EmbNum+: Effective, Efficient, and Robust Semantic Labeling for Numerical Values

Phuc Nguyen^{1,2} · Khai Nguyen² · Ryutaro Ichise^{1,2} · Hideaki Takeda^{1,2}

Received: 9 February 2019 / Accepted: 14 October 2019 / Published online: 4 November 2019
© The Author(s) 2019

Abstract

In recent years, there has been an increasing interest in numerical semantic labeling, in which the meaning of an unknown numerical column is assigned by the label of the most relevant columns in predefined knowledge bases. Previous methods used the p value of a statistical hypothesis test to estimate the relevance and thus strongly depend on the distribution and data domain. In other words, they are unstable for general cases, when such knowledge is undefined. Our goal is solving semantic labeling without using such information while guaranteeing high accuracy. We propose EmbNum+, a neural numerical embedding for learning both discriminant representations and a similarity metric from numerical columns. EmbNum+ maps lists of numerical values of columns into feature vectors in an embedding space, and a similarity metric can be calculated directly on these feature vectors. Evaluations on many datasets of various domains confirmed that EmbNum+ consistently outperformed other state-of-the-art approaches in terms of accuracy. The compact embedding representations also made EmbNum+ significantly faster than others and enable large-scale semantic labeling. Furthermore, attribute augmentation can be used to enhance the robustness and unlock the portability of EmbNum+, making it possible to be trained on one domain but applicable to many different domains.

Keywords Deep metric learning · Semantic labeling · Attribute matching

✉ Phuc Nguyen
phucnt@nii.ac.jp

Khai Nguyen
nhkhai@nii.ac.jp

Ryutaro Ichise
ichise@nii.ac.jp

Hideaki Takeda
takeda@nii.ac.jp

¹ SOKENDAI (The Graduate University for Advanced Studies), 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, Japan

² National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, Japan

Introduction

Thanks to the Open Data movement, a large number of table data resources have been published on the Web or Open Data portals. For example, 233 million tables were extracted from the July 2015 version of the Common Crawl [10].¹ Additionally, 200,000 tables from 232 Open Data portals were analyzed by Mitlohner et al. [12]. These resources could be integrated, and enabling them to be potentially useful for other applications such as table search [13, 17], table extension [11], completion [2], or knowledge base construction as used in DBpedia [32], YAGO [23], and Freebase [5].

However, these data resources are very heterogeneous. Each data resource is independently constructed by different people with different backgrounds, purposes, and contexts. Therefore, the use of vocabulary and schema structures might differ across various data resources. For example, one table attribute uses “population” as the table header label and another table attribute uses “number of people.” Do these two attributes labels share the same or different meanings? This “semantic heterogeneity” may lead to the propagation of misinformation in the data integration process.

To provide a unified view of the heterogeneous resources, one of the possible solutions is to assign a semantic label for each attribute in unknown resources. We categorize these semantic labeling approaches into three groups with respect to the data type: textual-based, numerical-based, and hybrid, which is a combination of the results of textual-based and numerical-based semantic labeling. The main focus of this paper is numerical-based approaches.

The most common approaches for semantic labeling use textual information, such as header labels, textual values, or table description. Previous studies used text-based entity linkage to search for similar concepts and entities in knowledge bases [1, 6, 21, 28, 29, 32]. Then, semantic labels can be inferred by using rich lexical and semantic information of matched concepts and entities in the knowledge base. However, many attributes do not have overlapping entity labels with knowledge bases. Even when overlapped, many entity labels do not have similar representation with the entities in knowledge bases because these are expressed as IDs, codes, or abbreviations [15]. Additionally, the numerical values of attributes rarely have the same set of values as the relevant values in knowledge bases; therefore, it is ineffective to straightforwardly apply linguistic approaches into these numerical attributes.

In a study on profiling Open Data portals, Mitlohner et al. showed that 50% of the table data extracted from Open Data portals contain numerical columns with missing or ambiguous headers [12]. Therefore, annotating semantic labels for these numerical columns is an important task in the data integration procedure. Prior studies proposed general workflows based on a retrieval setting where the label of an unknown attribute is assigned by the label of the most relevant attribute in labeled data with respect to a specific similarity metric [15, 19, 20, 26]. The most common

¹ <http://commoncrawl.org/>

approach is using the p value of statistical hypothesis tests as a metric to measure the similarity between lists of numerical attributes.

We argue that these p value-based metrics have a critical issue that needs to be addressed. The issue is how to choose the appropriate statistical test for hypothesis testing. In fact, the statistical hypothesis tests depend on the distribution and type of data being analyzed. For instance, these data attributes have to be drawn from a specific form of distribution (e.g., normal distribution or uniform distribution) or data types (e.g., continuous or discrete). However, determining the form of distributions and data types of unknown numerical attributes is a difficult challenge. As a result, a proper hypothesis test cannot be easily selected when we do not know the data distribution and data type. Therefore, we propose a neural numerical embedding approach, called EmbNum+, to learn a similarity metric directly from data without the need of making an assumption regarding data type and data distribution.

In this paper, we address the limitations of prior approaches based on three dimensions: *effectiveness*, *efficiency*, and *robustness*.

Effectiveness—EmbNum+ was inspired from deep metric learning approaches with which both representations and a similarity metric are learned without making any assumption regarding data. In particular, we used a representation network consisting of a triplet network and convolutional neural network (CNN) to map numerical attributes into feature vectors in an embedding space. The “semantic similarities” of numerical attributes are calculated on these feature vectors. In other words, the CNN network learns discriminate features from numerical attributes. The triplet network approximates the goal of similarity-metric learning, which involves pulling numerical attributes with the same semantic labels into nearby positions while pushing numerical attributes with different labels apart. The distance metric is directly learned from data without the need to make any assumption regarding data type or data distribution; hence, enabling EmbNum+ to be more generalized with a variety of data types and distributions.

Efficiency—Efficiency is an important factor to take into account the task of semantic labeling because of data velocity, and data volume in the current interest in the Open Data movement. We need a lightweight approach to support real-time and large-scale semantic labeling. EmbNum+ has two advantages in terms of efficiency compared with other approaches. First, EmbNum+ is promising for efficient data storage and memory usage since all numerical attributes are stored by their representations derived from the embedding model. In fact, the size of these attributes could be very large; up to a million of numerical values; hence, keeping entire numerical values is memory intensive. Second, EmbNum+ reduces computational overheads using the fast similarity calculation on these representations.

We also introduced an inverse transform sampling [30] to deal with the issues of varying input size of table numerical attributes. The representation network required a fixed size as the input, but the size of numerical attributes could vary from a few numerical values to thousands of them [12]. The sampling technique can capture the original distribution of numerical attributes, thereby overcoming the issue of varying the size of the input attributes. Moreover, the sampling technique also helps to speed up data processing since a small number of data values is considered instead of the entire values of attributes.

Robustness—To learn a robust EmbNum+, we introduce an attribute augmentation technique, which is automated to generate more training data from available data. Our experiments showed that EmbNum+ is robust in all tested datasets and it can be used to learn discriminant representations and a similarity metric from a single domain data and uses across multiple domains.

We evaluated the performance of EmbNum+ against two baseline approaches, i.e., SemanticTyper [20] and DSL [19] on four datasets: one standard dataset, e.g., City Data [20], two synthesis datasets, e.g., DBpedia Numerical Knowledge Base (NKB) and WikiData NKB, and one real-world dataset, e.g., Open Data extracted from five Open Data portals. The overall results show that EmbNum+ outperformed all other approaches in all experimental settings in terms of effectiveness, efficiency, and robustness.

EmbNum+ is an extended version of our work EmbNum [16]. In this paper, the extended version advances EmbNum in several ways.

