goals laid out for the Nexus PORTAL-DOORS System[3], however it is labor-intensive to manually annotate resources from the open-web. To remedy this, we have created CoVaSEA(Concept-Validating Search Engine Agent): an automated web crawler/query engine that can search and expand NPDS metadata records. With this system, users can apply semantic search to external biomedical resources, which may be valuable in a variety of tasks which benefit from a combination of a web-crawler's capability to search the open web with the versatility of a semantic query engine. .

Overview

As an expansion to the past section[4], CoVaSEA integrates several features which benefit NPDS including: **(A)** An implementation of SPARQL query based semantic search to allow retrieval and manipulation of linked data descriptions **(B)** Targeted web-crawling for relevant articles from external biomedical literature databases to expand NPDS records **(C)** Automated translation of free-form text abstracts into RDF triples to derive the semantic representations of lexical data. First the user inputs a Natural Language Query(NLQ) and a SPARQL query, which also has the option to be constructed for the user, into the CLI. These are passed to the interface component which calls the webcrawler. The webcrawler combs through the various databases and retrieves the raw text and the metadata which is passed to the natural language parser. On each article, the natural language parser transforms the text and metadata into semantic triples, which are stored in a local quadstore or DOORS directory. Then the interface components calls the SPARQL query engine. The SPARQL query engine searches through the local quadstore and DOORS directory and returns the requested results to the user. In terms of the structure of the program itself, the web crawler/search engine program consists of a number of interdependent components. Each of these components fulfills a general purpose. The methods are:

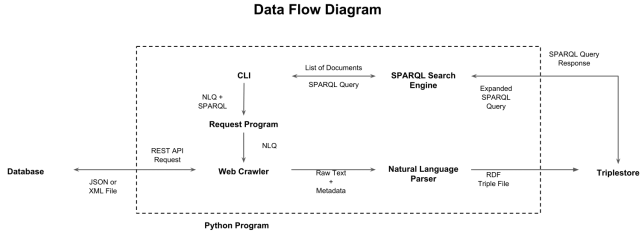


Fig. 1. Data flow diagram of the project and all of its included components

Document created June 20, 2018, revised September 30, 2018; typeset November 1, 2018.

- **Web Crawler** : Crawls through a multitude of sites in order to retrieve data. Currently the system can search through the article records on DOAJ, PubMed, Elsevier ScienceDirect, and CORE[5]. The webcrawler is similar in structure to Seung-Hong Bae. The webcrawler receives a requested query and then it searches through the available databases via REST API. The general method for searching through the databases is first a general lexical search query via the provided natural language query. The database then returns a number of articles. From there, the crawler searches through the list of returned articles and retrieves the basic-metadata from each one. It then passes the basic meta-data to the natural language parser. The basic meta-data constitutes of the abstract, title, author(s), database of origin, DOI(if available), and date of publication. The abstract will have its triples extracted and the rest will be included in the named graph of the article via the DublinCore ontology[6].
- **Natural Language Parser**: Receives the raw text from the web crawler and then parses it into logical form triples. These are put into RDF files and stored.

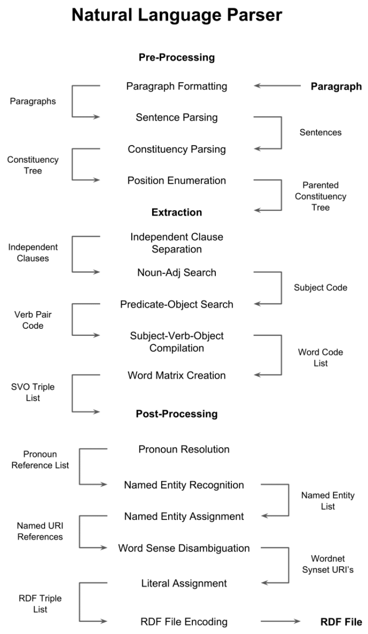


Fig. 2. Data flow diagram of the Natural Language to RDF parser

  - Pre-processing: Designed in order to pre-process the raw text to prepare it for parsing and also to perform tasks that depend on the raw texts such as Named Entity Recognition[7] and Co-Referencing[8]. Pre-processing preparation includes removing troublesome characters, re-spacing the text to ensure homogeneity, and parsing the paragraphs into sentences.
  - Extraction: Uses a constituency parse to extract relevant logical form triples. First independent clauses are separated from each other and a constituency parse occurs on each independent clause. Breadth-first and depth-first search are used to find subject and verb-object phrases. The breadth-first search is used to find the highest noun phrase in the tree. This noun phrase is then split up via conjunctions and any proximal adjectives are linked to the nouns. Then a two stage a breadth-first and depth-first search is used to find the verb. Breadth-first search is used to find the highest verb phrase and depth-first search is used to separate verb from the prepositional or adverb sections. The verbs and noun phrases are then broken into their individual parts and tagged with positions.
  - Post-processing: This step translates the logical form triples into RDF triples. The step primarily consists of graph compression. It does this by assigning URI's to each part of the logical form triples. First field-specific language and some named entities are assigned URI's via databases. Then standard words are assigned via WordNet[9] synsets using context and part of speech via Lesk[10]. This step aids in graph-compression and also allows for the full semantic meaning to be captured rather than just lexical meaning. Finally named entities and number are assigned to literals. These are all then packaged into a named graph.

- **SPARQL Analysis Engine**: Receives SPARQL query from user and then searches through the triple store to extract relevant data for the user. For a more conducive user experience, the query engine supports a SPARQL builder, thus circumventing the need to know SPARQL syntax. The system works by having the user create a series of either filtering, mandatory, or optional conditions in either an independent or nested connection. Thus the user can build a search query via simple fill-in-the-blank statements. Though it cannot replicate the full power of SPARQL syntax, it is a potent resource for users cannot use SPARQL. With the constructed or inputted query, CoVaSEA searches through the linked database. Due to the dynamic and heterogeneous nature of the data that CoVaSEA handles, the SPARQL query engine divides up the queries into sub-queries such that only the articles that are needed are compiled. This approach avoids the necessity of having to compile a computationally expensive local graph. Since the system is based off of named graphs, it is relatively easy to trace the article of origin of a particular triple if necessary. This iteration based approach helps to optimize runtime and memory-load on computers by eliminating the redundancy of loading unneeded graphs.

- **QuadStore**: To prevent redundant re-rendering of articles, CoVaSEA records the triples which are built by the web-crawler in either a local quadstore or a DOORS directory. Each of the articles in the records has its own named graph, allowing for the identification of the origin of triples. The graph is divided into two sections: the meta-data section and the semantic representation section. The meta-data section stores key information about the article such as author, title, publication date, and database of origin. The semantic representation section stores a set of triples which are a RDF portrayal of the article's abstract. The RDF files are sent and retrieved from the DOORS directory utilizing the Scribe API.

Results and Discussion

The primary point of possible expansion for the system is the Natural-Language Parser. Currently, the parser faces difficulty in 3 main areas: relevance of triples, accuracy of triples, and runtime. The first problem is the relevance of triples and how well they summarize the passages. There are several methods from which we can take inspiration from[11][12]. Most of these focus on document summarization by using a combination of the semantic graph structure of the document and pattern matching. The second issue is that the system cannot accurately parse advanced sentence structures. The best way to remedy this is to use a semantic frame solution such as FrameNet[13] or Boxer[14]. The final problem is that of optimization. The runtime for the parser in its current state is time-consuming in that it takes 30 seconds to parse a paragraph. This impedes us from performing whole articles parses with reasonable runtimes. The best way to solve this is to either use more advanced hardware or optimize the algorithms. Another section of improvement is the SPARQL query engine. SPARQL query expansion should be implemented to increase the breadth of SPARQL searches and to implement a reasoning engine such as RACER[15].

Conclusion

Here we presented a system in which resources from the open-web can be translated into machine-understandable semantic information and be searched via SPARQL. CoVaSEA has the capability to both search ”externally” with the web crawler for semantic data to expand the NPDS knowledge base and ”internally” with SPARQL to provide a method to navigate the data inside the DOORS directory. With the distinct advantage that the system is automated, thus can furnish large amounts semantic descriptions on a regular basis, CoVaSEA lays the groundwork for a variety of future NPDS applications for which linked data stores are a necessity along with providing a method to semantically search external biomedical literature databases.

Acknowledgment

I give my thanks to the other members of the Brain Health Alliance 2018 program for their assistance on technical and methodological points.

References

[1] C. Taswell, “A distributed infrastructure for metadata about metadata: The hdmm architectural style and portal-doors system,” *Future Internet*, vol. 2, no. 2, pp. 156–189, 2010.

[2] ——, “Doors to the semantic web and grid with a portal for biomedical computing,” *IEEE Transactions on Information Technology in Biomedicine*, vol. 12, no. 2, pp. 191–204, 2008.

[3] C. Taswell, G. TeleGenetics, and C. Ladera Ranch, “Portal-doors infrastructure system for translational biomedical informatics on the semantic web and grid,” *Proceedings of the American Medical Informatics Association Summit on Translational Bioinformatics, San Francisco, CA*, p. 43, 2008.

[4] S.-H. Bae, A. G. Craig, C. Taswell, *et al.*, “Expanding nexus diristries of dementia literature with the npds concept-validating search engine agent,”

[5] P. Knoth and Z. Zdrahal, “Core: Three access levels to underpin open access,” *D-Lib Magazine*, vol. 18, no. 11/12, 2012.

[6] S. Weibel, “The dublin core: A simple content description model for electronic resources,” *Bulletin of the American Society for Information Science and Technology*, vol. 24, no. 1, pp. 9–11, 1997.

[7] J. R. Finkel, T. Grenager, and C. Manning, “Incorporating non-local information into information extraction systems by gibbs sampling,” in *Proceedings of the 43rd annual meeting on association for computational linguistics*, Association for Computational Linguistics, 2005, pp. 363–370.

[8] K. Clark and C. D. Manning, “Entity-centric coreference resolution with model stacking,” in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, vol. 1, 2015, pp. 1405–1415.

[9] G. A. Miller, “Wordnet:a lexical database for english,” *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[10] S. Banerjee and T. Pedersen, “An adapted lesk algorithm for word sense disambiguation using wordnet,” in *International conference on intelligent text processing and computational linguistics*, Springer, 2002, pp. 136–145.

[11] J. Leskovec, M. Grobelnik, and N. Milic-Frayling, “Learning sub-structures of document semantic graphs for document summarization,” 2004.

[12] Y. Shang, Y. Li, H. Lin, *et al.*, “Enhancing biomedical text summarization using semantic relation extraction,” *PLoS one*, vol. 6, no. 8, e23862, 2011.

[13] C. F. Baker, C. J. Fillmore, and J. B. Lowe, “The berkeley framenet project,” in *Proceedings of the 17th international conference on Computational linguistics-Volume 1*, Association for Computational Linguistics, 1998, pp. 86–90.

[14] J. Bos, “Wide-coverage semantic analysis with boxer,” in *Proceedings of the 2008 Conference on Semantics in Text Processing*, Association for Computational Linguistics, 2008, pp. 277–286.

[15] V. Haarslev and R. Möller, “Racer: A core inference engine for the semantic web.,” in *EON*, vol. 87, 2003.

Analyzing and Expanding NPDS Biomedical Record Databases via a Semantic Search Engine

Shiladitya Dutta, Adam G. Craig, and Carl Taswell MD, PhD

***Abstract*—CoVaSEA (Concept-Validating Search Engine Agent) is an automated semantic search engine that is interoperable with the Nexus-PORTAL-DOORS System(NPDS). Utilizing NPDS, CoVaSEA serves to apply SPARQL-based search to external lexical databases by converting relevant biomedical resources to a linked data format which can be searched by the query engine. CoVaSEA consists of 3 components: the SPARQL query engine, the web-crawler, and the quadstore. Acting in concert, these can provide a composite open-web SPARQL search utility. For researchers, this opens the possibility to search external biomedical databases via a semantic web approach and is a solution for situations in which SPARQL queries need to be applied to non-linked data resources.**

***Index Terms*—Nexus-PORTAL-DOORS System, CoVaSEA, SPARQL, web-crawler, semantic web**

Introduction

The Nexus-PORTAL-DOORS System (NPDS)[1] is a system that provides the capability to publish both semantic and lexical resources regarding specific target areas. NPDS has inbuilt REST API services via the Scribe Registrar[2] along with a standardized messaging protocol, enabling exchange among client applications and servers. The structure of the Nexus PORTAL-DOORS System is sub-divided according to the Hierarchically Distributed Mobile Metadata architectural style into three parts: Nexus, PORTAL, and DOORS [3]. The PORTAL registry controls entities that have unique labels and their associated lexical meta-data. The DOORS directory contains semantic meta-data which is primarily comes in the form of RDF descriptions. The Nexus diristry (an aggregation of the terms DIRectory and regISTRY) is a single server that serves as a combination of the PORTAL registry and DOORS directory.

Integrated with NPDS, CoVaSEA (Concept-Validating Search Engine Agent) aims to allows users to apply a SPARQL-query based search to external biomedical literature databases. This approach may be valuable in a variety of tasks which benefit from a combination of a web-crawler's capability to search the open web with the versatility of a semantic query engine. CoVaSEA uses NPDS DOORS directories as a storage basis for the records that it has retrieved, which means that it can also serve as a utility to search and expand NPDS metadata records.

Overview

Acting as a successor to the past CoVaSEA [4], CoVaSEA integrates several features which augment the utility of NPDS including: **(A)** An implementation of SPARQL query based semantic search to allow retrieval and manipulation of linked data descriptions **(B)** Targeted web-crawling for relevant articles from external biomedical literature databases to expand NPDS records **(C)** Automated translation of free-form text abstracts into RDF triples to derive the semantic representations of lexical data. First, the user inputs a Natural Language Query (NLQ) and a SPARQL query, which also has the option to be constructed via a SPARQL builder form, into the CLI. These are passed to the webcrawler which combs through the various databases and retrieves the abstract text and metadata for a multitude of articles. On each article, the natural language parser transforms the text and metadata into semantic triples, which are stored in a local quadstore or DOORS directory. These are searched with the provided or constructed SPARQL query and the results are returned to the user. CoVaSEA consists of a number of interdependent components, each of which fulfills a general purpose. The methods are:

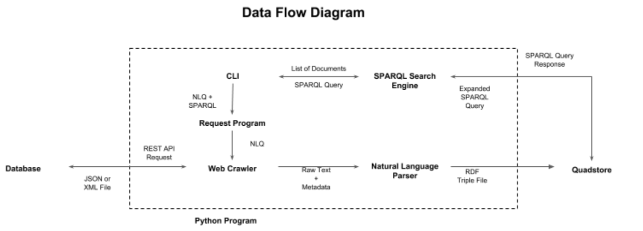


Fig. 1. Data flow diagram of CoVaSEA

- **Web Crawler**: The webcrawler combs through literature databases in order to retrieve relevant articles. Currently the system can search through the literature records on DOAJ, PubMed, Elsevier ScienceDirect, and CORE[5]. Using provided REST API

Document created June 20, 2018, revised September 30, 2018; typeset November 4, 2018.

services, the crawler obtains a list of relevant articles via a general lexical search. From there, the basic meta-data, consisting of the abstract, title, author(s), database of origin, DOI(if available), and date of publication, from each article is retrieved and passed to the natural language parser. An RDF representation of the abstract will be constructed and the rest of the meta-data will be included in the named graph of the article via the DublinCore ontology[6].

- **Natural Language Parser**: The Natural Language Parser receives the raw abstract text from the web crawler and parses it into logical form triples. This section utilizes the Stanford Core NLP library[7] and NLTK [8].

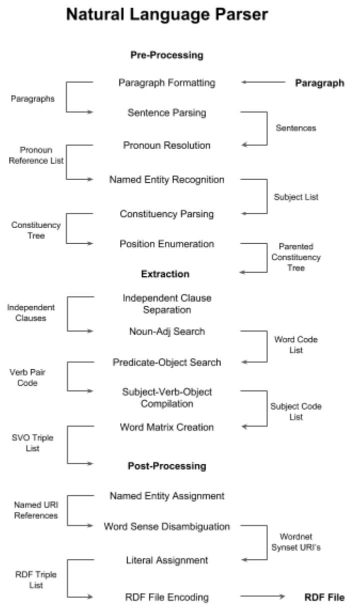


Fig. 2. Data flow diagram of the Natural Language to RDF parser

  - Pre-processing: A step in which the raw text is prepared for parsing and to perform tasks that depend on the text itself such as Named Entity Recognition[9] and Co-Referencing[10]. Preprocessing preparation includes removing troublesome characters, re-spacing the text to ensure homogeneity, and parsing the paragraphs into sentences.
  - Extraction: Independent clauses are separated from each other and a constituency parse occurs on each independent clause. Breadth-first and depth-first search are used to find subject and verb-object phrases. Breadth-first search is used to find the highest noun phrase in the tree. This noun phrase is split up if there are conjunctions and any adjectives are linked to the nouns. Then a two stage breadth-first and depth-first search is used to find the verb. Breadth-first search is used to find the highest verb phrase and depth-first search is used to separate the verb from the prepositional or adverb sections. The verbs and noun phrases are then tagged with positions and put into triples.
  - Post-processing: Graph Compression is performed and the logical form triples are translated into RDF triples. Unique Resource Identifiers(URI) are assigned to each part of the triples. First field-specific language and some named entities are assigned URI's via databases such as MeSH. Then standard words are assigned via WordNet[11] synsets based on context and part of speech using Lesk[12]. Finally named entities and numbers are assigned to literals. These steps aid in graph-compression and also allows for the full semantic meaning to be captured rather than just the lexical data. The triples are all packaged into the article's named graph.

