Semantic labeling: A domain-independent approach

Minh Pham, Suresh Alse, Craig A. Knoblock, and Pedro Szekely

University of Southern California,
{minhpham,alse}@usc.edu
{knoblock,pszekely}@isi.edu

Abstract. Semantic labeling is the process of mapping attributes in data sources to classes in an ontology and is a necessary step in heterogeneous data integration. Variations in data formats, attribute names and even ranges of values of data make this a very challenging task. In this paper, we present a novel domain-independent approach to automatic semantic labeling that uses machine learning techniques. Previous approaches use machine learning to learn a model that extracts features related to the data of a domain, which requires the model to be re-trained for every new domain. Our solution uses similarity metrics as features to compare against labeled domain data and learns a matching function to infer the correct semantic labels for data. Since our approach depends on the learned similarity metrics but not the data itself, it is domain-independent and only needs to be trained once to work effectively across multiple domains. In our evaluation, our approach achieves higher accuracy than other approaches, even when the learned models are trained on domains other than the test domain.

1 Introduction

Mapping attributes in data sources to a domain ontology is a necessary step in integrating different sources and mapping them to a domain ontology. The problem, which we call semantic labeling, requires annotating source attributes with classes and properties of ontologies. There has been a number of studies conducted to automate the process since labeling attributes manually is laborious and requires a sufficient amount of domain knowledge. However, automatic semantic labeling is difficult to perform accurately for several reasons. First, people have different ways to represent data of same labels. Table 1 shows different formats that *PlayerPosition* can be found in soccer data. On the other hand, data from different labels can be very similar. For example, data of *NumberOfGoalsScores* and *NumberOfGamesPlayed* in soccer data are very similar because both of them are in numeric format with values ranged mainly from 0 to 50. Therefore, a good semantic labeling approach needs to deal with two different issues: to distinguish similar labels and to recognize the same labels from different data, both of which generally make the problem very hard.

To address these issues, we present a domain-independent machine learning approach for semantic labeling. Our contribution is a novel way of using machine

Table 1. Different representations of *PlayerPosition*

Code	Abbreviation	Full form
1	GK	Goalkeeper
2	DF	Defender
3	MF	Midfieder
4	FW	Forward

learning to solve semantic labeling as a combination of many binary classification sub-problems. Our machine learning model uses similarity metrics as features and learns a matching function to determine whether attributes have the same labels to infer the correct semantic labels. Because the matching function is not related to specific labels, our model is independent from labels and thus independent from the domain ontologies.

We evaluate our approach on many datasets from different domains. When the machine learning models are trained on another domain, the system achieves an average mean reciprocal rank (MRR) [3] over 80% on 4 datasets. The results are even better if models are trained on the same domain. We also run experiments on the T2D Gold Standard data and achieve a higher F1-measure compared to the property-matching approach in the T2K system [12].

2 Motivating Example

In this section, we provide an example to explain the problem of mapping source attributes to semantic types in a domain ontology. Suppose that we want to map attributes in a data source named *WC2014* (Table 2), which contains information about players of national teams in World Cup 2014, to the DBpedia ontology. First, we define our target label, which we call *semantic type*, as a pair of values consisting of a domain class and one of its properties <*class, property*>. For example, in Table 2, the correct semantic types of column *player*, *height* and *position* are <*dbo:SoccerPlayer, dbo:birthName*>, <*dbo:SoccerPlayer, dbo:height*> and <*dbo:SoccerPlayer, dbo:draftPosition*>. Semantic labeling systems attempt to automatically identify these mappings. However, this cannot be done without knowing about these semantic types in a domain.

Table 2. Sample data from World Cup 2014 players (WC2014)

player	height	position
Alan PULIDO	176	Forward
Robin VAN PERSIE	186	Forward
Miiko ALBORNOZ	180	Defender
Marouane FELLAINI	194	Midfielder

Therefore, the semantic labeling problem refers to a situation where we have already mapped one or more sources to a common ontology and we want to label new sources using the same ontology. For example, we have the data source *EPL* containing information about all England Premier League players and it is already labeled with DBpedia semantic types (Table 3). Since we have information about the DBpedia ontology from the *EPL* source, we can label source attributes of *WC2014* based on this information. There are

