Reciprocal Mapping of Knowledge Graphs

by Mike Bergman - Wednesday, February 22, 2017

http://www.mkbergman.com/2024/reciprocal-mapping-of-knowledge-graphs/

An Advance Over Simple Mapping or Ontology Merging

The technical term for a knowledge graph is ontology. As artifacts of information science, artificial intelligence, and the semantic Web, knowledge graphs may be constructed to represent the general nature of knowledge, in which case they are known as upper ontologies, or for domain or specialized purposes. Ontologies in these senses were first defined more than twenty years ago, though as artifacts they have been used in computer science since the 1970s. The last known census of ontologies in 2007 indicated there were more than 10,000 then in existence, though today's count is likely in excess of 40,000 [1]. Because of the scope and coverage of these general and domain representations, and the value of combining them for specific purposes, key topics of practical need and academic research have been ontology mappings or ontology mergers, known collectively as ontology alignment. Mapping or merging make sense when we want to combine existing representations across domains of interest.

At Cognonto, ontology alignment is a central topic. Our KBpedia knowledge structure is itself the result of mapping nearly 30 different information sources, six of which are major knowledge bases such as Wikipedia and Wikidata. When applied to domain problems, mapping of enterprise schema and data is inevitably an initial task. Mapping techniques and options are thus of primary importance to our work with knowledge-based artificial intelligence (KBAI). When mapping to new sources we want to extract the maximum value from each contributing source without devolving to the tyranny of the lowest common denominator. We further need to retain the computability of the starting KBpedia knowledge structure, which means maintaining the logic, consistency, and coherence when integrating new knowledge.

We are just now completing an update to KBpedia that represents, we think, an important new option in ontology alignment. We call this option *reciprocal mapping*, and it represents an important new tool in our ontology mapping toolkit. I provide a high-level view and rationale for reciprocal mapping in this

article. This article accompanies a more detailed use case by Fred Giasson that discusses implementation details and provides sample code.

The Mapping Imperative and Current Mappings

Cognonto, like other providers of services in the semantic Web and in artificial intelligence applied to natural languages, views mapping as a central capability. This importance is because all real-world knowledge problems amenable to artificial intelligence best express their terminology, concepts, and relations between concepts in a knowledge graph. Typically, that mapping relies upon a general knowledge graph, or upper ontology, or multiples of them, as the mapping targets for translating the representation of the domain of interest to a canonical form. Knowledge graphs, as a form, can represent differences in semantic meaning well (you say *car*, I say *auto*) while supporting inference and reasoning. For these problems, mapping must be seen as a core requirement.

There is a spectrum of approaches for how to actually conduct these mappings. At the simplest and least accurate end of the spectrum are string matching methods, sometimes supplemented by regular expression processing and heuristic rules. An intermediate set of methods uses concepts already defined in a knowledge base as a way to "learn" representations of those concepts; while there are many techniques, two that Cognonto commonly applies are explicit semantic analysis and word embedding. Most of these intermediate methods require some form of supervised machine learning or other ML techniques. At the more state-of-the-art end of the spectrum are graph embeddings or deep learning, which also capture context and conceptual relationships as codified in the graph.

Aside from the simple string match approaches, all of the intermediate and state-of-the-art methods use machine learning. Depending on the method, these machine learners require developing either training sets or corpuses as a reference basis for tuning the learners. These references need to be manually scoped, as in the case of training corpuses for unsupervised learning, or manually scored into true and false positives and negatives for training sets for supervised learning. Cognonto uses all of these techniques, but importantly supplements them with logic tests and scripts applied to the now-modified knowledge graph to test coherence and consistency issues that may arise. The coherency of the target knowledge graph is tested as a result of the new mappings.

Items failing those tests are fixed or dropped. Though not a uniform practice by others, Cognonto also adheres to a best practice that requires candidate mappings to be scored and manually inspected before final commitment to the knowledge graph. The knowledge graph is a living artifact, and must also be supported by proper governance and understood workflows. Properly constructed and maintained knowledge graphs can power much work from tagging to classification to question answering. Knowledge graphs are one of the most valuable information assets an enterprise can create.