- **SPARQL Analysis Engine**: A SPARQL query is received and then used to search through the triple store to extract relevant data for the user. For a more conducive user experience, the query engine supports a SPARQL builder, thus circumventing the need to know SPARQL syntax. The system works by having the user create a series of either filtering, mandatory, or optional conditions in either an independent or nested connection. Thus the user can build a search query via simple fill-in-the-blank statements. Though it cannot replicate the full power of SPARQL syntax, it is a potent resource for users do not want to use SPARQL. Due to the dynamic and heterogeneous nature of the data that CoVaSEA handles, the SPARQL query engine divides up the queries into sub-queries such that only the articles that are needed are compiled. This federated approach avoids the necessity of having to compile a computationally expensive local graph. Since the system is based off of named graphs, it is relatively easy to trace the article of origin of a particular triple if necessary. This iteration based approach helps to optimize runtime and memory-load on computers by eliminating the redundancy of loading unneeded graphs.
- **QuadStore**: To prevent redundant re-rendering of articles, CoVaSEA records the triples which are built by the web-crawler in either a local quadstore or a DOORS directory. Each of the articles in the record has its own named graph, allowing for the identification of the origin of triples. The graph is divided into two sections: the meta-data section and the semantic representation section. The meta-data section stores key information about the article such as author, title, publication date, and database of origin. The semantic representation section stores a set of triples which are a RDF portrayal of the article's

abstract. The RDF files are sent and retrieved from the DOORS directory utilizing the Scribe API.

Results

CoVaSEA currently has a run time for a full search of 245.12 seconds for 20 articles and 1307.42 seconds for a 100 articles. The breakdown of the runtime for 20 articles is 102.32 seconds for the web-crawler to retrieve 20 articles from the 4 databases, 139.47 seconds to parse the abstracts into RDF files, and 3.21 seconds for the SPARQL query search. The breakdown for a 100 articles is 483.13 seconds for the web-crawler, 721.47 seconds for parser, and 102.82 seconds for the SPARQL query search. Note that all tests are run on a computer with 16 GB of RAM and a i7-8750H. The connection speeds are a download of 18.40 Mgbs and an upload of 1.68 Mgbs. It should be noted that most of the articles retrieved had relatively short abstracts of 1 paragraph.

Discussion

CoVaSEA offers capabilities that can facilitate a variety of tasks. Primarily, the role of CoVaSEA is a utility to apply a SPARQL-query based search to external biomedical literature databases. For researchers, this may be useful in situations for which the versatility and adaptability that is offered by a SPARQL query needs to be applied to lexical data on the open-web. However, CoVaSEA is not just limited to serving the user, but also, due to its relationship with NPDS, is a utility to search and expand semantic records on the DOORS directory. Since system is automated, and can furnish large amounts of semantic descriptions on a regular basis, CoVaSEA lays the groundwork for a variety of future NPDS applications(i.e. automated meta-analysis).

The primary point of possible expansion for CoVaSEA is the Natural-Language Parser. Currently, the parser faces difficulty in 3 main areas: relevance of triples, accuracy of triples, and runtime. The first problem is the relevance of triples and how well they summarize the passages. The second issue is the accurate parsing of advanced sentence structures. The final problem is that of optimization, which impedes us from performing whole articles parses with reasonable run-times. Another section of improvement is the SPARQL query engine. The two biggest improvements that could be made are SPARQL query expansion and a semantic reasoning engine, both of which increase the breadth of the search.

Conclusion

Here we presented CoVaSEA: a system in which resources from the open-web can be translated into machine-understandable semantic information and be searched via SPARQL. CoVaSEA has the capability to both search externally with the web crawler for semantic data to expand the NPDS knowledge base and internally with SPARQL to navigate the data inside the DOORS directory. These capabilities combine to offer a utility which can semantically search external biomedical literature databases. The application of the wide range of possibilities offered by semantic search to the array of lexical information present on the open-web renders many services, both for users and for the Nexus PORTAL-DOORS System.

References

[1] C. Taswell, G. TeleGenetics, and C. Ladera Ranch, “Portal-doors infrastructure system for translational biomedical informatics on the semantic web and grid,” *Proceedings of the American Medical Informatics Association Summit on Translational Bioinformatics, San Francisco, CA*, p. 43, 2008.

[2] A. Craig, S. H. Bae, T. Veeramacheneni, *et al.*, “Web service apis for scribe registrars, nexus diristries, portal registries and doors directories in the npd system.,” in *SWAT4LS*, 2016.

[3] C. Taswell, “A distributed infrastructure for metadata about metadata: The hdmm architectural style and portal-doors system,” *Future Internet*, vol. 2, no. 2, pp. 156–189, 2010.

[4] S.-H. Bae, A. G. Craig, C. Taswell, *et al.*, “Expanding nexus diristries of dementia literature with the npds concept-validating search engine agent,”

[5] P. Knoth and Z. Zdrahal, “Core: Three access levels to underpin open access,” *D-Lib Magazine*, vol. 18, no. 11/12, 2012.

[6] S. Weibel, “The dublin core: A simple content description model for electronic resources,” *Bulletin of the American Society for Information Science and Technology*, vol. 24, no. 1, pp. 9–11, 1997.

[7] C. Manning, M. Surdeanu, J. Bauer, *et al.*, “The stanford corenlp natural language processing toolkit,” in *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, 2014, pp. 55–60.

[8] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit*. ” O’Reilly Media, Inc.”, 2009.

[9] J. R. Finkel, T. Grenager, and C. Manning, “Incorporating non-local information into information extraction systems by gibbs sampling,” in *Proceedings of the 43rd annual meeting on association for computational linguistics*, Association for Computational Linguistics, 2005, pp. 363–370.

[10] K. Clark and C. D. Manning, “Entity-centric coreference resolution with model stacking,” in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, vol. 1, 2015, pp. 1405–1415.

[11] G. A. Miller, “Wordnet:a lexical database for english,” *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[12] S. Banerjee and T. Pedersen, “An adapted lesk algorithm for word sense disambiguation using wordnet,” in *International conference on intelligent text processing and computational linguistics*, Springer, 2002, pp. 136–145.

SPARQL-Based Search Engine and Agent for Finding Biomedical Literature and Converting References to NPDS Metadata Records

Shiladitya Dutta, Adam G. Craig, and Carl Taswell

***Abstract*—CoVaSEA (Concept-Validating Search Engine Agent) has been developed as a semantic search engine with automated search agents that interoperate with the Nexus-PORTAL-DOORS System (NPDS). CoVaSEA consists of 3 components. The focused crawler searches multiple external databases for scholarly articles potentially relevant to a query; downloads their titles, abstracts, and other lexical metadata; derives RDF triples from the metadata; and embeds these triples in NPDS records. The quadstore stores the RDF triples in named graphs to enable provenance-aware semantic search. The SPARQL engine searches the quadstore and returns an answer to the query along with a list of the articles used to derive it. Acting in concert, these provide a composite open-web SPARQL search utility, enabling researchers to use semantic queries to search external databases that lack built-in semantic markup. Here we describe the system architecture and implementation of CoVaSEA and demonstrate example results for queries involving neuroscience and other biomedical topics.**

***Index Terms*—Nexus-PORTAL-DOORS System, CoVaSEA, SPARQL, web-crawler, semantic web**

Introduction

The Nexus-PORTAL-DOORS System (NPDS) [1] is a system that provides the capability to publish both semantic and lexical resources regarding specific target areas. NPDS has inbuilt REST API services via the Scribe Registrar [2] along with a standardized messaging protocol, enabling exchange among client applications and servers. The structure of the Nexus PORTAL-DOORS System is sub-divided according to the Hierarchically Distributed Mobile Metadata architectural style into three parts: Nexus, PORTAL, and DOORS [3]. The PORTAL registry controls entities that have unique labels and their associated lexical metadata. The DOORS directory contains semantic metadata which is primarily comes in the form of RDF descriptions. The Nexus diristry (an aggregation of the terms DIRectory and regISTRY) is a single server that serves as a combination of the PORTAL registry and DOORS directory.

Integrated with NPDS, CoVaSEA (Concept-Validating Search Engine Agent) [4] aims to allows users to apply a SPARQL-query based search to external biomedical literature databases. This approach may be valuable in a variety of tasks which benefit from a combination of a web-crawler's capability to search the open web with the versatility of a semantic query engine. CoVaSEA uses NPDS DOORS directories as a storage basis for the records that it has retrieved, which means that it can also serve as a utility to search and expand NPDS metadata records.

Methods

Acting as a successor to the past CoVaSEA, CoVaSEA integrates several features which augment the utility of NPDS including: **(A)** An implementation of SPARQL query based semantic search to allow retrieval and manipulation of linked data descriptions **(B)** Targeted web-crawling for relevant articles from external biomedical literature databases to expand NPDS records **(C)** Automated translation of free-form text abstracts into RDF triples to derive the semantic representations of lexical data. First, the user inputs a Natural Language Query (NLQ) and a SPARQL query, which also has the option to be constructed via a SPARQL builder form, into the CLI. These are passed to the webcrawler which combs through the various databases and retrieves the abstract text and metadata for a multitude of articles. On each article, the natural language parser transforms the text and metadata into semantic triples, which are stored in a local quadstore or DOORS directory. These are searched with the provided or constructed SPARQL query and the results are returned to the user. CoVaSEA consists of a number of interdependent components, each of which fulfills a general purpose. The methods are:

Web Crawler: The webcrawler combs through literature databases in order to retrieve relevant articles. Currently the system can search through the literature records on DOAJ, PubMed, Elsevier ScienceDirect, and CORE [5]. Using provided REST API services, the crawler obtains a list of relevant articles via a general lexical search. From there, the basic metadata, consisting of the abstract, title, author(s), database of origin, DOI(if available), and date of publication, from each article is retrieved and passed to the natural language parser. An RDF representation of the abstract will be constructed and the rest of the metadata will be included in the named graph of the article via the DublinCore ontology [6].

Natural Language Parser: The Natural Language Parser receives the raw abstract text from the web

Document created October 10, 2018, revised November 4, 2018; typeset November 5, 2018.

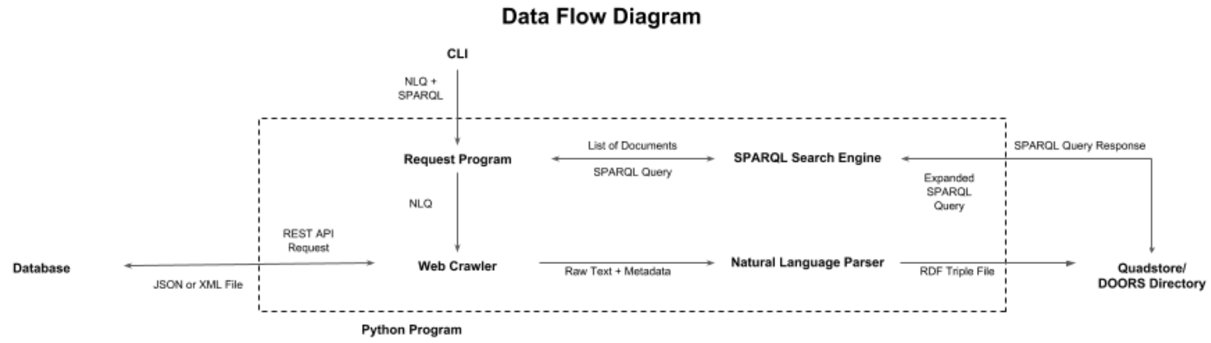


Fig. 1. Data flow diagram of CoVaSEA

crawler and parses it into logical form triples. This section utilizes the Stanford Core NLP library [7] and NLTK [8].

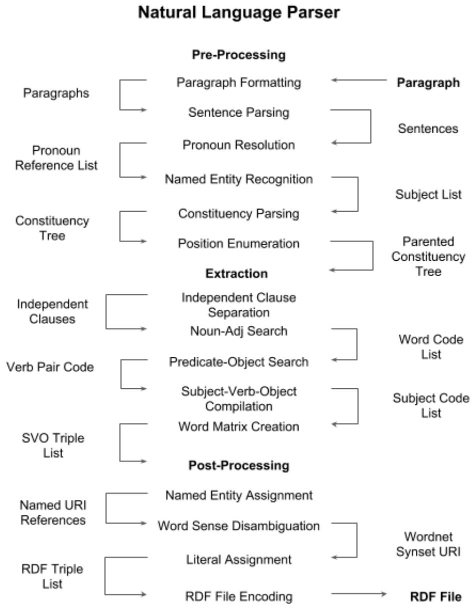


Fig. 2. Data flow diagram of the Natural Language to RDF parser

Pre-processing: A step in which the raw text is prepared for parsing and to perform tasks that depend on the text itself such as Named Entity Recognition [9] and Co-Referencing [10]. Pre-processing preparation includes re-spacing the text and parsing the paragraphs into sentences.

Extraction: A constituency parse is used to build a tree representation of the sentence. Independent clauses are separated into different trees. The highest noun phrase in the tree is found using breadth-first search. The noun phrase is split up if there are conjunctions and any adjectives are linked to the nouns. Then the highest verb phrase is found using breadth-first search and the verb is seperated from the prepositional or adverb sections with depth-first search. The verbs and noun phrases are then tagged with positions and put into triples.

Post-processing: The logical form triples are translated into RDF triples by assigning Unique Resource Identifiers(URI) to each part of the triples. Field-specific language and some named entities are assigned URI's via databases such as MeSH. Then standard words are assigned via WordNet [11] synsets using Lesk [12]. Finally, named entities and numbers are assigned to literals. These steps aid in graph-compression and also allows for the full semantic meaning to be captured rather than just the lexical data. The triples are all packaged into the article's named graph to be stored in a local quadstore or DOORS directory.

SPARQL Analysis Engine: A SPARQL query is received and then used to search through the triple store to extract relevant data for the user. The query engine supports a SPARQL builder, thus circumventing the need to know SPARQL syntax. The system works by having the user create a series of either filtering, mandatory, or optional conditions in either an independent or nested connection. Thus the user can build a search query via simple fill-in-the-blank statements. Though it cannot replicate the full power of SPARQL syntax, it is a potent resource for users who do not want to use SPARQL. Due to the dynamic and heterogeneous nature of the data that CoVaSEA handles, the SPARQL query engine divides up the queries into sub-queries such that only the articles that are needed are compiled. This federated approach avoids the necessity of having to compile a computationally expensive local graph.

QuadStore: To prevent redundant re-rendering of articles, CoVaSEA records the triples that are built by the web-crawler in either a local quadstore or a DOORS directory. Each of the articles has its own named graph, allowing for the identification of the origin of triples. The graph is divided into two sections: the metadata section and the semantic representation section. The metadata section stores key information about the article such as author, title, and publication date. The semantic representation section stores a set of triples which are a RDF portrayal of the article's abstract. The RDF files are sent and retrieved from the DOORS directory utilizing the Scribe API.

Results

**Computer spec: 16 GB RAM, i7-8750H. Connection speeds: Download of 18.40 Mgbs, Upload of 1.68 Mgbs.* → *Computer spec: 16 GB RAM, i7-8750H. Connection speeds: Download of 18.40 Mgbs, Upload of 1.68 Mgbs.*

**For all tests the SPARQL query requested the basic metadata and the NLQ was the name of the condition*

Input	Result	Runtime
Parkinson's Requested: 20 articles	20 articles 120 results	243.05 seconds
Parkinson's Requested: 100 articles	100 articles 581 results	1189.61 seconds
Dementia Requested: 20 articles	20 articles 120 results	245.12 seconds
Dementia Requested: 100 articles	100 articles 596 results	1307.42 seconds
Melanoma Requested: 20 articles	20 articles 118 results	232.12 seconds
Melanoma Requested: 100 articles	100 articles 587 results	1350.78 seconds

Discussion

CoVaSEA offers capabilities that can facilitate a variety of tasks. Primarily, the role of CoVaSEA is a utility to apply a SPARQL-query based search to external biomedical literature databases. For researchers, this may be useful in situations for which the versatility and adaptability that is offered by a SPARQL query needs to be applied to lexical data on the open-web. However, CoVaSEA is not just limited to serving the user, but also, due to its relationship with NPDS, is a utility to search and expand semantic records on the DOORS directory. Since system is automated, and can furnish large amounts of semantic descriptions on a regular basis, CoVaSEA lays the groundwork for a variety of future NPDS applications(i.e. automated meta-analysis).

The primary point of possible expansion for CoVaSEA is the Natural-Language Parser. Currently, the parser faces difficulty in 3 main areas: relevance of triples, accuracy of triples, and runtime. The long runtime of CoVaSEA impedes us from performing whole article parses with reasonable run-times. The largest reasons for the long run-times are co-reference resolution and constituency parsing. Another section of improvement is the SPARQL query engine. The two biggest improvements that could be made are SPARQL query expansion and a semantic reasoning engine, both of which increase the breadth of the search.