Table 3. Sample data from England Premier League (EPL)

first name *<SoccerPlayer, birthName>*	**position** *<SoccerPlayer, draftPosition>*	**height** *< SoccerPlayer, height>*
Hazard, Eden	Midfielder	172
Cahill, Gary	Defender	191
Felliani, Marouane	Midfielder	194
Oezil, Mesut	Midfielder	180

different ways to leverage domain data from labeled sources for semantic labeling. Previous work uses labeled sources such as *EPL* as training data to learn the characteristic of data in different attributes. Table 4 shows some feature values extracted from <*dbo:SoccerPlayer, dbo:birthName*> data. In our approach, we use *EPL* as our base data and compare attributes in *WC2014* with attributes in *EPL*. If these two attributes are similar such as column *first name* in EPL and column *player* in WC2014, we conclude that they have the same semantic types. Because we know that the semantic type of *first name* is <*dbo:SoccerPlayer, dbo:birthName*>, we infer that the semantic type of *player* is also <*dbo:SoccerPlayer, dbo:birthName*>.

Table 4. Some feature values extracted from <*SoccerPlayer, birthName*>

Feature	Value
all capitalized token	1
starts with char C	0.25
num len	0

The main difference between our approach and previous work is when faced with unseen semantic types. For example, consider the case where we have another labeled source named *BGL* containing information about players in Germany Bundesliga League (Table 5). *BGL* contains a column *salary* which is labeled as < *dbo:Person, dbo:salary* > - an unseen semantic type. In previous approaches, learned models need to be retrained to capture the data characteristic of < *dbo:Person, dbo:salary* > and this process needs to be repeated for every unseen semantic type. There are a huge number of data sources and semantic types, which makes the possibility of facing new semantic types very high and it is time-consuming to retrain the learning models each time. For our approach, we just need to store data with the new semantic types for later comparison with unlabeled attributes.

Table 5. Sample data from Germany Bundesliga League (GBL)

name < *SoccerPlayer, birthName*>	**salary** < *SoccerPlayer, salary*>
Neuer; Manuel	150,000
Boateng; Jerome	90,000
Dante	100,000

3 Approach

In this section, we explain our approach to determine similarities between unlabeled and labeled attributes and use machine learning techniques to find the correct semantic type. Section 3.1 describes various similarity metrics that we use as our features and how we compute them. Section 3.2 describes details of how we use machine learning for semantic labeling problem.

3.1 Similarity metrics

In our approach, we exploit different similarity metrics that measure how attributes are similar to others. In this section, we describe these similarity metrics and explain how they can help in semantic labeling.

Attribute Name Similarity In relational databases, web tables or spreadsheets, tabular structures usually have titles for each column. We consider these titles as attribute names and use them to compare similarities between two attributes.

Definition 1. *Given two attributes named a and b, we have A and B as sets of character tri-grams extracted from a and b. The* ***attribute name similarity*** *is calculated using Jaccard similarity [8] as follows:*

$$S(a,b) = \frac{|A \cap B|}{|A \cup B|} \tag{1}$$

In data sources, people usually name attributes based on the meaning of the data so that similarity in attribute names provides a good indication of the similarity in semantic types. However, as attribute names usually correspond only to ontology properties, using attribute names as the only metric can lead to false positives in labeling. For example, a column named *name* can refer to < *dbo:Person, dbo:birthName* > or < *dbo:SportsTeam, dbp:clubName* > depending on the sources. Collecting data sources from the web can also result in missing or noisy attribute names, which provide no information about the attributes.

Value Similarity Value similarity is the most common similarity metric, which is widely used in different matching systems. In semantic labeling, attribute values play an important role in identifying attributes that have the same semantic types because they usually contain similar values. In our approach, we compute three different value similarity metrics: Jaccard similarity and TF-IDF cosine similarity for textual data, as well as a modified version of Jaccard similarity for numeric values.

Definition 2. *Given two attributes named a and b with v_a and v_b as the corresponding sets of values, the* ***textual Jaccard similarity*** *[8] is computed as follows:*

$$S(a,b) = \frac{|v_a \cap v_b|}{|v_a \cup v_b|} \tag{2}$$

Definition 3. *Given set of attributes $\{a_1, a_2, ..., a_n\}$ with a corresponding sets of values $\{v_1, v_2, ..., v_n\}$, the* ***TF-IDF cosine similarity*** *[8] is computed using the following steps:*

1. *We concatenate the values in $\{v_1, v_2, ..., v_n\}$ by attribute to generate a set of documents: $\{D_1, D_2, ..., D_n\}$*
2. *For a document D_i, we calculate the corresponding TF-IDF vector W_i*
3. *We compute TF-IDF cosine similarity between two attributes a and b:*

$$S(a,b) = \frac{W_a \cdot W_b}{|W_a| \times |W_b|} \tag{3}$$

For numeric attributes, set-based similarity metrics such as Jaccard and cosine similarity do not work effectively because numeric data have continuous ranges of values. Therefore, we customize Jaccard similarity to work with range of values instead of sets of values.

