Semantic Enrichment of Data for AI Applications

Fatma Özcan[1]*, Chuan Lei[2], Abdul Quamar[2], Vasilis Efthymiou[3]*
[1]Google, 1600 Amphitheatre Parkway, Mountain View, CA, 94043
[2]IBM Research - Almaden, 650 Harry Road, San Jose, CA 95120
[3]FORTH - Institute of Computer Science, Heraklion, Crete, Greece
fozcan@google.com,chuan.lei@ibm.com,ahquamar@us.ibm.com,vefthym@ics.forth.gr

ABSTRACT

In this work, we use semantic knowledge sources, such as cross-domain knowledge graphs (KGs) and domain-specific ontologies, to enrich structured data for various AI applications. By enriching our understanding of the underlying data with semantics brought in from external ontologies and KGs, we can better interpret the data as well as the queries to answer more questions, provide more complete answers, and deal with entity disambiguation. To semantically enrich the data with external knowledge sources, we need to find the correspondences between the structured data and the entities in the cross-domain KGs and/or the domain-specific ontologies. In this paper, we break this problem into several steps, and provide detailed solutions for each step. We showcase the practical value of semantic enrichment of data using our proposed techniques in entity disambiguation, natural language querying and conversational interfaces to data, query relaxation, as well as query answering, with promising results.

ACM Reference Format:
Fatma Özcan, Chuan Lei, Abdul Quamar, Vasilis Efthymiou. 2021. Semantic Enrichment of Data for AI Applications. In *International Workshop on Data Management for End-to-End Machine Learning (DEEM'21), June 20–25, 2021, Virtual Event, China.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3462462.3468881

1 INTRODUCTION

Knowledge graphs (KGs), both hand-curated, as well as created from unstructured, semi-structured, and structured data sources, store information about the world in structured form and constitute the foundation of most modern AI systems. There are many cross-domain large-scale KGs such as DBpedia [8], Wikidata [47] and YAGO 4 [44], which provide well-structured, encyclopedic knowledge about a wide spectrum of entities, targeting coverage, but without much detail. Domain-specific ontologies, such as FIBO [2], FRO [3], SNOMED CT [6], and RxNorm [4], address this issue by providing more fine-grained information about entities, complex entity relationships and deep hierarchical relationships between them.

*Work done while at IBM Research.

DEEM'21, June 20–25, 2021, Virtual Event, China

ACM ISBN 978-1-4503-8486-5/21/06…$15.00
https://doi.org/10.1145/3462462.3468881

Encapsulating the rich semantic knowledge from domain-specific ontologies into a structured dataset, provides a better understanding the underlying data, and consequently, the ability to reason at a more abstract level. This deeper data understanding enables many AI applications, including natural language interfaces, entity disambiguation, entity resolution, query relaxation, as well as ontology-based query answering. Ontology-based natural language interfaces (e.g., [40, 42]), for example, augment linguistic patterns with domain knowledge and allow reasoning at the level of real-world entities, as opposed to tables and columns, for interpreting the user's utterance and converting it into a structured query. Use of external knowledge sources, such as ontologies, enable relaxing user queries[34] with synonymous terms, and expand the querying capabilities of the underlying systems. Ontology-based data access systems [52] capture the domain semantics and provide a standard description of the domain for applications to use. Ontologies and KGs are also used in entity resolution [37] and entity disambiguation [48] to find the correspondences between real-world entities. In summary, semantic enrichment of data enables deep reasoning capabilities that are needed in many applications.

In this work, we use both cross-domain KGs as well as domain-specific ontologies to enrich structured data for many AI applications. The main challenge in all these AI applications is to find the correspondences between the structured data and the entities in the cross-domain KG and/or the domain-specific ontologies. Figure 1 identifies the many problems that needs to be solved to augment structured data with KGs and ontologies for downstream AI applications.

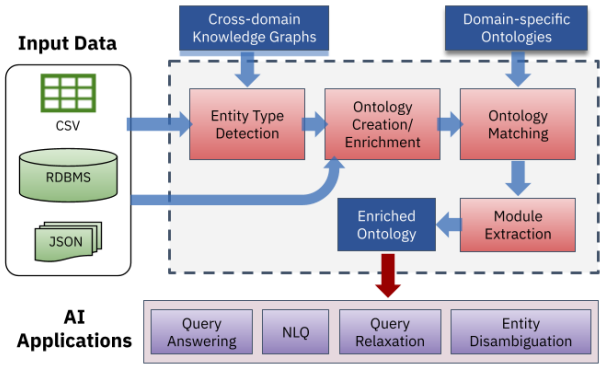

Figure 1: Semantic data enrichment for AI applications.

In most cases, a user starts with a structured data source, does not know the domain, and hence first needs to identify the entities in the data set. We call this the *e*ntity type detection problem.

Cross-domain KGs that contain high-level information about world entities is a rich source of information to leverage for this problem. Note that at this point the information about entities are still at high-level. We also observed that the accuracy of matching data to generic entities is limited without human intervention and is not as specific, matching higher level concepts in a is-a hierarchy.