Conclusion

Here we presented CoVaSEA: a system in which resources from the open-web can be translated into machine-understandable semantic information and be searched via SPARQL. CoVaSEA has the capability to both search externally with the web crawler for semantic data to expand the NPDS knowledge base and internally with SPARQL to navigate the data inside the DOORS directory. These capabilities combine to offer a utility which can semantically search external biomedical literature databases. The application of the wide range of possibilities offered by semantic search to the array of lexical information present on the open-web renders many services, both for users and for the Nexus PORTAL-DOORS System.

References

[1] C. Taswell, G. TeleGenetics, and C. Ladera Ranch, "Portal-doors infrastructure system for translational biomedical informatics on the semantic web and grid," *Proceedings of the American Medical Informatics Association Summit on Translational Bioinformatics, San Francisco, CA*, p. 43, 2008.

[2] A. Craig, S. H. Bae, T. Veeramacheneni, *et al.*, "Web service apis for scribe registrars, nexus diristries, portal registries and doors directories in the npd system.," in *SWAT4LS*, 2016.

[3] C. Taswell, "A distributed infrastructure for metadata about metadata: The hdmm architectural style and portal-doors system," *Future Internet*, vol. 2, no. 2, pp. 156–189, 2010.

[4] S.-H. Bae, A. G. Craig, C. Taswell, *et al.*, "Expanding nexus diristries of dementia literature with the npds concept-validating search engine agent,"

[5] P. Knoth and Z. Zdrahal, "Core: Three access levels to underpin open access," *D-Lib Magazine*, vol. 18, no. 11/12, 2012.

[6] S. Weibel, "The dublin core: A simple content description model for electronic resources," *Bulletin of the American Society for Information Science and Technology*, vol. 24, no. 1, pp. 9–11, 1997.

[7] C. Manning, M. Surdeanu, J. Bauer, *et al.*, "The stanford corenlp natural language processing toolkit," in *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, 2014, pp. 55–60.

[8] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

[9] J. R. Finkel, T. Grenager, and C. Manning, "Incorporating non-local information into information extraction systems by gibbs sampling," in *Proceedings of the 43rd annual meeting on association for computational linguistics*, Association for Computational Linguistics, 2005, pp. 363–370.

[10] K. Clark and C. D. Manning, "Entity-centric coreference resolution with model stacking," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, vol. 1, 2015, pp. 1405–1415.

[11] G. A. Miller, "Wordnet:a lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[12] S. Banerjee and T. Pedersen, "An adapted lesk algorithm for word sense disambiguation using wordnet," in *International conference on intelligent text processing and computational linguistics*, Springer, 2002, pp. 136–145.

Analyzing and Expanding NPDS Biomedical Record Databases via a Semantic Search Engine

Shiladitya Dutta and Carl Taswell MD, PhD

2018-October-31

Abstract: In this paper we describe CoVaSEA (Concept-Validating Search Engine Agent): an automated semantic search engine that is interoperable with the Nexus-PORTAL-DOORS System. The Nexus-PORTAL-DOORS System (NPDS) is a data management system that organizes repositories of lexical metadata (in PORTAL servers) and semantic representations (in DOORS servers) of resources. Integrated with NPDS, CoVaSEA serves to apply SPARQL-based search to external biomedical literature databases by converting relevant article resources to a linked data format which can be searched by the query engine. This capability can prove to be valuable in a variety of tasks which benefit from a combination of a web-crawler's capability to search the open web with the versatility of a semantic query engine. CoVaSEA, mirroring its dual nature, has a SPARQL-based search engine facing inwards and a focused web-crawler facing outwards. The search engine acts as an "internal" search, allowing a user to explore semantic records via SPARQL-based search. For a more conducive user experience, the query engine employs a SPARQL builder, thus circumventing the need to know SPARQL syntax. The utility functions by having the user input a series of either filtering, mandatory, or optional conditions in either an independent or nested connection. Thus the user can form a search query via simple fill-in-the-blank statements. Though it cannot replicate the full power of SPARQL syntax, it is a potent resource for users who do not wish to use SPARQL. With the constructed or inputted query, CoVaSEA searches through the linked database with the query engine. Due to the dynamic and heterogeneous nature of the data that CoVaSEA handles, the SPARQL query engine divides up the queries into sub-queries such that only the necessary articles are compiled. This iteration based approach optimizes runtime and memory-load on computers by eliminating the redundancy of loading unneeded triples into a computationally expensive local graph. In contrast to the query engine, the web-crawler provides "external" search functionality to render the capability to search for outside resources. The web crawler retrieves articles along with their basic metadata from several of biomedical literature databases via REST API. In addition to basic metadata, key RDF triples which describe the abstract are constructed in order to record a full semantic description of the data in each article. To prevent redundant re-rendering of articles, CoVaSEA saves the triples which are built by the web-crawler in either a local quadstore or a DOORS directory. Each articles in the record has a corresponding named graph, allowing for the identification of the origin of triples. The graph contains two sections: metadata and the semantic representation. The metadata section stores key information about the article such as author, title, publication date, and database of origin. The semantic representation section stores a set of triples which outline the key ideas of the article's abstract. A secondary benefit of this storage schema is that CoVaSEA is not limited to converting and searching external lexical databases, but can also perform as a semantic search engine for the DOORS directory. In a broad sense, CoVaSEA consists of 3 components: the federated query engine searching, the web-crawler expanding, and the quadstore storing. The query engine builds and applies SPARQL queries for users to explore semantic triples. The webcrawler adds semantically formatted records to NPDS from the open-web. The quadstore/DOORS directory logs the formatted records for future searches. Acting in concert, these can provide a composite open-web SPARQL search utility. For researchers, this opens the possibility to search external biomedical databases via a semantic web approach and is a solution for situations in which SPARQL queries need to be applied to non-linked data resources.

Analyzing and Expanding NPDS Biomedical Record Databases via a Semantic Search Engine

Shiladitya Dutta and Carl Taswell MD, PhD

2018-October-31

Abstract: In this paper we describe CoVaSEA (Concept-Validating Search Engine Agent): an automated semantic search engine that is interoperable with the Nexus-PORTAL-DOORS System. The Nexus-PORTAL-DOORS System (NPDS) is a data management system that organizes repositories of lexical metadata (in PORTAL servers) and semantic representations (in DOORS servers) of resources. Integrated with NPDS, CoVaSEA serves to apply SPARQL-based search to external biomedical literature databases by converting relevant article resources to a linked data format which can be searched by the query engine. This capability can prove to be valuable in a variety of tasks which benefit from a combination of a web-crawler's capability to search the open web with the versatility of a semantic query engine. CoVaSEA, mirroring its dual nature, has a SPARQL-based search engine facing inwards and a focused web-crawler facing outwards. The search engine acts as an "internal" search, allowing a user to explore semantic records via SPARQL-based search. For a more conducive user experience, the query engine employs a SPARQL builder, thus circumventing the need to know SPARQL syntax. The utility functions by having the user input a series of either filtering, mandatory, or optional conditions in either an independent or nested connection. Thus the user can form a search query via simple fill-in-the-blank statements. Though it cannot replicate the full power of SPARQL syntax, it is a potent resource for users who do not wish to use SPARQL. With the constructed or inputted query, CoVaSEA searches through the linked database with the query engine. Due to the dynamic and heterogeneous nature of the data that CoVaSEA handles, the SPARQL query engine divides up the queries into sub-queries such that only the necessary articles are compiled. This iteration based approach optimizes runtime and memory-load on computers by eliminating the redundancy of loading unneeded triples into a computationally expensive local graph. In contrast to the query engine, the web-crawler provides "external" search functionality to render the capability to search for outside resources. The web crawler retrieves articles along with their basic metadata from several of biomedical literature databases via REST API. In addition to basic metadata, key RDF triples which describe the abstract are constructed in order to record a full semantic description of the data in each article. To prevent redundant re-rendering of articles, CoVaSEA saves the triples which are built by the web-crawler in either a local quadstore or a DOORS directory. Each articles in the record has a corresponding named graph, allowing for the identification of the origin of triples. The graph contains two sections: metadata and the semantic representation. The metadata section stores key information about the article such as author, title, publication date, and database of origin. The semantic representation section stores a set of triples which outline the key ideas of the article's abstract. A secondary benefit of this storage schema is that CoVaSEA is not limited to converting and searching external lexical databases, but can also perform as a semantic search engine for the DOORS directory. In a broad sense, CoVaSEA consists of 3 components: the federated query engine searching, the web-crawler expanding, and the quadstore storing. The query engine builds and applies SPARQL queries for users to explore semantic triples. The webcrawler adds semantically formatted records to NPDS from the open-web. The quadstore/DOORS directory logs the formatted records for future searches. Acting in concert, these can provide a composite open-web SPARQL search utility. For researchers, this opens the possibility to search external biomedical databases via a semantic web approach and is a solution for situations in which SPARQL queries need to be applied to non-linked data resources.

SPARQL-Based Search Engine and Agent for Finding Brain Literature and Converting References to NPDS Metadata Records

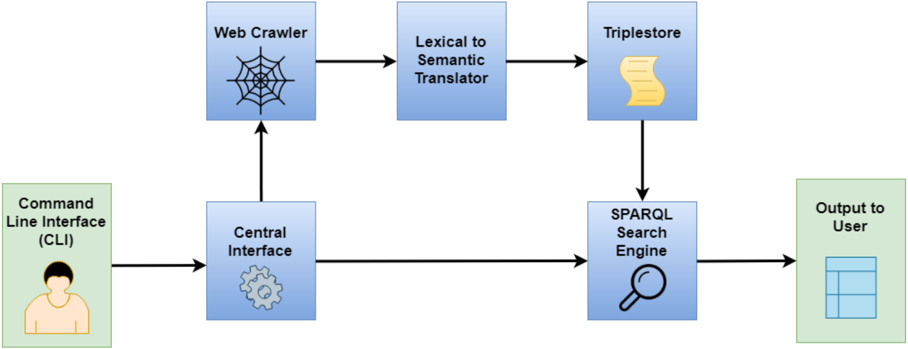
Shiladitya Dutta and Carl Taswell

December 2, 2018

Introduction

- The Nexus-PORTAL-DOORS System (NPDS) is a meta data management system that handles repositories of lexical and semantic representations of resources.
- Some tasks planned for NPDS require stores of brain literature descriptions and a method of retrieving and analyzing the semantic metadata
- To remedy this CoVaSEA (Concept-Validated Search Engine Agent) was developed
- Built via Python in Visual Studio using the NLTK and Stanford CoreNLP natural-language processing toolkits
- Consists of 3 main sections: Web crawler, Lexical to Semantic converter, and SPARQL query engine
- Combine to create a hybrid internal/external search system oriented towards brain literature and research

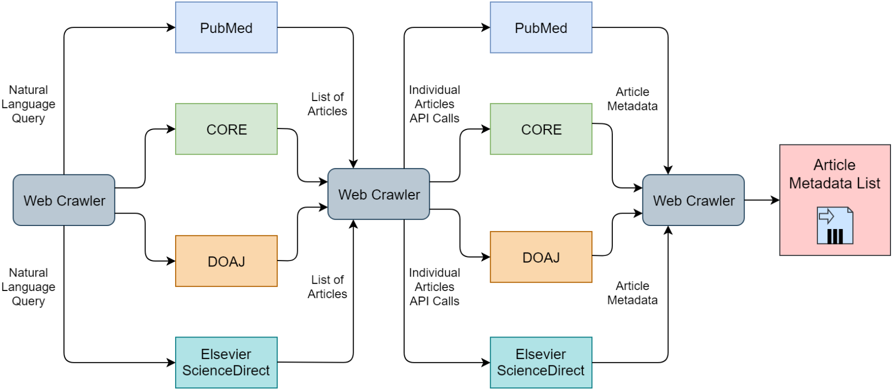
Data Flow Diagram



Web Crawler

- Based on the previous CoVaSEA web crawler developed in Javascript (Bae et al., 2017)
- The user inputs the database they want to search, the general search query that is used to search the literature database, and the number of articles they want to request
- Webcrawler has the option to go to 4 different brain literature databases: DOAJ, ScienceDirect, CORE, and PubMed
- Utilizes REST API to retrieve citation metadata (authors, name, publication date, etc.)
- First uses the inbuilt article search functionality with a user-provided general search query to find articles of interest
- Then retrieves the meta data for each article individually
- The meta data is passed to the lexical to semantic translator

Web Crawler Cont.



Overview of Lexical to Semantic Translation

- Objective is to translate the abstract received from crawler into Resource Description Framework (RDF) triples that describe the abstract
- Translation occurs in 3-steps:
  1. **Pre-processing:** Converts the abstract into sentence constituency trees
  2. **Extraction:** Derives subject-verb-object triples from the constituency trees
  3. **Post-processing:** Translates the subject-verb-object triples into RDF
- Creates a semantic description of the lexical data contained within the abstract for use by NPDS

Pre-processing

- First performs task that depend on or deal with the raw text of the abstract:
  - Edits the abstract so that it is in a standardized format for parsing (i.e. removing aberrant spacing or deleting tags from the beginning or end of abstract text).
  - Performs Co-reference resolution to determine what subjects the pronouns in the abstract refer to (Finkel et al., 2005)
  - Conducts Named Entity Recognition to identify proper nouns in the abstract (Recasens et al., 2013)
- Executes a constituency parse on the individual sentences to create a constituency tree of each sentence (Chen et al., 2014)
- The constituency tree acts as a tree-based syntactic representation of the sentence where the leaves are words and the nodes are grouping of the words (e.g. noun phrase, prepositional phrase, etc.)

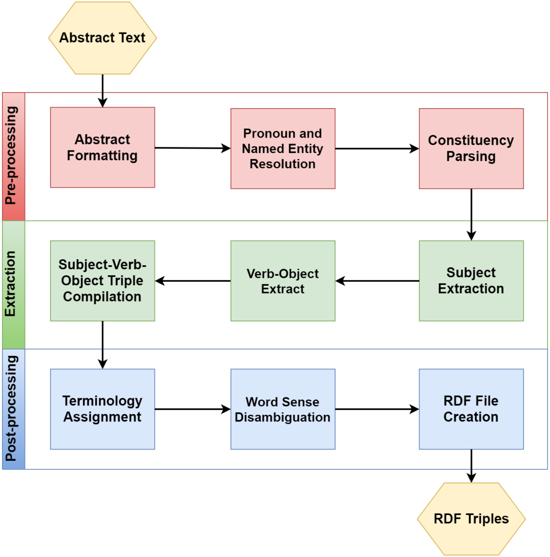
Extraction

- Derives Subject-Verb-Object triples from the constituency tree (Rusu et al., 2007)
- Breadth-first search is used to find the highest noun phrase in the tree
- The noun phrase is split into the individual subjects of the sentence and any adjectives in the noun phrase are linked to the subjects they are referring to
- Verb phrase is found by breadth-first search
- The verb is split from the object by using depth-first search on the verb phrase and the object is linked to its corresponding verb to form a verb-object pair
- The subject and verb-object pairs are combined into subject-verb-object triples

Post-processing

- Subject-verb-object triples are converted to RDF
- Each part of the subject-verb-object triples are assigned Unique Resource Identifiers (URI)
- Terminology is assigned a URI via domain-specific vocabulary databases
- Word Sense Disambiguation is performed by assigning standard nouns and verbs to WordNet synsets using the Lesk algorithm (Banerjee and Pederson, 2002)
- WordNet synsets (Miller, 1995) are groups of synonyms that are semantically equivalent for data retrieval purposes
- Names and numbers are put into RDF literals.
- The converted triples are encoded into RDF files for storage on the triplestore

Natural Language Processing Pipeline



Triple Store

- To store the semantic meta-data of the articles it has converted, CoVaSEA records the citation meta-data triples and the triples that are built by the natural-language parser
- Records can be stored in either a local triplestore and/or DOORS directory
- Graph consists of two sections: the citation metadata section and the semantic representation section.
- Citation metadata section stores basic metadata (author name, publication data, title, etc.)
- Semantic metadata stores triples derived from the abstract by the lexical to semantic translator

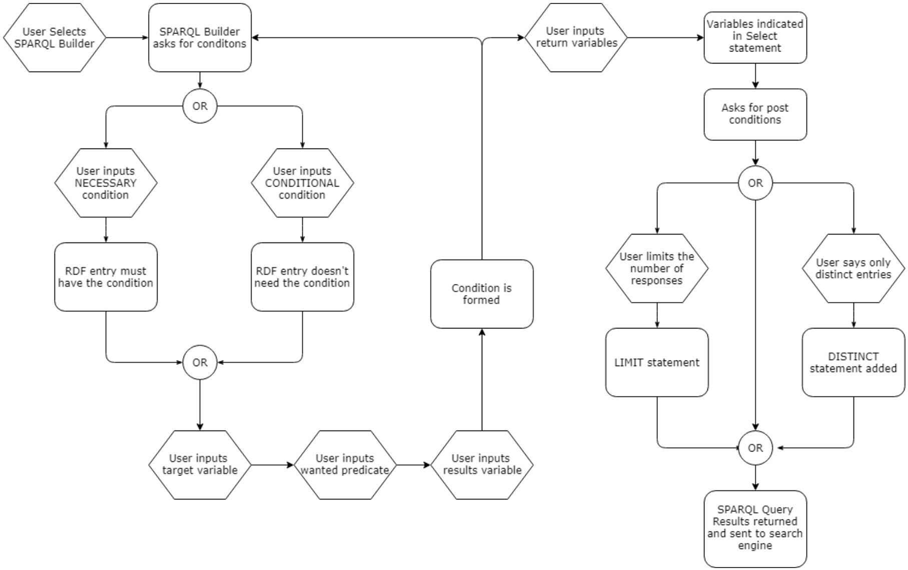
SPARQL Query Engine

- Performs a SPARQL Query search of the triplestore using RDFLib
- SPARQL allows for complex and versatile searches to be made by the user which query for specific data
- Compiles a local graph on the machine in order to perform the search
- The input has the option to be given directly by the user or be compiled via a SPARQL Query Builder
- The SPARQL Query builder is based on the Wikidata SPARQL Query builder (Vrandečić and Krötzsch, 2014)
- Main difference is that the CoVaSEA query builder allows for the user to specify the target variable of a condition whereas the Wikidata builder only allows for conditions to be targeted towards the article