Definition 4. *: Given two attributes named a and b with v_a and v_b as the corresponding sets of values, the* ***numeric Jaccard similarity*** *is computed as follows:*

$$S(a,b) = \frac{min(max(v_a), max(v_b)) - max(min(v_a), min(v_b))}{max(max(v_a), max(v_b)) - min(min(v_a), min(v_b))} \tag{4}$$

For example, the numeric Jaccard similarity s of two attributes with values in range [1912,1980] and [1940,2000] is computed as follows:

$$s = \frac{1980 - 1940}{2000 - 1912} = 0.45. \tag{5}$$

To reduce sensitivity to outliers, we only use the subsets containing values from first quartile to third quartile instead of the whole set of values in attributes.

Distribution Similarity For numeric data, there are semantic types that we are unable to distinguish by using value similarity because they have the same range of values. However, since they have different underlying meanings, their distribution of values may be different. For example, consider the example about *NumberOfGoalsScored* and *NumberOfGamesPlayed* in Section 1. Although they have the same range of values, *NumberOfGoalsScored* has skewed distribution because the high values are mostly distributed to *Forwards* and *Midfielders* while *NumberOfGamesPlayed* is more likely to follow a near-uniform distribution.

Therefore, we analyze the distribution of numeric values contained in the attributes using statistical hypothesis testing as one of the similarity metrics. For statistical hypothesis testing used in our approach, the null hypothesis is that the two sets of values are drawn from a same population (distribution), which may indicate that they come from a same semantic type. We use Kolmogorov-Smirnov test (KS test)[6] as our statistical hypothesis test based on evaluation of different statistical tests in Ramnandan et al's research [11].

Histogram Similarity For textual data, normal statistical hypothesis testing cannot be applied because there is no order in textual values. Moreover, we cannot use traditional correlation methods such as mutual information or KL-divergence since we are comparing attributes that do not appear in the same source. Therefore, we calculate value histograms in textual attributes and compare their histograms instead. The statistical hypothesis tests used for the histogram case is the Mann-Whitney test (MW test) [6]. The reason we use MW test instead of KS test is that histograms are not ordinal and using methods that compare two empirical value distributions such as KS test is not suitable. Mann-Whitney test computes distribution distances based on medians and, thus, is more appropriate to use for histograms.

When comparing a textual attribute with a numeric attribute, we also transform numeric data into histogram form and use MW test to compute histogram similarity. For the example of *PlayerPosition* in Table 1, even though they have different representations, they have similar histogram forms because every position usually have similar frequencies over the different sources of data. For instance, because every soccer team usually has 1 goalkeeper, 4 defenders, 4 midfielders and 2 forwards, the histogram frequencies are likely to be $[\frac{1}{11}, \frac{4}{11}, \frac{4}{11}, \frac{2}{11}]$.

Mixtures of numeric and textual data As we have described above, there are similarity measures that can only applied for the textual part of attribute values while some others only work on numeric parts (Table 6). Because textual similarity metrics are more important when comparing attributes with mostly text and numeric similarities are more important for attributes with numeric data, we need to adjust the values of these similarity measures based on the fraction of textual and numeric values contained in attributes.

Table 6. Similarity feature vector

Feature name	**Explanantion**	**Applied data types**
ATT NAME	Jaccard similarity for attribute names	All
TEXT JACCARD	Jaccard similarity for textual data	Textual
TF-IDF COSINE	TF-IDF cosine similarity for textual data	Textual
NUM JACCARD	Modified Jaccard similarity for numeric data	Numeric
NUM KS	KS statistical test for numeric data	Numeric
MW HISTOGRAM	MW test for histogram	All

Given r_1 and r_2 are fractions of textual data in the pair of attributes, the adjusted value of textual similarity value is computed as follows:

$$v_{adjusted} = \frac{(r_1 * r_2)}{(r_1 + r_2)} * v_{original} \tag{6}$$

On the other hand, the adjusted value of numeric similarity value is computed as follows:

$$v_{adjusted} = \frac{[(1 - r_1) * (1 - r_2)]}{[(1 - r_1) + (1 - r_2)]} * v_{original} \tag{7}$$

The adjusted value is the product of the harmonic mean over r_1, r_2 and the original value. The reason for using harmonic mean follows the intuition that the corresponding similarity values are more reliable when two attributes have similar fractions of textual data or numeric data and vice versa.

