Metadata Unification in Open Data with Gnomon

Christina Christodoulakis
christina@cs.toronto.edu
University of Toronto
Toronto, Canada

Moshe Gabel
mgabel@cs.toronto.edu
University of Toronto
Toronto, Canada

Angela Demke Brown
demke@cs.toronto.edu
University of Toronto
Toronto, Canada

ABSTRACT

Open Data authors often supplement their data sets with attribute metadata described in CSV files. However, authors rarely adhere to common metadata publishing standards. This makes consuming such metadata more difficult, for example, when trying to use them to integrate tables from different sources. In this work, we highlight the novel problem of *metadata unification*. We first formulate attribute metadata unification in Open Data, and show that classic schema matching techniques meant for matching data tables perform poorly in this setting. Next, we propose a unified model for Open Data attribute metadata. Finally, we describe Gnomon, a strong baseline approach for metadata unification. We manually annotate 218 files split for development and test, and show that Gnomon is 31-32 percentage points more accurate than standard schema matching approaches. We are making available both Gnomon and the metadata unification data set as open source.

1 INTRODUCTION

Data published in Open Data repositories are often accompanied by supplementary author-curated documentation with the goal of providing data consumers with the information necessary to understand and use it. This carefully curated documentation often describes what is known as tribal knowledge or business metadata, such as intended usage, context, collection methodologies, semantics of attributes, as well as value formatting, domains, and data types. Rich, standardized metadata has several benefits. Digital repositories rely on metadata for effective search [40]. Data consumers rely on metadata for effective and efficient decision-making, minimizing misinterpretation or data misuse [31].

Several community standards for Open Data metadata have been proposed. DCAT [45] supports metadata at a data set granularity, while Schema.org [39] provides a basic set of limited metadata properties that can be used to describe attributes of data. The W3C has proposed annotations at attribute granularity [10], The Data Documentation Initiative [43] is a metadata standard developed for the description of data from social, behavioral, and economic sciences. Finally, Frictionless Data [20] provides a lightweight specification for data packaging and metadata, primarily targeting data interoperability. In practice, however, authors of Open Data documentation often do not adhere to common metadata publishing standards [22]. Instead, they publish tabular documentation in ad-hoc, mutually-inconsistent schemas.

This diversity forces consumers to explore data sets manually on a case-by-case basis to benefit from metadata described in table documentation files. Several approaches have been proposed to avoid such tedious manual work. *Schema matching* approaches have previously been used to unify disparate data sources, and could be used to convert documentation into unified metadata. As we show in Figure 3, popular baseline approaches for general schema matching perform poorly in this setting. Approaches like Cupid [29] are designed to apply to hierarchical XML schemas, rather than a single flat table. Distribution-based approaches like Zhang et al. [47] and COMA[13, 14] were designed for tables containing data as opposed to metadata. Data tables are often large and follow a consistent form: multiple columns, each comprised of a header followed by lots of data of the same type and semantics. In addition, column semantics in data tables are typically well distinguishable. Those implicit assumptions do not hold for documentation tables, which aim to describe metadata. These are much smaller and less consistent: authors take liberties with value formatting within columns, and values are frequently free-form text. In addition, documentation fields frequently describe semantically similar concepts, making it challenging to distinguish the correct mapping to a unifying metadata model for methods like Similarity Flooding [30] or JaccardLevenshtein[23] (see Section 5 for per-method analysis). Many approaches for search and integration rely on *metadata generation* [1, 4–8, 11, 18, 23–28, 33, 41, 42, 48, 49], ignoring the existing carefully curated documentation. Finally, *ontological alignment* [3] works to align large, formal ontologies to improve matching. Ontologies, however, tend to have complex hierarchical structures, are rigorously defined, and are usually large. In contrast, the attribute metadata schemas we have observed in Open Data repositories are not formal ontologies. They are small, flat, and very loosely defined, with fields that can have overlapping semantics.

In summary, existing approaches are a poor match for the particular problem of using author-curated metadata from Open Data repositories: metadata generation simply ignores it, ontological alignment was not designed for it, and schema matching performs poorly due to the metadata's unique characteristics.

Our Contributions: In this paper, we identify the *metadata unification* problem – unifying author-curated attribute metadata in documentation files from Open Data portals, propose a strong baseline method to address it, and provide a data set for future researchers to expand on. Specifically, our contributions are:

(1) We formulate the problem of attribute *metadata unification* in Open Data documentation as a variant of schema matching, and show that classic schema matching approaches perform poorly in this setting.
(2) We define a formal metadata model for unifying diverse Open Data documentation curated by data authors.
(3) We propose Gnomon, a fast, simple, and easy-to-implement baseline approach for metadata unification based on classification followed by bipartite graph optimization.
(4) We collect 218 Open Data documentation CSV files, and annotate them with mappings to our metadata model. To our knowledge, this is the first data set for metadata unification.
(5) We show that Gnomon outperforms popular schema matching methods for the problem of unifying attribute metadata from Open CSV documentation files.