We have applied all of these techniques in various ways to map the six major knowledge bases that make up the core of KBpedia, plus to the other 20 common knowledge graphs mapped to KBpedia. These mappings are what is contained in our current version 1.20 of KBpedia, the one active on the Cognonto Web site. All of these nearly 30 mappings represent using the basis of KBpedia (A) as the mapping target for the contributing KBs (B). All version 1.20 mappings are of this B ? A form.

Reciprocal Mappings to Extend Scope

This is well and good, and is the basis for how we have populated what is already in the KBpedia knowledge graph, but what of the opposite? In other words, there are concepts and local graph structure within the contributing KBs that do not have tie-in points (targets) within the existing KBpedia. This is particularly true for Wikipedia with its (mostly) comprehensive general content. From the perspective of Wikipedia (B), KBpedia v 1.20 (A) looks a bit like Swiss cheese with holes and gaps in coverage. Any source knowledge graph is likely to have rich structure and detail in specific areas beyond what the target graph may contain.

As part of our ongoing KBpedia improvement efforts, the next step in a broader plan in support of KBAI, we have been working for some time on a form of reverse mapping. This process analyzes coverage in B to augment the scope of A (KBpedia). This kind of mapping, which we call *reciprocal mapping*, poses new challenges because candidate new structure must be accurately placed and logically tested against the existing structure. Fred recently wrote up a use case that covers this topic in part.

We are now nearly complete with this reciprocal mapping from Wikipedia to KBpedia (B ? (A+B)). Reciprocal mapping results in new concepts and graph structure being added to KBpedia (now A+B). It looks like this effort, which we will finalize and release soon, will add nearly 40% to the scope of KBpedia.

This expansion effort is across-the-board, and is not focused or limited to any particular domain, topic or vocabulary. KBpedia's coverage will likely still be inadequate for many domain purposes. Nonetheless, the effort does provide a test bed for showing how external vocabularies may be mapped and what benefits may arise from an expanded KBpedia scope, irrespective of domain. We will report here and on the Cognonto Web site upon these *before* and *after* results when we release.

Knowledge graphs are under constant change and need to be extended with specific domain information for particular domain purposes. KBpedia is perhaps not the typical ontology mapping case, since its purpose is to be a coherent, consistent, feature-rich structure to support machine learning and artificial intelligence. Yet, insofar as domain knowledge graphs aspire to support similar functionality, our new methods for reciprocal mapping may be of interest. In any case, effective means at acceptable time and cost must be found for enhancing or updating knowledge graphs, and reciprocal mapping is a new tool in the arsenal to do so.