3.2 Semantic Labeling

The overall framework is illustrated in Figure 1. The input of our system is an unlabeled attribute and a set of labeled attributes as domain data and the output is a set of top-k semantic types corresponding to the unlabeled attribute.

Overall Approach Given a set of attributes $\{a_1, a_2, ... a_n\}$, we compute M-dimensional feature vectors f_{ij} $(i \neq j)$. Each dimension k corresponds to a similarity metric, so $f[k]$ represents how similar attributes a_i and a_j are under metric k.

During training we label each f_{ij} as True/False, where True means that attributes a_i and a_j have the same semantic type and vice versa. To set up a new domain, we store a set of labeled attributes $\{a_1, a_2, ... a_n\}$ as domain data and use them to compare against new attributes to infer the semantic types.

Given a new attribute a_0, the algorithm computes f_{0j} for all j $(j \neq 0)$, and uses the learned classifier to label each f_{0j} as True/False. If the label of f_{0j} is yes, the algorithm says that the semantic type of a_0 is the semantic type that was recorded for a_j. From that, we can conclude the semantic type of a_0.

Previous approaches, tried to predict the semantic label of a_0 based on characteristic of recorded a_i. In contrast, our approach learns a classifier over similarity vectors. It is domain-independent because classification does not depend

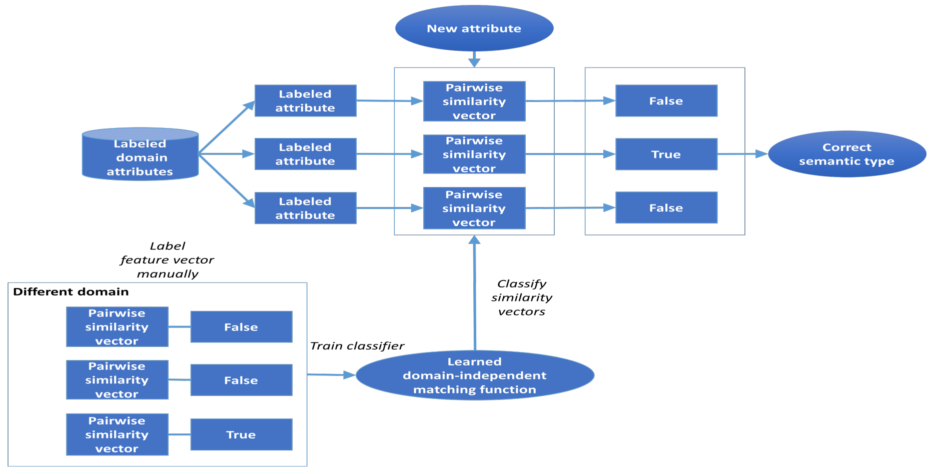
New attribute
Labeled domain attributes
Labeled attribute
Labeled attribute
Labeled attribute
Pairwise similarity vector
Pairwise similarity vector
Pairwise similarity vector
False
True
False
Correct semantic type
Label feature vector manually
Classify similarity vectors
Different domain
Pairwise similarity vector
False
Pairwise similarity vector
False
Pairwise similarity vector
True
Train classifier
Learned domain-independent matching function

Fig. 1. Overall framework of our semantic labeling system

on the values in attributes, but rather on the similarity scores of multiple metrics between the attributes.

Since there are no constraint on the number of True labels for each attribute, we develop a ranking method and only take the top-k results of semantic types. The ranking algorithm uses the predicted probabilities of the True class in classification as the confidence scores and ranks the candidate semantic types based on that.

Classifiers for Semantic Labeling To choose the best classifier for semantic labeling, we ran experiments on various of classifiers and compare the results. Because we use class probabilities of classifiers as confidence scores, classifiers need to have class probabilities calculated from the feature vector in order to be applicable. Therefore, we only consider Logistic Regression and Random Forests [2]. Details of the experiments are described in Section 4.2. According to the results from Table 8-10, Logistic Regression achieves the best performance and thus is the selected classifier in our system.

4 Evaluation

In our experiments, we use four different datasets: city [11], weather [1], museum [14], and soccer. The soccer data set was created to provide a wide variety of semantic types and consists of numerous real-world data sets about soccer. The purpose of using many datasets from different domains is to evaluate our classifiers when applying a single learned classifier to multiple domains. Table 7 shows the overall information about these data sets. The datasets and code used in our experiments have been published online[1].