Once we detect the entity types, we can identify the domain of the data set and use domain-specific ontologies to target specific concepts with rich domain information. In some cases, a user may already know the domain, and can skip entity type detection. We solve the problem of matching data to ontologies in two steps. In the first step, we first create an ontology [25, 35] from the data set, and then use ontology to ontology matching [24]. In many cases, a single ontology is not sufficient to cover all the data items, but require multiple ontologies. For example, only 15 entities in the MIMIC-III [29] dataset has matches in SNOMED CT. To match all entities, we need to use multiple domain-specific ontologies. An important observation is that the matches only cover a small subset of the domain-specific ontology. Using many large ontologies slows than the AI applications. Hence, it is important to identify and extract only the subset that matches. This problem is know as module extraction [11, 46] in the literature. The final step is the merging of these many subsets extracted from multiple ontologies to create the final enriched ontology that will power many AI applications.

In the rest of the paper, we briefly define each problem and provide our initial results for each, including entity type detection, ontology creation, ontology matching, and modular ontology reuse (Sections 2 and 3). We also describe four AI applications: query answering (Section 4.1.1), query relaxation (Section 4.1.2), NLQ (Section 4.2), and entity disambiguation (Section 4.3).

2 DATA TO ONTOLOGY MATCHING

In this section, we describe the problems and our solutions to enrich a given data set with external semantic knowledge sources, such as KGs and ontologies, by finding the correspondence between the data and ontologies. As shown in Figure 1, this involves various steps, including entity type detection, ontology creation, and ontology to ontology matching.

2.1 Entity Type Detection

There are many use cases where the domain of the source data is not known. In those cases, we first need to identify entity types, and the domain of the data set. In this section, we assume that we are given a data set stored in an RDBMS, with its metadata (tables, columns and a set of primary key-foreign key (PK-FK) constraints), and the goal is to map tables and columns to entity types. We exploit external knowledge sources such as Wikidata [47] and Schema.org [5] to identify high-level or generic entities such as Person, Organization, Product, Location, etc. that are domain agnostic. Wikidata is an open cross-domain knowledge graph which provides a central repository of structured data in terms of items, properties and values. Schema.org is an open type hierarchy initiated by Google, Microsoft and others, that describes schemas for structured data in terms of several different object/classes such as organization, person, place, product etc.

We propose a two-step approach for entity type detection (Figure 2), using the two external knowledge sources described above. In step 1, we map a column in a table to column-level entities in the target knowledge source. In step 2, we map each table in the relational schema to a table-level entity in the target knowledge source.

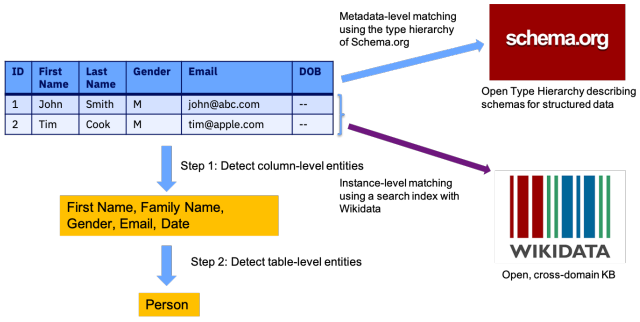

Figure 2: Entity Type Detection in two steps

Step 1: Column-level entity mapping. Column-level entity mapping is done using both instance data as well as the metadata associated with the relational schema of the source data set. First, we use instance level matching using a search index created from Wikidata [28]. For each column in a table, we match its data instances to the Wikidata search index using various similarity metrics, including lexical similarity metrics such as exact match, Levenshtein, BM25 and word embeddings using pre-trained language models such as BERT. We, then, aggregate the instance-level matches for each column to detect candidate column types. In addition to this, we also use regular expressions to detect certain column types such as date, email, etc. where the instance data is not available in the Wikidata index.

We also do meta-data level matching using the Schema.org type hierarchy to improve the accuracy. For each column in a table, we match the column name to the schema.org type hierarchy and detect candidate entity types using different similarity metrics, including Levenshtein and Q-gram as well as word embeddings. To combine the results of both instance level and meta-data level matching, we first consider the top matches using the instance level matching using Wikidata. For the columns where the instance level matching produces very low confidence or no results, we utilize the results obtained from the meta-data level matching using Schema.org. Combining the results of instance-level and metadata level matches, we pick the top-k[1] matches based on their similarity scores as the candidates for column-level entities. Using both instance data and meta-data level matching provides a robust method for detecting entity types.

Step 2: Table-level entity detection. The key idea here is that using the top-k column-level matches obtained, we identify the table level entities based on a voting algorithm. More specifically, for each column in a source table, we first list the top k column-level candidate entities identified in Step 1, and using each column-level entity match we map the table to a table-level entity in the

[1]We use k=5 in our initial results

schema.org type hierarchy. For each distinct source table, we count the number of times a destination table-level entity is identified for all column-level entities in the source table, and pick the entity type with the maximum count as the table-level entity match. In case of a tie, we use similarity metric rankings to break the tie.

