Linking Tabular Columns to Unseen Ontologies

Sarthak Dash[0000−0001−8765−4055], Sugato Bagchi[0009−0005−1173−600X],
Nandana Mihindukulasooriya[0000−0003−1707−4842], and Alfio Gliozzo[0000−0002−8044−2911]

IBM Research AI, Yorktown Heights, NY, USA
{sdash,bagchi,gliozzo}@us.ibm.com, nandana@ibm.com

Abstract. We introduce a novel approach for linking table columns to types in an ontology unseen during training. As the target ontology is unknown to the model during training, this may be considered a zero-shot linking task at the ontological level. This task is often a requirement for businesses that wish to semantically enrich their tabular data with types from their custom or industry-specific ontologies without the benefit of initial supervision. In this paper, we describe specific approaches and provide datasets for this new task: training models on open domain tables using a broad source ontology and evaluating them on increasingly difficult tables with target ontologies having different levels of type granularity. We use pre-trained Transformer encoder models and a range of encoding strategies to explore methods of encoding increasing amounts of ontological knowledge, such as type glossaries and taxonomies, to obtain better zero-shot performance. We demonstrate these results empirically through extensive experiments on three new public benchmark datasets.

1 Introduction

Enterprise data assets are now increasingly stored in various types of infrastructure, ranging from in-house database management systems on owned infrastructure to rented services on cloud infrastructure. In order to manage their distributed data assets, businesses invest in "data lakes" [5] that provide a unified view of the metadata associated with the assets regardless of their location. Although this metadata may include names of databases, tables, columns, and associated database schemas, it lacks mappings to an external ontology of interest. Such mappings allow a data consumer to discover, augment, and visualize information from this data lake. In addition, given multiple data consumer roles in an enterprise, there is a need to map columns to multiple custom ontologies, taxonomies, or business term glossaries often with thousands of business terms. Current approaches on linking columns to a target ontology are either rule-based or require training on the same ontology [34]. However, building such a system for each data consumer role is not feasible because each data consumer role within an enterprise may need to link columns to a different business glossary or an ontology. In such a setting, the rule-based approaches will suffer from recall/accuracy issues and supervised approaches will need manual annotations with high human effort/cost. Therefore, it is essential to find strategies to link tabular columns to any given target ontology in an unsupervised manner.

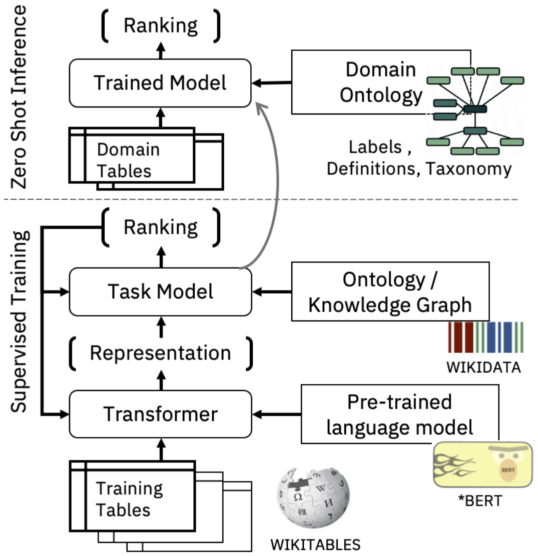

Fig. 1. System for column linking and inference. The task model is fine-tuned over pretrained transformer encoders using WikiTables with columns linked to Wikidata types. During inference, this trained model is applied as-is on tables and/or ontology from a different domain.

Existing approaches such as TABBIE [19], DODUO [31], TURL [11], SATO [33] propose table embedding models that are fine-tuned on either a subset of WebTables corpus from Viznet [16], or on the WikiTables corpus for Column Linking [11]. The ground truth in these datasets consists of only type labels, such as *Researcher*, *Institution*, etc. and does not use other associated ontological knowledge, such as glossary or taxonomy structure. Moreover, the size of the ground truth labels in these datasets is small, i.e., TABBIE and SATO are trained on 78 labels, whereas TURL is trained on 255 labels. Additionally, the ground truth labels in these datasets remain constant between the training and testing phases.

Alternatively, approaches such as [26, 30] or systems participating in the SemTab challenge [22] often couple this task together with linking cell values to entities in an ontology. These approaches rely on cell-linking performance and do not generalize when the target ontology has types but no entities as commonly observed in custom enterprise ontologies.

In this paper, we train Transformer-based deep learning models on tables with known mappings to an existing ontology and assess their performance when those models are applied to an unseen ontology. Figure 1 illustrates an example.

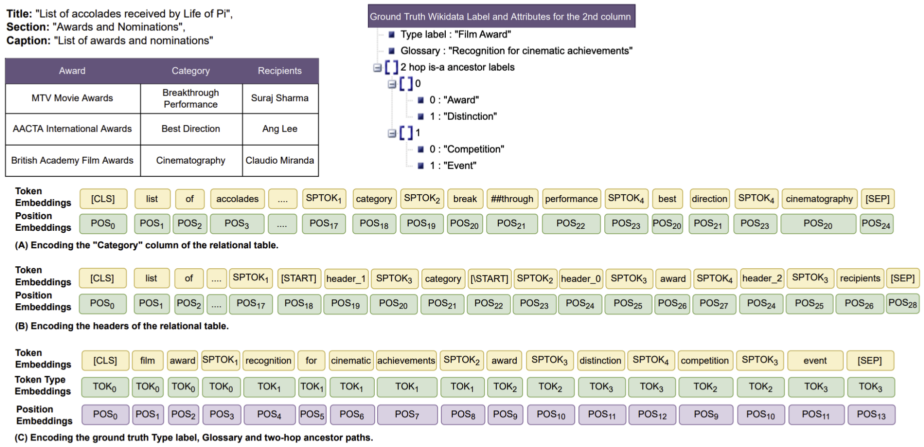
Title: "List of accolades received by Life of Pi",
Section: "Awards and Nominations",
Caption: "List of awards and nominations"
Award
Category
Recipients
MTV Movie Awards
Breakthrough Performance
Suraj Sharma
AACTA International Awards
Best Direction
Ang Lee
British Academy Film Awards
Cinematography
Claudio Miranda
Ground Truth Wikidata Label and Attributes for the 2nd column
Type label : "Film Award"
Glossary : "Recognition for cinematic achievements"
[ ] 2 hop is-a ancestor labels
[ ] 0
0 : "Award"
1 : "Distinction"
[ ] 1
0 : "Competition"
1 : "Event"
Token Embeddings
[CLS] list of accolades .... SPTOK$_1$ category SPTOK$_2$ break ##through performance SPTOK$_4$ best direction SPTOK$_4$ cinematography [SEP]
Position Embeddings
POS$_0$ POS$_1$ POS$_2$ POS$_3$ .... POS$_{17}$ POS$_{18}$ POS$_{19}$ POS$_{20}$ POS$_{21}$ POS$_{22}$ POS$_{23}$ POS$_{20}$ POS$_{21}$ POS$_{23}$ POS$_{20}$ POS$_{24}$
(A) Encoding the "Category" column of the relational table.
Token Embeddings
[CLS] list of .... SPTOK$_1$ [START] header_1 SPTOK$_3$ category [\START] SPTOK$_2$ header_0 SPTOK$_3$ award SPTOK$_4$ header_2 SPTOK$_3$ recipients [SEP]
Position Embeddings
POS$_0$ POS$_1$ POS$_2$ .... POS$_{17}$ POS$_{18}$ POS$_{19}$ POS$_{20}$ POS$_{21}$ POS$_{22}$ POS$_{23}$ POS$_{24}$ POS$_{25}$ POS$_{26}$ POS$_{27}$ POS$_{24}$ POS$_{25}$ POS$_{26}$ POS$_{28}$
(B) Encoding the headers of the relational table.
Token Embeddings
[CLS] film award SPTOK$_1$ recognition for cinematic achievements SPTOK$_2$ award SPTOK$_3$ distinction SPTOK$_4$ competition SPTOK$_3$ event [SEP]
Token Type Embeddings
TOK$_0$ TOK$_0$ TOK$_0$ TOK$_0$ TOK$_1$ TOK$_1$ TOK$_1$ TOK$_1$ TOK$_1$ TOK$_2$ TOK$_2$ TOK$_3$ TOK$_3$ TOK$_2$ TOK$_2$ TOK$_3$ TOK$_3$
Position Embeddings
POS$_0$ POS$_1$ POS$_2$ POS$_3$ POS$_4$ POS$_5$ POS$_6$ POS$_7$ POS$_8$ POS$_9$ POS$_{10}$ POS$_{11}$ POS$_{12}$ POS$_9$ POS$_{10}$ POS$_{11}$ POS$_{13}$
(C) Encoding the ground truth Type label, Glossary and two-hop ancestor paths.