Table 7. Data sets from different domains in experiments

Data set	No. sources	No. semantic types	No. attributes
museum	29	20	217
city	10	52	520
soccer	12	14	97
weather	4	11	44

4.1 Experimental Setup

In this section, we evaluate the performance of our system, which is called DSL (Domain-independent Semantic Labeler). . The evaluation metric that we measure is the mean reciprocal rank (MRR) [3]. The details of the experimental setup is as follows:

1. Choose a labeling dataset A.
2. Suppose A consists of n sources $\{s_1, s_2, ..., s_n\}$, choose the number of labeled sources m in the dataset ($m < n$).
3. For every source s_i in A, perform semantic labeling using m labeled sources from s_{i+1} to s_{m+i+1}.

For example, the soccer dataset has 12 sources. If we have one labeled source, we label s_1 with labeled data from s_2, label s_2 with labeled data from s_3 and so on. Likewise, if we have five labeled sources, we label s_1 with labeled data in set of sources s_2, s_3, ..., s_6 and continue through the entire data set.

For classifier training data, we follow the same process as above but we manually label the computed feature vectors generated instead of running semantic labeling. To assure that classifier training data is disjoint from labeling data, if labeling dataset and training dataset are the same, we choose distinct labeled sources for each process.

4.2 Classifier Analysis

In this experiment, we evaluate 2 classifiers: Logistic Regression and Random Forests to choose the best classifier for semantic labeling.

Table 8 - 10 lists results of two classifiers when being trained and tested on different datasets. We use city, museum and soccer datasets to train Logistic Regression and Random Forest since we can generate a sufficient amount of samples for training data. For semantic labeling, we use all 4 datasets: soccer,

Table 8. MRR scores of different classifiers when training on soccer

	soccer	museum	city	weather
Logistic Regression	0.814	0.863	0.944	0.951
Random Forests	0.794	0.799	0.947	0.86

Table 9. MRR scores of different classifiers when training on museum

	soccer	museum	city	weather
Logistic Regression	0.815	0.845	0.940	0.951
Random Forests	0.820	0.778	0.830	0.898

Table 10. MRR scores of different classifiers when training on city

	soccer	museum	city	weather
Logistic Regression	0.782	0.807	0.965	0.955
Random Forests	0.802	0.728	0.912	0.807

museum, city and weather with the numbers of labeled sources is 50% of the total numbers of sources in these datasets.

Overall, Logistic Regression achieves a comparable performance to Random Forests, which is a surprising result, because Random Forests have been shown to be the better classifiers in other research. However, because of the issue where we need to use class probabilities as confidence scores, the results can be explained.

Logistic Regression class probabilities are computed using the following function:

$$P(y = 1|x) = sigmoid(w^T x) \tag{8}$$

where x is the feature vector and w are its coefficients. Because $P(y = 1|x)$ is a monotonically increasing function of $w^T x$, $P(y = 1|x)$ increases when $w^T x$ increases. Thus, feature vectors with higher similarity values have higher class probabilities in Logistic Regression models.

Random Forests, on the other hand, calculate class probabilities based on fraction of samples of the same class in decision tree leaves. As long as the values are higher than splitting values in decision trees, feature vectors are split to the same branches and are likely to receive similar class probabilities. Therefore, using class probabilities of Random Forests as confidence scores performs worse.

Since the labeling accuracy of Logistic Regression and Random Forests are comparable, we consider the training time and labeling time of each classifier as additional measurements. Table 11 lists average system training time and labeling time of these classifiers.

The results in Table 11 show that Logistic Regression has a smaller training and labeling time. Although the differences are minor, it provides an advantage, especially in real-world scenarios with large amounts of data. Using Logistic Regression also provides more meaningful insights of features because of its linear

[1] https://github.com/minhptx/iswc-2016-semantic-labeling.git

Table 11. Training and labeling time of different classifiers

	Training time	Labeling time
Logistic Regression	144s	0.31s
Random Forests	157s	0.36s

combination compared with a randomized algorithm as Random Forests. Therefore, we use Logistic Regression as the classifier for the remaining experiments.

4.3 Feature Analysis

In machine learning classifiers, different features have different degrees of influence on the classification results. To analyze the importance of features in our similarity vectors, we train Logistic Regression on different datasets and extract coefficients of features. Table 12 shows coefficients of features when Logistic Regression models are trained on city, museum and soccer data.