The Reciprocal Mapping Process

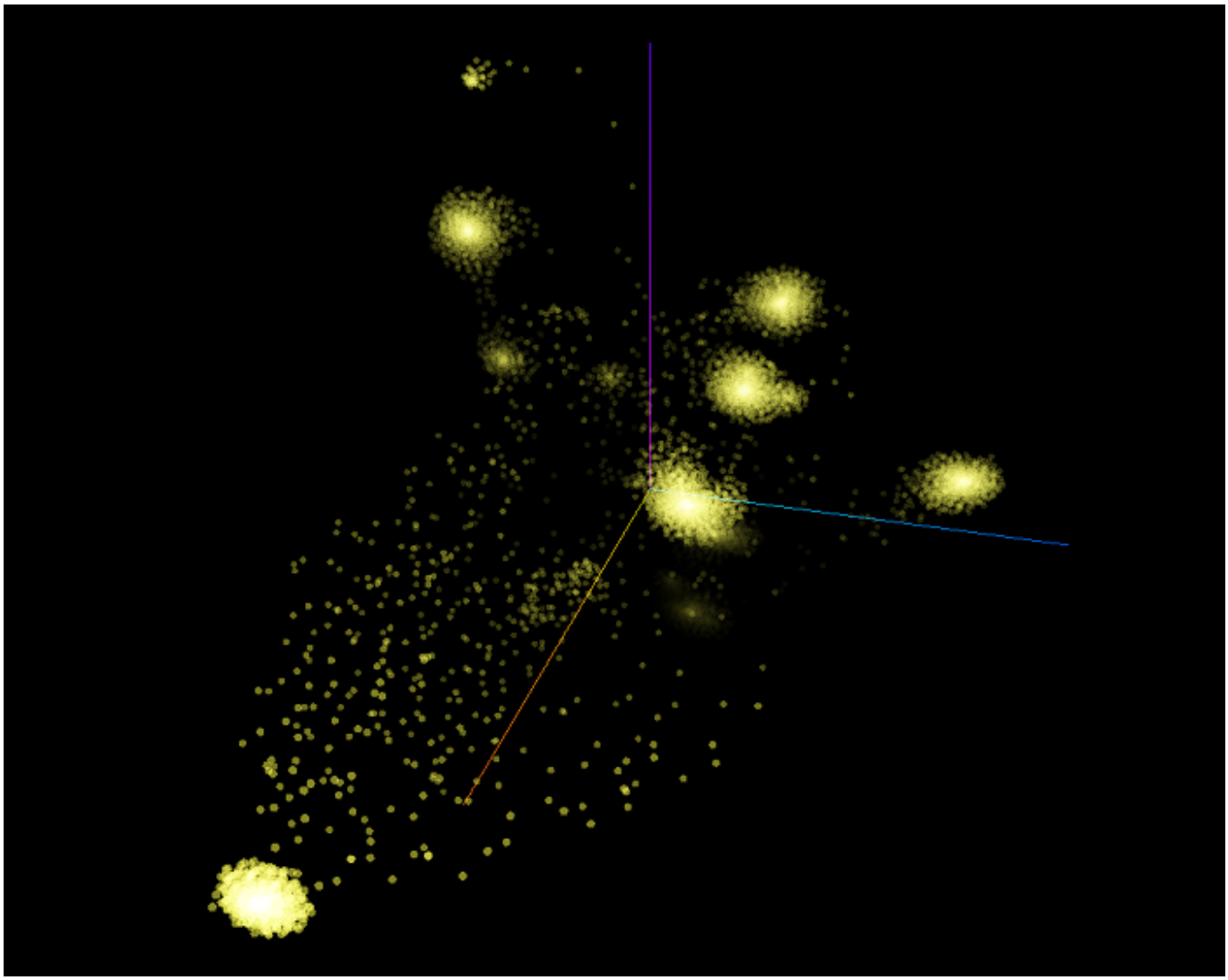
Wikipedia is a particularly rich knowledge base for the reciprocal mapping process, though other major KBs also are suitable, including smaller domain-specific ones. In order to accomplish the reciprocal mapping process, we observed there were several differences for this reverse mapping case from our standard B ? A mapping. First, categories within the source KB are the appropriate basis for mapping to the equivalent concept (class) structure in KBpedia. The category structure in the source KB is the one which also establishes a similar graph structure. Second, whatever the source knowledge base, we need to clean its categories to make sure they correspond to the actual subsumption bases reflective in the target KBpedia. Cleaning involves removing administrative and so-called "compound" categories, or ones of

convenience but not reflective of natural classes. In the case of Wikipedia, this cleaning results in the elimination of about 80% of the starting categories. Third, we also need to capture structural differences in the source knowledge graph (B). Possible category matches fall into three kinds: 1) *leaf* categories, which represent child extensions to existing KBpedia terminal nodes; 2) *near-leaf* categories, which also are extensions to existing KBpedia terminal nodes, but which also are parents to additional child structure in the source; and 3) *core* categories, which tie into existing intermediate nodes in KBpedia that are not terminal nodes. By segregating these structural differences, we are able to train more precise placement learners.