Evaluation. We evaluated our entity detection algorithms on two different data sets. The first data set contains *sales* data corresponding to sales of different products made by employees across different departments. The sales data is stored in a relational database with a total of 28 tables and 392 columns. We limited the scope of the evaluation to a subset of 14 columns across 6 different tables of this data set that were relevant to 5 generic entities: Person, Company, Product, Place and Order. Note that these are all generic entities that are domain agnostic. The second data set is a film data set containing information about actors, films, categories and rental histories. The data set consists of 15 tables and 240 columns. Again for this data set we limited the evaluation to a subset of 14 columns across 6 tables that were relevant to the 5 generic entities listed above.

Table 1: Entity type detection results (F1-scores)

	Sales Dataset	Films Dataset
Step 1	0.71	0.79
Step 2	1.0	1.0

Table 1 shows the results in terms of F1-scores for our two step approach for entity type detection for k=5. For Step 1, column-level entity detection we achieved F1-scores of 0.71 and 0.79 for the sales and films data sets, respectively. For Step 2, we evaluated the detection of 6 tables containing the selected columns for the two data sets. We were able to detect all the 6 table-level entities correctly for both the data sets achieving an F1-score of 1. We observe that even though we achieve a comparatively lower F1-score for column-level entities, the step-2 for table level entity detection using the voting based algorithm is quite robust and is able to identify all the table-level entities correctly.

2.2 Ontology Creation and Enrichment

There are many use cases where we know the domain of the data set. We can use this information to match the data to more detailed domain-specific entities. There is a lot of work in the literature for ontology to ontology matching [17, 26]. To take advantage of these works, we first create an ontology from the data set, followed by ontology to ontology matching. In the following sections, we describe both of these steps. We support creating ontologies from both relational data, as well as semi-structured data in JSON format. We also introduce an ontology enrichment technique to incorporate more semantic information before ontology matching.

2.2.1 Ontology generation from relational data. In earlier work, we built FINESSE [25, 35] to create an ontology from a relational schema describing the underlying data set. FINESSE uses a multi-step approach [35] for inferring entities, their data properties and relationships between these entities. These steps include: (1) Identifying entities. We map each table in the underlying relational schema to an entity in the inferred ontology[2] with the exception of a few tables that exist in the relational schema to enable m:n joins. More details on this can be found in [35]. (2) Identifying data properties. Each column in a table is mapped to a data property of the corresponding entity that represents the table. Exceptions include the foreign key columns. (3) Identifying relationships between entities. We use the primary key foreign key relationships in the underlying relational schema for inferring relationships between entities, their types and cardinalities. Further, in many cases the primary key foreign key constraints are not explicitly specified in the relational schema. Inferring semantically richer relationships such as parent-child (is-A) relationships between tables is also challenging. We have developed techniques to infer such relationships from the characteristics of underlying data distributions. Further details can be found in [35].

2.2.2 Ontology generation from JSON data. Semi-structured data is often represented in JSON format stored in document stores. We designed a multi-step process [35] to infer an ontology from the JSON documents. In the first step, we build a schema tree from the nested structure of the JSON data for each individual document. The individual schema trees extracted from different documents are merged into a single schema, similar to the data guide generation process of Goldman et al. [19]. In the second step, we use the following rules to convert the schema into an ontology.

```
Path A.b => Entity A, Property b of A
Path A.B.c => Entity A, Entity B,
              Relation A to B, Property c of B
```

2.2.3 Ontology enrichment. The methods described in Sections 2.2.1 and 2.2.2 create ontologies that capture schema-level details of the underlying data, but they are far less semantically rich than the standard ontologies created by experts. To alleviate this issue, we introduce entity and neighborhood augmentation heuristics to enrich the derived ontology in [24].

For each distinct value in the source data, the entity augmentation heuristic adds an instance-level entity to the created ontology, and connect these new entities to the existing schema-level ones via a new relationship *"instance of"*. The neighborhood augmentation heuristic adds the missing structural information to the created ontology. If two entities in the standard ontology have an edge, while their counterparts in the created ontology do not, a new edge is added to the created ontology. The enriched ontology contains rich semantic information from the instance data in the database, which can be matched to a standard ontology with higher precision.

2.3 Ontology to Ontology Matching

As described above, data to ontology matching is the process of finding semantic correspondences between tables in databases to standard ontologies. Once an ontology is derived and enriched from the source data, we can further map it to a standard domain-specific ontology. Ontology to ontology matching is an active area of research, with the Ontology Alignment Evaluation Initiative[3] organized each year. LogMap [26] and AML [17] employ various

[2] An entity corresponds to a class in an ontology. In this paper, we use the terms concepts and entities interchangeably

[3] http://www.cs.ox.ac.uk/isg/projects/SEALS/oaei/2020/

sophisticated features and domain-specific thesauruses to perform ontology matching. However, these hand-crafted features often are limited for a given data source and face the bottleneck of improvement. To address these shortcomings, in [24], we introduce MEDTO (Figure 3) to address ontology to ontology matching problem with two innovative techniques.