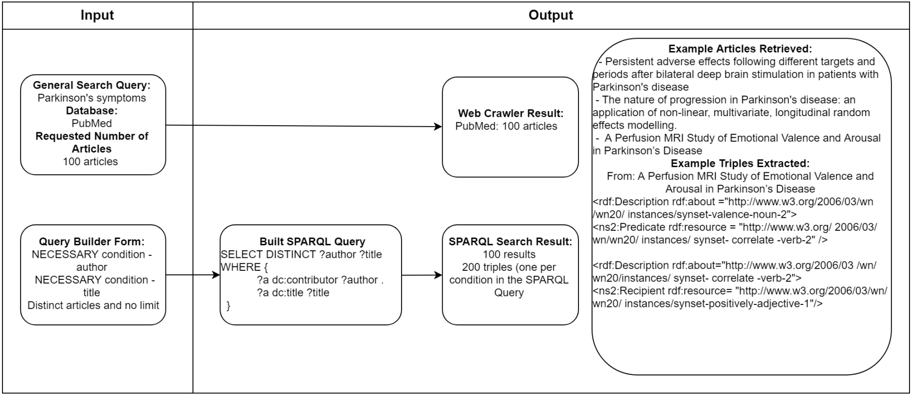
SPARQL Query Builder

- Resource for users who do not want to use SPARQL syntax to build a search query
- Is a SPARQL Builder form that helps users create their own queries
- First the user enters a series of conditions
- Each condition has four parts: type of condition, variable being searched, thing being searched for, and variable to which the result is assigned
- The type of condition can either be a necessary or optional
- Then the user decides which variables they want to return
- Finally, the user decides if they want want only distinct results and if they want to limit the amount of results
- Cannot replicate the full power of SPARQL syntax, but still a potent resource

SPARQL Query Builder Form


User Selects SPARQL Builder
SPARQL Builder asks for conditons
OR
User inputs NECESSARY condition
User inputs CONDITIONAL condition
RDF entry must have the condition
RDF entry doesn't need the condition
OR
User inputs target variable
User inputs wanted predicate
User inputs results variable
Condition is formed
User inputs return variables
Variables indicated in Select statement
Asks for post conditions
OR
User limits the number of responses
User says only distinct entries
LIMIT statement
DISTINCT statement added
OR
SPARQL Query Results returned and sent to search engine

Example Query



Input

Output

General Search Query:
Parkinson's symptoms
Database:
PubMed
Requested Number of Articles
100 articles

Web Crawler Result:
PubMed: 100 articles

Query Builder Form:
NECESSARY condition - author
NECESSARY condition - title
Distinct articles and no limit

Built SPARQL Query

```
SELECT DISTINCT ?author ?title
WHERE {
        ?a dc:contributor ?author .
        ?a dc:title ?title
 }
```

SPARQL Search Result:
100 results
200 triples (one per condition in the SPARQL Query

Example Articles Retrieved:

- Persistent adverse effects following different targets and periods after bilateral deep brain stimulation in patients with Parkinson's disease
- The nature of progression in Parkinson's disease: an application of non-linear, multivariate, longitudinal random effects modelling.
- A Perfusion MRI Study of Emotional Valence and Arousal in Parkinson's Disease

Example Triples Extracted:

From: A Perfusion MRI Study of Emotional Valence and Arousal in Parkinson's Disease

```
<rdf:Description rdf:about ="http://www.w3.org/2006/03/wn
/wn20/ instances/synset-valence-noun-2">
<ns2:Predicate rdf:resource = "http://www.w3.org/ 2006/03/
wn/wn20/ instances/ synset- correlate -verb-2" />

<rdf:Description rdf:about="http://www.w3.org/2006/03 /wn/
wn20/instances/ synset- correlate -verb-2">
<ns2:Recipient rdf:resource= "http://www.w3.org/2006/03/wn/
wn20/ instances/synset-positively-adjective-1"/>
```

Results

Note: Successful translation is defined by when the lexical to semantic translator is able to successfully convert an abstract to RDF triples

Database	# of Articles Requested	Query	# of Articles Received	Lexical to Semantic Abstract Translation	SPARQL Search Result	Runtime
DOAJ	10 articles	General Search: Parkinson's Symptoms SPARQL Search: Request for author and title	10 articles	Successfully Translated: 10 articles # of Triples: 64	10 results 20 triples	37.12 seconds
	100 articles	General Search: Parkinson's Symptoms SPARQL Search: Request for author and title	97 articles	Successfully Translated: 96 articles # of Triples: 573	96 results 192 triples	312.34 seconds
	1000 articles	General Search: Parkinson's Symptoms SPARQL Search: Request for author and title	983 articles	Successfully Translated: 967 articles # of Triples: 6854	966 results 1932 triples	2589.32 seconds
PubMed	10 articles	General Search: Parkinson's Symptoms SPARQL Search: Request for author and title	10 articles	Successfully Translated: 10 articles # of Triples: 93	10 results 20 triples	31.84 seconds
	100 articles	General Search: Parkinson's Symptoms SPARQL Search: Request for author and title	99 articles	Successfully Translated: 99 articles # of Triples: 745	99 results 198 triples	283.31 seconds
	1000 articles	General Search: Parkinson's Symptoms SPARQL Search: Request for author and title	1000 articles	Successfully Translated: 984 articles # of Triples: 8321	984 results 1968 triples	2391.12 seconds
Elsevier ScienceDirect	10 articles	General Search: Parkinson's Symptoms SPARQL Search: Request for author and title	10 articles	Successfully Translated: 10 articles # of Triples: 74	10 results 20 triples	45.31 seconds
	100 articles	General Search: Parkinson's Symptoms SPARQL Search: Request for author and title	97 articles	Successfully Translated: 96 articles # of Triples: 455	96 results 192 triples	332.14 seconds
	1000 articles	General Search: Parkinson's Symptoms SPARQL Search: Request for author and title	998 articles	Successfully Translated: 954 articles # of Triples: 7213	954 results 1908 triples	2431.21 seconds
CORE	10 articles	General Search: Parkinson's Symptoms SPARQL Search: Request for author and title	10 articles	Successfully Translated: 9 articles # of Triples: 50	9 results 18 triples	44.32 seconds
	100 articles	General Search: Parkinson's Symptoms SPARQL Search: Request for author and title	100 articles	Successfully Translated: 94 articles # of Triples: 398	94 results 188 triples	390.32 seconds
	1000 articles	General Search: Parkinson's Symptoms SPARQL Search: Request for author and title	973 articles	Successfully Translated: 912 articles # of Triples: 8434	911 results 1822 triples	2945.32 seconds

Discussion

- Translation success rate is not 100% due to either irregular formatting of results files or inability to properly parse abstracts that have more advanced sentence structure
- Number of abstract-derived semantic triples varies highly depending on the length of the abstract
- The runtime scaling between the 10, 100 and 1000 article parse is not linear due to the constant runtime of initializing the Stanford NLP Parser and the initial web crawler natural-language search.

Conclusion

- Here we present CoVaSEA: a Web crawler/SPARQL query engine
- Combines the capability to search externally on the open-web for articles and internally in its previously built semantic records with SPARQL
- CoVaSEA has 3 main parts:
  1. The web crawler retrieves articles from brain literature databases
  2. The lexical to semantic converter converts retrieved text into RDF triples and stores in a triplestore
  3. SPARQL query engine allows users to search through the triplestore
- These mesh together to create a system that can access and broaden semantic records for brain literature and research for NPDS

References

S. Banerjee and T. Pedersen, "An adapted lesk algorithm for word sense disambiguation using wordnet," in *International conference on intelligent text processing and computational linguistics*, Springer, 2002, pp. 136–145.

D. Chen and C. Manning, "A fast and accurate dependency parser using neural networks," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 740–750.

J. R. Finkel, T. Grenager, and C. Manning, "Incorporating non-local information into information extraction systems by gibbs sampling," in *Proceedings of the 43rd annual meeting on association for computational linguistics*, Association for Computational Linguistics, 2005, pp. 363–370.

G. A. Miller, "Wordnet: A lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

M. Recasens, M.-C. de Marneffe, and C. Potts, "The life and death of discourse entities: Identifying singleton mentions," in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2013, pp. 627–633.

D. Rusu, L. Dali, B. Fortuna, M. Grobelnik, and D. Mladenic, "Triplet extraction from sentences," in *Proceedings of the 10th International Multiconference" Information Society-IS*, 2007, pp. 8–12.

S.-H. Bae, A. G. Craig, C. Taswell, *et al.*, "Expanding nexus diristries of dementia literature with the npds concept-validating search engine agent,", 2017.

D. Vrandečić and M. Krötzsch, "Wikidata: A free collaborative knowledgebase," *Communications of the ACM*, vol. 57, no. 10, pp. 78–85, 2014.

Questions?

Brain Health Alliance Virtual Institute

Ladera Ranch, California

Shiladitya Dutta: sdutta@bhavi.us

Carl Taswell: ctaswell@bhavi.us

Developments to Project

Shiladitya Dutta

3 slides - 5 minutes

November 18, 2018

Query Builder

- SPARQL Query Builder utility for users
- Helpful to users who don't know SPARQL syntax
- Based of the user inputting a series of conditions to acquire a desired response
- Cannot replicate full functionality of SPARQL

Process of Inputting Condition

- First the user enters a series of conditions
- Each condition has four parts: type of condition, variable being searched, thing being searched for, and variable to which the result is assigned
- The type of condition can either be necessary or conditional
- One thing to note is that the last 3 conditions each correspond to a particular part of a condition in SPARQL
- Then the user decides which variables they want to return
- Finally, the user decides if they want want only distinct results and if they want to limit the amount of results

Discussion

- It is possible to replicate many of the functions in SPARQL utilizing this design
- Since the user can choose the variable to which the result is assigned, the user can link together queries
- Still lacks the capability to do more advanced actions such as UNION of queries and assigning variables
- The system also still depends on the underlying structure of SPARQL, so users will still need to know SPARQL reasoning and how conditions work
- Thus it is only helpful in situations where the user understands the logic of SPARQL but doesn't have a grasp on the syntax

A Web Crawler and SPARQL Query Search Agent to Expand and Navigate NPDS Brain Literature Records

Shiladitya Dutta, Carl Taswell M.D., Ph.D.

2019-September-4

Abstract: The Nexus-PORTAL-DOORS System (NPDS) manages independent repositories of lexical and semantic renditions of resources. Analogous to IRIS-DNS, NPDS is organized into 3 server types: PORTAL registries for lexical metadata, DOORS directories for semantic metadata, and Nexus diristries which act as hybrid PORTAL-DOORS servers. Many of the goals laid out for the Nexus PORTAL-DOORS System require records of semantic descriptions, however it is labor-intensive to manually annotate resources from the open-web. To remedy this, we have created CoVaSEA (Concept-Validated Search Engine Agent): an automated web crawler/query engine. The intention of CoVaSEA is to provide a method for a user to search and expand the semantic records of brain literature in the DOORS directories. To this end, CoVaSEA integrates multiple features which benefit NPDS including: (A) An implementation of SPARQL query based search to allow retrieval and manipulation of DOORS descriptions, (B) Targeted web-crawling for relevant articles from external biomedical literature databases to expand NPDS records, and (C) Automated translation of free-form text into RDF triples to derive the semantic representations of lexical data. The system consists of a pipeline in which a web crawler registers articles by converting the abstract text and descriptions to RDF, so that a SPARQL query engine can comb through the retrieved material. The web crawler searches for articles in 4 biomedical literature databases, PubMed, Elsevier ScienceDirect, DOAJ, and Springer, via REST API. Each article has its basic metadata (title, abstract, author(s), etc.) returned. In order to translate the lexical information in the abstract into semantic metadata, CoVaSEA develops RDF descriptions of the articles' abstracts. First, constituency parsing is performed to create a tree from which the logical-form triples are extracted by searching for the subject(s), predicate(s), and object(s). Then the triples are processed by coreference resolution and pronomial anaphora to ensure that unique entities have consistent references. Once the logical-form triple processing is finished, they are converted from lexical-based triples to valid RDF by assigning URI's to each part of the subject-predicate-object triples. This is accomplished by using various databases (i.e. MeSH) for field-specific terminology and select named entities, word sense disambiguation for standard words, and literals for numerals and names. Finally, the graph is converted to RDF, yielding a file that portrays the natural-language information in each abstract in a linked data format. These files are then stored in a DOORS directory via the Scribe API. When a SPARQL query is called, the program retrieves the graphs from the database to be queried via the SPARQL query engine. As a whole, CoVaSEA presents the capability to both search ”externally” with the web crawler for semantic data to expand the DOORS knowledge base and ”internally” with SPARQL to provide a method to navigate the data inside the DOORS directory. With the distinct advantage that the system is automated, CoVaSEA can furnish large numbers brain-related literature descriptions on a regular basis, thus laying the groundwork for a variety of future NPDS applications for which semantic metadata stores are a necessity.

A Web Crawler and SPARQL-Based Search Engine to Expand and Navigate Brain Literature Records

Shiladitya Dutta, Carl Taswell M.D., Ph.D.

2019-September-4

Abstract: In this paper we describe CoVaSEA (Concept-Validated Search Engine Agent): an automated web crawler/query engine that is designed for the Nexus-PORTAL-DOORS System. The Nexus-PORTAL-DOORS System (NPDS) is a data management system that handles repositories of lexical metadata (in PORTAL servers) and semantic representations (in DOORS servers) of resources. Due to the purpose built hybridized nature of NPDS, it is well-placed to perform a variety of lexical-semantic data analysis tasks. However, many of these tasks require records of semantic descriptions which are labor-intensive to create and maintain due to the substantial and rapidly increasing quantities of brain-literature available on the open web. To remedy this, we created CoVaSEA with the intention of providing a method for users to navigate and expand the semantic records of brain literature in the NPDS directories. To this end, CoVaSEA integrates multiple features which benefit NPDS including: (A) An implementation of SPARQL query based search to allow retrieval and manipulation of RDF descriptions, (B) Targeted web-crawling for relevant articles from external biomedical literature databases to broaden NPDS records, and (C) Automated translation of free-form text into RDF triples to derive the semantic portrayals of lexical data. CoVaSEA consists of three main components: the web-crawler, the lexical to semantic converter, and the SPARQL query engine. The web crawler retrieves articles along with their basic metadata (title, abstract, author(s), etc.) from several of biomedical literature databases via REST API. However, in order to create a full semantic description of the data in each article, the abstracts have key RDF triples extracted from them. First, each of the unique nouns in the passage are registered via coreference resolution and pronomial anaphora. Then the sentences are parsed into constituency tree format so that the subject(s), verb(s), and object(s) can be extracted. Once the SVO triples are extracted, they are transformed into valid RDF by assigning unique resource identifiers (URI) to each part of the triples. This is accomplished by using various databases (i.e. MeSH) for terminology and select named entities, word sense disambiguation for standard words, and literals for any other sections. These triples are stored via the Scribe API in a DOORS directory where they can be retrieved via the SPARQL query engine. In order to create a more conducive user experience, the query engine offers the capability to construct SPARQL queries from expressions in conjunctive normal form, thus circumventing the need to know SPARQL syntax. With the distinct advantage that the system is automated, CoVaSEA presents the capability to search "externally" to furnish large numbers brain-related literature descriptions on a regular basis and search "internally" to provide a method of retrieving those descriptions, thus laying the groundwork for a variety of future NPDS applications for which semantic metadata stores of brain literature are a necessity.

A Web Crawler and SPARQL-Based Search Engine to Expand and Navigate Brain Literature Records

Shiladitya Dutta, Carl Taswell M.D., Ph.D.