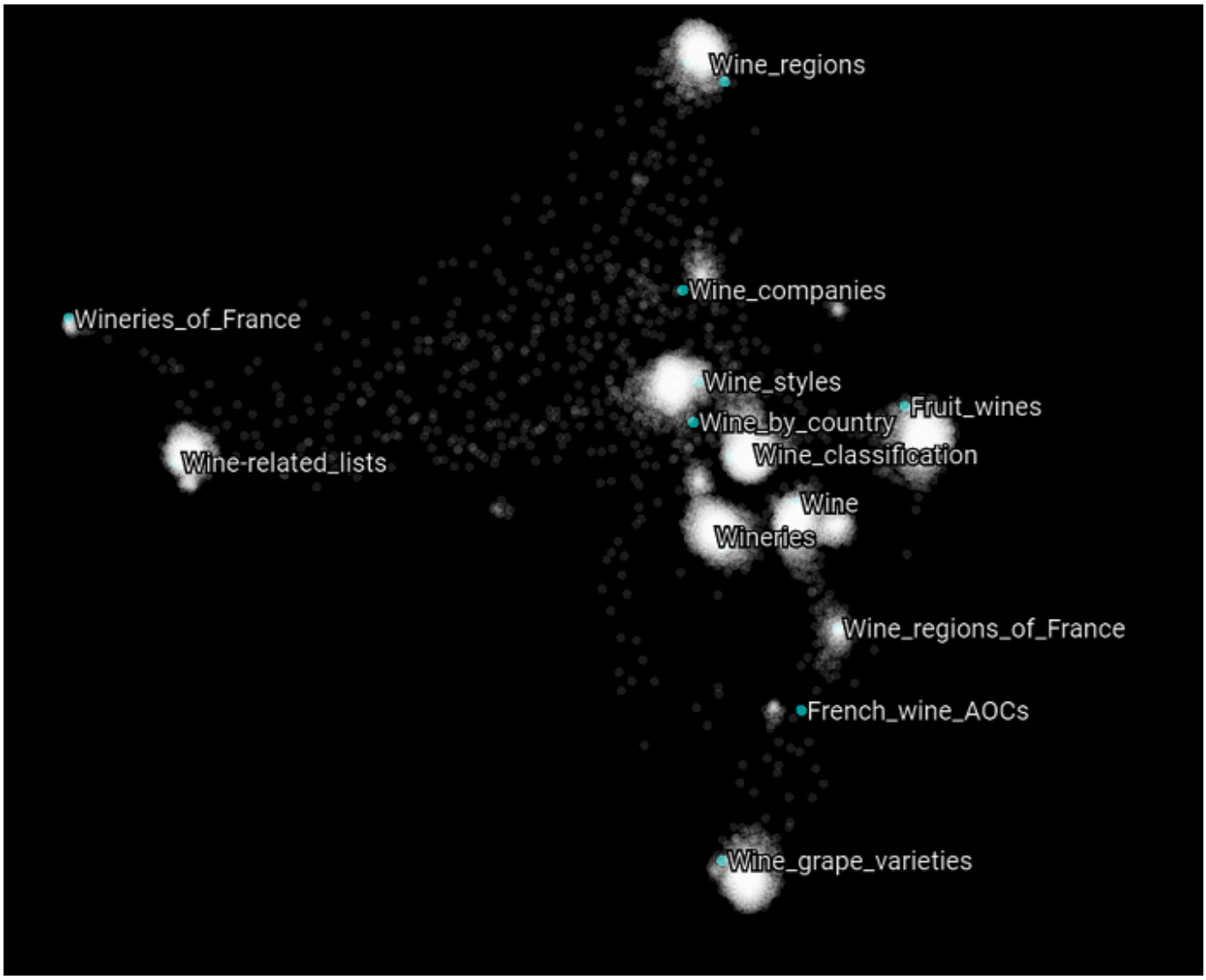
From our initial simple mapping (B ? A) we already have thousands of KBpedia reference concepts mapped to related Wikipedia categories. What we want to do is to use this linkage to propose a series of ***new*** sub-classes that we could add to KBpedia based on the sub-categories that exist in Wikipedia for each of these mappings. The challenge we face by proceeding in this way is that our procedure potentially creates tens of thousands of new candidates. Because the Wikipedia category structure has a completely different purpose than the KBpedia knowledge graph, and because Wikipedia's creation rules are completely different than KBpedia, many candidates are inconsistent or incoherent to include in KBpedia. A cursory inspection shows that most of the candidate categories need to be dropped. Reviewing hundred of thousands of new candidates manually is not tenable; we need an automatic way to rank potential candidates.

The way we automate this process is to use an SVM classifier trained over graph-based embedding vectors generated using the DeepWalk method [2]. DeepWalk learns the sub-category patterns that exists in the Wikipedia category structure in an unsupervised manner. The result is to create graph embedding vectors for each candidate node. Our initial B ? A maps enable us to quickly create training sets with thousands of pre-classified sub-categories. We split 75% of the training set for training, and 25% for cross-validation. We also employ some hyperparameter optimization techniques to converge to the best learner configuration.Once these three steps are completed, we classify all of the proposed sub-categories and create a list of potential sub-class-of candidates to add into KBpedia, which is then validated by a human. These steps are more fully described in the detailed use case.