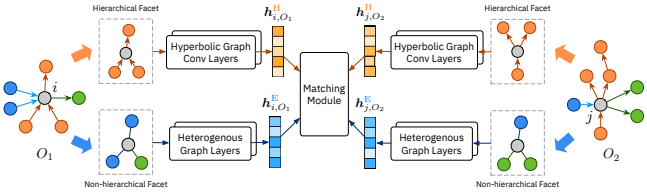

Figure 3: MEDTO system architecture (best viewed in color).

First, we employ hyperbolic graph convolution layers to encode the parent and child entities of each entity in the hyperbolic space, capturing the hierarchical characteristics in an ontology. Representing such hierarchical structures in the hyperbolic space would help identify matching entities and improve the accuracy of ontology matching. Second, we introduce heterogeneous graph layers to incorporate both the local structure and the global context into entity embeddings to enrich the features of each entity. The global context indicates the position of the entity within the broader context of the entire ontology, providing the high degree of separation between entities.

MEDTO leverages these two complementary representations (i.e., hierarchical and non-hierarchical views) to improve the ontology matching capabilities. As shown in [24], MEDTO significantly outperforms the state-of-the-art (e.g., LogMap [26] and AML [17]) methods on matching MIMIC-III to a standard medical ontology (i.e., SNOMED CT). Among 15 matching entitiess identified by domain experts, MEDTO was able to match 7 and 9 tables from MIMIC-III to SNOMED CT when Hits@10 and Hits@30, respectively. Furthermore, MEDTO consistently achieves results on par or even better than a variety of baselines (e.g., LogMap [26], AML [17], RDGCN [51], etc.) on a benchmark (i.e., Large BioMed Track) from the Ontology Alignment Evaluation Initiative.

3 MODULAR ONTOLOGY REUSE

Once we have matched our input data to entities defined in an external ontology, the next steps are to extract a small subset of the external ontology that is relevant to the matched entities (this subset is called a module) and then construct a unified, semantically enriched ontology for our data. In this section, we briefly describe those steps for the case in which a single external ontology is employed. This process, however, could be extended to incorporate multiple such external ontologies, which we plan to investigate in future work.

Module extraction. The external ontology O_2 typically contains a huge amount of information, most of which is not relevant to our constructed ontology O_1. For example, SNOMED CT contains hundreds of thousands of entities, and in the previous section, we presented a use-case in which only 15 of those entities were actually matched to MIMIC-III. Our goal with module extraction is to retain only the smallest possible subset of O_2 that is relevant to O_1, without losing its semantics. This is known in the literature as extracting a minimal module in O_2 [21]. Intuitively, for our downstream tasks, it makes no difference in the quality of the results if we use the whole O_2 or its minimal module O_2', but it makes a huge difference in efficiency. Therefore, this step is optional and it mainly targets performance improvements, as well as a more manageable visualization of the data semantics, if needed.

Ontology unification. In the unification step, we merge our constructed ontology O_1 with the module O_2' of the external ontology O_2. In our current implementation, the merge operation is a simple renaming of the terms in O_1 with the matching terms in O_2', which are typically standardized terms. We can also have entities in O_1, which are not matched to entities in O_2. For those entities, we retain their names as originally defined in O_1. In addition to the methodology described above, many other merge operations are also possible. For a comprehensive list, please refer to [15].

4 APPLICATIONS

In this section, we describe several AI applications that benefit from the semantic enrichment introduced in Sections 2 and 3.

4.1 Ontology-Enhanced Query Answering

With query answering, we refer to the problem of providing answers to user queries in a structured format (e.g., SQL, SPARQL), in (one-shot) natural language text, or in a conversational space. In all those settings, what is common is that by enriching our understanding of the underlying data with semantics brought in from external ontologies, we can better understand the queries, and also provide more complete answers, compared to just using the input data.

4.1.1 Ontology-enriched query answering on relational data. In [7], we show how we can enrich query answers on a relational database by exploiting an external ontology (e.g., SNOMED in the medical domain). First, we enrich the data-generated ontology with a module of the external ontology, as described in Section 2.2. Then, using the enriched ontology as a target schema, we follow the chase procedure from data exchange [16] to transform the input data into this schema. This allows the users to pose queries using the vocabulary of the external ontology, as well as that of the input data. Our experimental evaluation on 15 exemplar queries taken from the logs of a real conversational system, showed that the ontology-enriched framework provided more answers than just relying the knowledge coming the the underlying database in all 15 queries [7]. It could even provide answers to queries that could not have been understood without ontology enrichment, or queries for which we originally got 0 answers without ontology enrichment.

4.1.2 Query relaxation. In addition to providing enriched answers using an external ontology, we show how we can leverage an external ontology to alleviate the mismatch between the KB and the colloquial and imprecise terminology used in queries. In [34], we introduce a domain-specific query relaxation approach that leverages rich domain vocabularies and their semantic relationships from an external ontology to expand both the set of queries that we can answer, as well as the set of answers to the queries, over the KB. We introduce a lightweight adaptation method to customize

and incorporate external ontologies to work with the existing KB, and propose a novel similarity metric to leverage the information content in the KB, the structural information in the external knowledge source, and the contextual information from user queries. The experimental results and user study in [34] show that the proposed query relaxation method outperforms state-of-the-art methods, including deep learning-based ones, in precision and recall and improve the response quality of a query answering system.