Fig. 2. Overview of encoding query columns, table headers, and the target type for the column linking task. The goal is to classify the column titled CATEGORY within the left table to the type *Film Award*. A glossary and two-hop is-a ancestor labels, i.e., additional information from the target ontology, are also available for each target type label.

Here, we adapt the WikiTables dataset [4] to map table columns to Wikidata types. Moreover, we augment label of each type within our updated datasets with a glossary (using the description of the type) and verbalisation of a partial taxonomy structure (based on `rdfs:subClassOf` property) extracted from the corresponding ontology.

We introduce novel strategies to encode this ontological information and show that by doing so, we obtain models that can better link tabular columns to unseen ontologies. Here we use a Transformer's ability to encode a language model to learn the associations from columns and headers to types within an ontology.

Since most enterprise datasets and custom ontologies are highly proprietary and not openly available, in this work, we simulate a custom enterprise setting using publicly available ontologies. We use three datasets: WikiTables, BioDivTab [1], and Tough Tables [9], and three ontologies: Wikidata, DBpedia, and UMLS Semantic Network [24]. This allows for the evaluation on unseen ontologies in the same general knowledge domain (Wikidata and DBpedia) and on different domains (Wikidata and UMLS Semantic Network).

In summary, we make the following contributions,

1. We propose a new setting for the column linking task where the test ontology is unseen during the training.
2. We introduce three new datasets for the setting proposed above. These include WikiTables with columns mapped to Wikidata and DBpedia types and BioDivTab with columns mapped to UMLS Semantic Network types (Section 3). These datasets and ontologies allow for evaluation within and

across domains. We augment each type label in these datasets with additional glossary information and a partial taxonomy structure.

3. We propose novel strategies for encoding these semantically enriched types (Section 2) and empirically demonstrate their effectiveness on this task in Section 4.

2 Approach

In this section, we provide an overview of our proposed approach for the Column Linking task, also known as Column Type Annotation (CTA) in literature. We model this task as a *ranking* problem. Given a table containing a target column Q_c, headers H, and a list of one or more ground truth types $\mathcal{L}$, our training procedure uses Transformer encoders to learn how to associate a high similarity score between the query representation q_v and type representations from $\mathcal{L}$. The following sections describe our approach in detail.

2.1 Encoding the query column and the headers

This section describes our strategies for encoding the query column Q_c and the table headers H for a given relational table. We hypothesize that the table headers provide additional context to the query column and help determine its type more accurately. Consider the relational table (on the left) in Figure 2. Suppose that the query column Q_c is the second column with the header *Category*. The headers H consist of three entries: Award, Category, and Recipients.

We use six additional special tokens SPTOK_1 through SPTOK_4 and [START], [\START] for encoding using pre-trained Transformer encoder models and follow the strategy used by our previous work [10] to encode the query column Q_c and the headers H. For encoding Q_c, we first concatenate the title, section, and caption fields to generate the metadata text. We then append the column header, i.e., *Category*, to the metadata text using SPTOK_1 as the separator. Next, we concatenate the cell mentions in column Q_c using the separator SPTOK_4, and then append this sentence to the previous one using SPTOK_2. This strategy yields the overall pseudo sentence for the query column in yellow (Fig 2(A)).

For encoding table headers, we model all those headers not belonging to the query column Q_c as a collection of texts. Yet, we also respect that these headers are present in the relational table in a specific sequence (for better understandability). Therefore, we attach additional tags of the form "header_N" where N is a natural number greater than or equal to one. The SPTOK_3 token separates the term "header_k" from the actual value of the header, whereas SPTOK_2 has the same functionality as described before. We use the SPTOK_4 token to concatenate with other column headers and surround Q_c's header with [START], [\START] tokens. Figure 2(B) shows the pseudo sentence for the table headers.

We use the strategy introduced by [10] for the position embeddings. The SPTOK_2 token acts as the pivot: Word pieces to the left of the pivot get numerically increasing position IDs, whereas, for word pieces to the right, the position IDs are numerically increasing but are reset as soon as we hit the $\texttt{SPTOK}_4$ special token. The $\texttt{SPTOK}_4$ token always gets a position ID that is one less than the position ID of the [SEP] token. This assignment of position IDs is shown in green for Figure 2(A), (B). Also, we use the standard strategy for encoding Token Types and attention masks. Therefore, we have omitted their illustrations for brevity.

Finally, the [CLS] vector and the [START] vector corresponding to the final Transformer layer is used as the representations for the query column Q_c and the table headers H, respectively. This encoding strategy ensures that the column representation remains unchanged even if the cell values are randomly shuffled. In other words, the column representation is invariant to the ordering of the cell values within the query column.

2.2 Encoding the target type

In this section, we describe our strategy for encoding the target types. The JSON structure on the right in Figure 2 illustrates the target Wikidata type *Film Award*. Each target type also has a glossary and a list of two-hop is-a ancestor labels from the ontology. In this work, we consider the number of hops as a hyper-parameter that remains fixed throughout.