We have open sourced both Gnomon and the data set.[1]

2 THE METADATA UNIFICATION PROBLEM

Metadata refers to descriptive information about data, such as attributes, properties, tags, labels, and other contextual details that help describe and organize the data. We define *metadata unification* as the process of standardizing and consolidating metadata from different sources and formats into a single consistent model. Metadata unification can be challenging due to the heterogeneity of metadata schemas and conventions used in practice in author-curated Open Data attribute documentation. We thus identify the need for a standard metadata schema that adequately covers documentation requirements. In addition, we propose methods for unifying the documentation published under that common model so that it can be accessed with standard set-oriented tools and processes.

Throughout this work, we use the term *field* to refer to properties of the unifying metadata model, *attribute* to refer to the vertical components of a data table, and *column* to refer to the vertical components of a documentation table (which may be associated with a metadata field). The main step in the metadata unification problem is the semantic mapping of columns in author-curated documentation tables to the fields of a metadata model. This step is necessary to collect documentation instances from informally structured tables and persist them in a unified metadata format.

Metadata unification can be considered an instance of schema matching, with constraints. *Schema matching* [13, 29, 30, 36] is the problem of finding the confidence in a semantic correspondence between elements of different schemas. These techniques provide a final *mapping* at the end of the schema matching process, a specific assignment of elements from a *source* schema to elements of a *target* schema. Unfortunately, as we show in Section 5, existing schema matching approaches perform poorly for metadata unification.

We thus identify two problems for metadata unification. First, defining a suitable target model T for unifying metadata described in Open Data documentation tables (i.e., a standardized set of fields for describing the content in such tables. We address this problem in Section 3. Second, finding a "good" partial map from columns of a source documentation table S to T, i.e., a partial function that maps each column s_i to the correct metadata field t_j (including the helper field `UNMAPPED`). We address this problem in Section 4.

3 DATA SETS AND MODEL

To create a formal model for metadata unification, we first need to know what metadata look like. In our research, we focus on English documentation in CSV tables and limit the scope of this work to Open Data documentation tables that describe data attributes with no nesting (i.e., each row in a table describes a data attribute), which are by far the most common. We isolated documentation files by conducting a search on filenames for the terms `'documentation'`, `'metadata'`, `'dictionary'`, and `'guide'`. We developed and used a user interface to annotate mappings of columns to our proposed metadata model. We are unaware of a previously annotated data set for the task of unifying tabular metadata published in Open CSV. To develop and evaluate our metadata unification approach, we annotated attribute documentation tables in 218 English language Open CSV files randomly sampled from 6 Open Data repositories in Canada, Australia, and the UK.

[1] https://github.com/cchristodoulaki/gnomon

Table 1: Attribute metadata schema

Name	Description
`dataset`	The name of the file the described attribute exists in.
`name`	[mandatory] The attribute name, as seen in the data table.
`title`	A human-readable version of the attribute name.
`definition`	A text description of the attribute, may include context.
`datatype`	Specifies which type of value the attribute can have.
`scale`	For numeric attributes, the multiplier (e.g., millions).
`format`	A definition of the structure of data (e.g., 'YYYY-MM-DD').
`unit`	The unit of the associated values (e.g., meters, KWh).
`key`	An indication that the attribute is a primary key.
`nullable`	An indication that an attribute value is or isn't mandatory.
`schema`	Name/URL of an existing schema the attribute belongs to.
`examples`	Sampled values from the attribute domain.
`notes`	Text with miscellaneous extra information.

3.1 Attribute Metadata Model

To design a unifying model for Open Data metadata, we assigned 50 files to a development data set OpenData-Dev (see details in Section 3.2). Based on our analysis of the documentation tables in OpenData-Dev, we define the metadata model seen in Table 1 to unify the commonly seen attribute metadata. Our model reuses and extends Frictionless Data fields [20] and extends the JSON annotations for CSV tables introduced by Pytheas [9] inspired by the W3C CSV on the Web Working Group [10]. Annotations can be converted to RDF. We focus on fields that we believe will be useful in downstream applications, such as search and integration.

Not all fields are equally common. We always observe a column corresponding to the `name` attribute metadata field, which is unsurprising as consumers need to know which data attribute is described by the extracted metadata object. The columns corresponding to the metadata field `definition` are seen more frequently than `title`, but we sometimes observe both fields mapped to attributes in the same documentation table. We did not observe isolated columns corresponding to `scale`, `units`, and `format` in our development data set. However, we frequently observed documentation on scale and units embedded in the headers of the data table or the table context (this characteristic was also identified by Yi et al. [46]). We also observed that `format` information frequently occurs as a subset of information found in columns that map to fields `definition` or `examples`. We have added `scale`, `unit`, and `format` fields to our metadata model to support future work similar to Yi et al.

3.2 Annotated Data Sets

As annotating mappings from source tables to the target metadata model can be subjective, our performance metrics account for subjectivity in ground truth (Section 5.1). Moreover, we developed rules to ensure consistency: First, the `name` label must be used once and only once per table. Second, if the annotator judges that multiple columns could be mapped to a single target field, the most descriptive source column is selected for the mapping (e.g., in a table with columns `schema`, `schema rules`, and `schema comments`, only the `schema` column is mapped to the target field `schema`; the remaining two are unmapped). Third,

the column with text values containing the most descriptive attribute semantics is assigned to the metadata field `definition`. If a shorter description exists, it is assigned to the field `title`. Finally, a column describing the domain values or formatting instructions for the attribute is assigned to `examples`.