Table 12. Coefficients of features in Logistic Regression classifier

Feature	**Train on soccer**	**Train on museum**	**Train on city**
ATT NAME	4.41	6.08	0
TEXT JACCARD	1.88	0.88	9.16
TEXT TF-IDF	3.91	1.03	3.20
NUM JACCARD	4.21	3.28	12.68
NUM KS	1.78	0.78	7.25
MW HISTOGRAM	0.32	1.14	3.83

In general, all of our similarity features have positive correlation with the classification results, which means that higher values in these similarity metrics results in higher probabilities that the attributes have the same semantic type. As we can see from the results, value similarity features play the most important role in Logistic Regression classifiers regardless of training domain. Attribute names similarity has a good impact on soccer and museum data but not in city because city dataset does not have headers or titles for attributes. Distribution and histogram similarity metrics have higher coefficients in city data because city dataset contains mostly numeric attributes.

In conclusion, we have demonstrated that our similarity features contribute to the similarity in the semantic types of attributes. However, the importance of features in the learned classifiers can vary according to the training data as shown in Table 12.

4.4 Semantic Labeling

In this experiment, we evaluate performance of DSL (Domain-independent Semantic Labeler) in comparison with SemanticTyper [11]. To follow real-world scenarios where labeled sources are hard to find and manually labeling sources is tedious, our experiments run on configuration with only 1 to 5 labeled data sources for every dataset (Weather dataset has only 4 sources so the maximum number of labeled sources is 3). For DSL, we follow the setup in section 4.1 while having soccer, city and museum as our classifier training dataset iteratively. For SemanticTyper, the MRR scores reported are the MRR scores when being trained on the testing domains. The weather domain is only used in semantic labeling because it cannot provide a sufficient number of feature vectors for training classifiers.

Table 13. MRR scores of DSL and SemanticTyper on soccer dataset

Number of labeled sources	1	2	3	4	5
DSL (train on soccer)	0.625	0.782	0.777	0.800	0.815
DSL (train on city)	0.601	0.785	0.788	0.808	0.820
DSL (train on museum)	0.600	0.781	0.788	0.808	0.810
SemanticTyper	0.608	0.711	0.720	0.720	0.732

Table 14. MRR scores of DSL and SemanticTyper on museum dataset

Number of labeled sources	1	2	3	4	5
DSL (trained on soccer)	0.471	0.665	0.719	0.755	0.790
DSL (trained on museum)	0.463	0.652	0.709	0.752	0.792
DSL (trained on city)	0.472	0.659	0.706	0.713	0.730
SemanticTyper	0.491	0.615	0.656	0.699	0.697

Table 15. MRR scores of DSL and SemanticTyper on city dataset

Number of labeled sources	1	2	3	4	5
DSL (trained on soccer)	0.913	0.932	0.932	0.941	0.945
DSL (trained on museum)	0.912	0.927	0.928	0.941	0.944
DSL (trained on city)	0.914	0.928	0.930	0.939	0.944
SemanticTyper	0.856	0.893	0.893	0.913	0.919

Table 16. MRR scores of DSL and SemanticTyper on weather dataset

Number of labeled sources	1	2	3
DSL (trained on soccer)	0.899	0.951	0.977
DSL (trained on museum)	0.899	0.951	0.977
DSL (trained on city)	0.902	0.955	0.977
SemanticTyper	0.852	0.920	0.955

The results in Table 13 - 16 show that our approach outperforms SemanticTyper in all four evaluation datasets. Although there are slight changes in performance when the classifiers are trained on different domains, the changes are not significantly different and it shows that our approach is robust across multiple data datasets. According to the table, training the classifier from the same domain, which provides more information about the characteristic of data in domains, slightly improves the accuracy of the classifier.

We also evaluate our system on the T2D Gold Standard dataset[2] and compare our result with T2K system's approach for properties matching [12]. As described in Ritze's work, labeled sources are extracted from the DBpedia ontology. After that, they divided the T2D Gold Standard dataset into two equal-sized parts: an optimization set and an evaluation set. The optimization set is used to optimize the essential parameters for the system and the result are evaluated on the evaluation set. Although we are unable to reconstruct the exact experiment, we approximated the result by using the following configuration as an alternative:

1. Collect DBpedia ontology data in table format as labeled sources.
2. For every attribute in the ontology, extract only 1000 first values as the set of values for the attribute.
3. Train the classifiers on combination of soccer, museum and city datasets to enrich the training data.
4. Test semantic labeling (properties matching) on the entire T2D Gold Standard dataset.