4.2 Natural Language Querying and Conversational Systems

The use of ontologies to build natural language interfaces to data allows us reason about the underlying data in terms of real world entities and their relationships. This enables a richer semantic understanding of natural language queries and enable better query interpretation. This is in sharp contrast to working at the level of physical relational schema elements such as tables and columns, as real world entities tend to be spread across multiple such elements. Reasoning at the level of such physical schema level elements makes it much harder to interpret and respond to natural language queries.

4.2.1 Natural language querying. In [40, 42], we build ontology-based natural language interfaces, using domain knowledge to augment understanding of linguistic patterns and support natural language querying of structured data.

ATHENA [40] utilizes the information in a given ontology that represents the schema of the underlying data and an associated mapping between the individual elements of the ontology and the relational schema. A set of synonyms can be associated with each ontology element. When translating an input query into a SQL query, ATHENA uses an intermediate ontology query language (OQL) expressed over the ontology before subsequently translating it into SQL. ATHENA++ [42] extends ATHENA to support nested query detection and generation, both of which use the ontology for better accuracy. The extensive experiments show that ATHENA++ achieves 88.33% accuracy on the FIBEN benchmark [1] and 78.89% accuracy on Spider benchmark [54], consistently outperforming state-of-the-art systems.

In [23], we demonstrate the capability of our NLQ system to support querying process automation data stored in JSON format in Elasticsearch. We utilize process-specific artifacts and event logs to derive an ontology for querying event logs. We also introduce a novel translation algorithm to take a backend agnostic OQL query represented in terms of ontology and translate it to an executable DSL query over Elasticsearch.

4.2.2 Conversational systems. In addition to supporting one-shot NLQ systems like ATHENA++, we further demonstrate the effectiveness of using the semantically rich ontologies to build conversational systems [38, 39]. We propose an ontology-driven approach that leverages the domain-specific information captured in the ontology, to automatically bootstrap the conversational system in terms of intents (the questions that the system can identify and answer), entities (the system vocabulary) and the dialog to support the desired interaction with the system in natural language.

In [38] we propose an ontology-based conversational system for querying domain specific knowledge bases. In particular, we build algorithms that map common usage patterns to the elements of the domain ontology representing the underlying data, to identify user intents as well as the key entities that are involved. We automate the process of generating training samples to learn a model for the conversation service to interpret and respond to user queries against the knowledge base.

In [39] we extend the conversational system to support Business Intelligence (BI) applications. The ontology in this case is created from the OLAP cube definition (Business Model) against the underlying data stored in a relational database. The ontology provides an entity-centric view of the OLAP cube definition in terms of Measures (quantifiable attributes), Dimensions (categorical attributes), their hierarchies and relationships. We leverage the common access patterns observed in BI workloads and map them onto the semantically rich BI ontology to generate appropriate intents, their training examples and structure the dialog to support common BI operations.

4.3 Entity Disambiguation

Domain-specific ontologies, created and enriched from heterogeneous data sources, are expected to provide high-quality information to facilitate decision making. Entity disambiguation (also referred to as entity linking) leverages the wealth of such ontologies to identify entities in a data source and finds their correspondences in the ontology.

In [48], we introduce ED-GNN (Figure 4) based on three representative GNNs (GraphSAGE, R-GCN, and MAGNN) for entity disambiguation. The overall idea is to represent the entity mentions in a text snippet as a graph to capture their interdependence, since these entity mentions are likely to share similar or relevant context. We then model entity disambiguation as a graph matching problem. The proposed ED-GNN not only collectively learns the contextual information and structural interdependence of entity mentions in the given text snippets, but also captures discriminative contextual information of entities in an ontology.

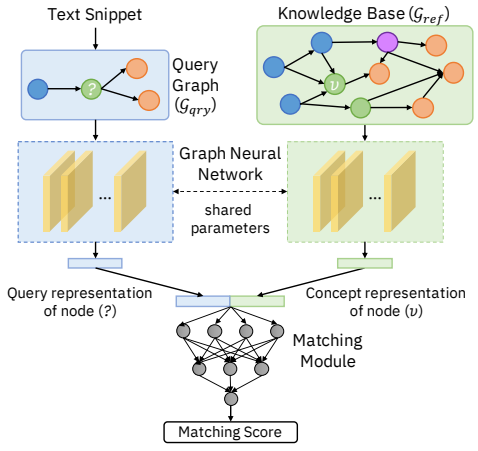

Figure 4: ED-GNN architecture (best viewed in color).

