Towards Implementing Semantic Literature-Based Discovery with a Graph Database

Dimitar Hristovski*, Andrej Kastrin†, Dejan Dinevski‡ and Thomas C. Rindflesch§
*Faculty of Medicine, University of Ljubljana, Ljubljana, Slovenia
†Faculty of Information Studies, Novo mesto, Slovenia
‡Faculty of Medicine, University of Maribor, Maribor, Slovenia
§National Library of Medicine, Bethesda, MD, USA

***Abstract*—Literature-based discovery (LBD) combines known facts in the scientific literature to generate discoveries, or hypotheses. Potential discoveries have the form of relations between concepts; for example, in the biomedical domain (on which we concentrate), a drug may be determined to treat a disease other than the one for which it was intended. We view the domain knowledge underpinning LBD as a network consisting of a set of concepts along with the relations connecting them. In the study presented here, we used SemMedDB, a database of semantic relations between biomedical concepts extracted with SemRep from MEDLINE. SemMedDB is distributed as a MySQL relational database, which is not optimal for dealing with network data. We transformed and uploaded SemMedDB into a Neo4j graph database, and implemented the basic LBD discovery algorithms with the Cypher query language. We conclude that storing the data needed for semantic LBD is facilitated by a graph database. Also, implementing LBD discovery algorithms is conceptually simpler with a graph query language when compared with standard SQL.**

Keywords–Data science; Databases; Data mining; Semantics; Literature-based discovery.

I. INTRODUCTION

The corpus of biomedical papers in online bibliographical repositories, nowadays referred to as the bibliome, is of considerable size and complexity. The amount of biomedical literature available is growing at an explosive speed, but a large amount of useful information in it remains undiscovered [1]. For example, MEDLINE contains over 24 million references to biomedical journals, with approximately 3000 references added each day. Exploiting this information effectively crucially depends on linking information from diverse sources into coherently interpretable knowledge. In this regard, development of automated knowledge discovery tools is of utmost importance. Computer-based methods can greatly complement manual literature management and knowledge discovery from biomedical data [2].

Literature-based discovery (LBD) is a mature text mining methodology for automatically generating hypotheses for scientific research by uncovering hidden, previously unknown relationships, from existing knowledge. The LBD methodology was pioneered by Swanson [3], who proposed that dietary fish oils might be used to treat Raynaud's disease because they lower blood viscosity, reduce platelet aggregation, and inhibit vascular reactivity. Swanson's hypothesis was validated by DiGiacomo et al. [4]. Swanson's approach is based on the assumption that there exist two nonintersecting scientific domains. Knowledge in one domain may be related to knowledge in the other, without the relationship being known. The methodology of LBD relies on the idea of concepts relevant to three literature domains: X, Y, and Z (Figure 1). For example, suppose a researcher has found a relationship between disease X and a gene Y. Further suppose that a different researcher has studied the effect of substance Z on gene Y. The use of LBD may suggest an XZ relationship, indicating that substance Z may potentially treat disease X. Many researchers have replicated Swanson's discoveries using various approaches: Gordon and Lindsay [5], [6], Webber et al. [7], Hristovski et al. [8], Srinivasan [9], Cameron et al. [10]. For a recent review of LBD tools and approaches, see Hristovski et al. [11].

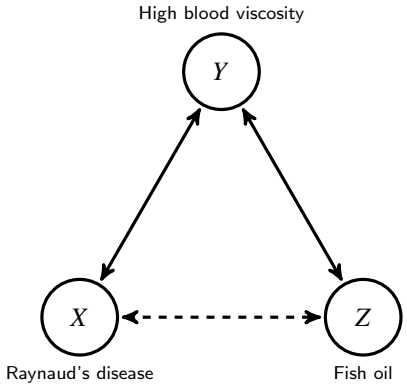

Figure 1. Swanson's XYZ discovery model

Current knowledge of a particular biomedical domain can be viewed as a set of concepts and the relationships among them [12]. For example, in pharmacogenomics relations among genes, diseases, and chemical substances constitute an important part of knowledge. These associations can be naturally represented as a graph consisting of nodes and edges, where the former represent concepts and the later relationships.