Table 17. MRR scores of DSL and T2K on T2D Gold Standard dataset

DSL	T2K (evaluation)	T2K (optimization)
0.773	0.730	0.700

Table 17 shows the results of DSL in comparisons with T2K. Although our approach is not optimized on the optimization set as T2K, we achieve a better accuracy on the dataset. Moreover, our classifiers have been trained on different domains and we only use 1000 values for every attribute as domain data instead of the entire set of values. Because we exploit more similarity features, our approach achieves better discriminative ability for the various semantic types. The evaluation also shows that we have a robust, domain-independent system that only needs to be trained once before using it for semantic labeling in a wide range of domains.

5 Related Work

Ramnandan et al. [11] describe an approach that captures and compares distributions and properties of data corresponding to semantic types as a whole. They apply heuristic rules to separate numeric and textual data and then use TF-IDF and KS as measures to compare the data. In our approach, we use more similarity features besides of TF-IDF and KS test, which enables our system to better discriminate between semantic types. Our similarity metrics can be applied to both textual and numeric attributes by the method described in Section 3.1.

[2] http://webdatacommons.org/webtables/goldstandard.html

Ritze et al. [12] propose a new approach for annotating HTML tables with DBpedia classes, properties, and entities. Their system, which is named T2K, use metrics like Jaccard, Levenshtein and deviation similarity to match attributes to properties and values. T2K also uses a iterative process to adjust property weights and filter the candidate sets until the similarity values converge. The system provides good results in entity and class matching but not in property matching. Since they exploit only value similarity for textual data and numeric similarity for numeric data for property matching, they face the same limitation as Ramnandan's work and achieve a lower performance compared to DSL.

A number of approaches have used probabilistic graphical models to solve the problem of semantic labeling. Goel et al. [4] exploit the underlying relationships between attributes and values with attribute characteristics as features and use Conditional Random Fields (CRF) to label attributes. They assign semantic types to every value in an attribute and then combine these semantic types to infer the semantic type for the whole attribute. Limaye et al. [7] use probabilistic graphical models in a broader problem as they annotate tables on the web by entities for cells, types for columns, and relationships for binary relations between columns. They exploit two feature functions that describe the dependency of column type with its values and header. The labels of all columns are then assigned simultaneously using a message passing algorithm to maximize the potential function formulated by features and their weights. Mulwad et al. [9] extend the work of Limaye et al. by proposing a novel *Semantic Message Passing* algorithm that uses Linked Open Data (LOD) knowledge to improve the existing semantic message algorithm. These approaches require the probabilistic graphical models to be retrained when handling new semantic types. The reason for this is that their feature weights are calculated associated with labels and need to be re-estimated for new semantic types. Also, graphical models do not scale well as the number of semantic types increases because of the explosion of different enumerations in the search space.

Mulwad et al. [10] also extend their work into a full system with multiple functions. They incorporated probabilistic semantic labeling with domain knowledge processing and data cleaning to produce a domain-independent semantic labeling system. However, their domain independence is limited in that it requires users to provide domain knowledge or apply preprocessing modules. In our semantic labeling system, the process is automatic and the domain-independent learning models only require a small amount of domain data to perform well on semantic labeling.

Venetis et al. [15] present an approach to annotate tables on the web by leveraging existing data on the web. An isA database in the form of {instance, class} is extracted from the web using linguistic patterns and is used to produce column labels. The column labels are assigned by a maximum likelihood estimator that assigns a column with a class label that maximize the fraction of column values in that label. Syed et al. [13] use Semantic Web data to infer the semantic models of tables. They annotate the table columns by using the column names if available and values inside the columns to build a query to Wikitology.

After that, columns are mapped to classes returned in the query result. Both the work of Venetis et al. and Syed et al. extract a huge amount of data from various sources to estimate the probability that a value belongs to a semantic type. Thus, their approach is restricted to domains where online data is widely available. In our approach, our learning model is not domain-specific and thus, we can use any domain as our training data and the system can still label data from other domains effectively.

Gunaratna et al. [5] address a related problem, which is called entity class resolution. Entity class resolution is similar to semantic labeling except that their targets are entity classes instead of semantic types. Their system, FACES, applies natural language processing (NLP) techniques to identify focus terms and uses text similarities to compare focus terms with entity class names in the ontology. Although FACES' approach works well in text documents because it is easy to detect focus terms in grammatical documents, it cannot be applied to most of web data such web tables, spreadsheets, or RDF stores because data values are mostly unstructured and do not follow grammar rules such as numbers and named entity mentions. In contrast, our approach does not rely on NLP algorithms so that it can perform effectively in noisy data sources from the web.