In ED-GNN, we develop two optimizations to improve its disambiguation capability. First, after constructing a *query graph* (representing the entity mentions in a text snippet), ED-GNN augments this graph with domain knowledge from the ontology. Second, ED-GNN is equipped with an effective negative sampling strategy, which challenges ED-GNN to learn from *difficult* samples to improve the model's disambiguation capability. The experimental results in [48] show that ED-GNN with the above two optimization techniques consistently outperforms the state-of-the-art entity disambiguation solutions in five real-world datasets by up to 16.4% in F1 score.

5 RELATED WORK

Ontology matching. Traditional feature-based approaches have been investigated for ontology matching, including terminological-based features, structural-based features and employing external semantic thesauruses for discovering semantically similar entities. LogMap [26] relies on lexical and structural indexes to enhance its scalability. AML [17] also employs various sophisticated features and domain-specific thesauruses to perform ontology matching. Feature-based methods mainly employ crafting features to achieve specific tasks. Unfortunately, these hand-crafted features will be limited for a given task and face the bottleneck of improvement.

Representation learning has limited impacts on ontology matching. DeepAlignment [31] is an unsupervised ontology matching system, which refines pre-trained word embeddings with the descriptions of entities, including synonymy and antonym extracted from general lexical resources and information captured implicitly in ontologies. Similar to DeepAlignment, a framework is introduced for medical ontology alignment [32], based on terminological embeddings. The retrofitted word vectors are learned from the domain knowledge encoded in ontologies and semantic lexicons.

Modular reuse of ontologies. Extracting a minimal module from an ontology for a specific set of target classes and properties from an ontologies comes from the theory of Description Logics, the fragment of first-order logic underlying ontologies. Depending on the expressivity of the target ontology, this problem may even be undecidable [21]. For that reason, efficient syntactic locality-based approximations [21, 27] and module extraction methods based on atomic decomposition [46] have emerged. In our query-answering setting[4], we have used the locality-based approach from an open-source library[5]. For a comprehensive literature review of this area, we refer the readers to [21, 46].

Graph neural networks. Graph representation learning has been shown to be extremely effective, achieving promising results in various domains over graph-structured data [22, 30, 36, 45, 53]. GCN [30] is a graph convolutional network via a localized first-order approximation of spectral graph convolutions. The seminal GNN framework, GraphSAGE [22], learns node embeddings through aggregating from a node's local neighborhood using inductive learning. Graph attention networks (GAT) [45] are introduced to learn the importance between nodes and their neighbors, and fuse the neighbors to perform node classification.

Heterogeneous graph embedding has also received much research attention recently [10, 18, 41, 49], as many KBs also fall under the general umbrella of heterogeneous graphs. For example, R-GCN [41] distinguishes different neighbors with relation-specific weight matrices. Heterogeneous graph attention network (HAN) [49] leverages a graph attention network architecture to aggregate information from the neighbors and then to combine various metapaths through the attention mechanism. Inspired by HAN, HetGNN [55] encodes the content of each node into a vector and then adopts a node type-aware aggregation function to collect information from the neighbors. MAGNN [18] captures all neighbor nodes and the metapath context using both intra-metapath aggregation and inter-metapath aggregation. Thus, the generated node embeddings preserve the comprehensive semantics in the heterogeneous graphs.

Entity disambiguation. For many years, entity disambiguation (also referred to as entity linking) has been an active field of research [43]. A related task, entity matching, has also been studied extensively in the context of structured data [12, 13, 33]. Recently, [20, 37] investigated various DL-based methods for entity matching, and concluded that although DL-based techniques do not offer significant advantages for structured data, they outperform current solutions considerably for textual entity matching. NCEL [9] applies graph convolutional network to integrate both local contextual features and global coherence information for entity linking. COM-AID [14] introduces a composite attentional encode-decode neural network in healthcare. It encodes an entity into a vector and decodes the vector into a text snippet with the help of textual and structural contexts. NormCo [50] is designed for disease normalization. It models entity mentions using a semantic model, which consists of an entity phrase model using word embeddings and a coherence model of other disease mentions using an RNN. The final model combines both sub-models trained jointly.

6 CONCLUSION

In this paper, we describe semantic enrichment of structured data using external semantic knowledge sources for various AI applications. We identify several sub-problems and provide a suite of solutions to address these sub-problems, including entity type detection, ontology creation, ontology matching, module extraction, and ontology unification. We discuss how deep semantic understanding of data empowers four applications: natural interfaces to data, ontology-based query answering, query relaxation, and entity disambiguation.

In general, a given dataset will have correspondences to multiple domain-specific ontologies. The techniques that we discussed in this paper match the data to one ontology at a time. We still need a mechanism to merge the modules extracted from multiple ontologies into one coherent ontology. We plan to address this problem as future work.

[4]https://github.com/IBM/ontology-enriched-query-answering
[5]https://github.com/ernestojimenezruiz/locality-module-extractor

REFERENCES

[1] FIBEN. https://github.com/IBM/fiben-benchmark. Accessed: 2021-06-01.
[2] FIBO. https://spec.edmcouncil.org/fibo/. Accessed: 2021-06-01.
[3] FRO. http://xbrl.squarespace.com/financial-report-ontology/. Accessed: 2021-06-01.
[4] RxNorm. https://www.nlm.nih.gov/research/umls/rxnorm/index.html. Accessed: 2021-06-01.