The great majority of existing LBD systems are co-occurrence based. Co-occurrence represents the simplest way to capture associations between biomedical concepts (nodes), but it does not express the meaning of the relationship between those concepts. Even widely used document retrieval systems, such as PubMed, typically have no access to the meaning of the text being processed [13]. To fill the gap between raw text and its meaning Rindflesch [14] developed the SemRep system. SemRep is a rule-based, symbolic natural language processing system that recovers semantic propositions from the biomedical research literature. The system relies on domain knowledge in the Unified Medical Language System (UMLS) [15] to provide partial semantic interpretation in the form of predications consisting of UMLS Metathesaurus concepts as arguments and UMLS Semantic Network relations as predicates. SemRep uses a partial semantic analysis based

 ISBN: 978-1-61208-408-4

on the SPECIALIST Lexicon [16] and MedPost tagger [17]. Each noun phrase in this analysis is mapped to a concept in the Metathesaurus using MetaMap [18]. Both syntactic and semantic constraints are employed to identify propositional assertions. For example, SemRep extracts three predications from the text "dexamethasone is a potent inducer of multidrug resistance-associated protein expression in rat hepatocytes":

1) Dexamethasone STIMULATES Multidrug Resistence-Associated Proteins
2) Multidrug Resistance-Associated Proteins PART_OF Rats
3) Hepatocytes PART_OF Rats

SemRep extracts 30 predicate types expressing assertions in clinical medicine (e.g., TREATS, ADMINISTERED_TO), substance interactions (e.g., INTERACTS_WITH, STIMULATES), genetic etiology of disease (e.g., CAUSES, PREDISPOSES), and pharmacogenomics (e.g., AUGMENTS, DISRUPTS). The program has been run on all of MEDLINE and the extracted predications deposited in a MySQL database named SemMedDB [19] updated quarterly and available to researchers.

In the last decade, various NoSQL (often interpreted as Not only SQL (Structured Query Language)) technologies for storing data have emerged. The term NoSQL refers to schemaless database technology such as key-value stores (e.g., Apache Cassandra), document stores (e.g., MongoDB), and graph databases (e.g., AllegroGraph, OpenLink Virtuoso, Neo4j). In this paper, we examine the Neo4j graph database as an alternative for LBD on semantic relations extracted from MEDLINE, because Neo4j is particularly useful for storing data structured as a graph.

II. METHODS

A. Exporting data from SemMedDB

SemMedDB contains detailed information about all the semantic relations extracted with SemRep from MEDLINE. The general form of the semantic relations is Subject-Relation-Object, for example Dopamine-TREATS-Parkinson Disease. This particular relation was extracted 1080 times from different text sentences. We will refer to each of these 1080 extractions as a semantic relation instance. In other words, we say that Dopamin-TREATS-Parkinson Disease is a semantic relation with 1080 instantces. For LBD we were interested in aggregated semantic relations, which means that for each semantic relation we wanted to have the corresponding number of instances. In SemMedDB, there is no single place where we could find aggregated semantic relations. Therefore, we exported only the semantic relation instances in a text CSV (Comma-Separated Values) format. For each instance, the following fields were exported: subject_concept_id, subject_name, relation_type, object_concept_id, and object_name. We exported all the 52 616 158 semantic relation instances.