- It improves semantic labeling effectiveness by using the attribute augmentation technique to generate more training data, therefore, it makes EmbNum+ more robust to overfitting. In “[Evaluation](#)” section, we show the advantages of attribute augmentation technique where EmbNum+ performs better than EmbNum [16] with the attribute augmentation is not taken into account.
- EmbNum+ can recognize whether a query numerical attribute is a new semantic label or not. In particular, we introduce a lightweight relevance-learning approach to model the relevant information between the labeled numerical attributes. Using the relevance model can help to filter the retrieval-ranking results. If the size of the ranking result is zero, it means that the query numerical attribute is a new semantic label.
- We evaluated EmbNum+ on the two new synthesis datasets, i.e., DBpedia NKB and WikiData NKB, that containing 372 semantic labels with more than 10 million numerical values extracted from DBpedia and Wikidata. These datasets are far larger than the two datasets, i.e., City Data and Open Data [16]; therefore, they enabled a more rigorous analysis of the effectiveness, efficiency, and robustness of EmbNum+ against other baseline approaches.

The rest of this paper is organized as follows. In “[Definitions and Notations](#)” section, we define the terms, concepts, and common notations used entire this paper as well as the semantic labeling problem for numerical attributes. In “[Related Work](#)” section, we discuss the related works on the task of semantic labeling for table data, and deep metric learning techniques. We present the EmbNum+ approach in “[Approach](#)” section. In “[Evaluation](#)” section, we describe the details of our evaluation, and experimental settings and then present the results. Finally, we summarize the paper and discuss the future direction in “[Conclusion](#)” section.

Table 1 List of frequently used mathematical notations in alphabetical order

Symbol	Description
(a_i, y_j)	A pair of numerical attribute and its corresponding semantic label, $(a_i, y_j) \in D$
a_i	A numerical attribute, $a_i \in A$
a_q	A unknown attribute
A	A list of numerical attributes
a	A numerical attribute
D	A numerical knowledge base
$\text{emb}(x)$	The embedding function for a tensor x , $\text{emb}(x) \in \mathbb{R}^k$
h	The output size of attribute transformation
k	The output size of embedding model
m	The number of semantic labels
n	The number of numerical attributes
p	The percentage probability
$\text{cdf}(v)$	The cumulative distribution functions (CDF) of a numerical value
$\text{icdf}(p)$	The inverse cumulative distribution functions of a probability p
$T_{i,j}$	A table cell
T	A table
$\text{trans}(V_a)$	The transformation for list of numerical values V_a
V_a	A list of numerical values of the attribute a
v	A numerical value
x	A tensor of numerical attributes, $x \in \mathbb{R}^h$
y_j	A semantic label, $y_j \in Y$
Y	A list of semantic labels

Definitions and Notations

In this section, we define the mathematical notations (“[Notations](#)” section) and general concepts regarding tables and table attributes (“[Definitions](#)” section) and then describe the problem statement of semantic labeling for numerical attributes (“[Problem Statement](#)” section).

Notations

We refer to Table 1 as the mathematical notations frequently used in this paper.

Definitions

In this section, we define the table, table attributes, numerical attributes, and textual attributes.

Definition 1 (*Table*). A table (denoted as T) is a two-dimensional tabular structure consisting of an ordered set of rows and columns. Each intersection between a row and column determines a cell T_{ij} , with its value as a number, string, or an empty value.

According to Nishida et al., there are six table types, i.e., vertical and horizontal relational tables, vertical and horizontal entity tables, matrix tables, and other tables [18]. In vertical tables, values of one attribute are listed on the vertical axis, while they are listed on the horizontal axis in horizontal tables. The relational tables list one or many attributes for a set of entities, the entity labels list one or many attributes for one entity, and the matrix tables have their cells measured in the same type at the intersection of a row and a column. We focused on those table types that have a list of numerical values, and all values in this list belong to the same type. These table types are equivalent to *vertical and horizontal relation tables*, and *matrix tables* in the study by Nishida et al. [18].

Definition 2 (*Table attribute*). A table attribute is a column of a vertical relation table, row of a horizontal relation table, or a row (column) of a matrix table.

We assume that all values in one table attribute have the same meaning and that shared meanings exist across tables. Table attributes can be categorized as either numerical attributes or textual attributes.

Definition 3 (*Textual attribute*). A textual attribute is a table attribute where all values of this attribute are textual.

Definition 4 (*Numerical attribute*). A numerical attribute is a table attribute where all values of this attribute are numerical.

We only consider a similarity metric between numerical attributes to infer semantic meaning. The similarity metric for a textual attribute is out of the scope of this paper. Notably, we did not tackle the problem of data scaling. For instance, if two attributes were found to have the same meaning, but they were expressed in different scales, we considered that the two attributes have a different meaning.

Problem Statement

The problem of semantic labeling for numerical values is defined as follows. Let $A = \{a_1, a_2, a_3, \dots, a_n\}$ be a list of n numerical attributes and $Y = \{y_1, y_2, y_3, \dots, y_m\}$ be a list of m semantic labels. Given (1) an unknown attribute a_q , and (2) a numerical knowledge base $D = \{(a_i, y_j) | a_i \in A, y_j \in Y\}$, where (a_i, y_j) is a data sample of a pair of a numerical attribute and its corresponding semantic label, the objective of semantic labeling is to identify the list of relevant semantic labels in D where these attributes are most likely closed to the unknown attribute a_q with respect to a numerical similarity metric.

Related Work

In this section, we present the previous approaches for semantic labeling with textual information (“[Semantic Labeling with Textual Information](#)” section) and numerical information (“[Semantic Labeling with Numerical Information](#)” section). Next, we will briefly describe related works on deep metric learning techniques (“[Deep Metric Learning](#)” section).

Semantic Labeling with Textual Information

Several attempts have been made to assign semantic labels to for table attributes using the information on header labels and textual values [6, 21, 32]. The most common approaches use entity linkage for mapping textual values of attributes to entities in a knowledge base. The schema of entities is then used to find the semantic concept for table attributes. Additional textual descriptions of tables have also been considered to improve the performance of the labeling task [1, 28, 29]. However, it is not easy to apply these approaches to numerical attributes because the unknown attribute rarely has the same set values with knowledge base values [20]. To build an effective integrated system, taking into account semantic labeling using numerical information is necessary.

Semantic Labeling with Numerical Information

Most researchers on semantic labeling with numerical information use statistical hypothesis tests as similarity metrics to compare the similarity of numerical attributes. Stonebraker et al. proposed a method for schema matching using decisions from four *experts* [26]. One decision used the t test statistic of the Welch’s t test [9] to measure the probability that two numerical attributes were drawn from the same distribution. The t test value is calculated based on the means and variances of two numerical attributes, which have to follow a normal distribution. However, numerical attributes do not always follow a normality assumption; therefore, applying a t test as a similarity metric is not appropriate for the non-normality numerical attributes.

To address the limitations of the t test, Ramnandan et al. [20] proposed SemanticTyper to find a better statistical hypothesis test. They tested on the Welch’s t test [9], the Mann–Whitney U test [14], and the Kolmogorov–Smirnov test (KS test) [9]. The authors reported that using the p value of the KS test archived the highest performance in their experiments than the two other tests. However, the Mann–Whitney U test and the KS test are used assuming that the two tested numerical attributes are continuous. Therefore, these tests are often not very informative when the distribution of tested attributes is assumed to be discrete.

Pham et al. [19] extended SemanticTyper (called DSL) by proposing a new similarity metric that is a combination of the KS test and two other metrics: the Mann–Whitney test (MW test) and the numeric Jaccard similarity. Their experiments

showed that using this combined metric provided better results over using only a KS test. However, performing multiple hypothesis tests is computationally intensive and it is not easy to learn a robust metric when we only have a small number of training data.