OpenData-Dev (Dev) contains 50 documentation files and was used in the development of our approach. The data set contains 681 attribute metadata instances and a total of 280 columns. There are 2–12 columns per table (6 on average), 2 unique-value columns per table on average, with 88% text and 12% numeric columns. Ground Truth annotation produced 204 mappings between the columns of the documentation table and the fields of the unified model (the remaining 76 were labeled `UNMAPPED`). The top 3 most common fields are `name`, `definition` and `datatype`. We observed 11 different variations of column names that were assigned to the metadata field `name`, 6 for `datatype` and 4 for `definition` – making them the most difficult to match. The remaining fields showed relatively little diversity in matches in our development sample.

After we finished Gnomon's development and finalized the feature set (see Section 5.2) using Dev, we manually annotated an additional set of 168 CSV documentation files, **OpenData-Test** (Test), using the same procedure as in Dev to create our test data set. As expected, this data set has a distribution similar to that seen in Dev. The data set was annotated only after our techniques were fully developed to avoid bias and was used to report the accuracy of our methods. The tables contain 2034 attribute metadata instances and a total of 913 columns with 2–13 columns per table (5 on average), 2 unique-valued columns per table on average, with 90% text, and 10% numeric columns. In total, 664 columns were assigned to fields of the metadata model. The top 3 fields continue to participate in most of the mappings and show even greater diversity in the names of mapped source columns. The Jaccard coefficient of column names in Dev and Test is 0.49, showing significant overlap.

4 METADATA UNIFICATION WITH GNOMON

With a well-defined metadata model as the target schema, we propose Gnomon, a two-phase schema-matching approach comprised of (1) a **column classification** step to discover <source column, target field> matches and (2) a **schema alignment** step for enforcing mapping constraints.

In detail, the process is as follows. We identify table elements in Open Data files with Pytheas [9] and isolate documentation tables. We process documentation tables by extracting features from the table columns. We use column features to quantify the match of the columns of each documentation table with each field of the metadata model in a *schema matching* step using classification. Finally, we use a bipartite graph matching technique to enforce mapping constraints in an *alignment* step, producing a final mapping.

We note that Gnomon is a combination of well-known techniques. We envision it as a strong baseline for future researchers: it is fast, accurate, easy to understand and implement, and provides a substantial improvement over popular prior work.

4.1 Column Features for Classification

For the classification step in the Gnomon pipeline, shown in Figure 1, we must first extract features for each column, which we source from both column headers and column values. From each source, we consider two feature categories (see Table 2).

Table 2: Final feature set used in Gnomon.

Source	Type	Description	Datatype
header	lexical	TF-IDF vector	float vector
	structure	column index	int
		character count in value	int
		token count in value	int
		value is upper case text	boolean
		value is lower case text	boolean
values	lexical	TF-IDF vector	float vector

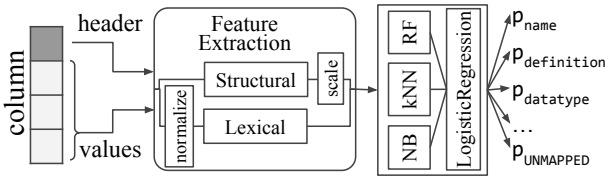

Figure 1: Gnomon classification step. We extract features from the header and values of a column, then use an ensemble of classifiers to assign a probability for each target field.

To produce lexical features we first normalize each column's header or text with a series of pre-processing steps: (1) Remove punctuation and trailing white space; (2) Split camelCase text and convert all text to lowercase; (3) Convert all text words to lemmas using `spaCy`[19]; and (4) Replace consecutive digits with '`__NUMBER__`'. We then generate two vectors of term frequency-inverse document frequency (TF-IDF) from the normalized text in each column header and its concatenated values. TF-IDF [37, 38] is a common term weighting scheme in information retrieval that has also found good use in document classification. We also considered a version of Gnomon with pre-trained GloVE [35] word embeddings. Our evaluation (Section 5) suggest it does not improve over TF-IDF.

To produce *structural features* (i.e., surface properties of text structure that do not capture meaning), we process the header and values of each column using the original text and compute features such as the length of text, the presence of nulls, the uniqueness of values, and the text case. Table 2 lists features that were used in the final configuration, omitting 16 structural features derived from column values that were not found to improve performance. We first compute lexical and structural features separately (normalizing text for lexical features with a pre-processing step). For classifiers using both feature types, we concatenate lexical features with scaled structural features (see Figure 1).

4.2 Classification

We treat the assignment of edge weights w_i as a multinomial *classification* problem: given the target metadata field names as distinct labels $c_1, ..., c_n$ plus a label c_{n+1} =`UNMAPPED`, we assign each source column s a probability $p(s = c_i)$ of being classified with each available label. We train a classifier L on a set of training examples $(x_1, c_{i,1}), ..., (x_m, c_{i,m})$, where x_j is the vector of features extracted from column j of a documentation table, and $c_{i,j}$ is the observed mapping of that column.