[5] Schema.org. https://schema.org. Accessed: 2021-06-01.

[6] SNOMED Clinical Terms. https://www.snomed.org/snomed-ct/what-is-snomed-ct. Accessed: 2021-06-01.

[7] S. Ahmetaj, V. Efthymiou, R. Fagin, P. G. Kolaitis, C. Lei, F. Özcan, and L. Popa. Ontology-enriched query answering on relational databases. In *IAAI*, page (to appear). AAAI Press, 2021.

[8] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. G. Ives. DBpedia: A nucleus for a web of open data. In *ISWC*, pages 722–735, 2007.

[9] Y. Cao, L. Hou, J. Li, and Z. Liu. Neural collective entity linking. In *COLING*, pages 675–686, 2018.

[10] Y. Cen, X. Zou, J. Zhang, H. Yang, J. Zhou, and J. Tang. Representation learning for attributed multiplex heterogeneous network. In *SIGKDD*, page 1358–1368, 2019.

[11] J. Chen, G. Alghamdi, R. A. Schmidt, D. Walther, and Y. Gao. Ontology extraction for large ontologies via modularity and forgetting. In *K-CAP*, pages 45–52, 2019.

[12] P. Christen. *Data Matching - Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Data-Centric Systems and Applications. Springer, 2012.

[13] V. Christophides, V. Efthymiou, T. Palpanas, G. Papadakis, and K. Stefanidis. An overview of end-to-end entity resolution for big data. *ACM Comput. Surv.*, 53(6):127:1–127:42, 2021.

[14] J. Dai, M. Zhang, G. Chen, J. Fan, K. Y. Ngiam, and B. C. Ooi. Fine-grained concept linking using neural networks in healthcare. In *SIGMOD*, pages 51–66, 2018.

[15] A. Doan, A. Y. Halevy, and Z. G. Ives. *Principles of Data Integration*. Morgan Kaufmann, 2012.

[16] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: semantics and query answering. *Theor. Comput. Sci.*, 336(1):89–124, 2005.

[17] D. Faria, C. Pesquita, E. Santos, M. Palmonari, I. F. Cruz, and F. M. Couto. The agreementmakerlight ontology matching system. In *OTM*, pages 527–541, 2013.

[18] X. Fu, J. Zhang, Z. Meng, and I. King. Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding. In *WWW*, page 2331–2341, 2020.

[19] R. Goldman and J. Widom. Dataguides: Enabling query formulation and optimization in semistructured databases, 1997.

[20] Y. Govind, P. Konda, P. S. G. C., P. Martinkus, P. Nagarajan, H. Li, A. Soundararajan, S. Mudgal, J. R. Ballard, H. Zhang, A. Ardalan, S. Das, D. Paulsen, A. S. Saini, E. Paulson, Y. Park, M. Carter, M. Sun, G. M. Fung, and A. Doan. Entity matching meets data science: A progress report from the magellan project. In *SIGMOD*, pages 389–403, 2019.

[21] B. C. Grau, I. Horrocks, Y. Kazakov, and U. Sattler. Modular reuse of ontologies: Theory and practice. *J. Artif. Intell. Res.*, 31:273–318, 2008.

[22] W. L. Hamilton, R. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *NIPS*, pages 1024–1034, 2017.

[23] X. Han, L. Hu, J. Sen, Y. Dang, B. Gao, V. Isahagian, C. Lei, V. Efthymiou, F. Özcan, A. Quamar, Z. Huang, and V. Muthusamy. Bootstrapping natural language querying on process automation data. In *2020 IEEE International Conference on Services Computing, SCC 2020, Beijing, China, November 7-11, 2020*, pages 170–177. IEEE, 2020.

[24] J. Hao, C. Lei, V. Efthymiou, A. Quamar, F. Ozcan, Y. Sun, and W. Wang. Medto: Medical data to ontology matching using hybrid graph neural networks. In *SIGKDD*, 2021.

[25] M. Jammi, J. Sen, A. R. Mittal, et al. Tooling framework for instantiating natural language querying system. *PVLDB*, 11(12):2014–2017, 2018.

[26] E. Jiménez-Ruiz and B. C. Grau. Logmap: Logic-based and scalable ontology matching. In *ISWC*, pages 273–288, 2011.

[27] E. Jiménez-Ruiz, B. C. Grau, U. Sattler, T. Schneider, and R. B. Llavori. Safe and economic re-use of ontologies: A logic-based methodology and tool support. In *ESWC*, volume 5021, pages 185–199, 2008.

[28] E. Jiménez-Ruiz, O. Hassanzadeh, V. Efthymiou, J. Chen, and K. Srinivas. Semtab 2019: Resources to benchmark tabular data to knowledge graph matching systems. In A. Harth, S. Kirrane, A. N. Ngomo, H. Paulheim, A. Rula, A. L. Gentile, P. Haase, and M. Cochez, editors, *The Semantic Web - 17th International Conference, ESWC 2020, Heraklion, Crete, Greece, May 31-June 4, 2020, Proceedings*, volume 12123 of *Lecture Notes in Computer Science*, pages 514–530. Springer, 2020.