Neumaier et al. [15] created a numeric background knowledge base from DBpedia. Given an unknown numerical attribute, they also used the p value of the KS test as a similarity metric to compare with each labeled attribute in the numeric background knowledge base. In fact, the size of attributes varies from a few numerical values to a few hundred thousand numerical values. Therefore, keeping the original numerical values of attributes in background knowledge bases is memory intensive. Moreover, it is very time-consuming to conduct the KS test between the unknown numerical attributes with all attributes in the background knowledge base since the computational overheads of the KS test are strongly based on the size of compared attributes.

Overall, the similarity metrics used with these approaches are hypothesis tests, which are conducted under a specific assumption regarding data type, and data distribution. In contrast to these approaches, we propose a neural numerical embedding model to learn a similarity metric directly from data without making any assumption about data type and data distribution. Moreover, EmbNum+ is promising for efficient memory processing and data storage as well as fast similarity metric calculation across numerical attributes because all computations are made on these representations are derived from numerical attributes by using the embedding model.

Deep Metric Learning

Deep metric learning has achieved considerable success in extracting useful representations of data by introducing deep neural networks [3, 24, 31]. Deep metric learning utilizes the advantages of the nonlinear representation learning of a convolutional neural network (CNN) and the discrimination ability of metric learning; therefore, it is widely used in many tasks of computer vision. There are a large number of published studies (the siamese network [27], the triplet network [22], the quadruplet network [4]) proposed loss functions to learn better representations. In this study, we used a representation network, which is a combination of a CNN network and the triplet network, to jointly learn representations and a metric to measure the similarity between numerical attributes. Different from images data, EmbNum+ models the discrimination features from the distribution presentations of numerical attributes.

Approach

In this section, we present the overall framework of EmbNum+ in “[Overview](#)” section. The detail of each component is described from “[Attribute Augmentation](#)” section to “[Attribute Transformation](#)” section.

Fig. 1 General architecture of semantic labeling for numerical attributes with EmbNum+. Overall workflow consists of four phases: **a** representation learning, **b** relevance learning, **c** semantic labeling offline, and **d** semantic labeling online

Overview

Figure 1 depicts the semantic labeling task with EmbNum+. The general workflow is composed of four phases: *representation learning*, *relevance learning*, *semantic labeling offline*, and *semantic labeling online*.

The overall workflow of semantic labeling starts with the *representation learning* as shown in Fig. 1a (“Representation Learning” section) that jointly learn the discriminative representations and a similarity metric across numerical labeled attributes. Given numerical *labeled attributes*, new labeled data samples are first generated with the attribute augmentation (“Attribute Augmentation” section). Second, *attribute transformation* (“Attribute Transformation” section) converts these augmented labeled attributes from numerical values into distribution presentations. Third, these distribution presentations will be used as the input for *representation learning*. Finally, the output of the *representation learning* is an *embedding model* which is used as the *feature extraction* module in the later phases.

To determine whether a query attribute is a new semantic label, we introduced the *relevance learning* as shown in Fig. 1b (“Relevance Learning” section). Specifically, we used the logistic regression to learn a *relevance model* that predicts the relevance probabilities from pair-wise similarities of the labeled attributes. To calculate the pair-wise similarity of the labeled attributes, the *attribute augmentation* and the *attribute transformation* are also carried out and *feature extraction* is done to derive the feature vectors using the *embedding model* learned in the representation-learning phase. The pair-wise similarities are calculated on those feature vectors. The *relevance model* will be used to filter the ranking results in the *semantic-labeling-online* phase. If the size of the ranking results is zero after filtering, the query attribute is a new semantic label.

Fig. 2 Illustration of augmentation on a numerical attribute. The input is the original attribute (Left), and the output is the augmented attributes (Right). We use all data (original and augmented attributes) during the training process

As mentioned above, there are two *semantic-labeling* phases as shown in Fig. 1c (“[Semantic Labeling](#)” section): *offline* and *online* (hereafter, offline phase and online phase, respectively). The *offline phase* involves data preparation, while the *online phase* is actually semantic labeling for an unknown attribute. In the *offline phase*, *labeled attributes* are standardized with attribute transformation and derived feature vectors with feature extraction. These feature vectors are stored in the *knowledge base* for future similarity comparison.

In the *online phase* of semantic labeling as shown in Fig. 1d, the first two steps are similar to those of the *offline phase* where an unknown attribute is standardized with *attribute transformation* and feature vectors are derived with *feature extraction*. Then, the *similarity searching* module is used to calculate the similarities between the feature vector of the unknown attribute and all the feature vectors in the *knowledge base*. After this process, we have a ranking list of semantic labels ordered by their corresponding similarity scores. Then, the *relevance model* is used on this ranking list to select only the relevant semantic labels based on these scores. Finally, the output is a ranking list of the most relevant attributes.

In the following sections, we describe the details of each component in the overall workflow starting with the attribute augmentation, then, attribute transformation, the two learning phases: representation learning and relevance learning, and the two semantic-labeling phases: online and offline. Readers may refer to “[Notations](#)” section for the explanation of mathematical notations and equations.

Attribute Augmentation

In this section, we introduce a technique for augmenting attributes from originally labeled attributes to create more labeled attributes for learning. In principle, we need many labeled attributes to learn a robust embedding model and relevance model; however, the lack of labeled data is a common problem with this task. The size of datasets for this task is not so large, for example, the City Data [20] contains 300 numerical columns with 30 semantic labels, and the Open

Fig. 3 Attribute augmentation for the numerical attribute of *areaLandSqMi* in City Data [20]. Blue line represents 8327 original values, and dotted lines represent the augmented samples from original values

Data [16] contains 500 numerical columns with 50 semantic labels. The reason for this issue is that it is very time-consuming and costly to manually assign the semantic labels for table data. Therefore, we need to augment existing data to increase data size.

To address this issue, we introduced an attribute augmentation technique based on the intuition that the semantic label of an attribute does not change whether we have several or thousands of numerical values (Fig. 2). Therefore, we can generate samples by changing the size of attributes and randomly choose the numerical values in the original attributes. While using the attribute augmentation technique for an attribute, the distribution of augmented attributes is slightly different from the original one. Therefore, it increases the variety of attribute distribution and addresses the issue of lacking training data.

The attribute augmentation technique is described as follows. Given (1) an attribute a having n numerical values $V_a = [v_1, v_2, v_3, \dots, v_n]$, and (2) a number of new data samples aug_size , this attribute augmentation technique first randomly determines a new size $n' \in [min_size, n]$ for the new sample, then randomly selects n' values from V_a to V'_a , where $V'_a \subseteq V_a$. The process is repeated aug_size times to create new aug_size samples for a .

Figure 3 illustrates an example of attribute augmentation for the numerical attribute of *areaLandSqMi* in City Data [20]. The figure depicts the probability distribution of the original values of *areaLandSqMi* (blue line) and five augmented samples (dotted lines). The five augmented samples vary in terms of attribute size and shape, increasing the variety of training data, and reducing the overfitting for representation learning and relevance learning.

Attribute Transformation

In this section, we describe the transformation of numerical attributes to standardize the input size, for the representation learning. Attribute transformation is an important module because the representation learning requires a standardized input size, and the size of numerical attributes could vary from a few to thousands of values. We use inverse transform sampling [30] (“[Inverse Transform Sampling](#)” section) to standardize the input size and transform numerical values into forms of distribution presentations. This technique is chosen because it retains the original distribution of a given list of numerical values. In “[Transformation Analysis](#)” section, we empirically showed that the output from the inverse transform sampling is better than the usual random-choice sampling technique. After transformation, the list of numerical values is sorted in a specific order to leverage the capability of the CNN network to learn representations from distribution presentations.

Given an attribute a having numerical values $V_a = [v_1, v_2, v_3, \dots, v_n]$, the objective of attribute transformation is the $\text{trans}(V_a)$ function, which is defined as follows.

$$x = \text{trans}(V_a) = x_{\text{icdf}} \quad (1)$$

The transformation function $\text{trans}(\cdot)$ converts V_a into x , where $x \in \mathbb{R}^h$. The list of values $x_{\text{icdf}} \in \mathbb{R}^h$ is obtained by the transformation using the inverse transform sampling (“[Inverse Transform Sampling](#)” section) on numerical values. The inverse transform sampling is described as follows.