We use visualization and reference "gold" standards as means to further speed the time to get to a solution. We visualize interim passes and tests using the TensorFlow Projector web application. Here, for example, is a 3D representation of the some of the clusters in the concepts:



Another way we might visualize things is to investigate the mappings by topic. Here, for example, we highlight the Wikipedia categories that have the word "wine" in them.



While the visualizations are nice and help us to understand the mapping space, we are most interested in finding the learner configurations that produce the "best" results. Various statistics such as *precision*, *recall*, *accuracy* and others can help us determine that. The optimization target is really driven by the client. If you want the greatest coverage and can accept some false positives, then you might favor recall. If you only want a smaller set of correct results, then you would likely favor precision. Other objectives might emphasize other measures such as accuracy or AUM.

The reference "gold" standards in the scored training sets provide the basis for computing all of these statistics. We score the training sets as to whether a given mapping is true or false (correct or not). (False mappings need to be purposefully introduced.) Then, when we parse the test candidates against the training set, we note whether the learner result is either positive or negative (indicated as correct or indicated as not correct). When we match the test to the training set, we thus get one of four possible scores: true positives (TP), false positives (FP), true negatives (TN) and false negatives (FN). Those four simple scoring categories are sufficient for calculating any of the statistical measures.

We capture the reciprocal mapping process using a repeatable pipeline with the reporting of these various

statistical measures, enabling rapid refinements in parameters and methods to achieve the best-performing model, according to how we have defined "best" per the discussion above. Once appropriate candidate categories are generated using this optimized model, the results are then inspected by a human to make selection decisions. We then run these selections against the logic and coherency tests for the now-modified graph, and keep or modify or drop the final candidate mappings depending on how they meet the tests. The semi-automatic methods in this use case can be applied to extending KBpedia with any external schema, ontology or vocabulary.

This semi-automated process takes 5% of the time it would normally take to conduct this entire process by comparable manual means. We know, since we have been doing the manual approach for nearly a decade.

The Snake Eating Its Tail

As we add each new KB to KBpedia or expand its scope through these mapping methods, verified to pass all of our logic and satisfiability tests, we continue to have a richer structure for testing coherence and consistency for the next iteration. The accretive process that enables these growing knowledge bases means there is more structure, more assertions, and more relations to test new mappings. The image that comes to mind is that of ouroboros, one of the oldest images of fertility. The snake or serpent eating its tail signifies renewal.

KBpedia has already progressed through this growth and renewal process hundreds of times. Our automated build scripts mean we can re-generate KBpedia on a commodity machine from scratch in about 45 minutes. If we add all of the logic, consistency and satisfiability checks, a new build can be created in about two hours.This most recent reciprocal mapping effort adds about 40% more nodes to KBpedia's current structure. Frankly, this efficient and clean build structure is remarkable for one of the largest knowledge structures around with about 55,000 nodes and 20 million mapped instances. Remarkably, using the prior KBpedia as the starting structure, we have been able to achieve this expansion with even better logical coherence of the graph in just a few hundred hours of effort.

The mapping methods discussed herein can extend KBpedia using most any external source of knowledge, one which has a completely different structure than KBpedia and one which has been built completely differently with a different purpose in mind than KBpedia. A variety of machine learning methods can reduce the effort required to add new concepts or structure by 95% or more. Machine learning techniques can filter potential candidates automatically to reduce greatly the time a human reviewer has to spend to make final decisions about additions to the knowledge graph. A workable and reusable pipeline leads to fast methods for testing and optimizing parameters used in the machine learning methods. The systematic approach to the pipeline and the use of positive and negative training sets means that tuning the approach can be fully automated and rapidly vetted.

This is where the real value of KBpedia resides. It is already an effective foundation for guiding and testing new domain extensions. KBpedia now shows itself to be the snake capable of consuming (mapping to), and thereby growing from, nearly any new knowledge base.

[1] A simple Google search of https://www.google.com/search?q=filetype:owl (OWL is the Web Ontology Language, one of the major ontology formats) shows nearly 39,000 results, but there are multiple ontology languages available, such as RDF, RDFS, and others (though use of any of these languages does

not necessarily imply the artifact is a vocabulary or ontology).

[2] Perozzi, B., Al-Rfou, R., & Skiena, S. (2014, August). Deepwalk: Online learning of social representations. In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 701-710). ACM.