2019-September-14

Abstract: In this paper we describe CoVaSEA (Concept-Validated Search Engine Agent): an automated web crawler/query engine that is inter-operable with the Nexus-PORTAL-DOORS System. The Nexus-PORTAL-DOORS System (NPDS) is a data management system that manages repositories of the lexical metadata (in PORTAL servers) and semantic representations (in DOORS servers) of resources. Due to the purpose built hybridized nature of NPDS, it is well-placed to perform a variety of lexical-semantic data analysis tasks. However, many of these tasks require records of semantic descriptions which are labor-intensive to create and maintain due to the substantial and rapidly increasing quantities of brain-related literature available on the open web. To remedy this, we have created CoVaSEA with the intention of providing a method for users to navigate and expand the semantic records of brain literature in the NPDS directories. To this end, CoVaSEA integrates multiple features which benefit NPDS including: (A) An implementation of SPARQL query based search to allow retrieval and manipulation of RDF descriptions, (B) Targeted web-crawling for relevant articles from external biomedical literature databases to broaden NPDS records, and (C) Automated translation of free-form text into RDF triples to derive the semantic portrayals of lexical data. CoVaSEA consists of three principal components: the web-crawler, the lexical to semantic converter, and the SPARQL query engine. The web crawler retrieves articles along with their basic metadata (title, abstract, author(s), etc.) from several of biomedical literature databases via REST API. However, in order to create a full semantic description of the data in each article, key RDF triples which describe the abstracts are constructed. First, each of the unique nouns in the passage are registered via coreference resolution and pronomial anaphora. Then the sentences are parsed into constituency tree format so that the subject(s), verb(s), and object(s) can be extracted. Once the SVO triples are extracted, they are transformed into valid RDF by assigning unique resource identifiers (URI) to each part of the triples. This is accomplished by using various databases (i.e. MeSH) for terminology and select named entities, word sense disambiguation for standard words, and literals for any other sections. These triples are stored via the Scribe API in a DOORS directory where they can be retrieved via the SPARQL query engine. In order to create a more conducive user experience, the query engine supports the capability to construct SPARQL queries from expressions in conjunctive normal form, thus circumventing the need to know SPARQL syntax. With the distinct advantage that the system is automated, CoVaSEA presents the capability to search "externally" to furnish large numbers brain-related literature descriptions on a regular basis and search "internally" to provide a method of retrieving those descriptions, thus laying the groundwork for a variety of future NPDS applications for which semantic metadata stores of brain literature are a necessity.

SPARQL-Based Search Engine and Agent for Finding Brain Literature and Converting References to NPDS Metadata Records

Shiladitya Dutta and Carl Taswell

2018-September-16

Abstract: In this paper we describe CoVaSEA (Concept-Validating Search Engine Agent): an automated web crawler/query engine that is interoperable with the Nexus-PORTAL-DOORS System. The Nexus-PORTAL-DOORS System (NPDS) is a data management system that organizes repositories of lexical metadata (in PORTAL servers) and semantic representations (in DOORS servers) of resources. Due to the purpose built hybridized nature of NPDS, it is well placed to perform a variety of data analysis tasks. However, many of these tasks require records of semantic descriptions which are labor intensive to create and maintain due to the substantial and rapidly increasing quantities of brain related literature available on the open web. To remedy this, we created CoVaSEA with the intention of providing an automated method for users to navigate and expand the semantic records of brain literature in the NPDS directories. To this end, CoVaSEA integrates multiple features which benefit NPDS including: (A) An implementation of SPARQL query based search to allow retrieval and manipulation of RDF descriptions, (B) Targeted web-crawling for relevant articles from external biomedical literature databases to broaden NPDS records, and (C) Translation of free-form text into RDF triples to derive the semantic portrayals of lexical data. CoVaSEA consists of three principal components: the web-crawler, the lexical to semantic converter, and the SPARQL query engine. The web crawler retrieves articles along with their basic metadata (title, abstract, author(s), etc.) from several of biomedical literature databases via REST API. However, in order to capture a full semantic description of the data in each article, key RDF triples which describe the abstract are constructed. First, each of the unique nouns in the passage are registered via coreference resolution and pronomial anaphora. Then the sentences are parsed into constituency tree format so that the subject(s), verb(s), and object(s) can be extracted. Once the SVO triples are extracted, they are transformed into valid RDF by assigning unique resource identifiers (URI) to each part of the triples. This is accomplished by using various databases (i.e. MeSH) for terminology and select named entities, word sense disambiguation for standard words, and literals for any other sections. These triples are stored via the Scribe API in either a DOORS directory or a localized triplestore where they can be retrieved via the SPARQL query engine. In order to create a more conducive user experience, the query engine supports the capability to construct SPARQL queries from expressions in conjunctive normal form, thus circumventing the need to know SPARQL syntax. With the distinct advantage that the system is automated, CoVaSEA presents the capability to search "externally" to furnish large numbers of brain-related literature descriptions on a regular basis and search "internally" to provide a method of retrieving those descriptions, thus laying the groundwork for a variety of future NPDS applications for which semantic metadata stores of brain literature are a necessity.

Unit Testing Methodology

Shiladitya Dutta

BHAVI 2018

July 5, 2018

Summary of Unit Testing

- Unit testing is the practice of testing individual units of a larger program.
- Typically is testing is automated, however can be done manually
- Serves to streamline development process as to avoid large debugging times
- Is a central component of Test-Driven Development

Test-Driven Development

- Also called Test-Driven Design
- Method of software programming that involves a process of unit testing, programming, and refactoring
- Includes unit testing, integration testing, and system or end-to-end testing
- Reduces the time spent debugging and testing the code at the endpoint because the code is continuously tested during the design process to ensure the functionality of any new changes.

Reasons to Perform Unit Testing

- Stops the creation of compound errors (errors that stem from failure in multiple components)
- Allows for easy identification of the point of failure in a given program
- Helps to ensure the program still functions as the code base changes
- Simplifies integration of multiple sections of code

How to perform Unit Testing

- Unit Testing is most commonly performed by an automated tool
- The particular tool usually depends on the language
- Ex: Pyunit for Python, Junit for Java, Ctest for C++, etc.

Steps of Unit Testing

1. Divide the program into set methods or units and name them
2. Develop a set of simple tests and make some ”dummy” data to test with
3. Make the simplest possible program that is capable of passing the unit test
4. Refactor and develop the code ensuring that with each new change the unit test is still passed.

Challenges with Unit Testing

1. One of the biggest challenges with unit testing is dividing the program into testable units
2. This is because the code must be carefully planned such that there are not units that are unable to be tested
3. The code developed in the TDD process is only as good as the tests used, and thus the unit tests must be as accurate to real-life use as possible

User Interfaces to Serve as Inspiration for CoVaSEA

Shiladitya Dutta

January 20, 2018

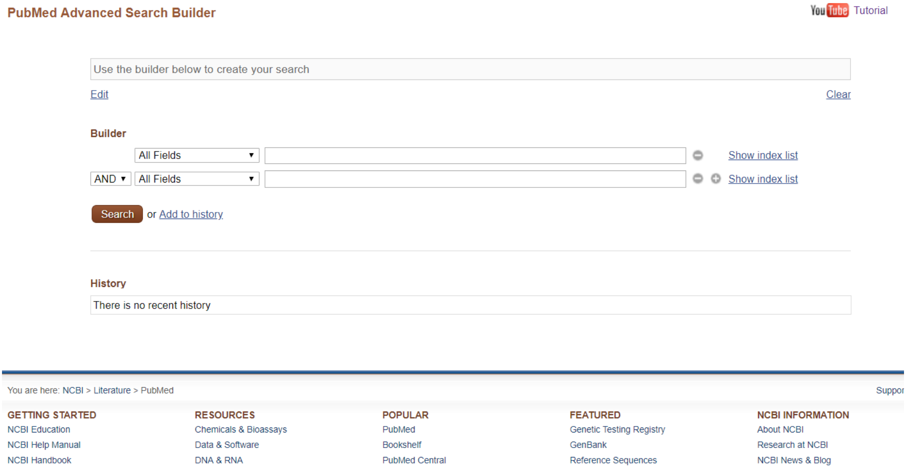
Introduction

- Analyze query builders offered by 4 major research search systems: NCBI PubMed, Elsevier ScienceDirect, Retraction Watch, and Google Scholar
- All of these offer advanced search functionality in some form or another which is designed to provide superior accuracy to that of a general search query
- Analysis consists of the role of the advanced search query and user review of the query
- These will hopefully serve as an inspiration for CoVaSEA's query builder, both for the general and semantic query

NCBI PubMed

- A series of conditions are chained together via AND/OR statements in order to create a search query.
- Also has the option of being inputted directly into a search box. [Ex: (sensory OR language OR motor OR behavior) AND (onset) AND (neurodegenerative OR dementia)]
- This method is relatively intuitive for a user to use once the basic concept is grasped
- Also has a lot more functionality than the conventional method because of the MEDLINE database along with integration with MeSH headings and subheading. (Lowe and Barnett, 1994)
- Overall, this is an extremely proficient system which empowers users of the advanced search functionality greatly.
- System was mirrored by CoVaSEA for the concept-validating system (Bae et. al, 2017).

NCBI PubMed Cont.



PubMed Advanced Search Builder

YouTube Tutorial

Use the builder below to create your search

Edit

Clear

Builder

All Fields ▾ — Show index list

AND ▾ All Fields ▾ — Show index list

Search or Add to history

History

There is no recent history

You are here: NCBI > Literature > PubMed

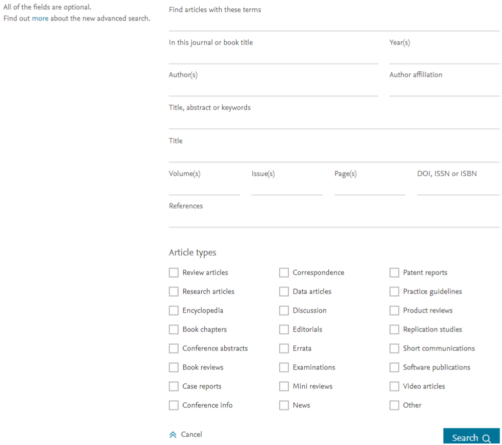
Suppor

GETTING STARTED	RESOURCES	POPULAR	FEATURED	NCBI INFORMATION
NCBI Education	Chemicals & Bioassays	PubMed	Genetic Testing Registry	About NCBI
NCBI Help Manual	Data & Software	Bookshelf	GenBank	Research at NCBI
NCBI Handbook	DNA & RNA	PubMed Central	Reference Sequences	NCBI News & Blog

Elsevier ScienceDirect

- Elsevier ScienceDirect's advanced search works off of a more conventional system of having the option to input a few more conditions
- However, there is one interesting feature of Elsevier's approach. This is the option to specify which type of content the user would like
- The advanced search option fields an exhaustive list of possible content types which can be accessed
- This allows users to specify with high accuracy what general type of content they would like.
- This method may not be exactly transferable to CoVaSEA because of the focus on the categorization of the content rather than the content itself, however it is nevertheless an effective approach.

Elsevier ScienceDirect Cont.



All of the fields are optional.
Find out more about the new advanced search.

Find articles with these terms

In this journal or book title

Year(s)

Author(s)

Author affiliation

Title, abstract or keywords

Title

Volume(s)

Issue(s)

Page(s)

DOI, ISSN or ISBN

References

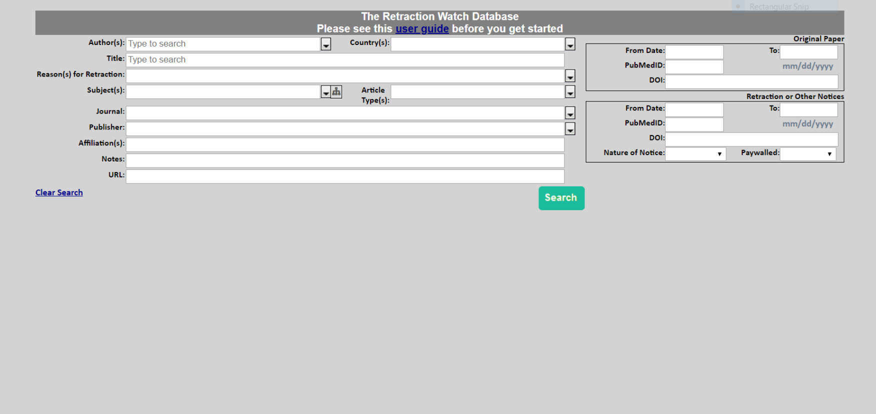
Article types

- □ Review articles
- □ Research articles
- □ Encyclopedia
- □ Book chapters
- □ Conference abstracts
- □ Book reviews
- □ Case reports
- □ Conference info
- □ Correspondence
- □ Data articles
- □ Discussion
- □ Editorials
- □ Errata
- □ Examinations
- □ Mini reviews
- □ News
- □ Patent reports
- □ Practice guidelines
- □ Product reviews
- □ Replication studies
- □ Short communications
- □ Software publications
- □ Video articles
- □ Other

Retraction Watch

- Retraction Watch's search system is less focused on searching based on the content of a paper and more focused on narrowing down based on citation metadata.
- Matches with objective of Retraction Watch which is to provide a database of retracted scientific literature.
- Also has support for entering the PubMedID in order to search for redacted papers.
- Should be noted to be a bit daunting at first glance to an average user because there is no option for a simpler search
- The search method is definitely effective, if somewhat lacking on surface level elegance

Retraction Watch Cont.



The Retraction Watch Database
Please see this user guide before you get started

Author(s): Type to search
Country(s):
Title: Type to search
Reason(s) for Retraction:
Subject(s):
Article Type(s):
Journal:
Publisher:
Affiliation(s):
Notes:
URL:

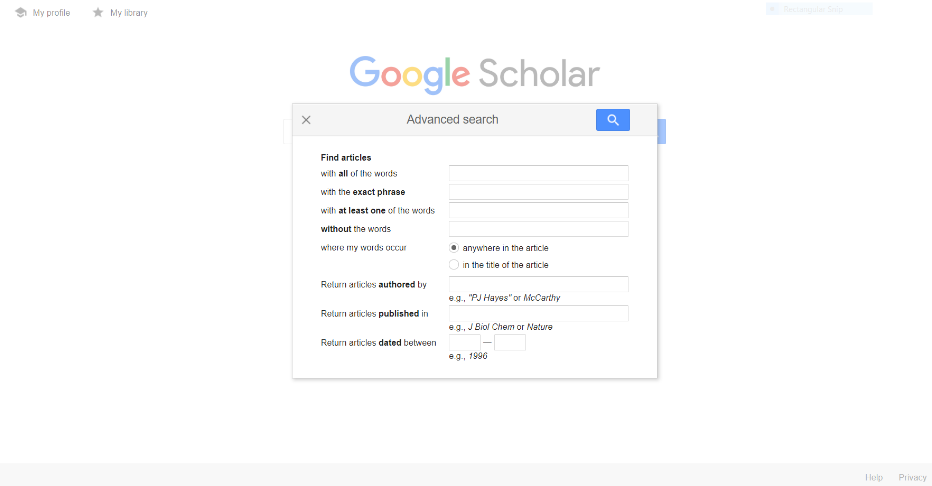
Original Paper
From Date: To: mm/dd/yyyy
PubMedID:
DOI:

Retraction or Other Notices
From Date: To: mm/dd/yyyy
PubMedID:
DOI:
Nature of Notice: ▾ Paywalled: ▾

Google Scholar

- Google Scholar is the search system offered by Google
- Google Scholar differs somewhat from the other systems in that it isn't focused purely on scientific literature, but also patents and books.
- This wide breadth of content makes it significantly more difficult for an effective advanced search builder to be made because there are a wide variety of factors to take into account
- Google solves this by opting for a more generalized version of having multiple conditions
- This generalized system comes in the form of a series of options regarding word matching.
- This method is heavily dependent on the lexical analysis side, which makes sense since lexical search is Google's main business.
- However, this won't be transferable to CoVaSEA as this search method isn't necessarily that effective for users due to the lack of ability to narrow down based on citation data or on general topic.

Google Scholar Cont.



Find articles

with **all** of the words

with the **exact phrase**

with **at least one** of the words

without the words

where my words occur

● anywhere in the article

○ in the title of the article

Return articles **authored** by

e.g., *"PJ Hayes"* or *McCarthy*

Return articles **published** in

e.g., *J Biol Chem* or *Nature*

Return articles **dated** between —

e.g., *1996*

Inspiration from Examples

- CoVaSEA ideally should be a combination of Retraction Watch's approach and PubMed's approach
- Retraction Watch provides an excellent variety of conditions to narrow down results with, which should be useful for the web crawler section of CoVaSEA
- PubMed offers inspiration for the SPARQL Query builder since both are inherently condition based.
- Mirroring PubMed's approach of listing conditions should be very effective for a simple to use SPARQL Query builder
- There are also other SPARQL query builders to take inspiration from such as Wikidata's SPARQL Query builder and SPARQL Views (Vrandečić et. al, 2014) (Clark, 2010)
- Most of these visual SPARQL Query builder use a similar condition listing approach

Future Expansions

- One of the main expansions to CoVaSEA would be to integrate the general search query with the SPARQL search query
- This would be accomplished via the translation of the general search query to a SPARQL Search Query
- Research has been done on this topic (Ngonga Ngomo et. al, 2013) (Wang et. al, 2007) (Yahya et. al, 2012)
- Most of these depend upon the parsing of generic search query into a tree for translation which can then have a SPARQL Query extracted from it.
- This would then lead naturally into the final stage of the CoVaSEA user interface: text-to-speech recognition
- The numerous off the shelf solutions which are available would make text-to-speech relatively easy to implement once the natural language to SPARQL Query translation is complete

References

S.-H. Bae, A. G. Craig, C. Taswell, *et al.*, “Expanding nexus diristries of dementia literature with the npds concept-validating search engine agent,”,

H. J. Lowe and G. O. Barnett, “Understanding and using the medical subject headings (mesh) vocabulary to perform literature searches,” *Jama*, vol. 271, no. 14, pp. 1103–1108, 1994.