Figure 2(C) illustrates our strategy for encoding the target type. First, we *linearize* the two-hop is-a ancestor labels as follows. As observed in Figure 2, the two-hop is-a ancestor labels can be viewed as a list of lists wherein the inner list (or a path) contains at most two type label strings. In this example, there are two inner lists (or paths) which are *Award, Distinction* and *Competition, Event*, respectively. We concatenate the type label strings for each path using the special token $\texttt{SPTOK}_3$ as the separator. This action results in a single list of *linearized* paths. All the paths are then concatenated together using the special token $\texttt{SPTOK}_4$ as the separator. This sequence of steps yields a linearized view of the two-hop is-a ancestor label attribute for a given target type.

Finally, we concatenate the type label and the glossary using the special token $\texttt{SPTOK}_1$ as the separator. We then concatenate the result with the linearized view of the two-hop is-a ancestor labels. For the second step, we use the special token $\texttt{SPTOK}_2$ as the separator. This process yields the overall pseudo sentence in yellow (Figure 2(C)).

We use a total of *four* token type IDs TOK_0 through TOK_3 in our architecture. As shown in Figure 2(C), we use the token ID *zero* for the type label, *one* for the glossary, *two* for the parent type labels and *three* for the grand-parent type labels. We initialize TOK_0 and TOK_1 to the two pre-trained token type embeddings of a BERT-*like* model. By BERT-*like* model, we refer to a pre-trained Transformer encoder model that has two Token Type IDs. For TOK_2 and TOK_3, we discovered that initializing it with the pre-trained token type embedding for token ID *one* of a BERT-*like* model generates the best results. We also tried out other initialization strategies, whose results are shown in Table 2.

Besides, we use a similar strategy as Section 2.1 for assigning Position IDs. Moreover, we use the standard approach for encoding attention masks, which is why we have omitted them from Figure 2.

2.3 Putting it all together

Our model architecture is illustrated in Figure 3, which puts together the contents of the previous *two* subsections. The query column Q_c and the table headers H are first encoded using the strategy described in Section 2.1. Once the encoding is complete, it is processed through two Transformer encoder models named $\text{Transformer}_\text{A}$ and $\text{Transformer}_\text{B}$ respectively. The `[CLS]` vector from $\text{Transformer}_\text{A}$ and the `[START]` vector from $\text{Transformer}_\text{B}$ are considered as the representations for Q_c and H respectively. Both these representations are then concatenated together, processed through a linear layer first, followed by a GELU layer [14] to yield the final query vector q_v.

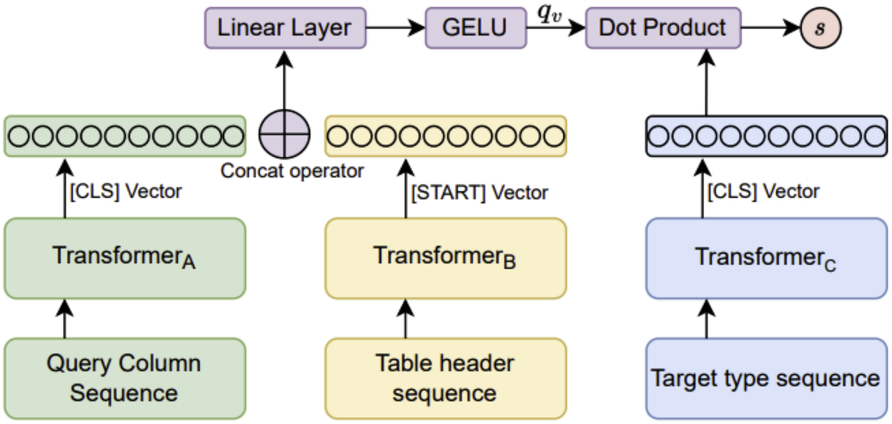

Fig. 3. Overview of our model architecture. Given a query column Q_c and table headers H, Section 2.1 describes how to generate the Query column and Table header sequences. Section 2.2 describes how to build the target type sequence.

On the other hand, the target type labels, together with the associated glossary and two-hop is-a ancestor labels, are first encoded using the strategy described in Section 2.2. It is then processed through $\text{Transformer}_\text{C}$. The overall score s is calculated as a *dot product* between the query vector q_v and the `[CLS]` vector from $\text{Transformer}_\text{C}$.

During the training phase, each instance, i.e., column Q_c, table headers H, and associated table metadata, may have one or more positive ground truth (GT) types. We use *random* sampling to generate *negative* types and then score both positive and negative types using the architecture described in Figure 3. Furthermore, we use a binary cross-entropy loss for training.

During scoring, each test instance is provided a list of all possible type labels, a glossary, and information regarding two-hop is-a ancestor labels from the target ontology. This target ontology need not be the same ontology used during training; it can be a different ontology altogether bought in for scoring purposes only, which is unseen during training. As long as the target ontology has labels, a glossary, and is-a ancestor labels, we can use a Transformer model to encode this information as described above.

Since the list of all possible types from the target ontology stays fixed across all test instances, it can be processed once through Transformer$_{\mathrm{C}}$, and the target type representations can be pre-computed. We can then re-use these pre-computed target type representations to generate a ranked list of types for each test instance. Alternatively, one can also use an off-the-shelf library for large-scale vector-similarity search, such as FAISS [23] for faster scoring. We use Mean Reciprocal Rank (MRR) and Hits@k metrics for evaluations.

3 Dataset Construction

We use three tabular datasets: WikiTables, BioDivTab, and Tough Tables, and three ontologies: Wikidata, DBpedia, and UMLS Semantic network [24]. All of our experiments below are done in English only; we leave multi-lingual column linking models as a source of future work. For training our models, we use the variant of the WikiTables corpus [4] released by [11] under Apache License 2.0, which consists of 580,171 tables. We did an 80:10:10 split of the tables to build the train, valid, and test folds. Splitting at a table level ensures that all the columns from one table fall entirely within a fold.

For the scoring phase, we use WikiTables and Tough Tables datasets with target types from DBpedia, and BioDivTab with target types from UMLS Semantic Network. The following sections describe the annotation procedures for the benchmarks mentioned above.

3.1 Building datasets using WikiTables

The WikiTables dataset contains Wikipedia page links for cell mentions, wherever available. We collect all the available Wikipedia page links for a given column and map them to Wikidata entity QIDs via a SPARQL query. Once the Wikidata QIDs are known, accumulating their corresponding types as a collection $\mathcal{T}$ is another SPARQL query. Assuming this column contains at least three Wikipedia page links, we find the *lowest common ancestor* (LCA) type of $\mathcal{T}$ within four hops. The *lowest common ancestor* (LCA) for K unique types is the closest type in the ontology graph which is a superclass to all the k types. Because the types that are too generic such as *Thing* or *Concept* are less useful for the downstream tasks, only 4 hops of super-types for each type are looked up for finding a common ancestor. If such an LCA type exists, we annotate it as the target type. Note that there can be multiple LCA types, in such cases, we consider all of them as target types.