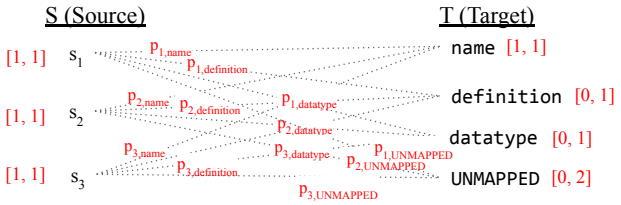

Figure 2: Example mapping three columns $S = s_1, s_2, s_3$ to $|T| - 1$ fields of a simplified metadata model, shown as a weighted bipartite graph with weights per edge and vertex capacities with lower and upper bounds per vertex.

We initially observed that different classifiers were better at predicting specific classes. Inspired by previous work on schema reconciliation [15], we design Gnomon as a *stacking* classifier [44]: an ensemble learning approach that combines multiple base classifiers (learners) to improve predictive performance. Stacking can potentially capture patterns and relationships that may not be obvious to individual base classifiers. Our evaluation (Section 5.2) found that in practice the advantage of stacking is modest.

Gnomon uses Logistic Regression as a meta-estimator with balanced class weights on the predictions made by *base classifiers*. Since our classification problem is a multi-class problem, we use a One-vs-Rest strategy with multiple binary classifiers, one for each class, where each classifier distinguishes between one class and the rest [32]. We use Naive Bayes (NB), k-Nearest Neighbors (kNN), and Random Forest (RF) as base classifiers in Gnomon. We use OpenData-Dev to tune parameters (Section 5.2).

We use the annotated (s_i, t_j) pairs (Section 3.2) to train the base estimators, whose predictions are in-turn used to train the meta-estimator. During inference, each base classifier predicts the probability of the data being labeled with each class respectively.

4.3 Alignment for Constraint Satisfaction

A straightforward way of producing a mapping given discovered matches would be to select the class prediction with the highest probability for each source column, effectively assigning the **best match**. However, this does not take into account mapping constraints. For example, that every schema mapping must include one and only one mapping of a column to the `name` field, or that at most one source column can be mapped to the `datatype` target field.

We observed that for many source schemas the ground truth mapping will be *one-to-one* (1-1) between elements in the source and target schemas, e.g., `VARIABLE` → `name`, however, other source schemas may have more complex mappings of *many-to-one* (N-1), such as when multiple source columns do not correspond to a target field. Additionally, we know that one and only one source column must map to the `name` target field. The best match approach does not enforce such constraints.

Because many documentation columns may remain unmapped to the metadata model, our classifiers support this with the additional label `UNMAPPED`; for clarity, we assume the set T includes this label. The corresponding bipartite graph contains nodes corresponding to columns in the documentation table on the one side and $|T| - 1$ nodes corresponding to metadata fields *plus an additional node* corresponding to the `UNMAPPED` class on the other. If more than one column of the documentation table is not assigned to the model, this will result in multiple edges ending at the `UNMAPPED` node. When nodes on one side of the bipartite graph can be matched with more than one node on the other, this is an instance of the **weighted bipartite B-matching** problem (WBbM) [21]. In a bipartite graph with weighted edges and vertices with capacities of lower and upper bounds b_l and b_u, WBbM seeks alignment that maximizes total weight, allowing any vertex i to be matched at least $b_l(i)$ and at most $b_u(i)$ times (see Figure 2). The edge weights $w : E \rightarrow R^+$ representing the degree of similarity $P(s, t)$ are computed for all possible edges in the bipartite graph $G(S, T, E)$. A valid mapping must not violate cardinality constraints.

Based on our observations and model design, Gnomon enforces several mapping constraints in the second and final step of schema matching, expressed as WBbM cardinality constraints. First, attribute names must be unique in a table's schema. Attribute documentation refers to the corresponding data attributes by their name, making this field mandatory. Exactly one column of an attribute documentation table (source) may be mapped to the metadata `name` field (target). Second, there may exist up to $|S| - 1$ columns in an attribute documentation table that do not get matched to fields in the metadata model (only the name field is a hard requirement in mapping). This is supported by the `UNMAPPED` node and sets the upper bound of the incoming edges to the node. Finally, there may exist up to $|T| - 2$ fields in the metadata model for which no documentation column is mapped. When mapping the columns S of a source documentation table to the $|T| - 1$ fields of the target unifying metadata model (or designating them `UNMAPPED`), one and only one column of S will be mapped to the field `'name'`. The remaining $|S| - 1$ documentation columns are not guaranteed to be assigned to a node in the $|T| - 1$ metadata fields.

In summary, given the output of the classifier step, we define a WBbM optimization problem, solve it, and use the resulting mapping as the alignment solution.

5 EVALUATION

We design an experimental evaluation of methods for attribute metadata unification, and compare Gnomon to popular schema matching approaches. We implement Gnomon classifiers using `scikit-learn` [34]. Our weighted bipartite b-matching (WBbM) alignment solver is based on the implementation offered by Ahmadi et al. [2]. We use the Valentine schema matching suite [23] to measure the performance of popular matchers using default values recommended by the API for method parameters.