D. Vrandečić and M. Krötzsch, “Wikidata: A free collaborative knowledgebase,” *Communications of the ACM*, vol. 57, no. 10, pp. 78–85, 2014.

A.-C. Ngonga Ngomo, L. Bühmann, C. Unger, J. Lehmann, and D. Gerber, “Sorry, i don’t speak sparql: Translating sparql queries into natural language,” in *Proceedings of the 22nd international conference on World Wide Web*, ACM, 2013, pp. 977–988.

C. Wang, M. Xiong, Q. Zhou, and Y. Yu, “Panto: A portable natural language interface to ontologies,” in *European Semantic Web Conference*, Springer, 2007, pp. 473–487.

M. Yahya, K. Berberich, S. Elbassuoni, M. Ramanath, V. Tresp, and G. Weikum, “Natural language questions for the web of data,” in *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, Association for Computational Linguistics, 2012, pp. 379–390.

L. Clark, “Sparql views: A visual sparql query builder for drupal,” in *9th International Semantic Web Conference, ISWC*, 2010.

A Web Crawler and SPARQL Query Search Agent to Expand and Navigate NPDS Semantic Metadata Records

Shiladitya Dutta, Degrees[1], Firstname B. Lastname, Degrees[2]
[1]Institution, City, State, Country (if applicable); [2]Institution, City, State, Country (if applicable)

Introduction

The Nexus PORTAL-DOORS System (NPDS)[1] manages lexical metadata and semantic descriptions of resources. It has 3 principal components: Nexus diristry, PORTAL registry, and DOORS directory. This project interfaces with the DOORS directory and the semantic metadata stored within it. Semantic metadata descriptions are a prerequisite for many of the goals laid out for NPDS such as automated meta-analysis. However, linked data semantic descriptions are time consuming to manually annotate. Thus, we have developed a system that can extract semantic metadata from a variety of biomedical resources and provide a method of searching through that metadata via SPARQL.

Web Crawler Component

The web crawler component retrieves articles and article metadata from DOAJ, Elsevier ScienceDirect, CORE[2], and PubMed via REST API. The websites' databases are searched through via their inbuilt natural language search functionality. Each article has its title, abstract, DOI (if available), author(s), and publication date returned as basic metadata.

Triple Extraction Process

The program extracts RDF triples from the unstructured text of the articles' abstracts utilizing NLTK[3] and the Stanford Core NLP[4] modules. First, constituency parsing is performed to create a composition tree from which the subject(s), predicate, and object(s)/adjective(s) are recorded. Then co-reference resolution and pronomial anaphora occurs to recognize unique entities and ensure that their references are consistent throughout the graph. Once entities are identified, named entity extraction is completed by referencing the MeSH ontology(for biomedical terminology) and Wikipedia(for select named entities). Then, word sense disambiguation is performed on predicates and any remaining subjects to determine the WordNet synset for graph compression.

SPARQL Query Engine

The in-memory graphs are converted to turtle format. These turtle files are then stored in a DOORS directory via the Scribe API. When a SPARQL query is called the program retrieves the .ttl files from the DOORS directory and merges them into a singular in-memory graph. This singular graph can be queried via SPARQL.

Conclusion

Here we presented a method for searching through and retrieving semantic metadata from the open web to expand the Nexus PORTAL-DOORS System (NPDS) records. In order to perform this task a pipeline was developed consisting of a web crawler component to retrieve article metadata from external databases, a triple extraction process to derive logical form triples from the unstructured text of the each article's abstract, and a SPARQL query engine to facilitate semantic metadata retrieval. These components mesh together to create a system that can furnish large amounts of linked semantic metadata for use by NPDS with relatively low time investment.

References

1. Taswell C, TeleGenetics G, Ladera Ranch CA. PORTAL-DOORS infrastructure system for translational biomedical informatics on the semantic web and grid. Proceedings of the American Medical Informatics Association Summit on Translational Bioinformatics, San Francisco, CA. 2008 Mar:43.
2. Knoth P, Zdrahal Z. CORE: three access levels to underpin open access. D-Lib Magazine. 2012 Nov;18(11/12).

3. Bird S, Klein E, Loper E. Natural language processing with Python: analyzing text with the natural language toolkit. " O'Reilly Media, Inc."; 2009 Jun 12.

4. Manning C, Surdeanu M, Bauer J, Finkel J, Bethard S, McClosky D. The Stanford CoreNLP natural language processing toolkit. InProceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations 2014 (pp. 55-60).

A Web Crawler and SPARQL Query Search Agent to Expand and Navigate NPDS Semantic Metadata Records

Shiladitya Dutta, Degrees[1], Carl Taswell, Degrees[2]
[1]Brain Health Alliance, Ladera Ranch, CA, USA

Introduction

The Nexus PORTAL-DOORS System (NPDS)[1] manages lexical metadata and semantic descriptions of resources. It has 3 principal components: Nexus diristry, PORTAL registry, and DOORS directory. This project interfaces with the DOORS directory and the semantic metadata stored within it. Semantic metadata descriptions are a prerequisite for many of the goals laid out for NPDS such as automated meta-analysis. However, linked data semantic descriptions are time consuming to manually annotate. Thus, we have developed a system that can extract semantic metadata from a variety of biomedical resources and provide a method of searching through that metadata via a SPARQL query.

Web Crawler and SPARQL Query Engine

The web crawler component retrieves articles and article metadata from DOAJ, Elsevier ScienceDirect, CORE[2], and PubMed via REST API. The websites' databases are searched through via their inbuilt natural language search functionality. Each article has its title, abstract, DOI (if available), author(s), and publication date returned as basic metadata. From the abstract, triples are extracted and placed in a graph on a document-to-document basis. The in-memory graphs are converted to turtle format. These turtle formatted files are then stored in a DOORS directory via the Scribe API. When a SPARQL query is called the program retrieves the graphs from the database to be queried via the SPARQL query engine.

Triple Extraction Process

The program extracts RDF triples from the unstructured text of the articles' abstracts utilizing NLTK[3] and the Stanford Core NLP[4] modules. First, constituency parsing is performed to create a composition tree from which the subject(s), predicate, and object(s)/adjective(s) are recorded. Then co-reference resolution and pronomial anaphora occurs to recognize unique entities and ensure that their references are consistent throughout the graph. Once entities are identified, named entity extraction is completed by referencing multiple databases (i.e. MeSH and Wikipedia) to label unique entities or jargon. Then, word sense disambiguation is performed on predicates and any remaining subjects to determine the WordNet synset for graph compression.

Conclusion

Here we presented a method for searching through and retrieving semantic metadata from the open web to expand the Nexus PORTAL-DOORS System (NPDS) records. In order to perform this task a pipeline was developed consisting of a web crawler component to retrieve article metadata from external databases, a triple extraction process to derive logical form triples from the unstructured text of each article's abstract, and a SPARQL query engine to facilitate semantic metadata retrieval. These components mesh together to create a system that can furnish large amounts of linked semantic metadata for use by NPDS with relatively low time investment.

References

1. Taswell C, TeleGenetics G, Ladera Ranch CA. PORTAL-DOORS infrastructure system for translational biomedical informatics on the semantic web and grid. Proceedings of the American Medical Informatics Association Summit on Translational Bioinformatics, San Francisco, CA. 2008 Mar:43.
2. Knoth P, Zdrahal Z. CORE: three access levels to underpin open access. D-Lib Magazine. 2012 Nov;18(11/12).
3. Bird S, Klein E, Loper E. Natural language processing with Python: analyzing text with the natural language toolkit. " O'Reilly Media, Inc."; 2009 Jun 12.

4. Manning C, Surdeanu M, Bauer J, Finkel J, Bethard S, McClosky D. The Stanford CoreNLP natural language processing toolkit. InProceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations 2014 (pp. 55-60).

A Web Crawler and SPARQL Query Search Agent to Expand and Navigate NPDS Semantic Metadata Records

Shiladitya Dutta[1]**, Carl Taswell, MD, PhD**[2]
[1]**Brain Health Alliance, Ladera Ranch, CA, USA**

Introduction

The Nexus PORTAL-DOORS System (NPDS)[1] manages lexical metadata and semantic descriptions of resources. It has 3 principal components: Nexus diristry, PORTAL registry, and DOORS directory. This project interfaces with the DOORS directory and the semantic metadata stored within it. Semantic metadata descriptions are a prerequisite for many of the goals laid out for NPDS such as automated meta-analysis. However, linked data semantic descriptions are time consuming to manually annotate. Thus, we have developed a system that can extract semantic metadata from a variety of biomedical resources and provide a method of searching through that metadata via a SPARQL query.

Web Crawler and SPARQL Query Engine

The web crawler component retrieves articles and article metadata from DOAJ, Elsevier ScienceDirect, CORE[2], and PubMed via REST API. The websites' databases are searched through via their inbuilt natural language search functionality. Each article has its basic metadata(title, abstract, author(s), etc.) returned. From the abstract, semantic triples are extracted, placed in a graph, and converted to turtle format on a document-to-document basis. These turtle formatted files are then stored in a DOORS directory via the Scribe API. When a SPARQL query is called the program retrieves the graphs from the database to be queried via the SPARQL query engine.

Triple Extraction Process

The program extracts RDF triples from the unstructured text of the articles' abstracts utilizing NLTK[3] and the Stanford Core NLP[4] modules. First, constituency parsing is performed to create a tree from which the subject(s), predicate, and object(s)/adjective(s) are recorded. Then co-reference resolution and pronomial anaphora occur to recognize unique entities and ensure that their references are consistent throughout the graph. Once the logical-form triples are compiled, they are converted from lexical-based triples to valid RDF by identifying each part of the subject-verb-object triples. This is accomplished by using various databases (i.e. MeSH) for field-specific "jargon" and select named entities, word sense disambiguation for standard words, and literals for numerals and unlinked names.

Conclusion

Here we presented a method for searching through and retrieving semantic metadata from the open web to expand the Nexus PORTAL-DOORS System (NPDS) records. In order to perform this task a pipeline was developed consisting of a web crawler component to retrieve article metadata from external databases, a triple extraction process to derive logical form triples from the unstructured text of each article's abstract, and a SPARQL query engine to facilitate semantic metadata retrieval. These components mesh together to create a system that can furnish large amounts of linked semantic metadata for use by NPDS with relatively low time investment.

References

1. Taswell C, TeleGenetics G, Ladera Ranch CA. PORTAL-DOORS infrastructure system for translational biomedical informatics on the semantic web and grid. Proceedings of the American Medical Informatics Association Summit on Translational Bioinformatics, San Francisco, CA. 2008 Mar:43.
2. Knoth P, Zdrahal Z. CORE: three access levels to underpin open access. D-Lib Magazine. 2012 Nov;18(11/12).
3. Bird S, Klein E, Loper E. Natural language processing with Python: analyzing text with the natural language toolkit. " O'Reilly Media, Inc."; 2009 Jun 12.
4. Manning C, Surdeanu M, Bauer J, Finkel J, Bethard S, McClosky D. The Stanford CoreNLP natural language processing toolkit. InProceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations 2014 (pp. 55-60).

A Web Crawler and SPARQL Query Search Agent to Expand and Navigate NPDS Semantic Metadata Records

Shiladitya Dutta[1], Carl Taswell, MD, PhD[2]
[1]Brain Health Alliance, Ladera Ranch, CA, USA

Introduction

The Nexus PORTAL-DOORS System (NPDS)[1] manages lexical metadata and semantic descriptions of resources. Out of the three principal components,the Nexus diristry, PORTAL registry, and DOORS directory, this project primarily interfaces with the DOORS directory and the semantic metadata stored within it. Semantic metadata descriptions are a prerequisite for many of the goals laid out for NPDS such as automated meta-analysis. However, linked data semantic descriptions are time consuming to manually annotate. Thus, we have developed a automated system by which semantic metadata can be extracted from external biomedical articles and furnished for use in NPDS for a variety of analytical application with relatively low time investment.

Web Crawler and SPARQL Query Engine

The system consists of 3 main sections: a web crawler to retrieve articles, a triple extraction process to extract semantic metadata, and a SPARQL query engine to allow the descriptions to be searched and retrieved. The web crawler component retrieves articles and article metadata from DOAJ, Elsevier ScienceDirect, CORE[2], and PubMed via REST API. The websites' databases are searched through via their inbuilt natural language search functionality. Each article has its basic metadata(title, abstract, author(s), etc.) returned. From the unstructured text of the abstract, semantic triples are extracted, placed in a graph, and converted to turtle format on a document-to-document basis. These turtle formatted files are then stored in a DOORS directory via the Scribe API. When a SPARQL query is called the program retrieves the graphs from the database to be queried via the SPARQL query engine.

Triple Extraction Process

In order to translate the lexical metadata into semantic metadata, the program extracts RDF triples from the unstructured text of the articles' abstracts. First, constituency parsing is performed to create a tree from which the subject(s), predicate, and object(s)/adjective(s) are recorded. Then co-reference resolution and pronomial anaphora occur to recognize unique entities and ensure that their references are consistent throughout the graph. Once the logical-form triples are compiled, they are converted from lexical-based triples to valid RDF by identifying each part of the subject-verb-object triples. This is accomplished by using various databases (i.e. MeSH) for field-specific ”jargon” and select named entities, word sense disambiguation for standard words, and literals for numerals and names. This process yields a set of RDF triples that portray the natural-language information in each abstract in a linked data format.

Conclusion

Here we presented a method for searching through and retrieving semantic metadata from the open web to expand the Nexus PORTAL-DOORS System (NPDS) records. In order to perform this task a pipeline was developed consisting of a web crawler component to retrieve article metadata from external databases, a triple extraction process to derive logical form triples from the unstructured text of each article's abstract, and a SPARQL query engine to facilitate semantic metadata retrieval. These components mesh together to create a system by which resources from the open-web can have their lexical information translated into machine-understandable semantic information with minimal labor requirement from humans, thus laying the groundwork for a variety of applications for which linked data stores are a necessity.

References

1. Taswell C, TeleGenetics G, Ladera Ranch CA. PORTAL-DOORS infrastructure system for translational biomedical informatics on the semantic web and grid. Proceedings of the American Medical Informatics Association Summit on Translational Bioinformatics, San Francisco, CA. 2008 Mar:43.
2. Knoth P, Zdrahal Z. CORE: three access levels to underpin open access. D-Lib Magazine. 2012 Nov;18(11/12).

A Focused Web Crawler and SPARQL Query Search Agent to Expand and Navigate NPDS Semantic Metadata Records

Shiladitya Dutta, Carl Taswell, M.D., Ph.D.
Brain Health Alliance, Ladera Ranch, CA, USA

Abstract: Semantic descriptions are a prerequisite for many of the goals laid out for the Nexus PORTAL-DOORS System, however it is labor-intensive to manually annotate resources from the open-web. To remedy this, we have created CoVaSEA: an automated web crawler/query engine that can search and expand NPDS metadata records. With this system, articles from external biomedical databases can have semantic descriptions extracted from them for use in a variety of computational tasks.

Introduction: The Nexus PORTAL-DOORS System (NPDS)[1] manages lexical and semantic renditions of resources. It has 3 main components: the Nexus diristry, PORTAL registry, and DOORS directory. In order to both expand the DOORS records along with providing a method to search NPDS with a SPARQL query, a semantic section of CoVaSEA(Concept-Validated Search Engine Agent) was developed. Acting as an addition to past lexical-based CoVaSEA systems[2], we hope to furnish a means of expansion and search for NPDS linked data descriptions.

Methods: CoVaSEA integrates several features which benefit NPDS including: (A) An implementation of SPARQL query based semantic search to allow retrieval and manipulation of DOORS linked data descriptions (B) Targeted web-crawling for relevant article metadata from external biomedical literature databases to expand NPDS records (C) Automated translation of free-form text abstracts into RDF triples to derive the semantic representations of lexical data. The system consists of a pipeline in which a web crawler registers articles by converting the abstract text and description to RDF, so that a SPARQL query engine can comb through the retrieved material. The web crawler component searches for articles and their metadata in biomedical literature databases via REST API. Each article has its basic metadata(title, abstract, author(s), etc.) returned. In order to translate the lexical information in the abstract into semantic metadata, CoVaSEA develops RDF triples from the unstructured text of the articles' abstracts. First, constituency parsing is performed to create a tree from which the subject(s), predicate, and object(s) are recorded in a method similar to Rusu D et al.[3]. Then co-reference resolution and pronomial anaphora occur to recognize unique entities and ensure that their references are consistent. Once the logical-form triple post-processing is finished, they are converted from lexical-based triples to valid RDF by identifying each part of the subject-verb-object triples. This is accomplished by using various databases (i.e. MeSH) for field-specific "jargon" and select named entities, word sense disambiguation for standard words, and literals for numerals and names. Finally, the RDF graph is converted to the turtle format, yielding a file that portrays the natural-language information in each abstract in a linked data format. These turtle formatted files are then stored in a DOORS directory via the Scribe API. When a SPARQL query is called the program retrieves the graphs from the database to be queried via the SPARQL query engine.