During this annotation procedure, if the query column has less than three Wikipedia page links, or no *lowest common ancestor* exists within four hops, we assign a special NoType for this column. Moreover, generic Wikidata types such as *class* or *entity* are ignored. Additionally, we under-sample columns from the NoType class to reduce skewness in the training distribution. Specifically, we ensure that the size of the NoType class equals the size of the most frequent Wikidata type within the training fold.

For the validation and test folds, we use an Elastic-search-based Wikidata Lookup to generate the top twenty candidate types for each query column. If this lookup doesn't return any candidates for a query column, then that query column is removed from the fold. If this step returns less than twenty candidate types, we keep the query column and candidate types within the fold. During inference time, the model is asked to rank these candidate types for every query column. We have this lookup step because Wikidata has many types ($\geq$ 50K), and scoring all the Wikidata types for each query column is computationally expensive. In the future, we plan on investigating alternative approaches to this task that do not require a candidate generation step while working with Wikidata.

The annotation procedure, as described above, yields Wikidata types for the WikiTables dataset, whose statistics are as follows: We have 491,739 query columns for train, 61,204 for valid, and 69,135 for the test fold.

Next, we take the test fold tables of the WikiTables dataset and re-annotate the columns to types from DBpedia. We follow a similar strategy as described above. During the construction of this dataset, if we cannot find a ground truth DBpedia type for a particular column, then that column is dropped altogether.

Additionally, because DBPedia has a total of 782 types, we do not employ an Elastic-search-based candidate generation strategy; instead, we ask the model to rank all of the 782 types while scoring for a query column. This procedure yields an additional test dataset containing 47,849 columns from WikiTables test fold annotated DBpedia types.

3.2 Building datasets using BioDivTab

The BioDivTab dataset [2] based on biodiversity research data is a domain-specific tabular dataset consisting of 50 tables and is initially annotated with Wikidata types. Because we want to evaluate on types from an ontology different from the one used for training, we reannotated the columns within this dataset to types from UMLS Semantic Network. We had three annotators for this task; they had access to the ground truth Wikidata type and were asked to manually map the Wikidata type to the best-matching UMLS Semantic Network type. The disagreements in annotations were resolved via majority voting. Because our training dataset did not contain columns with floating point values, we removed such columns from this dataset. We leave the linking of columns containing numerical values to a type within an ontology as a future task.

This annotation procedure yielded a test set with 205 columns in total. Like DBpedia, while scoring for a query column, we ask the model to rank all 127 types belonging to the UMLS Semantic network.

3.3 Tough Tables dataset

The Tough Tables (2T) dataset [9] is based on tables from multiple heterogeneous sources. This dataset is annotated with DBpedia type labels which are manually verified and puts particular emphasis on tables with ambiguous and misspelled mentions. This dataset has 540 query columns overall.

Table 1 below denotes the differences in the distribution of the type labels between datasets used for training and scoring. While scoring on DBpedia types, the model is given 782 types for each query instance to generate a ranking. Of the 782 type labels, only 268, i.e., 34.3%, have a non-empty description. For the others, we use an empty string.

Ontology Property	WD Train	DBP	UMLS SN
Type labels	5,327	782	127
Labels with non-empty glossary	4,588	268	127
Avg 1hop ancestor *is-a* labels per type label	1.94	0.92	0.98
Avg 2hop ancestor *is-a* labels per type label	3.67	0.73	0.95
Labels with $\geq$ *five* 1hop ancestor *is-a* labels	125	0	0
Labels with $\geq$ *five* 2hop ancestor *is-a* labels	1,567	0	0

Table 1. Statistics for WD (Wikidata) Train, DBP (DBpedia) and UMLS SN (Semantic Network). WD train represents statistics from our training dataset.

For training using Wikidata, we have non-empty descriptions for 86.1% of the Wikidata labels. Each Wikidata label, on average, has 3.67 two-hop ancestor is-a type labels. Additionally, 2.3% and 29.4% of the train Wikidata labels have at least five one-hop and two-hop ancestor is-a type labels, respectively. In contrast, DBpedia and UMLS SN have *less than one* two-hop ancestor is-a type label per target type.

4 Experiments and Analysis

We train a model once using the WikiTables training dataset annotated with Wikidata labels. We initialize using a pre-trained Transformer encoder and then fine-tune during our training procedure. Once training is complete, we evaluate independently on *three* test datasets having labels from DBpedia (same broad domain as Wikidata) and UMLS Semantic network (a more specialized biomedical domain).

4.1 Results on WikiTables and DBpedia

Table 2 shows the results on the Column Type annotation (CTA) task for the WikiTables test fold. We show results using both Wikidata (supervised task) and DBpedia, i.e., evaluating on a new ontology unseen during training.

Using Type labels only We describe our results and analysis in *three* blocks depending upon the *Ontological artifacts used* (See Table 2) to build the model. In the *first* block, we use the Type labels only to encode the target type. In this baseline setting, we have *five* models depending upon the encoders used (See Fig 3). The *first* two rows use TinyBERT [21] and BERTBase [12] to encode the query column, the table header, and the target type label.

Ontological artifacts used	Encoder(s)	Wikidata		DBpedia		
		MRR	H@1	MRR	H@1	H@3
Type labels only (Baselines)	TinyBERT	0.84	0.741	0.465	0.306	0.569
	BERTBase	0.891	0.816	0.395	0.294	0.448
	BERTBase + WD-to-DBP lookup	0.891	0.816	0.192	0.154	0.210
	TAPASBase + BERTBase	0.884	0.804	0.378	0.292	0.422
	TABBIE + BERTBase	0.702	0.533	0.279	0.167	0.348
Type labels + **Glossary**	TinyBERT	0.857	0.764	0.489	0.374	0.531
	BERTBase	0.891	0.814	0.380	0.260	0.444
	TAPASBase + BERTBase	0.853	0.759	0.435	0.287	0.537
Type labels + Glossary + **is-a parent labels**	TinyBERT + GraphConv (h=2)	0.859	0.767	0.395	0.244	0.478
	TinyBERT + GATConv (h=2)	0.857	0.766	0.445	0.313	0.497
	TinyBERT (*Baseline*1 encoding, h=2)	0.843	0.746	0.468	0.325	0.562
	TinyBERT (*Baseline*2 encoding, h=2)	0.859	0.770	0.424	0.267	0.520
	TinyBERT (*Our* encoding, h=1)	0.847	0.750	0.496	0.375	0.563
	TinyBERT (*Our* encoding, h=2)	0.854	0.762	**0.533**	**0.404**	**0.601**

Table 2. Results for the CTA task on the WikiTables test fold with labels from Wikidata (supervised task) and DBpedia (linking to unseen ontology). h=k indicates parent *is-a* labels upto k hops. H@k denotes Hits at k metric.