We compare Gnomon to several baseline schema matching methods implemented in the Valentine suite[23]. A baseline matching approach with Jaccard-Levenshtein uses the column values to discover mappings between columns of a source table to a target table. The method outputs a ranked list of column pairs, along with their respective similarity score. Other approaches offered in the suite are Distribution-based matching [23, 47], Similarity Flooding [23, 30], Cupid [23, 29], and COMA [13, 14, 23]. Such techniques were designed for schema matching of *data* tables, which tend to have more distinct semantics between source columns compared to the application of *metadata unification* observed in Open Data documentation. We also consider GloVe-kNN , a baseline matcher which feeds pretrained GloVe word embeddings [35] of headers and values into a kNN classifier, and Gnomon-GloVe, a variant of Gnomon that replaces the TF-IDF vectors (Section 4.1) with pretrained GloVe word embeddings.

With the exception of COMA, each approach can output multiple potential matches from source columns to the target metadata

model, each match with a corresponding match similarity. For COMA, Valentine produces only the top match for each source column. For columns for which no matches are returned (e.g., in the case where no matches have a similarity above the similarity threshold), we assign the mapping `UNMAPPED`.

5.1 Metrics

To evaluate the quality of the first step of metadata unification (i.e., column classification), we use the F-measure [12, 16, 17]. Let P be *precision*, the fraction of all classifier predictions that turned out to be correct, and let R be *recall*, the fraction of all mappings in the dataset correctly labeled by the classifier. The F-measure $F = 2\frac{PR}{P+R}$ measures how well a model predicts the target field, balancing precision and recall. We use the macro F-measure, which is the unweighted mean of per-class F-measures, to aggregate F-measure over all classes without biasing towards the more common ones.

F-measure, however, is ill-suited for describing the accuracy of schema mapping: it focuses on each column in isolation, without considering other columns of the same source table nor alignment constraints. We thus propose two novel metrics for metadata unification meant to describe the accuracy of an entire mapping.

First, a_{top3}, which measures the mapping accuracy of the top three target fields: $F_{\text{top3}} = \{\texttt{name}, \texttt{definition}, \texttt{datatype}\}$. We focus on these fields because they are the most commonly used fields in the data set and because the `name` field is critical for correctly linking the metadata to relevant data attribute. Formally, let $R = \{S_1, S_2, \dots\}$ be a set of source schemas. We consider a mapping of schema $S \in R$ to be correct if it correctly finds all top 3 columns (meaning the mapping matches the ground truth for these fields), and no other column is mapped to those target fields. Note this definition ignores fields not in F_{top3}, but that even a single mistake in mapping to F_{top3} means that the mapping of the schema S is considered incorrect. On the rare cases where the ground truth for S indicates there is no mapping for one of the fields in F_{top3}, then a correct mapping should not have any column mapped to it. We can now define the metric: $a_{\text{top3}} = \frac{N_{\text{top3}}}{|R|}$, where N_{top3} is the number of schemas in R correctly matched with $F_{\texttt{top3}}$.

We also introduce a softer metric, a_{soft}, that considers the percentage of correct column mappings in a proposed mapping on source schema S across all target fields, while still reflecting the importance of the accurate mapping to the top 3 fields. Unlike a_{top3} where a single error renders the entire mapping incorrect, a_{soft} allows some error when mapping non-top fields. A less strict metric is desirable since ground truth mappings can sometimes be subjective, especially when going beyond the top 3 fields (Section 3.2). Formally, let $M(S)$ be the set of proposed column mappings to *all* fields of the metadata model in a mapping on schema S; let $M_C(S)$ be the set of *correct* column mappings in $M(S)$ if all mappings to F_{top3} are correct, otherwise $M_C(S) = \emptyset$. We define $a_{\text{soft}} = \frac{1}{|R|}\sum_{S\in R}\frac{|M_C(S)|}{|M(S)|}$. Note a_{soft} is continuous in the range $[0, 1]$, and that $a_{\text{soft}} \leq a_{\text{top3}}$.

5.2 Configuring Gnomon

We tuned Gnomon on Dev using 5-fold group cross-validation. To account for class imbalance, we performed stratified group sampling, partitioning annotated documentation tables into folds such that each fold contains roughly the same distribution of target metadata fields. Each table acts as a logical unit, and columns of the same table may not be split into different folds. We used the macro F-measure to select the best configuration. We omit detailed exploration of these results due to space constraints.

When configuring base classifiers we considered different combinations of three cases of feature source (header, values, or both), and three cases of feature type (structure, lexical, or both). For k-Nearest Neighbors we considered $k \in \{3, 5, 7, 9, 11, 13\}$. For Random Forest, we considered $n_{\text{trees}} \in \{10, 50, 100, 150, 200\}$.

Naive Bayes (NB) : We account for class imbalance in the NB classifier by computing class weights, giving more importance to minority classes during training to balance their impact on the model's learning process. The best NB classifier used lexical features sourced from both column headers and values, achieving an average macro F-measure of 84.85% (6.23 std).

k-Nearest Neighbors (kNN): To account for class imbalance, we weight the k neighbors by the inverse of their distance to the query point, meaning a close neighbor from a rare class can still have more influence than distant neighbors from a common class. The highest performing configuration had $k = 3$ neighbors and with lexical features sourced from both column headers and column values, achieved an average F-measure of 94.19% (5.8 std).

Random Forest (RF): The best performance is achieved using both lexical and structural features from column headers using $n_{\text{trees}} = 100$ with a macro F-measure of 95.56% (6.22 std).