Conclusion: Here we presented a system in which resources from the open-web can be translated into machine-understandable semantic information and be searched via SPARQL. CoVaSEA has the capability to both search "externally" with the web crawler for semantic data to expand the NPDS knowledge base and "internally" with SPARQL to provide a method to navigate the data inside the DOORS directory. With the distinct advantage that the system is automated, thus can furnish large amounts semantic descriptions on a regular basis, CoVaSEA lays the groundwork for a variety of future NPDS applications for which linked data stores are a necessity.

References

1. Taswell C, TeleGenetics G, Ladera Ranch CA. PORTAL-DOORS infrastructure system for translational biomedical informatics on the semantic web and grid. Proceedings of the American Medical Informatics Association Summit on Translational Bioinformatics, San Francisco, CA. 2008 Mar:43.

2. Bae SH, Craig AG, Taswell C. Expanding Nexus Diristries of Dementia Literature with the NPDS Concept-Validating Search Engine Agent.
3. Rusu D, Dali L, Fortuna B, Grobelnik M, Mladenic D. Triplet extraction from sentences. InProceedings of the 10th International Multiconference" Information Society-IS 2007 Jul (pp. 8-12).

Report BHA-2018-09: A SPARQL Query Search Agent for Finding Web Resources and Creating NPDS Semantic Metadata Records Describing Them

Shiladitya Dutta, Adam Craig and Carl Taswell

***Abstract*—Semantic descriptions are a prerequisite for many of the goals laid out for the Nexus PORTAL-DOORS System, however it is labor-intensive to manually annotate resources from the open-web. To remedy this, we have created CoVaSEA: an automated web crawler/query engine that can search and expand NPDS metadata records. With this system, articles from external biomedical databases can have semantic descriptions extracted from them for use in a variety of computational tasks.**

***Index Terms*—Nexus-PORTAL-DOORS System, CoVaSEA, SPARQL, web-crawler, semantic web**

Introduction

The Nexus-PORTAL-DOORS System (NPDS) is a software system that provides the capability to publish both semantic and lexical resources regarding specific target areas. NPDS has inbuilt REST API services via the Scribe Registrar for record search, retrieval, and publication along with a standardized messaging protocol, enabling exchange among client applications and servers. All remote components are organized according to the Hierarchically Distributed Mobile Metadata architectural style**taswell2010distributed**. The structure of the Nexus PORTAL-DOORS System is sub-divided, in a method analogous to IRIS-DNS, into 3 primary parts: Nexus, PORTAL, and DOORS. The PORTAL registry controls entities that have unique labels and their associated lexical meta-data, which can come in the form of tags, vocabulary terms, or cross references. The DOORS directory contains the semantic meta-data which is primarily comes in the form of RDF descriptions. The Nexus diristry (an aggregation of the terms DIRectory and regISTRY) is a single server that serves as a combination of the PORTAL registry and DOORS directory.**taswell2008doors** The NPDS project also has several other features built in such as Hypothesis and Concept-Validating ontologies, each of which serve to avoid large and cumbersome ontologies,**skarzynski2015solomoncraig2017bridging** and also the capability of having individuals maintain their own Nexus, PORTAL, or DOORS servers. Semantic descriptions are a prerequisite for many of the goals laid out for the Nexus PORTAL-DOORS System**taswell2008portal**, however it is labor-intensive to manually annotate resources from the open-web. To remedy this, we have created CoVaSEA: an automated web crawler/query engine that can search and expand NPDS metadata records. With this system, articles from external biomedical databases can have semantic descriptions extracted from them for use in a variety of computational tasks.

Methods

As an expansion to the past section**baeexpanding**, CoVaSEA integrates several features which benefit NPDS including: **(A)** An implementation of SPARQL query based semantic search to allow retrieval and manipulation of DOORS linked data descriptions **(B)** Targeted web-crawling for relevant article metadata from external biomedical literature databases to expand NPDS records **(C)** Automated translation of free-form text abstracts into RDF triples**lassila1999resource** to derive the semantic representations of lexical data. The system consists of a pipeline in which a web crawler registers articles by converting the abstract text and description to RDF, so that a SPARQL query engine can comb through the retrieved material. The web crawler component searches for articles and their metadata in biomedical literature databases via REST API. Each article has its basic metadata(title, abstract, author(s),

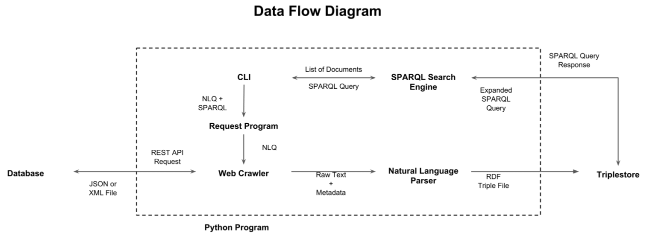


Fig. 1. Data flow diagram of the project and all of its included components

Document created June 20, 2018, revised September 30, 2018; typeset December 30, 2018.

TABLE I
A TABLE OF THE DIFFERENT FEATURES OF THE SEMANTIC AND LEXICAL CoVaSEA

	Semantic	Lexical	Both
Internal	DOORS Directory	PORTAL Registry	Nexus Diristry
External	Semantic Webcrawler Natural Language to RDF Parser	Lexical Webcrawler	Webcrawler
Both	Semantic Search Engine Natural Language to SPARQL Query UI (User Interface)	Lexical Search Engine UI (User Interface)	UI (Possibly)

etc.) returned. In order to translate the lexical information in the abstract into semantic metadata, CovaSEA uses Stanford Parser**manning2014stanford** and NLTK**bird2009natural** to develop RDF triples from the unstructured text of the articles' abstracts. First, constituency parsing is performed to create a tree from which the subject(s), predicate, and object(s) are recorded in a method similar to Rusu D et al.**rusu2007triplet**. Then co-reference resolution and pronomial anaphora occur to recognize unique entities and ensure that their references are consistent. Once the logical-form triple post-processing is finished, they are converted from lexical-based triples to valid RDF by identifying each part of the subject-verb-object triples. This is accomplished by using various databases (i.e. MeSH) for field-specific ”jargon” and select named entities, word sense disambiguation for standard words, and literals for numerals and names. Finally, the RDF graph is converted to the turtle format, yielding a file that portrays the natural-language information in each abstract in a linked data format. These turtle formatted files are then stored in a DOORS directory via the Scribe API. When a SPARQL query is called the program retrieves the graphs from the database to be queried via the SPARQL query engine. In terms of the structure of the program itself, the web crawler/search engine program consists of a number of interdependent components. Each of these components fulfills a general purpose. The methods are:

- **Web Crawler** : Crawls through a multitude of sites in order to retrieve data. Currently the system can search through the article records on DOAJ, PubMed, Elsevier ScienceDirect, and CORE**knoth2012core**. The webcrawler is similar in structure to Seung-Hong Bae**baeexpanding**. The webcrawler receives a requested query and then it searches through the available databases via REST API. The general method for searching through the databases is first a general lexical search query via the provided natural language query. The database then returns a number of articles. From there, the crawler searches through the list of returned articles and retrieves the basic-metadata from each one. It then passes the basic meta-data to the natural language parser. The basic meta-data constitutes of the abstract, title, author(s), database of origin, DOI(if available), and date of publication. The abstract will have its triples extracted and the rest will be included in the named graph of the article via the DublinCore ontology **weibel1997dublin**. It should be noted that the general search query returns twice the amount of requested queries by the user. This is due to the fact that, in general, the provided lexical search API doesn't have filtering capabilities. This means that we receive many articles that aren't suitable for a variety of reasons (i.e. unsupported format or different language than English). Thus we have to remove those, but in order to retrieve the amount of articles requested we must ensure that we have enough that it is more or less assured that only one general query will retrieve enough valid articles.
- **Natural Language Parser**: Receives the raw text from the web crawler and then parses it into logical form triples. Then from the logical form triples they are parsed into RDF/XML triples and put into a graph.

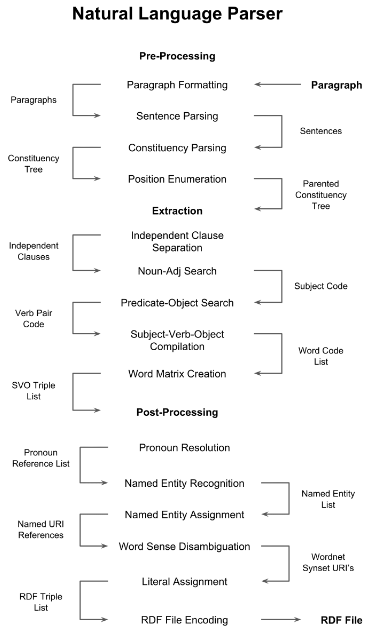
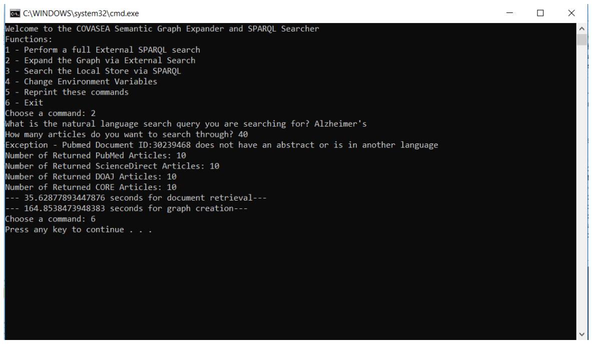


Fig. 2. Data flow diagram of the Natural Language to RDF parser

  - Pre-processing: Designed in order to pre-process the raw text to prepare it for parsing and also to perform tasks that depend on the raw texts such as Named Entity Recognition**finkel2005incorporating** and Co-Referencing**clark2015entity**. Pre-processing preparation includes removing troublesome characters, re-spacing the text to ensure homogeneity, and parsing the paragraphs into sentences.
  - Extraction: Uses a constituency parse to extract relevant logical form triples. First independent clauses are extracted by separating out the phrases that have a subject and a verb phrase. Note that this system of separation only works on clauses that are completely separate and not clauses that are nested within one another or are interlaced. First a constituency parse occurs on each sentence. Then breadth-first and depth-first search are used to find subject and verb-object phrases. The breadth-first search is used to find the highest noun phrase in the tree. This noun phrase is then split up via conjunctions and any proximal adjectives are linked to the nouns. Then a two stage breadth first, depth first search is used to find the verb. Breadth first search is used to find the highest verb phrase. Then depth first search is used to find the verb specifically so that it can be separated from the prepositional or adjective section of the verb phrase. The verb-object phrases are paired since in general the verb and object need to be specifically related by a link in order to maintain their relative positions. These phrases are then broken into their individual parts and tagged with positions. For the nouns, any proximal adjectives are put into a specialized adjectives list. This is so that they can be directly linked to their representative nouns.
  - Post-processing: This step translates the logical form triples into RDF triples. The step primarily consists of graph compression. It does this by assigning URI's to each part of the logical form triples. First ”jargon” and some named entities are assigned URI's via databases. Then standard words are assigned via WordNet**miller1995wordnet** synsets using context and part of speech**toutanova2000enriching**. This step is important since it applies to most of the parts of the sentence and is vital to capturing the full meaning behind the sentence. This also is extremely useful in graph compression so that the graph isn't purely lexical-based but rather has references that are broad. The algorithm that is used in order to perform this word sense disambiguation is Lesk**banerjee2002adapted**. Finally named entities and number are assigned to literals. These are all then packaged into a named graph.

- **SPARQL Analysis Engine**: Receives SPARQL query from user and then searches through the triple store to extract relevant data for the user. In order to increase ease-of-use for a user a feature is implemented to allow for constructing a SPARQL query in normal conjunctive form. This is where a series of statements conjoined by AND/OR statements create a query. However, this system doesn't allow yet for the same level of detail that can be achieved by inputting a raw SPARQL query.
- **Triplestore/QuadStore**: This triple store is used in order to store RDF triples that describe documents. The triplestore/quadstore consists of named graphs (in the .ttl format) each describing an article. The graphs consists of two different sections. The first section describes the generic metadata of the article and the second section describes the semantic meaning of the abstract.
- **CLI**: The CLI (Command Line Interface) is how the user interacts with the program. It includes various features to interface with the program. However, its primary objective is to get input and display output. It will take a SPARQL Query and search query for an input and return a list of relevant documents as an output. The CLI interface consists of 6 different options. The first is to perform a full external SPARQL Query search utilizing the web crawler and the SPARQL query engine. The second option is just to expand the stores via the web crawler. The third option is to search the existing stores via SPARQL. The fourth option is to change the environmental variables which will be a necessity for first time setup. Their are three variables to change: the Java installation, StanfordNLP installation, and location of local stores. The fifth command is to reprint the previous commands and the sixth command is to exit the program.



```
Welcome to the COVASEA Semantic Graph Expander and SPARQL Searcher
Functions:
1 - Perform a full External SPARQL search
2 - Expand the Graph via External Search
3 - Search the Local Store via SPARQL
4 - Change Environment Variables
5 - Reprint these commands
6 - Exit
Choose a command: 2
What is the natural language search query you are searching for? Alzheimer's
How many articles do you want to search through? 40
Exception - Pubmed Document ID:30239468 does not have an abstract or is in another language
Number of Returned PubMed Articles: 10
Number of Returned ScienceDirect Articles: 10
Number of Returned DOAJ Articles: 10
Number of Returned CORE Articles: 10
--- 35.62877893447876 seconds for document retrieval---
--- 164.8538473948383 seconds for graph creation---
Choose a command: 6
Press any key to continue . . .
```

Fig. 3. A screenshot of the CLI interface

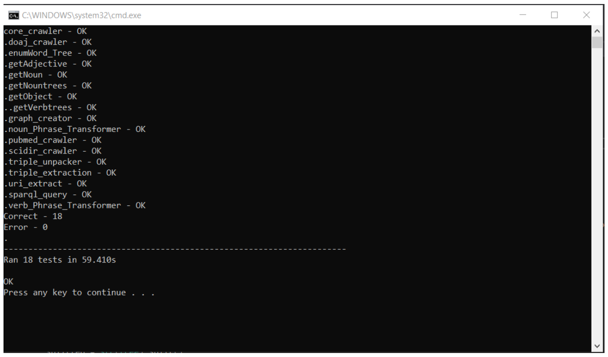
- **Interface Component**: The central component is the juncture between the Command Line Interface and the other components. The central component handles interface display and also acts as a center point to pass arguments to the other components.

These items integrate to form a whole system. First the user inputs a Natural Language Query(NLQ) and a SPARQL query into the CLI components. This is then passed to the interface component which calls the webcrawler. The webcrawler goes through the various databases and retrieves the raw text and the metadata which is passed to the natural language parser. On each articles, the natural language parser transforms the text and metadata into semantic triples. These triples are then stored in .ttl files in a triplestore/quadstore. Then the interface components calls the SPARQL query engine. The SPARQL query engine searches through the triplestore/quadstore and returns a list of articles which is outputted to the user via the CLI. The system is built in Python via the Visual Studio Code environment. The format of the project is such that each of the major sections, natural language parser, web crawler, unit test, central interface, and SPARQL query engine, each have their own Python class. The classes, with the exception of the central interface, don't call each other in a pipeline, but rather the central interface calls all of the individual classes and passes data between them. However, the interface doesn't change anything so effectively the flow is between the components. The system uses a variety of external code libraries in order to complete its objectives. NLTK and the Stanford Parser are used in the natural language parser. NLTK is used primarily for the tree manipulation, sentence parsing, Part-Of-Speech(POS) tagging, and the word sense disambiguation. The Stanford Parser is used for constituency parsing, co-reference resolution, and named entity recognition. It should be noted that the Stanford Parser is used from Java by instantiating a server. The system also uses RDFLib in order to manipulate the RDF triples. RDFLib is used in order to extract, build, and search the RDF files. The system currently uses turtle for its RDF format. The web crawler uses requests to make and format the REST API calls and xmljson in order to convert returned XML to JSON for easy searching. The system uses the default unittest system to perform unit testing on all of the components.

Results

Note that all tests are run on a computer with 16 GB of RAM and a six-cored i7-8750H with a base-clock of 2.20 GHz. The process uses only one core. The connection speeds are a download speed of 18.40 Mgbs (Megabytes per second) and an upload speed of 1.68 Mgbs. One result in particular is that the setting for the only key triples and all triples is similar. This is a setting in which if on only the key triples or triples describing subject-object relations are transferred and those triples that refer to subject-adjective relations are filtered out. The run time is probably similar due to the selection process and the added time needed to process the extra triples balancing out.