The *third* row in Table 2 illustrates the CTA results for a *pipeline* approach. In this approach, we use the previous BERTBase encoder approach to generate a ranked list of Wikidata (WD) types for each query instance in the test set. For evaluating on DBpedia, we map the WD types to DBP types through a sequence of `owl:sameAs` and `rdf:type` properties using a SPARQL query to the DBpedia endpoint. We refer to this mapping as "WD-to-DBP lookup". The *fourth* row uses a pretrained TAPASBase [15] model to encode the query column and BERTBase to encode the target type label. The *fifth* row uses a pretrained TABBIE [20] model to encode the query table and separate BERTBase models to encode the table metadata and the target type label. TABBIE is the current state-of-the-art on *supervised* column linking and is table-centric, i.e., it encodes the entire table and then gets the embedding for the query column. In contrast, other models are column-centric, i.e., encode only the query column directly. TABBIE performs similarly to TaBERT on the CTA task over Viznet Web Tables [19] and outperforms SATO [33]. Thus, in our analysis, we compare against TABBIE only.

For predicting Wikidata types, we observe that BERTBase has a performance gain of roughly *five* points over TinyBERT on the MRR metric. The TAPAS-Base+BERTBase setup performs roughly similarly to BERTBase. In contrast, using a pretrained TABBIE model to encode columns, as described before, yields poorer results. Unlike [19] wherein a pretrained TABBIE is fine-tuned for CTA over a dataset containing 78 type labels only, in our experiments, we fine-tune over a training dataset containing 5,327 Wikidata Types (See Table 1). Additionally, unlike [19], our overall architecture is a Siamese network architecture designed to learn meaningful representations for the query column and the type labels to map the query column to the correct type label. The increase in the total number of target types, combined with a change in architecture, could be why fine-tuned TABBIE-based models yields poor results.

However, when evaluated on DBpedia target labels, we observe that the TinyBERT has the best performance of the lot. A plausible reason behind this could be that since BERTBase, TAPASBase, and TABBIE are bigger in size than TinyBERT, they are likely overfitting on the training distribution and are not generalizing as well as TinyBERT.

We observe that the *pipeline* approach (*third* row) performs very poorly when evaluated on DBpedia. Compared to Wikidata, this drop in performance can be attributed to errors within the "WD-to-DBP lookup". This lookup has insufficient coverage. Because it maps 50K Wikidata (WD) types to 782 DBpedia (DBP) types, many predicted WD types do not have an equivalent DBP type. Moreover, the alignments within this lookup are not consistent. For example, the WD type Q40357 (label: *prison*) does not map to the *dbo:Prison* DBP type. Due to these issues, using a human-mapped alignment can cause errors while linking to unseen ontologies. Moreover, based on the target unseen ontology, such alignments may or may not exist.

Using Type labels and glossary In Rows *six* through *eight* of Table 2, we use Type labels and related glossary information to encode the target type sequence. In this setup, the type label gets the token ID zero and the glossary gets the token ID one (Sec 2.2). Here, we use TinyBERT only, BERTBase only, and TAPASBase + BERTBase, all of which have been introduced before.

Here we are using additional glossary information compared to the *first* block in Table 2. Therefore, we believe that the models' performance *with* glossary information would be better than *without* glossary information. For TinyBERT, it is true, i.e., MRR on both Wikidata and DBpedia targets improves *with* glossary. For BERTBase, the MRR stays roughly the same *with* glossary compared to *without*. For TAPASBase + BERTBase, the MRR under DBpedia target improves *with* glossary. While comparing models in this block, we observe that TinyBERT still has the best MRR compared to others when scored against unseen DBpedia types.

Using Type labels, glossary, and is-a parent labels In rows *nine* through *thirteen* of Table 2, we use Type labels, glossary, and is-a parent labels, i.e., a partial Taxonomy structure to encode the target type. We use TinyBERT for all experiments in this setting because of its smaller size, i.e., TinyBERT is roughly 7.5x-12x smaller than other Transformer encoders introduced in Sec 4.1. We are interested in exploring how much we could push the envelope on the CTA task by using semantically enriched types with a smaller-sized transformer encoder.

Ontological artifacts used	Encoder(s)	BioDivTab		2T	
		MRR	Hits at 10	MRR	Hits at 1
Type labels only (Baseline)	TinyBERT	0.243	0.444	0.325	0.222
	BERTBase	0.262	0.537	0.296	0.219
	TAPASBase + BERTBase	0.253	0.409	0.296	0.228
	TABBIE + BERTBase	0.089	0.234	0.145	0.059
Type labels + Glossary + **2hop is-a parent labels**	TinyBERT + GATConv	**0.264**	0.444	**0.350**	0.231
	TinyBERT (*Our* Encoding)	0.193	**0.566**	**0.350**	**0.263**

Table 3. Results on BioDivTab and 2T datasets with types from UMLS Semantic network and DBpedia respectively.

The *ninth* and *tenth* rows of Table 2 use GraphConv [25] and GATConv [32] layers to encode the two-hop is-a partial taxonomy structure. This partial taxonomy structure can be modeled as a Graph wherein the nodes correspond to the types and edges correspond to “is-a” relation between the types. Only the target type node in this graph has a label and a glossary; all other type nodes only have a label string. We process the nodes through TinyBERT and use the `[CLS]` vector at the final layer as the node embedding. While building these models, we employ *two* layers of either GraphConv or GATConv, with a ReLU and a drop-out layer in between. The target type node embedding is then used to score against the query vector q_v as shown in Figure 3.

The *eleventh* row denotes *Baseline*1 setup. Here, we use parent type labels up to two hops, and all the tokens in the target type sequence (Sec 2.2) use a token type ID of *zero*. The *twelfth* row denotes the *Baseline*2 encoding. Compared to *Baseline*1, here we use *four* token type IDs, one each for the target type label, glossary, parent, and grand-parent types. Additionally, we initialize the two additional token type IDs (Sec 2.2) using Xavier Initialization [13] prior to fine-tuning.

The final *two* rows denote models implementing our proposed strategy for encoding semantically enriched types. Our encoding strategy (Row *fourteen*), even when applied on TinyBERT, yields the best performance. Comparisons amongst *Baseline*1, *Baseline*2, and *Our* encodings, i.e., rows eleven, twelve, and fourteen, illustrate the ablation studies detailing the impact of different encoding and initialization strategies on this task. These results indicate that encoding richer ontological knowledge via meaningful representations results in better performance on an unseen ontology within the same general knowledge domain, i.e., from Wikidata to DBpedia in this case.

4.2 Results on BioDivTab and 2T

Table 3 illustrates the performance of our trained model when evaluated on the BioDivTab and 2T datasets with labels from UMLS Semantic Network and DBpedia respectively.

Following Sec 4.1, we divide the results into two groups and use a total of *six* encoders for analysis. The sixth row uses TinyBERT to implement our strategy for encoding semantically enriched types. We separately analyze our results on BioDivTab and 2T datasets in the following sections.

Analysis on BioDivTab As described before, the tables in BioDivTab belong to a biodiversity research domain and not an open domain like WikiTables. Moreover, the target UMLS Semantic network is from the biomedical domain which is more specialized than Wikidata, which was used for training. Thus, the results in Table 3 are much lower than those in Table 2.

We observe that our encoding strategy using TinyBERT yields the best Hits at 10 scores, outperforming baselines employing much larger encoders. However, in this case, TinyBERT+GATConv has the best performance overall in terms of MRR.