Inverse Transform Sampling

Let a be an attribute with numerical values $V_a = [v_1, v_2, v_3, \dots, v_n]$. We treat V_a as a discrete distribution so that the CDF of $v \in V_a$ is $\text{cdf}_{V_a}(v)$ and expressed as follows.

$$\text{cdf}_{V_a}(v) = P(V_a \leq v), v \in V_a, \text{cdf}_{V_a} : \mathbb{R} \rightarrow [0, 1] \quad (2)$$

where $P(V_a \leq v)$ represents the probability of values in V_a less than or equal to v . The inverse function of $\text{cdf}_{V_a}(\cdot)$ takes the probability p as input and returns $v \in V_a$ as follows.

$$\text{icdf}_{V_a}(p) = \text{cdf}_{V_a}^{-1}(p) = \min\{v : \text{cdf}_{V_a}(v) \geq p\}, p \in [0, 1] \quad (3)$$

We select h numbers (“[Implementation and Settings](#)” section) from V_a where each number is the output of the inverse distribution function $\text{icdf}_{V_a}(p)$ with probability $p \in \mathcal{P} = \{\frac{i}{h} | i \in \{1, 2, 3, \dots, h\}\}$. For example, when the input size $h = 100$, then we have $\mathcal{P} = \{0.01, 0.02, 0.03, \dots, 1\}$. For each attribute $a \in A$, we have a list of values $x_{\text{icdf}} = \{v_1, v_2, v_3, \dots, v_h\}$ that correspond to the given list of probabilities $\mathcal{P}$.

Fig. 4 Analysis of inverse CDF and random-choice sampling technique on the *decRainDays* (a) and the *aprHighF* (b) properties of City Data

Fig. 5 Representation-learning architecture

Transformation Analysis

To understand how well the samples of the inverse transform sampling fit the original data, we analyzed two attributes in City Data using the inverse transform sampling and the random-choice sampling technique, which generates a uniform random sample from a given list of numerical values.

Figure 4 depicts the transformation results of two techniques on the *decRainDays* and the *aprHighF* properties of City Data. The distribution using the inverse transform sampling (blue curve) clearly better fits the original distribution (red circles) than the random-choice sampling technique (green curve). Therefore, the inverse transform sampling is better to simulate the original distribution.

Table 2 The comparison of different CNN architectures on model size and model depth

CNN networks	#Parameters	#Layers
ResNet 18	3,854,164	18
ResNet 34	7,228,500	34
ResNet 50	15,995,220	50
ResNet 101	28,302,676	101
ResNet 152	38,441,300	152
VGG 11	34,628,436	11
VGG 13	34,690,452	13
VGG 16	36,463,764	16
VGG 19	38,237,076	19

Representation Learning

In this section, we describe the representation-learning phase (Fig. 5) to learn the embedding models.

Given an list of n transformed attribute $X = \{x_1, x_2, x_3, \dots, x_n\}$ and their m semantic labels $Y = \{y_1, y_2, y_3, \dots, y_m\}$, we first performed triplet sampling to create a list of triplets. A triplet (x, x^+, x^-) with $(x, x^+, x^- \in X)$ is a combination of an anchor x where the semantic label is y , a positive attribute x^+ where the semantic label is y , and a negative attribute x^- where the semantic label is not y . In Fig. 5, the blue square and green square indicate the semantic label y_1 and the semantic label y_2 , respectively, where $y_1 \neq y_2$. We used the hard negative sampling method [22] (negative samples close to anchor) to select triplets for training. This method involves choosing the closest sample to an anchor among the dissimilar attributes in a mini-batch of learning and helps the training process to converge faster.

We used the triplet network and CNN to learn a mapping function $\text{emb}(\cdot)$ for numerical attributes of triplets into an embedding space so that the distance between a positive pair must be less than that between a negative pair $d_{\text{emb}}(x, x^+) < d_{\text{emb}}(x, x^-)$ [22]. Recall a numerical attribute x , $\text{emb}(x)$ is the output of a representation learning network to convert x into a k dimensions embedding space $\text{emb}(x) \in \mathbb{R}^k$. We used the two-dimensional Euclidean plane as the embedding space because it is the most common space used in the literature. It is noticed that we also can use any of other spaces as the embedding space. The distance between two numerical attributes x_i and x_j is calculated by using the Euclidean distance between $\text{emb}(x_i)$ and $\text{emb}(x_j)$:

$$d_{\text{emb}}(x_i, x_j) = d(\text{emb}(x_i), \text{emb}(x_j)) = \|\text{emb}(x_i) - \text{emb}(x_j)\|_2^2 \quad (4)$$

The triplet loss function for the triplet network is defined as follows.

$$L = \max(0, \alpha + d_{\text{emb}}(x, x^+) - d_{\text{emb}}(x, x^-)) \quad (5)$$

where α is a hyper-parameter that regularizes between positive pair distance and negative pair distance.

Fig. 6 Relevance learning architecture

Embedding model with CNN architecture We use a CNN architecture to learn the embedding model since it can capture latent features directly from data. Instead of designing a CNN architecture from scratch, we adopted the popular CNN architectures such as VGG [25] and ResNet [7] architectures. Since the input data were one-dimensional, we modified the convolutional structure to be one-dimensional on the convolutional, batch normalization, and pooling layers. The output of the CNN is a vector with k dimensions (“Implementation and Settings” section).

Table 2 depicts a comparison of different CNN architectures on model size, and model depth of the adapted VGG and ResNet (one-dimensional data). Intuitively, the more layers of the network have, the better information capacity model can carry [7]. Finally, we came up with the ResNet 18 since it requires fewer parameters but deeper comparing to other networks. It is noticed that we adopted the ResNet 18 as a practical reason, and we can use other CNN architectures instead. The optimal designing CNN architectures are left as future work.

Relevance Learning

In this section, we describe the relevance-learning architecture in detail. Figure 6 illustrates the procedure of relevance learning.

Given the list of n transformed attributes $X = \{x_1, x_2, x_3, \dots, x_n\}$ and their m semantic labels $Y = \{y_1, y_2, y_3, \dots, y_m\}$, we first used the learned embedding model to derive the feature vectors for those attributes (apply mapping function $\text{emb}(\cdot)$ on each transformed attribute $x \in X$). We then carried out the pair-wise similarities (all combination) across attributes with the Euclidean distance on their embedding vectors (Eq. 4). Each pair-wise similarity is used as an input variable, while its output label (relevance or non-relevance) is determined by the semantic labels. If two attributes have the same semantic label, the data label of the pair-wise similarity is relevance, else the label is non-relevance. All pairs of input variables and output labels are used as training data for the relevance model.

We used the logistic regression (binary classification) to learn the *relevance model*. It is notice that we can use other binary classification algorithms. Regarding the logistic regression, a sigmoid function is used to map the pair-wise similarities to probabilities as Eq. 6.

Fig. 7 Offline-semantic-labeling architecture

Fig. 8 Online-semantic-labeling architecture

$$\sigma(d) = \frac{1}{1 + e^{-(w_1 * d + w_0)}} \quad (6)$$

where w_1, w_0 are the learning parameters for logistic regression model, d is a pair-wise similarity. We used θ as the decision boundary parameter. If the probability $\sigma \geq \theta$, the prediction is relevance, else the prediction is non-relevance.

Semantic Labeling

The *semantic labeling* has two phases: *offline* and *online*. The *offline phase* involves data preparation, while the *online phase* involves actual semantic labeling for an unknown attribute.