6 Conclusion and Future Work

In this paper, we presented a novel domain-independent approach for semantic labeling that leverages similarity measures and machine learning techniques. In our system, we capture the patterns of matching decisions given the similarity scores between unlabeled attributes and labeled data to find the correct semantic types. The approach allows us to train the machine learning model only once and use it in multiple different domains. Moreover, our similarity features are independent within a semantic type and across other semantic types. We can compute feature vectors using a parallel and distributed implementation which reduces the running time while maintaining labeling accuracy.

In the future, we plan to exploit transfer learning to incorporate some specific information about the domain data to adjust the weights of our features. For example, if a domain contains mostly numeric data, we may give more weight to numeric features. In view of the machine learning model, we can leverage data from Linked Open Data to enrich our learning models. In this way, the model can have information about many difficult cases and, therefore, it will be more likely to generalize well. Finally, although our approach allows new semantic types to be easily integrated, it lacks the ability to detect whether the true semantic type exists in the labeled data. This inability can lead to incorrect mappings in unseen cases and decrease the overall system accuracy. One of the directions of future work is to have the machine learning model detect these cases.

Acknowledgments This material is based upon work supported in part by the United States Air Force and the Defense Advanced Research Projects Agency (DARPA) under Contract No. FA8750-16-C-0045. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force and DARPA.

References

1. Ambite, J.L., Darbha, S., Goel, A., Knoblock, C.A., Lerman, K., Parundekar, R., Russ, T.: Automatically Constructing Semantic Web Services from Online Sources. In: Proceedings of the 8th International Semantic Web Conference. pp. 17–32. Springer-Verlag (2009)
2. Breiman, L.: Random Forests. Machine Learning 45, 5–32 (2001)
3. Craswell, N.: Mean reciprocal rank. In: Encyclopedia of Database Systems, pp. 1703–1703. Springer (2009)
4. Goel, A., Knoblock, C.A., Lerman, K.: Exploiting structure within data for accurate labeling using conditional random fields. In: Proceedings of the 14th International Conference on Artificial Intelligence (ICAI). vol. 69 (2012)
5. Gunaratna, K., Thirunarayan, K., Sheth, A., Cheng, G.: Gleaning Types for Literals in RDF Triples with Application to Entity Summarization. In: The Semantic Web. Latest Advances and New Domains, pp. 85–100 (2016)
6. Lehmann, E.L., Romano, J.P.: Testing Statistical Hypotheses (Springer Texts in Statistics) (2005)
7. Limaye, G., Sarawagi, S., Chakrabarti, S.: Annotating and Searching Web Tables Using Entities, Types and Relationships. Proc. VLDB Endow. 3, 1338–1347 (2010)
8. Manning, C.D., Raghavan, P., Schtze, H.: Introduction to Information Retrieval. Cambridge University Press, New York, NY, USA (2008)
9. Mulwad, V., Finin, T., Joshi, A.: Semantic message passing for generating linked data from tables. In: The Semantic Web–ISWC 2013, pp. 363–378. Springer (2013)
10. Mulwad, V.V.: TABEL - A Domain Independent and Extensible Framework for Inferring the Semantics of Tables. Ph.D. thesis, University of Maryland, Baltimore County (2015)
11. Ramnandan, S.K., Mittal, A., Knoblock, C.A., Szekely, P.: Assigning Semantic Labels to Data Sources. In: The Semantic Web. Latest Advances and New Domains, pp. 403–417. Springer International Publishing (2015)
12. Ritze, D., Lehmberg, O., Bizer, C.: Matching HTML Tables to DBpedia. In: Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics. pp. 10:1–10:6. WIMS '15, ACM, New York, NY, USA (2015)
13. Syed, Z., Finin, T., Mulwad, V., Joshi, A.: Exploiting a web of semantic data for interpreting tables. In: Proceedings of the Second Web Science Conference (2010)
14. Taheriyan, M., Knoblock, C.A., Szekely, P., Ambite, J.L.: Learning the semantics of structured data sources. Web Semantics: Science, Services and Agents on the World Wide Web (2016)
15. Venetis, P., Halevy, A., Madhavan, J., Paca, M., Shen, W., Wu, F., Miao, G., Wu, C.: Recovering semantics of tables on the web. In: Proceedings of the VLDB Endowment. pp. 528–538 (2011)