B. Aggregating the semantic relations and loading into Neo4j

Originally, we thought of aggregating the semantic relations with shell tools, such as Awk, sort and uniq. However, we decided to put Neo4j to a test. We decided to load and aggregate the data in Neo4j with the `LOAD CSV` command, which is used for loading external text data (Figure 2). This command is part of Cypher, which is Neo4js graph query language. The command reads the input file line by line. Each input line is split into fields. After loading, we used the `MERGE` command for the subject part of the instance. If the concept that appeared as subject was not found, then, a node with label "Concept" was created with the corresponding concept_unique_id, name and semantic type, and the frequency counter was set to 1. All this was done with the `ON CREATE` part of the `MERGE` command. However, if there already was a node for the subject concept, then only its frequency counter was increased with the `ON MATCH` part of the `MERGE` command. A similar procedure was repeated for the object concept of the semantic relation instance with another `MERGE` command. And finally, a `MERGE` command was used for the semantic relation itself. If there was no relation between the current subject and object nodes, a new relation was created with the relation type, and the frequency counter was set to 1 within the `ON CREATE` part of the commend. If the relation already existed then its frequency counter was increased with the `ON MATCH` part. After the end of this procedure, we expected to have a graph in which the nodes were the subjects and/or objects of the semantic relations, with the nodes having relations between them that corresponded to the semantic relations between the arguments. This procedure worked well with a few million relation instances. Unfortunatelly, it was not able to load the entire set of semantic relation instances, despite the fact that we used the `USING PERIODIC COMMIT` command as instructed in the Neo4j documentation.

We then used a few other approaches. For example, loading and aggregating only the nodes as described above worked well and finished successfully. However, we could not load all the relations at once. We then tried loading the relations in batches by adding corresponding `START` and `LIMIT` parameters to the `LOAD CSV` command. In addition to this being too slow, there was another problem with this command. It does not allow setting the label of a node based on a field value in the input line, which we wanted to do. We wanted the semantic type of the nodes to become one of its labels. For example, we wanted to have nodes such as "c:Concept:dsyn" which in Neo4j terminology is read as node "c" with labels "Concept" and "dsyn" (where "dsyn" is our abbreviation for the concept semantic type "Disease or Syndrome").

```
LOAD CSV FROM 'semmed_sub_rel_obj.txt'
  AS line
WITH line
MERGE (c1:Concept {cui: line[0]})
ON CREATE SET c1.name=line[1],
  c1.type=line[2], c1.freq=1
ON MATCH SET c1.freq = c1.freq + 1
MERGE (c2:Concept {cui: line[4]})
ON CREATE SET c2.name=line[5],
  c2.type=line[6], c2.freq=1
ON MATCH SET c2.freq = c2.freq + 1
MERGE (c1)-[r:Relation {type:
  line[3]}]->(c2)
ON CREATE SET r.freq = 1
ON MATCH SET r.freq = r.freq + 1;
```

Figure 2. Loading semantic relation instances with the `LOAD CSV` Neo4j command.

 ISBN: 978-1-61208-408-4

We next tried to aggregate the instances with several Awk scripts and with the sort and join shell commands. During this process, for each semantic relation, all the instances of that semantic relation were replaced with a single text line, which, in addition to the instance level fields, also contained the total number of instances. We noticed that sometimes the same concepts appeared with different semantic types in different instances. All these semantic types were also aggregated as a comma-delimited list. To load the data into Neo4j we used a stand-alone batch-import tool [20]. This tool requires two files as input: a file describing the nodes, and a file describing the relations between the nodes. From the aggregated relation file prepared in the previous step we prepared the required two files with two more Awk scripts. Additionally, for each node, these scripts calculated in the number of relations in which the node (biomedical concept) occurs as an argument (subject or object). That number becomes a node property. The batch import tool for Neo4j then loaded the prepared files into a new database very quickly. It was also able to create labels for the nodes from the list of semantic types for each node. Finally, in Neo4j, we created an index on the node properties "name" and "cui", which is an abbreviation for "concept unique identifier" (in the UMLS).

III. RESULTS

Currently, there are $269\,047$ unique concepts and $14\,150\,952$ distinct relationships between them in our Neo4j graph database. These relationships originated from $52\,616\,158$ SemMedDB semantic relation instances. We illustrate the use of this database with some LBD examples.