Offline Phase

In the *offline phase*, *labeled attributes* are standardized with attribute transformation (“Attribute Transformation” section) and feature vectors are derived with feature

Table 3 Statistical description about the number of numerical values per semantic label in four datasets: City Data, Open Data, Wikidata NKB, and DBpedia NKB

Dataset	m	n	#Values of each labels				
			All	Min	Max	Med	Avg
City Data	30	300	192,820	40	22,510	1,130	6,427.33
Open Data	50	500	7,329,815	120	1,671,455	12,506	146,596.3
Wikidata NKB	169	1690	5,762,389	55	1,535,812	1,055	34,096.98
DBpedia NKB	203	2030	4,467,716	56	955,957	1,632	22,008.45

extraction (“[Representation Learning](#)” section). These feature vectors are stored in the *knowledge base* for future similarity comparisons.

Figure 7 depicts an example of semantic labeling for an unknown attribute. Given labeled attributes and their semantic labels $\{(a_1, y_1), (a_2, y_2), (a_3, y_3), \dots, (a_n, y_m)\}$, the data is first transformed into $\{(x_1, y_1), (x_2, y_2), (x_3, y_3), \dots, (x_n, y_m)\}$ using the attribute transformation module. After that, the mapping function $\text{emb}(\cdot)$ is used to map all the labeled data into the embedding space $\{(\text{emb}(x_1), y_1), (\text{emb}(x_2), y_2), (\text{emb}(x_3), y_3), \dots, (\text{emb}(x_n), y_m)\}$. These data are stored in a knowledge base D for the next comparison in the semantic-labeling online phase.

Online Phase

In this section, we describe the online phase of semantic labeling for unknown attributes. Figure 8 depicts the process of the online phase. Given an unknown attribute a_q , the two first steps are similar to those of the *offline phase* where an unknown attribute is standardized to x_q with *attribute transformation* (“[Attribute Transformation](#)” section) and $\text{emb}(x_q)$ features are derived with *feature extraction* (“[Representation Learning](#)” section). Then, the *similarity searching* module is used to calculate the similarities (Equation 4) between the feature vector of the unknown attribute and all the feature vectors in the *knowledge base* D . After this process, we have a ranking list of semantic labels ordered by their corresponding similarity scores. Then, the *relevance model* is used on those similarity scores to predict the semantic labels is relevance or non-relevance using Equation 6 (“[Relevance Learning](#)” section). Then, we remove all the non-relevance semantic labels from the ranking list. Finally, the output is a ranking list of the most relevant attributes.

Evaluation

In this section, we first describe the benchmark datasets in “[Benchmark Datasets](#)” section and evaluation metrics in “[Evaluation Metrics](#)” section. Next, we present the implementation details and experiment settings in “[Implementation](#)” section.

Table 4 Overall quantile ranges of City Data, Open Data, DBpedia NKB, and Wikidata NKB

Dataset	From	To
City Data	−10e2	10e8
Open Data	−10e6	10e14
Wikidata NKB	−10e5	10e12
DBpedia NKB	−10e10	10e16

and Settings” section. Finally, we explain the results and detailed analysis on semantic labeling for numerical values and of in “Experimental Results” section.

Benchmark Datasets

To evaluate EmbNum+, we used four datasets, i.e., City Data, Open Data, DBpedia NKB, and Wikidata NKB. City Data is the standard data used in the previous studies [19, 20] while Open Data, DBpedia NKB, and Wikidata NKB are newly built datasets extracted from Open Data portals, DBpedia and Wikidata, respectively. The datasets are available at <https://github.com/phucty/embnum>.

The detailed statistics of each dataset are shown in Table 3. m denotes for the number of semantic labels in a dataset. n denotes for the number of columns in a dataset. In each dataset, each semantic label has 10 columns in the same semantic labels. The columns of City Data, DBpedia NKB, and Wikidata NKB are randomly generated using 10 partitions splitting, while the columns of Open Data are the real table columns from Open Data portals. The number of semantic labels of the new datasets is larger than City Data, enabling rigorous comparisons between EmbNum+ and other baseline approaches.

Table 4 reports the overall quantile ranges of the four datasets. DBpedia NKB is the most complex dataset in terms of the largest semantic labels (206 semantic labels) as well as the range of numerical values (the range of $[-10e10, 10e16]$). Moreover, the overlapping rate of numerical attributes in DBpedia NKB is also higher than other datasets. The detailed distributions of quantile ranges of each numerical attribute in City Data, Open Data, DBpedia NKB, and Wikidata NKB are depicted in Figs. 12, 13, 14, and 15 of “Appendix A,” respectively.

DBpedia NKB and City Data have the same source of data as DBpedia. Therefore, there is high overlapping of attributes between these data. The two other datasets of Wikidata NKB and Open Data are different from DBpedia NKB and City Data. Wikidata NKB is constructed from Wikidata; it is an independence project manually annotated by the community. The source of Wikidata NKB is different from Wikipedia where DBpedia is extracted from; therefore, Wikidata NKB and DBpedia NKB are different. Open Data is extracted from five Open Data portals which are different about the domain of data with other datasets.

City Data

City Data [20] has 30 numerical properties extracted from the city class in DBpedia. The dataset consists of 10 sources; each source has 30 numerical attributes associated with 30 data properties.

Open Data

Open Data has 50 numerical properties extracted from the tables in five Open Data portals. We built the dataset to test semantic labeling for numerical values in the open environment.

To build the dataset, we extracted table data from five Open Data portals, i.e., Ireland (data.gov.ie), the UK (data.gov.uk), the EU (data.europa.eu), Canada (open.canada.ca), and Australia (data.gov.au). First, we crawled CSV files from the five Open Data portals and selected files whose size is less than 50 MB. Then, we analyzed tables in CSV files and selected only numerical attributes. After that, we created attribute categories based on the clustering of the numerical attributes with respect to the textual similarity of column headers. A category contains many numerical columns with the same semantic labels. We got 7,496 categories in total.

We manually evaluated these categories with two criteria: (1) The first criterion was to pick up categories with a certain frequency. By examining the collection of data, we found that high-frequency and low-frequency categories are often unclear on their semantics. We decided to pick up the categories with ten attributes by following the setting of City Data. (2) The second criterion was removing the categories where column headers had too general meanings such as “ID,” “name,” or “value.”

Finally, we chose 50 categories as semantic labels; each semantic label had ten numerical attributes. Following the guideline of City Data, we also made 10 data sources by combining each numerical attributes from each category.

Wikidata NKB

The Wikidata NKB was built from the most frequently used numerical properties of Wikidata. At the time of processing, there were 477 numerical properties² but we only selected 169 numerical properties which are used more than 50 times in Wikidata.

DBpedia NKB

To build the DBpedia NKB, we collected numerical values of the 634 of DBpedia properties directly from their SPARQL query service.³ Finally, we obtained 206 of

² <https://query.wikidata.org/>

³ <http://live.dbpedia.org/sparql>

the most frequently used numerical properties of DBpedia where each attribute has at least 50 values.

Evaluation Metrics

We used the mean reciprocal rank score (MRR) to measure the effectiveness of semantic labeling. The MRR score was used in the previous studies [19, 20] to measure the probability correctness of a ranking result list. To measure the efficiency of EmbNum+ over the baseline methods, we evaluated the run-time in seconds of the semantic labeling process.

Implementation and Settings

We have different interests in each dataset to evaluate the performance of EmbNum+. The DBpedia NKB is the most complex and complete dataset with the largest semantic labels as well as a wide range of values. It provides a discriminative power to train the representation model as well as relevance model EmbNum+. Therefore, we use the DBpedia NKB dataset for these two learning modules. The details of the learning settings are described in “[Representation and Relevance Learning](#)” section. We use City Data as the standard data to fairly compare with other existing approaches. The Wikidata NKB is challenging in terms of large scale and transfer capacity setting where the embedding model is learned from DBpedia NKB. Finally, Open Data is used to evaluate the real-world setting where numerical attributes are extracted from the five Open Data portals.