[29] A. E. Johnson, T. J. Pollard, L. Shen, et al. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3:160035, 2016.

[30] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[31] P. Kolyvakis, A. Kalousis, and D. Kiritsis. DeepAlignment: Unsupervised ontology matching with refined word vectors. In *NAACL*, pages 787–798, 2018.

[32] P. Kolyvakis, A. Kalousis, B. Smith, and D. Kiritsis. Biomedical ontology alignment: an approach based on representation learning. *J. Biomed. Semant.*, 9(1):21:1–21:20, 2018.

[33] H. Köpcke and E. Rahm. Frameworks for entity matching: A comparison. *Data Knowl. Eng.*, 69(2):197–210, 2010.

[34] C. Lei, V. Efthymiou, R. Geis, and F. Ozcan. Expanding query answers on medical knowledge bases. In *EDBT*, pages 567–578, 2020.

[35] C. Lei, F. Özcan, A. Quamar, A. R. Mittal, J. Sen, D. Saha, and K. Sankaranarayanan. Ontology-based natural language query interfaces for data exploration. *IEEE Data Eng. Bull.*, 41(3):52–63, 2018.

[36] Y. Li, D. Tarlow, M. Brockschmidt, and R. S. Zemel. Gated graph sequence neural networks. In *ICLR*, 2016.

[37] S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, G. Krishnan, R. Deep, E. Arcaute, and V. Raghavendra. Deep learning for entity matching: A design space exploration. In *SIGMOD*, page 19–34, 2018.

[38] A. Quamar, C. Lei, D. Miller, F. Ozcan, J. Kreulen, R. J. Moore, and V. Efthymiou. An ontology-based conversation system for knowledge bases. In *SIGMOD*, pages 361–376, 2020.

[39] A. Quamar, F. Özcan, D. Miller, R. J. Moore, R. Niehus, and J. Kreulen. Conversational BI: an ontology-driven conversationsystem for business intelligence applications. *Proc. VLDB Endow.*, 13(12):3369–3381, 2020.

[40] D. Saha, A. Floratou, K. Sankaranarayanan, U. F. Minhas, A. R. Mittal, and F. Özcan. Athena: an ontology-driven system for natural language querying over relational data stores. *PVLDB*, 9(12):1209–1220, 2016.

[41] M. S. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling. Modeling relational data with graph convolutional networks. In *ESWC*, pages 593–607, 2018.

[42] J. Sen, C. Lei, A. Quamar, F. Özcan, V. Efthymiou, A. Dalmia, G. Stager, A. R. Mittal, D. Saha, and K. Sankaranarayanan. ATHENA++: natural language querying for complex nested SQL queries. *Proc. VLDB Endow.*, 13(11):2747–2759, 2020.

[43] W. Shen, J. Wang, and J. Han. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Trans. Knowl. Data Eng.*, 27(2):443–460, 2015.

[44] T. P. Tanon, G. Weikum, and F. M. Suchanek. YAGO 4: A reason-able knowledge base. In *ESWC*, pages 583–596, 2020.

[45] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. In *ICLR*, 2018.

[46] C. D. Vescovo, M. Horridge, B. Parsia, U. Sattler, T. Schneider, and H. Zhao. Modular structures and atomic decomposition in ontologies. *J. Artif. Intell. Res.*, 69:963–1021, 2020.

[47] D. Vrandecic. Wikidata: a new platform for collaborative data collection. In *WWW*, pages 1063–1064, 2012.

[48] A. Vretinaris, C. Lei, V. Efthymiou, X. Qin, and F. Özcan. Medical entity disambiguation using graph neural networks. *CoRR*, abs/2104.01488, 2021.

[49] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu. Heterogeneous graph attention network. In *WWW*, page 2022–2032, 2019.

[50] D. Wright, Y. Katsis, R. Mehta, and C.-N. Hsu. NormCo: Deep disease normalization for biomedical knowledge base construction. In *AKBC 2019*, 2019.

[51] Y. Wu, X. Liu, Y. Feng, Z. Wang, R. Yan, and D. Zhao. Relation-aware entity alignment for heterogeneous knowledge graphs. In *IJCAI*, pages 5278–5284, 2019.

[52] G. Xiao, D. Calvanese, R. Kontchakov, D. Lembo, A. Poggi, R. Rosati, and M. Zakharyaschev. Ontology-based data access: A survey. In *IJCAI*, pages 5511–5519, 2018.

[53] K. Xu, L. Wu, Z. Wang, Y. Feng, and V. Sheinin. Graph2seq: Graph to sequence learning with attention-based neural networks. *CoRR*, abs/1804.00823, 2018.

[54] T. Yu, R. Zhang, K. Yang, et al. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *EMNLP*, pages 3911–3921, 2018.

[55] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla. Heterogeneous graph neural network. In *SIGKDD*, page 793–803, 2019.