Stacking (SC): The stacking classifier performs marginally better than the RF classifier alone, with an average macro F-measure of 95.72% (5.95 std). In the few examples for which both stacking and RF are incorrect (i.e., the class with the highest probability is the wrong class), SC was less confident than RF, suggesting that stacking may do a better job of producing match weights.

When evaluated on Test, classifiers maintained their relative performance, with a macro F-measure of 78% for NB, 79% for kNN, 82% for RF, and 82% for Stacking of all three. NB struggled particularly with correctly classifying `name`, `title`, and `notes` fields (recording class F-measures of 87%, 26% and 72%). kNN improved the class F-measure significantly for class `name` (93%), however maintained poor performance for `notes` and `title` classes (80% and 26% respectively). Surprisingly, RF recorded a drop in F-measure for `dataset`, significantly improving for `name` (98%) and `title` (42%).

5.3 Schema Mapping Accuracy

We used Dev to train Gnomon, and Test to compare metadata unification performance to baseline schema matchers using the a_{top3} and a_{soft} mapping accuracy metrics. Since compared approaches are not designed for metadata matching with domain constraints, we augment them with our WBbM alignment step when possible. We include augmented versions of the baseline approaches (JL^a, DB^a, SF^a, Cpd^a, Coma^a, GVkNN^a). With the exception of COMA, each approach can output multiple potential matches from source columns to the target metadata model. We apply Gnomon's WBbM postprocessing alignment step to these matches. For COMA, Valentine produces only the top match for each source column, so Coma^a is the same as COMA. For completeness, we also evaluate SC: a classifier-only version of Gnomon without WBbM alignment.

Figure 3 shows that Gnomon (Gnm) outperforms compared baselines. Moreover, by comparing top3 WBbM performance with the best-match performance, we observe that WBbM alignment improves mapping performance for all approaches; top3

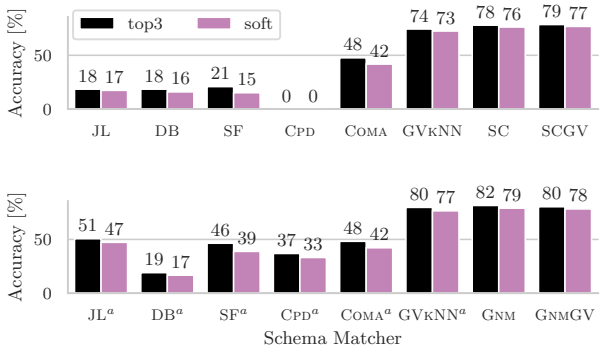

Figure 3: Accuracy of schema mapping methods trained on Dev and applied on Test (top row showing best-match, bottom row including WbBM post-processing alignment step).

accuracy for SC is 78% compared to Gnm's 82%. Analysis of Gnm mapping errors indicates that `definition` columns were misclassified as `title` in 16% of the cases, explained by the significantly more variations of source column names mapped to `definition` in Test compared to the much smaller Dev. Despite the novel data points in Test, Gnm maintains a high mapping accuracy (likely due to the use of TF-IDF vectors).

For this problem, JaccardLevenshtein (JL) performs poorly when evaluated columns have non-string data types, which is rare in our data. More importantly, this method also fails if the columns require understanding of semantic similarity beyond simple string edits and value overlaps, which is often the case in documentation data, as much of it is free-form text.

Distribution-based approaches like DistributionBased (DB) and COMA may work well for matching columns with numeric or limited categorical values (e.g., `datatype`, `scale`, and `unit`), but do not work well on columns with long, indistinct text values and therefore are likely unsuitable for this task. SimilarityFlooding (SF) uses the Levenshtein string distance to indicate columns with similar headers [23, 30], which does not work well in simple table-to-table matches when headers are quite dissimilar and the relational structure is flat, as seen in this problem setting.

Cupid (Cpd) performs poorly in this setting, as it is a schema-based approach initially developed for XML, and relies heavily on the tree structure seen in XML. Although the linguistic matching used in Cupid should intuitively be helpful, the semantic differences between some fields in the metadata model are nuanced and are difficult to differentiate with WordNet. Given the simplicity of the source and target relational tables, Cupid's structural matching (i.e., calculating similarity of columns based on context) is not effective.

While GloVe-kNN (GVkNN) is more accurate than classic schema matching approaches, without including the WBbM alignment step (GVkNN^a) it performs poorly compared to Gnm (with a *soft* accuracy of 73% VS Gnm's 79%). GVkNN^a benefits from alignment postprocessing with WBbM (*soft*=77%), with performance rising close to Gnm and GnmGV. We find that replacing TF-IDF with pre-trained GloVe word embeddings in Gnomon-GloVe (GnmGV) does not improve performance, and in fact slightly harms it. We attribute this to the fact that some metadata fields and values have subtle semantic differences that embeddings struggle to differentiate, and column headers often contain terms or abbreviations not found in natural English language. Fine-tuning a large language model may improve accuracy but requires overcoming the issues above and acquiring significantly larger amounts of diverse training data.