We manually analyzed the top-ranked predictions returned by the TinyBERT model trained using our proposed encoding strategy (Row *six* of Table 3). Unlike WikiTables, which contains English cell values, this dataset contains quite a few query columns having Latin cell values (corresponding to species names) which confuses our models trained on WikiTables. Therefore it makes errors while differentiating between organism types based on Latin cell values. Moreover, BioDivTab does not contain any metadata information to provide context. For columns containing numerical values, this absence of context promotes misclassifications, such as, our model predicts *Age Group* when the true label is *Quantitative Concept*. We show a few examples of the outputs generated by our proposed approach in Section 8.1.

Analysis on 2T From the results in Table 3, we observe that our proposed encoding strategy, even when applied on a smaller Transformer encoder, yields the best performance compared to baseline approaches, thereby indicating the effectiveness of our approach. Because 2T contains misspelled mentions, it likely confuses the models since they were trained on instances that did not have misspells. This could be a plausible reason why the performance on this dataset is lower than that of the WikiTables test fold with DBpedia target ontology (Table 2).

We manually analyzed the top-ranked predictions returned by the TinyBERT model trained using our proposed encoding strategy (Row *six*). We notice that, quite often, the model outputs are at a different granularity than the ground truth types. For example, there are four query columns having mentions of lakes with a ground truth type *place*; however, our above model correctly classifies it as *lake*. In another example, the model identifies *four* columns containing names of monarchs as *monarch*, with the ground truth label being *person*. Additionally, six columns contain names of cities with a ground truth type *place*. In this case, our model predicts it as *community*. We believe that the top-ranked predictions for the first two examples are better than the existing ground truth label. In contrast, it is worse for the last example.

5 Related Work

Linking tabular columns to types in an ontology is a crucial task for many business intelligence tasks such as semantic retrieval, data exploration, knowledge discovery, etc. [6, 7, 11] have studied this task based on cell values only, i.e., using only the available information in a table for linking columns without doing entity linking first. [11] introduce a new dataset for the supervised CTA task by annotating WikiTables columns with Freebase types.

[16] introduced Viznet, a large-scale corpus of over 31 million datasets. The dataset of particular interest here is the WebTables dataset. [33] introduce SATO, a supervised model that combines topic modeling and structured learning with single-column type prediction based on Sherlock [17]. SATO is trained on a subset of the WebTables dataset comprising relational tables with valid headers only.

The Column Type Annotation task within the SemTab challenge [22] benchmarks systems that annotate tabular data to types within a target ontology. For this task, participating systems are not given training data. They are asked to output a type for each query instance within a collection of held-out tables. Systems such as MTab [27], JenTab [1], Kepler-aSI [3], DAGOBAH [18], etc., participating in this challenge follow a pipeline architecture. The first step links cell mentions to entities within the target ontology. The second step predicts the most likely type for the query column based on the linking results. MTab and DAGOBAH also use additional information from the graph, such as entity relations, to improve cell linking accuracy. These approaches rely on the cell linking performance and suffer if the target ontology does not contain any entities to link to, which is very common in industry-specific ontologies.

Entity typing in a zero-shot fashion over textual data is a well-studied problem. FIGER [28] leverages Wikipedia descriptions of types to learn a multi-class classifier given an entity mention and its context. MZET [35] uses a memory network that models the relationship between the entity mention and the entity type and transfers the knowledge from seen entity types to the zero-shot ones. NZFET [29] uses a bi-LSTM architecture and employs entity type attention to focus on information relevant to the entity type.

MSF [8] uses additional auxiliary information such as WordNet glossaries, type hierarchy, and prototype mentions to get richer type representations. This approach independently models the interaction between the query mention and each auxiliary information. It obtains fused results in the next step. In contrast, our method jointly encodes the type label, the glossary, and the partial taxonomy structure to yield more meaningful type representations.

6 Conclusion

This paper introduces a novel approach for linking tabular columns to types in a new ontology unseen during training. We argue that this task is vital for business intelligence applications wherein data discovery needs within enterprise data lakes vary from one consumer role to another. We further argued that current academic datasets used for research in Table Understanding are unsuitable for this task and therefore introduced three new datasets. We enriched the ground truth type label within each new dataset with a glossary and a partial taxonomy structure. We show that encoding this auxiliary information for types via our proposed approach yields meaningful representations, which we can combine to build models that perform well on this task. Through extensive evaluation and analysis on these new datasets, we demonstrate the effectiveness of our proposed approach.

7 Limitations

In this work, we have used English WikiTables as the source of tabular datasets and trained a column linking model using Wikidata types. We then apply this trained model to score query columns against target ontologies having different levels of type granularities. As evidenced by results in Sec 4.2, models trained on broad-domain WikiTables might not be reliably used to handle domain-specific jargon when working with the BioDivTab dataset, which contains cell values from the biodiversity research domain. Alternatively, if the tabular dataset contains misspells (e.g., 2T dataset), which is characteristic of many real-world datasets, hoping that a model trained on WikiTables only will generalize in such cases, might not hold.

In this work, we have shown results using Relational Tables only, i.e., tables with a well-defined 2D structure. If a user brings in, let's say, a non-relational table and queries it for Column linking against our WikiTables-only trained model, the results returned might be far from ideal. In other words, the scope of the empirical evaluations covers 2D Relational Tables only.

Summarizing the above points, we believe that, with this implementation, the Zero-shot performance on the Column linking task, when evaluated on domains far apart from the one used during training, would be less than ideal. In such cases, either building more extensive training data spanning multiple domains and using it for training or employing a human-in-the-loop approach to provide feedback to model predictions and then taking necessary corrective actions could help alleviate the limitations described above.

Supplemental Material Statement: The variant of the WikiTables corpus used in our work (and as released by [11]) are available at: https://github.com/sunlab-osu/TURL. Additionally, the original dataset files for BioDivTab annotated with Wikidata, and Tough Tables annotated with DBPedia are available at https://zenodo.org/record/6461556 and https://zenodo.org/record/6211551 respectively.

We have attached the WikiTables benchmark, annotated with Wikidata and DBPedia types, and the BioDivTab benchmark, annotated with UMLS Semantic Network types, as supplemental materials to this submission. We will provide the source code corresponding to this work, once the anonymity period ends.

The following enumerates the experimental details and the hyperparameters used in all our experiments. We performed a grid search using the validation fold of the WikiTables dataset containing Wikidata labels.

For the learning rate, we performed a grid search over {2e-5, 3e-5, 5e-5} and decided to use 2e-5. All of our experiments are run for a maximum of 40 epochs, with an early stopping criterion of 5 epochs. We had a fixed batch size of 64 in all our experiments. We used *random* negative sampling in all our experiments to generate negatives. For the transformer models in this task, we performed a grid search for the max tokenizer length parameter over {128, 256} and decided to use 128. We used a *longest first* truncation strategy, i.e., any sequence longer than 128 tokens is truncated to a maximum token length of 128.