LBD is conducted using one of two modes: open and closed discovery. In open discovery, a concept X (e.g., a disease) is used to start, and the task is to find a new discovery Z (e.g., a drug) regarding X through some intermediate concept Y (e.g., a gene associated with X). In closed discovery both the starting concept X and final concept Z are know in advance. The goal is to supply Y an explanation for the relation between X and Z. Closed discovery can be used, for example, to elucidate statistically determined relations between X and Z, which do not have an explanation.

Figure 3 shows the most general LBD implementation with a Cypher query. The relations between the concepts X, Y and Z can be matched regardless of the direction of the relations. We require a relation between X and Y, as well as a relation between Y and Z, but we are interested only in those X and Z concepts that are not already related. If we instantiate X with a specific concept (e.g., Curcumin) and leave Z uninstantiated, then we have an open discovery mode. If we instantiate both the X and Z concepts with particular concepts then we have closed discovery mode.

```
MATCH (x:Concept)--(y:Concept)--
  (z:Concept)
WHERE NOT (x)--(z)
RETURN x, y, z;
```

Figure 3. The most general LBD implementation in Cypher.

UMLS semantic types are often used in LBD for concept filtering or for working with a whole class of concepts. Figure 4 shows how we can use semantic types for node (concept) filtering. We implemented concept semantic types as Neo4j labels. In our current implementation, each concept has the label "Concept" and additionally all its semantic types appear as additional labels. The query in Figure 4 searches for drugs (concepts with semantic type "phsu" (abbreviation for Pharmacologic Substance) which are related to some concepts Y which are related to some diseases (concepts with semantic type "dsyn" (abbreviation for Disease or Syndrome)). Moreover, we add an additional constraint that the drug is not already used to treat the disease. This is a declarative query for discovering novel treatments. It can be used in several ways depending on what kind of input data is provided. If a specific drug is provided, the query finds diseases that have not been treated with that particular drug before. If a specific disease is provided, the query finds drugs that have not been used to treat that particular disease before. And finally, in this query we show how we can rank the potentially novel treatments by the count of intermediate concepts Y, as it is usually done in LBD.

```
MATCH (drug:Concept:phsu)-[r1]->(y)-
  [r2]->(disease:Concept:dsyn)
WHERE NOT (drug)-[:TREATS]->(disease)
RETURN drug, disease, count(y) AS y_count
  DESC;
```

Figure 4. A generic Cypher query for finding novel treatment rela-tions between drugs and diseases.

A discovery pattern [21] reduces the number of false positive discoveries and supports explanation by stipulating a set of conditions which enhance the likelihood for a good discovery candidate. Such conditions refer to the semantic types of concepts and the relations between them. We show here how to implement the LBD discovery pattern "inhibit the cause of the disease", which can be used to find novel treatments or explain why certain drugs might be beneficial for certain diseases [22]. This discovery pattern is more specific than the one shown before because it also takes into account the semantic relations between the concepts. The idea of the discovery pattern is to find drugs that inhibit some genes that are etiologically related to a disease. Additionally, we are interested only in drugs that have not been already used to treat the disease. In this Cypher query, "phsu", "gngm" and "dsyn" are UMLS semantic type abbreviations; and drug, gene and disease are variables that are instantiated to particular values when the query is run. Figure 5 shows a generic implementation of this discovery pattern.

```
MATCH (drug:phsu)-[:INHIBITS]->
  (gene:gngm)-[:CAUSES]-> (disease:dsyn)
WHERE NOT (drug)-[:TREATS]->(disease)
RETURN drug, gene, disease;
```

Figure 5. Generic Cypher implementation of the "inhibit the cause of the disease discovery pattern".