Representation and Relevance Learning

To train EmbNum+, we used the numerical attributes of DBpedia NKB as the training data. We randomly divided DBpedia NKB into two equal parts: 50% for the two learning tasks and 50% for evaluating the task of semantic labeling. The first part was used for the representation learning of EmbNum+. It is noticed that we made using the attribute augmentation technique to generate training samples. Therefore, the actual training data is not the same as the original data. We also use this part to train the relevance model by using the pair-wise distance between these original training samples. This data was also used to learn the similarity metric for DSL. We follow the guideline that using logistic regression to train the similarity metrics where training samples are the pairs of numerical attributes [19].

We used PyTorch (<http://pytorch.org>) to implement representation learning. The network uses the rectified linear unit (ReLU) as a nonlinear activation function. To normalize the distribution of each input features in each layer, we also used batch normalization [8] after each convolution and before each ReLU activation function. We trained the network using stochastic gradient descent (SGD) with back-propagation, a momentum of 0.9, and a weight decay of $1e-5$. We started with a learning rate of 0.01 and reduced it with a step size of 10 to finalize the model. We set the dimension of the attribute input vector h and the attribute output vector k as 100.

Table 5 Semantic labeling setting with 10 data sources

Experiment	1	2	3	4	5	6	7	8	9	...	90
Queries	s_1	s_1	s_1	s_1	s_1	s_1	s_1	s_1	2	...	s_9
Knowledge base	s_2	s_2	s_2	s_2	s_2	s_2	s_2	s_2	s_1	...	s_1
		s_3	s_3	s_3	s_3	s_3	s_3	s_3		...	s_2
			s_4	s_4	s_4	s_4	s_4	s_4		...	s_3
				s_5	s_5	s_5	s_5	s_5		...	s_4
					s_6	s_6	s_6	s_6		...	s_5
						s_7	s_7	s_7		...	s_6
							s_8	s_8		...	s_7
								s_9		...	s_8

Table 6 Number of unseen semantic labels of DBpedia NKB, City Data, Wikidata NKB, and Open Data

Dataset	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%
DBpedia NKB	0	20	40	60	81	101	121	142	162	182
City Data	0	3	6	9	12	15	18	21	24	27
Wikidata NKB	0	16	33	50	67	84	101	118	135	152
Open Data	0	5	10	15	20	25	30	35	40	45

We trained the EmbNum+ with 20 iterations. In each iteration, we used the attribute augmentation technique to generate a *aug_size* of 100 samples for each semantic labels. The numerical values of the augmented samples are randomly selected from the list of numerical values of the attributes. The size of each augmented sample ranges from *min_size* of 4 to the size of its original attribute. Then, the representation learning was trained with 100 epochs. After each epoch, we evaluated the task of semantic labeling on the MRR score using the original training data. We saved the learned model having the highest MRR score. All experiments ran on Deep Learning Box with an Intel i7-7900X-CPU, 64 GB of RAM, and three NVIDIA GeForce GTX 1080 Ti GPU.

The training time of EmbNum+ is 29,732 seconds, while the training time of DSL is 2965 s. It is clear that EmbNum+ uses the deep learning approach and needs more time to train the similarity metric than DSL, which uses logistic regression. However, the similarity metric is only needed to train once, and it could be applied to other domains without retraining. The detailed experimental result on EmbNum+ robustness is reported in “[Semantic Labeling: Effectiveness](#)” section.

Semantic Labeling

In this section, we describe the detailed experimental setting to evaluate the semantic labeling task. We follow the evaluation setting of SemanticTyper [20] and DSL [19]. This setting is based on cross-validation, but it was modified to observe how

the number of numerical values in the knowledge base will affect the performance of the labeling process. The detail of the experimental setting is described as follows.

Suppose a dataset $S = \{s_1, s_2, s_3, \dots, s_d\}$ has d data sources. One data source was retained as the unknown data, and the remaining $d - 1$ data sources were used as the labeled data. We repeated this process d times, with each of the data source used exactly once as the unknown data.

Additionally, we set the number of sources in the labeled data increasing from one source to $d - 1$ sources to analyze the effect of an increment of the number of labeled data on the performance of semantic labeling. We obtained the MRR scores and labeling times on $d \times (d - 1)$ experiments and then averaged them to produce the $d - 1$ estimations of the number of sources in the labeled data.

Table 5 depicts the semantic learning setting with 10 data sources. From the first experiment to eighth experiment, s_1 is assigned as the queries of unknown sources, and the remaining sources are considered as the labeled sources in the knowledge base. We conducted a similar approach for the remaining experiments. Overall, we performed 90 experiments on the 10 sources of a dataset.

Unseen Semantic Labeling

In this section, we describe the setting of unseen semantic labeling. We split data to d partitions and used the setting of d -fold cross-validation for evaluation. To analyze the changing of EmbNum+ performance regarding the number of unseen semantic labels, we linearly increased the percentage of unseen semantic labels from 0 to 90% of all labels in knowledge bases. For details, Table 6 depicts the number of unseen semantic labels of City Data, Open Data, DBpedia NKB, and Wikidata NKB.

The performance is evaluated with the MRR score on the four datasets. When a query is an unseen attribute, the reciprocal rank (RR) is 1 if the ranking result is empty, and 0 otherwise.

Ablation Study

We also conducted ablation studies to evaluate the impact of the representation learning and the attribute augmentation on the task of semantic labeling. For the setting of EmbNum+ without using the representation learning, we created three methods which ignore the representation learning: *Num_l1*, *Num_l2*, and *Num_l∞*. The similarities between numerical attributes are directly calculated from the *tran(.)* without using the embedding model. The *Num_l1* used the Manhattan distance, *Num_l2* used the Euclidean distance, and *Num_l∞* used Chebyshev distance. For the setting of EmbNum+ without using the attribute augmentation, we call this method as EmbNum+ NonAu.

We conducted this ablation study based on d -fold cross-validation. Given a dataset with d data sources, one data source was retained as the query set, and the remaining $d - 1$ data sources were used as a knowledge base. We conducted semantic labeling for the query set on the knowledge base. We repeated this process d times, with each of the data source used exactly once as the unknown data.

Table 7 Semantic labeling in the MRR score of SemanticTyper [20], DSL [19], and EmbNum+ on City Data, Open Data, DBpedia NKB, and Wikidata NKB

Dataset	Method	Number of sources in labels sources in training data								
		1	2	3	4	5	6	7	8	9
DBpedia NKB	SemanticTyper [20]	0.7123	0.7804	0.8082	0.8239	0.831	0.8364	0.8452	0.8447	0.8488
	DSL [19]	0.626	0.7845	0.818	0.8279	0.8319	0.8356	0.8411	0.8424	0.8439
	EmbNum+	0.9183	0.9331	0.9389	0.944	0.9468	0.946	0.9496	0.9494	0.9483
City Data	SemanticTyper [20]	0.8318	0.8646	0.8799	0.8841	0.8888	0.8916	0.8974	0.8962	0.8993
	DSL [19]	0.8643	0.8893	0.8949	0.8997	0.9027	0.9108	0.9149	0.9074	0.9105
	EmbNum+	0.9165	0.923	0.9366	0.9357	0.9431	0.9453	0.9437	0.9497	0.9518
Wikidata NKB	SemanticTyper [20]	0.6734	0.7605	0.7912	0.8102	0.82	0.8263	0.8351	0.8453	0.8477
	DSL [19]	0.6085	0.7703	0.8037	0.8132	0.8141	0.8205	0.8268	0.829	0.835
	EmbNum+	0.8321	0.8637	0.8821	0.8924	0.8997	0.905	0.9077	0.9121	0.9175
Open Data	SemanticTyper [20]	0.3243	0.3715	0.4071	0.4257	0.4446	0.4564	0.4647	0.4687	0.4711
	DSL [19]	0.5385	0.5663	0.5844	0.5976	0.6067	0.6147	0.6202	0.6266	0.6319
	EmbNum+	0.6382	0.668	0.6807	0.6796	0.69	0.6863	0.6899	0.698	0.7093

Fig. 9 Semantic labeling in the MRR score on City Data, Open Data, DBpedia NKB, and Wikidata NKB

Experimental Results

In this section, we report the experimental results of semantic labeling in terms of effectiveness, robustness (“[Semantic Labeling: Effectiveness](#)” section), and efficiency (“[Semantic Labeling: Efficiency](#)” section). “[Unseen Semantic Labeling](#)” section reports the experimental result of the setting of unseen semantic labeling. Finally, we report the result of the ablation study in “[Ablation Study](#)” section.