For experiments using Graph Neural Networks (GNN), we used *two* GNN layers overall. We also used a ReLU layer and a drop-out layer between the two GNN layers. For the drop-out value, we performed a grid search over {0.1, 0.2, 0.3, 0.5} and used 0.1 for GATConv, and 0.2 for GraphConv. For experiments using TABBIE [20], we used a batch size of 2.

Furthermore, unless otherwise mentioned, we impose a max compute budget of 48 hours (or 72 hours if a pretrained TABBIE model is used). If a training run did not converge after 48 hours (or 72 hours for TABBIE), we used its last serialized checkpoint for scoring. The compute budget ensures that the carbon footprint corresponding to these training runs is bounded.

We used PyTorch v1.8.1, torch-scatter v2.0.9, torch-geometric v2.1.0.post1, and torch-sparse v0.6.12 for running our experiments on Linux RHEL v8.5 operating system using Intel x86 CPU and NVIDIA A100 GPU machines. All of our experiments use a maximum RAM of 32G and a random seed value of 73.

References

1. Abdelmageed, N., Schindler, S.: Jentab meets semtab 2021's new challenges. In: SemTab@ ISWC. pp. 42–53 (2021)
2. Abdelmageed, N., Schindler, S., König-Ries, B.: BiodivTab: A Tabular Benchmark based on Biodiversity Research Data. In: SemTab@ISWC, submitted (2021)
3. Baazouzi, W., Kachroudi, M., Faiz, S.: Kepler-asi at semtab 2021. In: SemTab@ ISWC. pp. 54–67 (2021)
4. Bhagavatula, C.S., Noraset, T., Downey, D.: Tabel: Entity linking in web tables. In: Arenas, M., Corcho, Ó., Simperl, E., Strohmaier, M., d'Aquin, M., Srinivas, K.,

Groth, P., Dumontier, M., Heflin, J., Thirunarayan, K., Staab, S. (eds.) The Semantic Web - ISWC 2015 - 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9366, pp. 425–441. Springer (2015). https://doi.org/10.1007/978-3-319-25007-6_25, https://doi.org/10.1007/978-3-319-25007-6_25

5. Bogatu, A., Fernandes, A.A.A., Paton, N.W., Konstantinou, N.: Dataset discovery in data lakes. 2020 IEEE 36th International Conference on Data Engineering (ICDE) pp. 709–720 (2020)
6. Chen, J., Jiménez-Ruiz, E., Horrocks, I., Sutton, C.: Colnet: Embedding the semantics of web tables for column type prediction. In: The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019. pp. 29–36. AAAI Press (2019). https://doi.org/10.1609/aaai.v33i01.330129, https://doi.org/10.1609/aaai.v33i01.330129
7. Chen, J., Jiménez-Ruiz, E., Horrocks, I., Sutton, C.: Learning semantic annotations for tabular data. In: Kraus, S. (ed.) Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019. pp. 2088–2094. ijcai.org (2019). https://doi.org/10.24963/ijcai.2019/289, https://doi.org/10.24963/ijcai.2019/289
8. Chen, Y., Jiang, H., Liu, L., Shi, S., Fan, C., Yang, M., Xu, R.: An empirical study on multiple information sources for zero-shot fine-grained entity typing. In: Moens, M., Huang, X., Specia, L., Yih, S.W. (eds.) Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021. pp. 2668–2678. Association for Computational Linguistics (2021). https://doi.org/10.18653/v1/2021.emnlp-main.210, https://doi.org/10.18653/v1/2021.emnlp-main.210
9. Cutrona, V., Bianchi, F., Jiménez-Ruiz, E., Palmonari, M.: Tough tables: Carefully evaluating entity linking for tabular data. In: International Semantic Web Conference. pp. 328–343. Springer (2020)
10. Dash, S., Bagchi, S., Mihindukulasooriya, N., Gliozzo, A.: Permutation invariant strategy using transformer encoders for table understanding. In: Findings of the Association for Computational Linguistics: NAACL 2022. pp. 788–800. Association for Computational Linguistics, Seattle, United States (Jul 2022). https://doi.org/10.18653/v1/2022.findings-naacl.59, https://aclanthology.org/2022.findings-naacl.59
11. Deng, X., Sun, H., Lees, A., Wu, Y., Yu, C.: TURL: table understanding through representation learning. Proc. VLDB Endow. **14**(3), 307–319 (2020). https://doi.org/10.5555/3430915.3442430, http://www.vldb.org/pvldb/vol14/p307-deng.pdf
12. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: Burstein, J., Doran, C., Solorio, T. (eds.) Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers). pp. 4171–4186. Association for Computational Linguistics (2019). https://doi.org/10.18653/v1/n19-1423, https://doi.org/10.18653/v1/n19-1423
13. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Teh, Y.W., Titterington, D.M. (eds.) Proceedings of the Thirteenth International Conference on Artificial Intelligence

and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010. JMLR Proceedings, vol. 9, pp. 249–256. JMLR.org (2010), http://proceedings.mlr.press/v9/glorot10a.html

14. Hendrycks, D., Gimpel, K.: Gaussian error linear units (gelus). arXiv preprint arXiv:1606.08415 (2016)
15. Herzig, J., Nowak, P.K., Müller, T., Piccinno, F., Eisenschlos, J.: TaPas: Weakly supervised table parsing via pre-training. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. pp. 4320–4333. Association for Computational Linguistics, Online (Jul 2020). https://doi.org/10.18653/v1/2020.acl-main.398, https://aclanthology.org/2020.acl-main.398
16. Hu, K., Gaikwad, N., Bakker, M., Hulsebos, M., Zgraggen, E., Hidalgo, C., Kraska, T., Li, G., Satyanarayan, A., Demiralp, Ç.: Viznet: Towards a large-scale visualization learning and benchmarking repository. In: Proceedings of the 2019 Conference on Human Factors in Computing Systems (CHI). ACM (2019)
17. Hulsebos, M., Hu, K.Z., Bakker, M.A., Zgraggen, E., Satyanarayan, A., Kraska, T., Demiralp, Ç., Hidalgo, C.A.: Sherlock: A deep learning approach to semantic data type detection. In: Teredesai, A., Kumar, V., Li, Y., Rosales, R., Terzi, E., Karypis, G. (eds.) Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019. pp. 1500–1508. ACM (2019). https://doi.org/10.1145/3292500.3330993, https://doi.org/10.1145/3292500.3330993
18. Huynh, V.P., Liu, J., Chabot, Y., Deuzé, F., Labbé, T., Monnin, P., Troncy, R.: Dagobah: Table and graph contexts for efficient semantic annotation of tabular data. In: SemTab@ ISWC. pp. 19–31 (2021)
19. Iida, H., Thai, D., Manjunatha, V., Iyyer, M.: TABBIE: pretrained representations of tabular data. In: Toutanova, K., Rumshisky, A., Zettlemoyer, L., Hakkani-Tür, D., Beltagy, I., Bethard, S., Cotterell, R., Chakraborty, T., Zhou, Y. (eds.) Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021. pp. 3446–3456. Association for Computational Linguistics (2021). https://doi.org/10.18653/v1/2021.naacl-main.270, https://doi.org/10.18653/v1/2021.naacl-main.270
20. Iida, H., Thai, D., Manjunatha, V., Iyyer, M.: TABBIE: Pretrained representations of tabular data. In: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 3446–3456. Association for Computational Linguistics, Online (Jun 2021). https://doi.org/10.18653/v1/2021.naacl-main.270, https://aclanthology.org/2021.naacl-main.270
21. Jiao, X., Yin, Y., Shang, L., Jiang, X., Chen, X., Li, L., Wang, F., Liu, Q.: Tinybert: Distilling BERT for natural language understanding. In: Cohn, T., He, Y., Liu, Y. (eds.) Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020. Findings of ACL, vol. EMNLP 2020, pp. 4163–4174. Association for Computational Linguistics (2020). https://doi.org/10.18653/v1/2020.findings-emnlp.372, https://doi.org/10.18653/v1/2020.findings-emnlp.372
22. Jiménez-Ruiz, E., Hassanzadeh, O., Efthymiou, V., Chen, J., Srinivas, K.: Semtab 2019: Resources to benchmark tabular data to knowledge graph matching systems. In: European Semantic Web Conference. pp. 514–530. Springer (2020)