We would like to show how more advanced versions of this discovery pattern can be implemented. In Figure 6, first,

 ISBN: 978-1-61208-408-4

we find all the drugs in the "Antipsychotic Agents" class by using the ontological relation IS_A. We restrict the class of diseases to neoplasms by using the semantic type "neop".

```
MATCH (drug:Concept:phsu)-[:ISA]->
  (m:Concept {
    name:"Antipsychotic Agents"})
WITH drug
MATCH (drug)-[:INHIBITS]->
  (gene:gngm)-[:CAUSES]->(s:neop)
WHERE NOT (drug)-[:TREATS]->(s)
RETURN drug, count(distinct gene),
  count(distinct s);
```

Figure 6. More specific version of the "inhibit the cause of the disease" discovery pattern.

IV. Discussion

We faced some chalenges when using Neo4j and its declarative graph query language Cypher. The `LOAD CSV` Cypher command, although elegant and consise, was not able to load our data in a reasonable amount of time. There is confusion regarding indexing in the current version of Neo4j (2.1.6) because two types of indexes exist: "schema indexes" and "legacy indexes". The "schema indexes" are recommended by Neo4j, however they do not provide full text indexing and they only index node properties and not relation properties. The "legacy indexes" are not recommended by Neo4j, but they provide full text indexing and relation properties can also be indexed. However, they are more cumbersome to create and use. We did not evaluate how fast Cypher was when answering queries. Our subjective observation was that it was fast with a small number of starting nodes and no aggregation. The queries become much slower when dealing with a set of starting nodes and when aggregation was required.

In the future we will conduct performance evaluation of the proposed approach in terms of execution speed and memory consumption. We will also implement the same algorithms in a RDF triple store such as Virtuoso, and compare its performance to traditional relational database (MySQL) and Neo4j.

V. Conclusion

Research in LBD can be facilitated by considering the relevant literature as a graph of interacting semantic predications, such as those extracted from MEDLINE using SemRep. Implementing this graph using a graph database such as Neo4j has several advantages over the use of SQL technology for this task. We have illustrated this advantage by showing how the graph query language Cypher naturally supports the use of discovery patterns, a powerful mechanism for limiting the number of false positive "discoveries" that must be human reviewed and for providing explanation.

Acknowledgment

This work was supported by the Slovenian Research Agency and by the Intramural Research Program of the U.S. National Institutes of Health, National Library of Medicine.

References

[1] L. Cheng, H. Lin, F. Zhou, Z. Yang, and J. Wang, "Enhancing the accuracy of knowledge discovery: a supervised learning method," BMC Bioinformatics, vol. 15, no. Suppl 12, 2014, p. S9.

[2] D. Rebholz-Schuhmann, A. Oellrich, and R. Hoehndorf, "Text-mining solutions for biomedical research: Enabling integrative biology," Nature Reviews. Genetics, vol. 13, no. 12, 2012, pp. 829–839.

[3] D. R. Swanson, "Fish oil, Raynaud's syndrome, and undiscovered public knowledge," Perspectives in Biology and Medicine, vol. 30, no. 1, 1986, pp. 7–18.

[4] R. A. DiGiacomo, J. M. Kremer, and D. M. Shah, "Fish-oil dietary supplementation in patients with Raynaud's phenomenon: a double-blind, controlled, prospective study." The American journal of medicine, vol. 86, no. 2, 1989, pp. 158–64.

[5] M. D. Gordon and R. K. Lindsay, "Toward discovery support systems: A replication, re-examination, and extension of Swanson's work on literature-based discovery of a connection between Raynaud's and fish oil," Journal of the American Society for Information Science, vol. 47, no. 2, 1996, pp. 116–128.

[6] R. K. Lindsay, "Literature-based discovery by lexical statistics," Journal of the American Society for Information Science, vol. 50, no. 7, 1999, pp. 574–587.