Semantic Labeling: Effectiveness

We tested SemanticTyping [20], DSL [19], and EmbNum+ on the semantic labeling task using the MRR score to evaluate the effectiveness. The results are shown in Table 7 and Fig. 9.

Table 8 Paired sample t test between EmbNum+ and SemanticTyper, DSL on DBpedia NKB, City Data, Wikidata NKB, and Open Data

Dataset	SemanticTyper	DSL
DBpedia NKB	3.46e−6	1.40e−4
City Data	2.21e−5	5.99e−5
Wikidata NKB	1.31e−5	2.25e−4
Open Data	9.10e−9	4.27e−8

The MRR scores obtained by three methods steadily increase along with the number of labeled sources. It suggests that the more labeled sources in the database, the more accurate the assigned semantic labels are. DSL outperformed SemanticTyper in City Data and Open Data but was comparable with SemanticTyper in DBpedia NKB and Wikidata NKB. In the DBpedia NKB and Wikidata NKB, there are more semantic labels as well as a high level of range overlapping between numerical attributes; therefore, these features (KS Test, numerical Jaccard, and MW test) proposed by DSL become less effective.

EmbNum+ learned directly from the empirical distribution of numerical values without making any assumption on data type and data distribution and, hence, outperformed SemanticTyper and DSL on every dataset. The similarity metric based on a specific hypothesis test, which was used in SemanticTyper and DSL, is not optimized for semantic meanings with various data types and distributions in DBpedia NKB and Wikidata NKB.

The performance of semantic labeling systems is different according to datasets. In particular, semantic labeling on City Data, DBpedia NKB, and Wikidata NKB yields higher performance than Open Data. The performance differences occur because of data quality. The City Data, DBpedia NKB, and Wikidata NKB are synthesis data, where each numerical values of attributes are normalized in terms of data scaling. Open Data is the real-world data where we usually do not know the meaning of attributes; therefore, it is difficult to do normalization operation. In this paper, we do not tackle the issue of data scaling, it is left as our future work.

Although EmbNum+ was trained on 50% of DBpedia NKB, the performance of EmbNum+ consistent yields the best performance in the four datasets, especially the two different datasets: Wikidata NKB and Open Data. It means that EmbNum+ is promising for semantic labeling in a wide range of data domains.

To understand whether EmbNum+ does significantly outperform SemanticTyper and DSL, we performed a paired sample t test on the MRR scores between EmbNum+ and SemanticTyper, DSL. Table 8 shows the result of the paired t test on City Data, Open Data, DBpedia NKB, and Wikidata NKB. We set the cutoff value for determining statistical significance to 0.01. The results of the paired t test revealed that EmbNum+ significantly outperforms SemanticTyper and DSL on all four datasets (all the p values < 0.01).

Table 9 Run-time in seconds of semantic labeling of SemanticTyper [20], DSL [19], and EmbNum+ on City Data, Open Data, DBpedia NKB, and Wikidata NKB

Datasets	Methods	Number of sources in labels sources in training data								
		1	2	3	4	5	6	7	8	9
DBpedia NKB	SemanticTyper [20]	0.3241	0.3914	0.4645	0.5216	0.5942	0.6725	0.7506	0.8125	0.9069
	DSL [19]	0.9259	1.0972	1.2621	1.4398	1.5774	1.7269	1.8918	2.0245	2.1777
	EmbNum+	0.009	0.0178	0.0265	0.0354	0.044	0.0531	0.0617	0.0704	0.0787
City Data	SemanticTyper [20]	0.0296	0.0315	0.0337	0.0364	0.0388	0.0404	0.0429	0.045	0.0475
	DSL [19]	0.1803	0.1835	0.1829	0.1839	0.1898	0.1959	0.1999	0.2056	0.2074
	EmbNum+	0.0016	0.0027	0.0042	0.0055	0.0071	0.0091	0.011	0.0121	0.013
Wikidata NKB	SemanticTyper [20]	0.34	0.4539	0.5587	0.6919	0.8483	0.9895	1.0416	1.1243	1.2437
	DSL	1.4199	1.7652	1.9741	2.1329	2.5746	2.7915	2.9679	3.1139	3.3529
	EmbNum+	0.0091	0.0171	0.0256	0.0336	0.0417	0.0508	0.0595	0.0667	0.0741
Open Data	SemanticTyper [20]	0.1663	0.239	0.315	0.3929	0.4819	0.5571	0.638	0.7217	0.8063
	DSL [19]	0.5047	0.637	0.7708	0.9042	1.0378	1.1723	1.3113	1.445	1.5806
	EmbNum+	0.0028	0.0056	0.0092	0.0131	0.0158	0.0208	0.0225	0.0268	0.0292

Fig. 10 Run-time in seconds of semantic labeling on DBpedia NKB, City Data, Wikidata NKB, and Open Data

Semantic Labeling: Efficiency

In this experiment, we also used the same setting with the previous experiment but the efficiency is evaluated by the run-time of semantic labeling. Table 9 and Fig. 10 depict the run-time of semantic labeling of SemanticTyper, DSL, and EmbNum+ on the four dataset: DBpedia NKB, Wikidata NKB, City Data, and Open Data.

The run-time of semantic labeling linearly increases with the number of labeled sources. The run-time of DSL was extremely high when the number of labeled data sources increased because three similarity metrics were needed to be calculated. The run-time of SemanticTyper was less than DSL because it only used the KS test as a similarity metric. Semantic labeling with EmbNum+ is significantly faster than SemanticTyper (about 17 times) and DSL (about 46 times). EmbNum+ outperforms the baseline approaches in run-time since the similarity metric of EmbNum+ was calculated directly on extracted feature vectors instead of all original values.

Table 10 Unseen semantic labeling in the MRR score of EmbNum+ on DBpedia NKB, City Data, Wikidata NKB, and Open Data

Dataset	Methods	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%
DBpedia NKB	EmbNum+ NonRe	0.9483	0.8615	0.7679	0.6747	0.5774	0.4869	0.3935	0.2956	0.1977	0.1021
	EmbNum+	0.9483	0.8649	0.7748	0.688	0.5967	0.5179	0.4649	0.4079	0.411	0.4765
City Data	EmbNum+ NonRe	0.9518	0.8674	0.7707	0.6725	0.5696	0.4809	0.3944	0.2983	0.1983	0.1
	EmbNum+	0.9518	0.8774	0.8107	0.7225	0.6762	0.6409	0.6278	0.6217	0.645	0.7867
Wikidata NKB	EmbNum+ NonRe	0.9175	0.8383	0.749	0.6599	0.5621	0.4793	0.3831	0.2913	0.1972	0.0982
	EmbNum+	0.9175	0.8401	0.7573	0.6723	0.5792	0.5095	0.4293	0.383	0.3807	0.4905
Open Data	EmbNum+ NonRe	0.7093	0.6458	0.5777	0.5277	0.4524	0.3811	0.3209	0.2559	0.1663	0.0882
	EmbNum+	0.7093	0.6478	0.5977	0.5557	0.4844	0.4431	0.4169	0.4379	0.4063	0.5202

Fig. 11 Unseen semantic labeling in the MRR score of EmbNum+ on DBpedia NKB, City Data, Wikidata NKB, and Open Data

Table 11 Ablation study result of EmbNum+ on City Data, Open Data, DBpedia NKB, and Wikidata NKB

Methods	City Data	Open Data	DBpedia NKB	Wikidata NKB
Num_I1	0.9320	0.6972	0.9026	0.8934
Num_I2	0.9225	0.6891	0.8964	0.8919
$Num_I\infty$	0.9090	0.6721	0.8879	0.8985
EmbNum+ NonAu	0.9412	0.6998	0.9165	0.9085
EmbNum+	0.9518	0.7093	0.9483	0.9175

Unseen Semantic Labeling

In this section, we report the experiment results of unseen semantic labeling. Table 10 and Fig. 11 report the MRR score of EmbNum+ when using (EmbNum+) and not using (EmbNum+ NonRe) relevance model. When the number of unseen

semantic labels increases, the performance of semantic labeling decreases if we do not use the relevance model. When we use the relevance model, the performance of EmbNum+ changes considerably.