23. Johnson, J., Douze, M., Jégou, H.: Billion-scale similarity search with gpus. CoRR **abs/1702.08734** (2017), http://arxiv.org/abs/1702.08734
24. McCray, A.T.: An upper-level ontology for the biomedical domain. Comparative and Functional Genomics **4**, 80 – 84 (2003)
25. Morris, C., Ritzert, M., Fey, M., Hamilton, W.L., Lenssen, J.E., Rattan, G., Grohe, M.: Weisfeiler and leman go neural: Higher-order graph neural networks. In: The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, USA, January 27 - February 1, 2019. pp. 4602–4609. AAAI Press (2019). https://doi.org/10.1609/aaai.v33i01.33014602, https://doi.org/10.1609/aaai.v33i01.33014602
26. Mulwad, V., Finin, T., Syed, Z., Joshi, A.: Using linked data to interpret tables. In: Hartig, O., Harth, A., Sequeda, J.F. (eds.) Proceedings of the First International Workshop on Consuming Linked Data, Shanghai, China, November 8, 2010. CEUR Workshop Proceedings, vol. 665. CEUR-WS.org (2010), http://ceur-ws.org/Vol-665/MulwadEtAl_COLD2010.pdf
27. Nguyen, P., Yamada, I., Kertkeidkachorn, N., Ichise, R., Takeda, H.: Semtab 2021: Tabular data annotation with mtab tool. In: Jiménez-Ruiz, E., Efthymiou, V., Chen, J., Cutrona, V., Hassanzadeh, O., Sequeda, J., Srinivas, K., Abdelmageed, N., Hulsebos, M., Oliveira, D., Pesquita, C. (eds.) Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching co-located with the 20th International Semantic Web Conference (ISWC 2021), Virtual conference, October 27, 2021. CEUR Workshop Proceedings, vol. 3103, pp. 92–101. CEUR-WS.org (2021), http://ceur-ws.org/Vol-3103/paper8.pdf
28. Obeidat, R., Fern, X., Shahbazi, H., Tadepalli, P.: Description-based zero-shot fine-grained entity typing. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). pp. 807–814 (2019)
29. Ren, Y., Lin, J., Zhou, J.: Neural zero-shot fine-grained entity typing. In: Seghrouchni, A.E.F., Sukthankar, G., Liu, T., van Steen, M. (eds.) Companion of The 2020 Web Conference 2020, Taipei, Taiwan, April 20-24, 2020. pp. 846–847. ACM / IW3C2 (2020). https://doi.org/10.1145/3366424.3382725, https://doi.org/10.1145/3366424.3382725
30. Ritze, D., Lehmberg, O., Bizer, C.: Matching HTML tables to dbpedia. In: Akerkar, R., Dikaiakos, M.D., Achilleos, A., Omitola, T. (eds.) Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics, WIMS 2015, Larnaca, Cyprus, July 13-15, 2015. pp. 10:1–10:6. ACM (2015)
31. Suhara, Y., Li, J., Li, Y., Zhang, D., Demiralp, Ç., Chen, C., Tan, W.C.: Annotating columns with pre-trained language models. arXiv preprint arXiv:2104.01785 (2021)
32. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings. OpenReview.net (2018), https://openreview.net/forum?id=rJXMpikCZ
33. Zhang, D., Suhara, Y., Li, J., Hulsebos, M., Demiralp, Ç., Tan, W.: Sato: Contextual semantic type detection in tables. Proc. VLDB Endow. **13**(11), 1835–1848 (2020), http://www.vldb.org/pvldb/vol13/p1835-zhang.pdf
34. Zhang, S., Balog, K.: Web table extraction, retrieval, and augmentation: A survey. ACM Trans. Intell. Syst. Technol. **11**(2), 13:1–13:35 (2020). https://doi.org/10.1145/3372117, https://doi.org/10.1145/3372117

35. Zhang, T., Xia, C., Lu, C.T., Philip, S.Y.: Mzet: Memory augmented zero-shot fine-grained named entity typing. In: Proceedings of the 28th International Conference on Computational Linguistics. pp. 77–87 (2020)

8 Appendix

8.1 Model Predictions

The following tables below shows examples of predictions returned by our proposed model built using a pretrained TinyBERT encoder. This model is trained using Wikidata labels and is asked to predict from the DBpedia target ontology for the top two tables. For the bottom two tables, the model predicts from the UMLS Semantic Network (UMLS SN).

The *first* row in the block titled **Top model prediction** returns model predictions using Type labels only. The *second* row returns predictions using Type labels and associated glossaries. The *final* row in this block returns predictions using our proposed encoding strategy. Note that the BioDivTab benchmark does not contain table metadata.

PageTitle SecTitle	1993-94 NBA Season Statistics leaders
Header	Player
Query column	David Robinson Dennis Rodman John Stockton Dikembe Mutombo Shaquille O'Neal
Dataset Ontology	WikiTables DBpedia
Top model prediction	agent american football coach basketball player
True label	basketball player

PageTitle SecTitle	1970 TANFL season 1970 TANFL Ladder
Header	Team
Query column	Sandy Bay Clarence New Norfolk North Hobart Glenorchy
Dataset Ontology	WikiTables DBpedia
Top model prediction	sports club sports club australian football team
True label	australian football team

Header	Species Group
Query column	plants bryophytes lichens moths heteroptera
Dataset Ontology	BioDivTab UMLS SN
Top model prediction	organism attribute human organism
True label	organism

Header	genus
Query column	ambloplites catostomus chrosomus notropis clinostomus
Dataset Ontology	BioDivTab UMLS SN
Top model prediction	organism attribute organism function bacterium
True label	fish