6 DISCUSSION

In this work, we identify *metadata unification*, a distinct variant of schema matching, as an important step for effective use of Open Data. We show that popular baseline schema matching approaches perform poorly on this task, and provide a unified model for metadata unification and a strong baseline approach, Gnomon, for tackling it. Our goal in this work was to highlight the problem and provide tools for future research. Such research could focus on multi-language documentation, methods to incorporate metadata embedded in data tables, fine-tuning large-language models for metadata unification, and normalizing metadata values.

ACKNOWLEDGMENTS

This work was supported by an NSERC Discovery grant. We thank Mariano Consens, Periklis Andritsos, and Yannis Ioannidis for insightful comments and constructive feedback on early versions of this work.

REFERENCES

[1] Vibhav Agarwal, Akansha Bhardwaj, Paolo Rosso, and Philippe Cudré-Mauroux. 2021. ConvTab: A Context-Preserving, Convolutional Model for Ad-Hoc Table Retrieval. In *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, 5043–5052.

[2] Sina Ahmadi, Mihael Arcan, and John McCrae. 2019. Lexical sense alignment using weighted bipartite b-matching. In *Proceedings of the LDK 2019 Workshops*. 2nd Conference on Language, Data and Knowledge (LDK 2019).

[3] Z. Bellahsène, A. Bonifati, and E. Rahm. 2011. Schema matching and mapping. (2011). https://doi.org/10.1007/978-3-642-16518-4

[4] Alex Bogatu, Alvaro AA Fernandes, Norman W Paton, and Nikolaos Konstantinou. 2020. Dataset discovery in data lakes. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 709–720.

[5] Michael J. Cafarella, Alon Halevy, and Nodira Khoussainova. 2009. Data Integration for the Relational Web. *PVLDB* 2, 1 (2009), 1090–1101. https://doi.org/10.14778/1687627.1687750

[6] Zhiyu Chen, Haiyan Jia, Jeff Heflin, and Brian D Davison. 2020. Leveraging schema labels to enhance dataset search. In *European Conference on Information Retrieval*. Springer, 267–280.

[7] Zhiyu Chen, Mohamed Trabelsi, Jeff Heflin, Yinan Xu, and Brian D Davison. 2020. Table search using a deep contextualized language model. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 589–598.

[8] Zhiyu Chen, Mohamed Trabelsi, Jeff Heflin, Dawei Yin, and Brian D Davison. 2021. MGNETS: Multi-Graph Neural Networks for Table Search. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2945–2949.

[9] Christina Christodoulakis, Eric B Munson, Moshe Gabel, Angela Demke Brown, and Renée J Miller. 2020. Pytheas: pattern-based table discovery in CSV files. *VLDB* 13, 12 (2020).

[10] CSV on the Web Working Group. 2015. Model for Tabular Data and Metadata on the Web. https://www.w3.org/TR/2015/REC-tabular-data-model-20151217/. Accessed: 2024-08-30.

[11] Anish Das Sarma, Lujun Fang, Nitin Gupta, Alon Halevy, Hongrae Lee, Fei Wu, Reynold Xin, and Cong Yu. 2012. Finding related tables. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. 817–828.

[12] Hong-Hai Do, Sergey Melnik, and Erhard Rahm. 2003. Comparison of schema matching evaluations. In *Web, Web-Services, and Database Systems: NODe 2002 Web-and Database-Related Workshops Erfurt, Germany, October 7–10, 2002 Revised Papers 4*. Springer, 221–237.

[13] Hong-Hai Do and Erhard Rahm. 2002. COMA—a system for flexible combination of schema matching approaches. In *VLDB'02: Proceedings of the 28th International Conference on Very Large Databases*. Elsevier, 610–621.

[14] Hong-Hai Do and Erhard Rahm. 2007. Matching large schemas: Approaches and evaluation. *Information Systems* 32, 6 (2007), 857–885.

[15] AnHai Doan, Pedro Domingos, and Alon Y Halevy. 2001. Reconciling schemas of disparate data sources: A machine-learning approach. In *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*. 509–520.

[16] Fabien Duchateau, Zohra Bellahsene, and Ela Hunt. 2007. XBenchMatch: a benchmark for XML schema matching tools. In *The VLDB Journal*, Vol. 1. Springer Verlag, 1318–1321.

[17] Jérôme Euzenat, Pavel Shvaiko, et al. 2007. *Ontology matching*. Vol. 18. Springer.

[18] Maryam Habibi, Johannes Starlinger, and Ulf Leser. 2020. Tabsim: A siamese neural network for accurate estimation of table similarity. In *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 930–937.

[19] Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. (2017). To appear.

[20] Open Knowledge International. [n.d.]. Frictionless Data Specifications and Software: Data Package. https://frictionlessdata.io/specs/data-package/. Accessed: 2024-08-30.

[21] Tony Jebara and Vlad Shchogolev. 2006. B-matching for spectral clustering. In *Machine Learning: ECML 2006: 17th European Conference on Machine Learning Berlin, Germany, September 18-22, 2006 Proceedings 17*. Springer, 679–686.

[22] Jim Tasevski. 2018. Metadata Standards for Open Data. https://salsa.digital/insights/metadata-standards-for-open-data. Accessed: 2023-11-19.

[23] Christos Koutras, George Siachamis, Andra Ionescu, Kyriakos Psarakis, Jerry Brons, Marios Fragkoulis, Christoph Lofi, Angela Bonifati, and Asterios Katsifodimos. 2021. Valentine: Evaluating Matching Techniques for Dataset Discovery. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 468–479.