Unit testing has been performed on all methods in the CoVaSEA library to ensure functionality and to facilitate test-driven development as described in Software Design, Development, and Coding **dooley2017software**.



```
core_crawler - OK
.doaj_crawler - OK
.enumWord_Tree - OK
.getAdjective - OK
.getNoun - OK
.getNountrees - OK
.getObject - OK
..getVerbtrees - OK
.graph_creator - OK
.noun_Phrase_Transformer - OK
.pubmed_crawler - OK
.scidir_crawler - OK
.triple_unpacker - OK
.triple_extraction - OK
.uri_extract - OK
.sparql_query - OK
.verb_Phrase_Transformer - OK
Correct - 18
Error - 0
.
----------------------------------------------------------------------
Ran 18 tests in 59.410s

OK
Press any key to continue . . .
```

Fig. 4. Screenshot of Unit Testing Results. Some results are cutoff by the Resource Exceptions

Discussion

The primary point of possible expansion for the system is the Natural-Language Parser. Currently, the parser faces difficulty in 3 main areas: relevance of triples, accuracy of triples, and runtime. The first problem is the relevance of triples and how good they summarize the arguments. There are several methods which we can take inspiration from**leskovec2004learningshang2011enhancing**. Most of these focus on document summarization by extracting the semantic graph structure of the document. Most of these approaches use some form of pattern recognition to identify potentially valuable triples. It is possible to use a system similar to this, though it would require a training set to be effective. The second issue is of sentence format. The system currently can only extract triples from a small set of sentence formats. Thus it is unable to take on sentence formats that are more advanced in nature (i.e. ”In spite of a lack of operational knowledge, which could be expected of any animal, the monkey was able to miraculously, and without any training, operate the machine”). In the previous example the parser will not be able to extract the nested statements. The best way to remedy this is to use a semantic frame solution such as FrameNet**baker1998berkeley** or Boxer**bos2008wide**, which was the approach taken by FRED**gangemi2017semantic**. The final problem is that of optimization. The runtime for the parser in its current state is relatively extreme in that it takes 30 seconds to parse a paragraph. This impedes us from performing whole articles parses with reasonable runtime. The best way to solve this is to either use more advanced hardware or rewrite certain sections of code to optimize. The next section of improvement is the SPARQL query engine. The SPARQL query engine relies upon local graph download which could lead to large run times later down the road. This leads to runtime constraints with large graphs. This could be remedied by implementing the capability to access distributed

TABLE II
RUN TIMES AND OUTPUTS OF COVASEA

Function	Input	Result	Run Time

Full (Internal + External)	Number Requested = 20
	Query = Dementia,
	SELECT DISTINCT ?author ?title WHERE {?a dc:contributor ?author . ?a dc:title ?title}
		20 files
		20 results
			245.120 seconds

	Number Requested = 100
	Query = Dementia,
	SELECT DISTINCT ?author ?title WHERE {?a dc:contributor ?author . ?a dc:title ?title}
		100 files
		98 results
			1307.42 seconds

SPARQL Internal	Query = See Above # of Articles = 225	220 results	5.731 seconds
Web Crawler Expansion	Number Requested = 20 Query = Parkinson's	20 files	102.324 seconds
	Number Requested = 100 Query = Parkinson's	100 files	483.126seconds
Triple Extraction	Only Key Triples = True Sentence = Parkinson's is a serious brain disorder. It takes multiple years to develop. The cause of Parkinson's is a lack of dopamine receptors in the brain. Research has been done to cure and diagnose Parkinson's.	3 triples	5.388 seconds
	Only Key Triples = False Sentence = Parkinson's is a serious brain disorder. It takes multiple years to develop. The cause of Parkinson's is a lack of dopamine receptors in the brain. Research has been done to cure and diagnose Parkinson's.	7 triples	5.699 seconds

graphs**quilitz2008querying**. In addition SPARQL query expansion should be implemented to increase the breadth of SPARQL searches and to implement a reasoning engine such as RACER**haarslev2003racer** or F-OWL**zou2004f**.

Conclusion

Here we presented a system in which resources from the open-web can be translated into machine-understandable semantic information and be searched via SPARQL. CoVaSEA has the capability to both search ”externally” with the web crawler for semantic data to expand the NPDS knowledge base and ”internally” with SPARQL to provide a method to navigate the data inside the DOORS directory. With the distinct advantage that the system is automated, thus can furnish large amounts semantic descriptions on a regular basis, CoVaSEA lays the groundwork for a variety of future NPDS applications for which linked data stores are a necessity.

Acknowledgment

I give my thanks to the other members of the Brain Health Alliance 2018 program for their assistance on technical and methodological points.

BHAVI 2018 Fall Quarter Update[*]

Shiladitya Dutta

created December 14, 2018, revised December 28, 2018

Primary Research Project

- Description of progress on your primary research project;
- goals expected at beginning of quarter;
- tasks completed by end of quarter;
- citations for any relevant literature found during the quarter not previously listed in your prior report references for the project;
- STEMM competitions, conferences or journals considered for possible publication of project;
- report title for each submission planned or completed;
- reports already submitted and status – rejected, accepted or published;
- request to continue primary research project versus change to different primary research project.

Most of the development on the primary research project has been in the area of developing the software. The software development has been focused on improving the usability of CoVaSEA. This includes optimizing the code and making it easier for a user to download CoVaSEA on their system. Most of the runtime optimization has been in the lexical to semantic converter. This includes creating an easier to use user interface, allowing the user to change the location of meta-data storage, and creating an instruction set. It also includes the development of a SPARQL Builder form to allow users to build a search query if they don't know SPARQL. The starting point of the software in the beginning of the Fall Quarter is detailed in the end of summer technical report: BHA201809ShiladityaDutta0930 [1].

The goals in the beginning of the quarter were to make CoVaSEA more reliable and improve the usability of CoVaSEA. This includes improving the web crawler, allowing for more search options for the user, and reconfiguring the code so that it could more easily be transferred onto another system. I believe that I have achieved many of these goals for CoVaSEA. The web crawler was optimized so that it could access a larger amount of references. The inclusion of the SPARQL Builder form allows for users to build SPARQL queries without SPARQL syntax. Finally, the code has been improved such that it is smoother for a user to download CoVaSEA.

Relevant Citations that have not been previously listed:

- Paper describing the Scribe API of the NPDS system [2]
- Paper describing the co-referencing method used by the StanfordCoreNLP toolkit [3]
- Paper describing the dependency parsing method that is used by the StanfordCoreNLP toolkit. [4]
- Example of paper that utilizes semantic triple extraction for summarization of texts that are relevant to the biomedical field. [5]

[*] Document created December 14, 2018, revised December 28, 2018; typeset December 30, 2018.

- Another example of a paper describing a method to extract RDF triples from a paper and discern the key triples via a trained machine learning classifier.[6]
- Paper describing the part of speech tagger that is used by NLTK. [7]
- Paper describing the DublinCore ontology which is used in order to describe citation meta-data in the RDF files.[8]

One competition that I wish to submit to is the Golden Gate Science Fair (GGSF). This science fair is the science fair for the San Francisco area and provides a pathway to participation at the Intel International Science and Engineering Fair (ISEF). One note that should be made is that GGSF doesn't preside over my region. Rather students in Pleasanton would normally participate in the Alameda County Science and Engineering Fair (ACSEF). However, ACSEF has went on a 1-year hiatus, thus all science projects are being routed to GGSF.

Titles:

- AMIA: A Focused Web Crawler and SPARQL Query Search Agent to Expand and Navigate NPDS Semantic Metadata Records
- Brain Informatics: SPARQL-Based Search Engine and Agent for Finding Brain Literature and Converting References to NPDS Metadata Records
- IEEE BIBM: SPARQL-Based Search Engine and Agent for Finding Biomedical Literature and Converting References to NPDS Metadata Records

The first submission was planned to be to the AMIA conference. However, the submission was decided not to be sent in at the last minute. The second submission was to the Brain Informatics 2018 Conference as an abstract/ oral slide presentation. The submission was accepted and presented by Dr. Taswell at the Brain Informatics 2018 Conference on December 9, 2018.[9] The third submission was a 3-page poster paper to the International Conference on Bioinformatics and Biomedicine (BIBM). The submission was rejected from the conference.

I would like to continue the development of CoVaSEA into the Winter Quarter. I have a few objectives that I will pursue during Winter Quarter. The highest priority objective is to create an efficient CLI and GUI for CoVaSEA. There should be two version of the GUI: the one for expert semantic web engineers who are proficient with SPARQL and the one for users who aren't familiar with the technical nuances of SPARQL. The approach for this is to review the search GUIs of popular literature databases and take note of features which are beneficial. Also purchase various textbooks on good GUI design. A GUI is absolutely necessary for the completion of the software so that users can more easily use CoVaSEA. The second objective is the expansion of the web crawler. This mainly consists of expanding the capabilities of the search system of the web crawler such that it can handle more search parameters. The third objective is the adding of more supported sentence formats to the lexical-to-semantic converter. This would allow the converter to parse more advanced sentences accurately.

Number of Records entered into the Nexus Diristry: **16**. All of the records were entered into the Davinci Diristry.

Secondary Research Project

- Description of progress on your secondary research project;
- goals expected at beginning of quarter;
- tasks completed by end of quarter;
- citations for any relevant literature found during the quarter not previously listed in your prior report references for the project;
- STEMM competitions, conferences or journals considered for possible publication of project;

- report title for each submission planned or completed;
- reports already submitted and status – rejected, accepted or published;
- request to continue secondary research project versus change to different secondary research project.

No Secondary Research Project. Originally was to be second author on Arnav Bansal's project. However, Arnav Bansal left the BHAVI program thus I have not had a secondary research project. For a secondary research project, if possible, I wish to work on a research project in the semantic web field. The most appropriate project is the FAIR metric project since my RDF triple extraction has significance in that project area.

Education Project

- CSSE = Computational Sciences & Software Engineering
- CSSE course work completed at school None
- CSSE course work completed independently online Linked Data Engineering by Open HPI: https://open.hpi.de/courses/semanticweb2016?locale=en
- CSSE textbooks purchased and read The Algorithm Design Manual by: Steven S. Skiena [10]

Discussion

Any general comments, suggestions, requests regarding the BHAVI program?

One comment that I have is that it would be beneficial to have a list of mantras of Brain Health Alliance made available to the BHAVI students. There are many mantras in BHAVI including "RTFM" and "Real software that really works". These are general guiding principals for participants in the program, thus is would be beneficial to have a list of mantras for students, especially new students, to read and understand.

References

[1] C. T. Shiladitya Dutta Adam Craig, "A sparql query search agent for finding web resources and creating npds semantic metadata records describing them," Brain Health Alliance Virtual Institute, Technical Report, 2018.

[2] A. Craig, S. H. Bae, T. Veeramacheneni, *et al.*, "Web service apis for scribe registrars, nexus diristries, portal registries and doors directories in the npd system.," in *SWAT4LS*, 2016.

[3] K. Clark and C. D. Manning, "Entity-centric coreference resolution with model stacking," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, vol. 1, 2015, pp. 1405–1415.

[4] J. R. Finkel, T. Grenager, and C. Manning, "Incorporating non-local information into information extraction systems by gibbs sampling," in *Proceedings of the 43rd annual meeting on association for computational linguistics*, Association for Computational Linguistics, 2005, pp. 363–370.

[5] Y. Shang, Y. Li, H. Lin, *et al.*, "Enhancing biomedical text summarization using semantic relation extraction," *PLoS one*, vol. 6, no. 8, e23862, 2011.

[6] J. Leskovec, M. Grobelnik, and N. Milic-Frayling, "Learning sub-structures of document semantic graphs for document summarization," 2004.

[7] K. Toutanova and C. D. Manning, "Enriching the knowledge sources used in a maximum entropy part-of-speech tagger," in *Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics-Volume 13*, Association for Computational Linguistics, 2000, pp. 63–70.

[8] S. Weibel, "The dublin core: A simple content description model for electronic resources," *Bulletin of the American Society for Information Science and Technology*, vol. 24, no. 1, pp. 9–11, 1997.

[9] S. Dutta and C. Taswell, “Sparql-based search engine and agent for finding brain literature and converting references to npds metadata records,” in *11th International Conference on Brain Informatics*, 2018.

[10] S. S. Skiena, *The algorithm design manual: Text*. Springer Science & Business Media, 1998, vol. 1.

BHAVI 2019 Winter Quarter Report*

Shiladitya Dutta

created March 19, 2019, revised March 19, 2019

Primary Research Project

Description: The main objective at the beginning of Winter Quarter was to make CoVaSEA more usable. I believe that this goal has been accomplished in a variety of ways. Currently, the main progress that has been made during Winter Quarter is the development of a GUI for CoVaSEA. This GUI consists of 4 main options. The first option is for a complete search which includes an external web crawler search and an internal SPARQL query search. The second option is for only an external web crawler search to convert records and the third option is an internal SPARQL query search to analyze existing records. The last option is the ability to change the environment variables of CoVaSEA to allow for easier set-up. In addition, the SPARQL query builder form has been finalized. It consists of three main sections. The first section is choosing the conditions for the SPARQL query search. The second section is choosing the outputs for the search and the last section is choosing any post-conditions.

In addition to the addition of new features, I have attempted to make this project more easy-to-use and reproducible. Firstly, I have added an external environment variables file so that the environment variables can be changed without having to change anything in the code. I have also written a user installation manual such that it is easier to learn how to install the system. There has also been other minor changes such as moving the semantic metadata records file to the CoVaSEA program file so that the environment variable doesn't need to be changed. I have also tested these features with another student. Other than these, I have also continued work on the lexical-to-semantic translator. Most notably, I have spent time to improve the parsing capabilities and runtime.

Citations:

- [1] - A paper describing a semantic reasoning engine using direct logic to make inferences about semantic graphs in a database.
- [2] - This paper describes Apache Jena, a semantic triplestore database. Jena could be a possible triplestore system for CoVaSEA.
- [3] - SHACL is supposed to act as a potential replacement for SPARQL. SHACL differs from SPARQL in SPARQL is condition based while SHACL is shape based.
- [4] - This paper describe a linear logic theorem prover. This is another possible approach to semantic reasoning in the CoVaSEA internal search engine.
- [5] - A paper on PELLET a semantic reasoning engine. PELLET is a relatively well-known OWL 2 based semantic reasoner.
- [6] - This article details Google's approach to lexical-to-semantic markup translation. Currently, most translations systems, including CoVaSEA's, utilize a multi-stage discourse representation theory parser. However, Google has taken a new approach by simply having a neural network model perform the parsing. This approach is very interesting and elegant, though it does have its own fair share of problems.

*Document typeset March 19, 2019.

- [7] - An article describing an open source software library which implements a neural machine parse: Opennmt.

Conferences/Competitions:

- GGSF - March 23rd (Already included in Fall Quarter report, however competition technically occurring in Spring Quarter) Also it should be noted that California State Science Fair (CSSF) and/or Intel International Science and Engineering Fair (ISEF) may occur depending on my performance at GGSF. Both of these competitions should occur during Spring Quarter.

Statement: I wish to continue my current primary research project.

Secondary Research Project

Description: Nothing to report. I didn't have a secondary research project since it was originally planned to be with Arnav Bansal. However, Arnav Bansal left the program thus I didn't have a secondary research project.

Statement: For my secondary research project, I wish to contribute to Adam Craig's research on FAIR metrics. This is due to its potential relevance to my current work and my involvement in the project thus far.

Software Education Project

- I am not currently enrolled in any CSSE coursework at my school.
- I have not completed or am currently enrolled in any CSSE coursework online. However, I am currently doing USA Computing Olympiad (USACO). USACO is a competitive coding competition in the same vein as TopCoder, ACM ICPC, or HackerRank. USACO administers a series of 4-hour online tests from December to March. Each test consists of 3-4 algorithmic coding problems to complete. Though not necessarily coursework, it is worth noting this activity.
- I have continued reading the Algorithm Design Manual. Most notably I have spent more time reading about graph theory and tree traversal algorithms. This is because these could possibly be useful for the tree parsing section of CoVaSEA. I have also read more about data structures in an attempt to learn possible ways to optimize the runtime of CoVaSEA. [8]

Discussion

I would suggest the compilation of a complete list of BHAVI sayings and phrases. This would include phrases such as ”Real software that really works”, ”RTFM”, ”Deliver the deliverables”, and ”Elite program for elite students”. This would provide a simple and somewhat humorous introduction to the cardinal values of BHAVI for new students. I have no other suggestions or requests in regards to the BHAVI program.

References

[1] N. Bassiliades, G. Antoniou, and I. Vlahavas, “A defeasible logic reasoner for the semantic web,” *International Journal on Semantic Web and Information Systems (IJSWIS)*, vol. 2, no. 1, pp. 1–41, 2006.

[2] J. J. Carroll, I. Dickinson, C. Dollin, *et al.*, “Jena: Implementing the semantic web recommendations,” in *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, ACM, 2004, pp. 74–83.

[3] J. Corman, J. L. Reutter, and O. Savković, “Semantics and validation of recursive shacl,” in *International Semantic Web Conference*, Springer, 2018, pp. 318–336.

[4] J. Rao, P. Küngas, and M. Matskin, “Composition of semantic web services using linear logic theorem proving,” *Information Systems*, vol. 31, no. 4-5, pp. 340–360, 2006.

[5] E. Sirin, B. Parsia, B. C. Grau, *et al.*, “Pellet: A practical owl-dl reasoner,” *Web Semantics: science, services and agents on the World Wide Web*, vol. 5, no. 2, pp. 51–53, 2007.

[6] Y. Wu, M. Schuster, Z. Chen, *et al.*, “Google’s neural machine translation system: Bridging the gap between human and machine translation,” *arXiv preprint arXiv:1609.08144*, 2016.

[7] G. Klein, Y. Kim, Y. Deng, *et al.*, “Opennmt: Open-source toolkit for neural machine translation,” *arXiv preprint arXiv:1701.02810*, 2017.

[8] S. S. Skiena, *The algorithm design manual: Text*. Springer Science & Business Media, 1998, vol. 1.