[7] M. Weeber, H. Klein, L. T. W. De Jong-Van Den Berg, and R. Vos, "Using concepts in literature-based discovery: Simulating Swanson's Raynaud-fish oil and migraine-magnesium discoveries," Journal of the American Society for Information Science and Technology, vol. 52, no. 7, 2001, pp. 548–557.

[8] D. Hristovski, J. Stare, B. Peterlin, and S. Dzeroski, "Supporting discovery in medicine by association rule mining in Medline and UMLS." Studies in health technology and informatics, vol. 84, no. Pt 2, 2001, pp. 1344–8.

[9] P. Srinivasan, "Text mining: Generating hypotheses from MEDLINE," Journal of the American Society for Information Science and Technology, vol. 55, no. 5, 2004, pp. 396–413.

[10] D. Cameron, O. Bodenreider, H. Yalamanchili, T. Danh, S. Vallabhaneni, K. Thirunarayan, A. P. Sheth, and T. C. Rindflesch, "A graph-based recovery and decomposition of Swanson's hypothesis using semantic predications." Journal of biomedical informatics, vol. 46, no. 2, 2013, pp. 238–51.

[11] D. Hristovski, T. Rindflesch, and B. Peterlin, "Using literature-based discovery to identify novel therapeutic approaches," Cardiovascular & Hematological Agents in Medicinal Chemistry, vol. 11, no. 1, 2013, pp. 14–24.

[12] M. E. Bales and S. B. Johnson, "Graph theoretic modeling of large-scale semantic networks," Journal of Biomedical Informatics, vol. 39, no. 4, 2006, pp. 451–464.

[13] T. Rindflesch and H. Kilicoglu, "Semantic MEDLINE: An advanced information management application for biomedicine," Information Services & Use, vol. 31, no. 1-2, 2011, pp. 15–21.

[14] T. C. Rindflesch and M. Fiszman, "The interaction of domain knowledge and linguistic structure in natural language processing: Interpreting hypernymic propositions in biomedical text." Journal of Biomedical Informatics, vol. 36, no. 6, 2003, pp. 462–477.

[15] O. Bodenreider, "The Unified Medical Language System (UMLS): integrating biomedical terminology." Nucleic acids research, vol. 32, no. Database issue, 2004, pp. D267–70.

[16] A. T. McCray, S. Srinivasan, and A. C. Browne, "Lexical methods for managing variation in biomedical terminologies." in Proceedings of the Eighteenth Annual Symposium on Computer Application in Medical Care, J. G. Ozbolt, Ed. Washington, DC: Hanley & Belfus, 1994, pp. 235–239.

[17] L. Smith, T. Rindflesch, and W. J. Wilbur, "MedPost: A part-of-speech tagger for bioMedical text," Bioinformatics, vol. 20, no. 14, 2004, pp. 2320–2321.

[18] A. R. Aronson and F.-M. Lang, "An overview of MetaMap: historical perspective and recent advances." Journal of the American Medical Informatics Association : JAMIA, vol. 17, no. 3, 2010, pp. 229–236.

[19] H. Kilicoglu, D. Shin, M. Fiszman, G. Rosemblat, and T. C. Rindflesch, "SemMedDB: A PubMed-scale repository of biomedical semantic predications," Bioinformatics, vol. 28, no. 23, 2012, pp. 3158–3160.

 ISBN: 978-1-61208-408-4

[20] "Neo4j (CSV) Batch Importer," visited on 2015-05-14. [Online]. Available: https://github.com/jexp/batch-import

[21] D. Hristovski, C. Friedman, T. C. Rindflesch, and B. Peterlin, "Exploiting semantic relations for literature-based discovery." AMIA Annual Symposium proceedings, 2006, pp. 349–53.

[22] C. B. Ahlers, D. Hristovski, H. Kilicoglu, and T. C. Rindflesch, "Using the literature-based discovery paradigm to investigate drug mechanisms." AMIA Annual Symposium proceedings, 2007, pp. 6–10.

 ISBN: 978-1-61208-408-4