[24] Oliver Lehmberg and Christian Bizer. 2017. Stitching Web Tables for Improving Matching Quality. *Proc. VLDB Endow.* 10, 11 (Aug. 2017), 1502–1513. https://doi.org/10.14778/3137628.3137657

[25] Girija Limaye, Sunita Sarawagi, and Soumen Chakrabarti. 2010. Annotating and Searching Web Tables Using Entities, Types and Relationships. *Proc. VLDB Endow.* 3, 1–2 (Sept. 2010), 1338–1347. https://doi.org/10.14778/1920841.1921005

[26] Xiao Ling, Alon Y Halevy, Fei Wu, and Cong Yu. 2013. Synthesizing union tables from the web. In *Twenty-Third International Joint Conference on Artificial Intelligence*.

[27] Ying Liu. 2009. *Tableseer: automatic table extraction, search, and understanding*. Ph.D. Dissertation. The Pennsylvania State University.

[28] Ying Liu, Kun Bai, Prasenjit Mitra, C Lee Giles, et al. 2007. Tablerank: A ranking algorithm for table search and retrieval. In *Proceedings of the National Conference on Artificial Intelligence*, Vol. 22. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 317.

[29] Jayant Madhavan, Philip A. Bernstein, and Erhard Rahm. 2001. Generic Schema Matching with Cupid. In *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB '01)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 49–58.

[30] Sergey Melnik, Hector Garcia-Molina, and Erhard Rahm. 2002. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *Proceedings 18th international conference on data engineering*. IEEE, 117–128.

[31] Michael Vandi. 2021. Understanding the Importance of Data Documentation. Baltimore Neighborhood Indicators Alliance - Jacob France Institute. https://bniajfi.org/2021/01/14/understanding-the-importance-of-data-documentation/ Accessed: 2023-11-19.

[32] Kevin P Murphy. 2012. *Machine learning: a probabilistic perspective*. MIT press.

[33] Fatemeh Nargesian, Erkang Zhu, Ken Q. Pu, and Renée J. Miller. 2018. Table Union Search on Open Data. *PVLDB* 11, 7 (2018), 813–825. https://doi.org/10.14778/3192965.3192973

[34] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[35] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543. http://www.aclweb.org/anthology/D14-1162

[36] Erhard Rahm and Philip A Bernstein. 2001. A survey of approaches to automatic schema matching. *the VLDB Journal* 10 (2001), 334–350.

[37] Anand Rajaraman and Jeffrey David Ullman. 2011. Mining of massive datasets: Data mining (ch01). *Min. Massive Datasets* 18 (2011), 114–142.

[38] Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information Processing & Management* 24, 5 (1988), 513–523. https://doi.org/10.1016/0306-4573(88)90021-0

[39] Schema.org. 2023. Schema.org: Structured Data on the Web. https://schema.org/ Accessed: 2024-08-30.

[40] Nigel Shadbolt and Tim Berners-Lee. 2008. Web science emerges. *Scientific American* 299, 4 (2008), 76–81.

[41] Mohamed Trabelsi, Zhiyu Chen, Brian D Davison, and Jeff Heflin. 2020. Relational graph embeddings for table retrieval. In *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 3005–3014.

[42] Petros Venetis, Alon Halevy, Jayant Madhavan, Marius Paşca, Warren Shen, Fei Wu, Gengxin Miao, and Chung Wu. 2011. Recovering Semantics of Tables on the Web. *Proc. VLDB Endow.* 4, 9 (June 2011), 528–538. https://doi.org/10.14778/2002938.2002939

[43] Joachim Wackerow. 2008. The Data Documentation Initiative (DDI). In *Proceedings of the 2008 International Conference on Dublin Core and Metadata Applications (DCMI '08)*. Dublin Core Metadata Initiative, 206.

[44] David H Wolpert. 1992. Stacked generalization. *Neural networks* 5, 2 (1992), 241–259.

[45] World Wide Web Consortium (W3C). 2024. Data Catalog Vocabulary (DCAT) - Version 3. https://www.w3.org/TR/vocab-dcat/. Accessed: 2024-08-30.

[46] Yang Yi, Zhiyu Chen, Jeff Heflin, and Brian D Davison. 2018. Recognizing quantity names for tabular data. In *ProfS/KG4IR/Data: Search@ SIGIR*.

[47] Meihui Zhang, Marios Hadjieleftheriou, Beng Chin Ooi, Cecilia M. Procopiuc, and Divesh Srivastava. 2011. Automatic Discovery of Attributes in Relational Databases. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data (SIGMOD '11)*. Association for Computing Machinery, New York, NY, USA, 109–120. https://doi.org/10.1145/1989323.1989336

[48] Shuo Zhang and Krisztian Balog. 2018. Ad hoc table retrieval using semantic similarity. In *Proceedings of the 2018 world wide web conference*. 1553–1562.

[49] Erkang Zhu, Fatemeh Nargesian, Ken Q. Pu, and Renée J. Miller. 2016. LSH Ensemble: Internet-Scale Domain Search. *Proc. VLDB Endow.* 9, 12 (aug 2016), 1185–1196. https://doi.org/10.14778/2994509.2994534