Interestingly, the trend of MRR score changed from decreasing to increasing at 80% unseen semantic labels in knowledge bases. This result is promising in practice since we usually do not have so many labeled data. Detecting unseen semantic labels assists domain experts in terms of simplifying and reducing the time for the manually labeling process.

Ablation Study

Table 11 reports the ablation study of EmbNum+ on City Data, Open Data, DBpedia NKB, and Wikidata NKB.

The method of *Num_l1*, *Num_l2*, *Num_l∞* is EmbNum+ without using the representation learning. Among these methods of EmbNum+ without using representation learning, *Num_l1* outperforms *Num_l2*, *Num_l∞*. It indicates that the Manhattan distance has more advantages than Euclidean and Chebyshev distance. By removing the representation learning, we can see the MRR score is significantly reduced in the four datasets. This validates our assumption that the representation learning is a necessary module in the semantic labeling procedure.

EmbNum+ NonAu is a system of EmbNum+ without using the attribute augmentation module. The performance of EmbNum+ is higher than the EmbNum+ NonAu; therefore, we verify that the module of attribute augmentation is necessary for our proposed approach.

Conclusion

This work introduces EmbNum+, a semantic labeling method that annotates semantic labels for numerical attributes of tabular data. EmbNum+ advances other baseline approaches in many ways. First, EmbNum+ learns directly from numerical attributes without making any assumption regarding data; therefore, it is particularly useful to apply on general data when we do not have any knowledge about data domain and distribution. Second, EmbNum+ significantly boosts the efficiency against other baseline approaches because all the calculations are performed on the representation of numerical attributes. Third, EmbNum+ can recognize the unseen query which is extremely helpful in practice. Additional, we also introduced two new synthesis datasets, i.e., Wikidata NKB and DBpedia NKB, and one real-world dataset, i.e., Open Data. These datasets will be useful to evaluate the scalability as well as the robustness of semantic labeling for numerical values.

In future work, we plan to extend our work in the two directions. The first direction is to extend the similarity metric to interpret multiple scales. In this study, we assumed that two numerical attributes are similar when they are expressed on the same scale. In fact, two attributes with the same meaning could be measured using different scales. For instance, the numerical attributes “human

height” could be expressed in “centimeters” or “feet.” The second direction is to extend the metric to interpret the hierarchical representation of numerical data. The current presentation using the Euclidean distance cannot reflect the hierarchical structure. Building a similarity metric that is hierarchy aware can help to make a more fine-grained semantic labeling.

Acknowledgements We thank Tony Ribeiro and Patrik Schneider for their helpful comments. We are extremely grateful to the reviewers for their comments and feedback.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

Appendix A: Dataset Analysis

In this section, we illustrate the quantile range distribution of numerical attributes in City Data, Open Data, Wikidata NKB, and DBpedia NKB (Figs. 12, 13, 14, 15).

Fig. 12 Quantile range distribution of numerical attributes in City Data

Fig. 13 Quantile range distribution of numerical attributes in Open Data

Fig. 14 Quantile range distribution of numerical attributes in Wikidata NKB

Fig. 15 Quantile range distribution of numerical attributes in DBpedia NKB

References

- Adelfio, M.D., Samet, H.: Schema extraction for tabular data on the web. *Proc. VLDB Endow.* **6**(6), 421–432 (2013). <https://doi.org/10.14778/2536336.2536343>
- Ahmadov, A., Thiele, M., Eberius, J., Lehner, W., Wrembel, R.: Towards a hybrid imputation approach using web tables. In: 2015 IEEE/ACM 2nd International Symposium on Big Data Computing (BDC), pp. 21–30 (2015). <https://doi.org/10.1109/BDC.2015.38>
- Bengio, Y.: Learning deep architectures for AI. *Found. Trends® Mach. Learn.* **2**(1), 1–127 (2009)

4. Chen, W., Chen, X., Zhang, J., Huang, K.: Beyond triplet loss: a deep quadruplet network for person re-identification. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1320–1329 (2017). <https://doi.org/10.1109/CVPR.2017.145>
5. Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmman, T., Sun, S., Zhang, W.: Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 601–610. ACM (2014)
6. Ermilov, I., Auer, S., Stadler, C.: User-driven semantic mapping of tabular data. In: Proceedings of the 9th International Conference on Semantic Systems, I-SEMANTICS '13, pp. 105–112. ACM, New York, NY, USA (2013). <https://doi.org/10.1145/2506182.2506196>
7. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27–30, 2016, pp. 770–778 (2016). <https://doi.org/10.1109/CVPR.2016.90>
8. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. In: Proceedings of the 32nd International Conference on International Conference on Machine Learning—Volume 37, ICML'15, pp. 448–456. JMLR.org (2015)
9. Lehmann, E.L., Romano, J.P.: Testing Statistical Hypotheses. Springer Texts in Statistics, third edn. Springer, New York (2005). <https://doi.org/10.1007/0-387-27605-X>
10. Lehmberg, O., Ritze, D., Meusel, R., Bizer, C.: A large public corpus of web tables containing time and context metadata. In: Proceedings of the 25th International Conference Companion on World Wide Web, WWW '16 Companion, pp. 75–76. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland (2016). <https://doi.org/10.1145/2872518.2889386>
11. Lehmberg, O., Ritze, D., Ristoski, P., Meusel, R., Paulheim, H., Bizer, C.: The mannheim search join engine. *Web Semant.* **35**(P3), 159–166 (2015). <https://doi.org/10.1016/j.websem.2015.05.001>
12. Mitlöhner, J., Neumaier, S., Umbrich, J., Polleres, A.: Characteristics of open data csv files. In: 2016 2nd International Conference on Open and Big Data (OBD), pp. 72–79 (2016). <https://doi.org/10.1109/OBD.2016.18>
13. Nargesian, F., Zhu, E., Pu, K.Q., Miller, R.J.: Table union search on open data. *Proc. VLDB Endow.* **11**(7), 813–825 (2018). <https://doi.org/10.14778/3192965.3192973>
14. Neuhäuser, M.: Wilcoxon–Mann–Whitney Test, pp. 1656–1658. Springer, Berlin (2011). https://doi.org/10.1007/978-3-642-04898-2_615
15. Neumaier, S., Umbrich, J., Parreira, J.X., Polleres, A.: Multi-level semantic labelling of numerical values. In: Groth, P., Simperl, E., Gray, A., Sabou, M., Krötzsch, M., Lecue, F., Flöck, F., Gil, Y. (eds.) *The Semantic Web—ISWC 2016*, pp. 428–445. Springer, Cham (2016)
16. Nguyen, P., Nguyen, K., Ichise, R., Takeda, H.: Embnum: semantic labeling for numerical values with deep metric learning. In: Ichise, R., Lecue, F., Kawamura, T., Zhao, D., Muggleton, S., Kozaki, K. (eds.) *Semantic Technology*, pp. 119–135. Springer, Cham (2018)
17. Nguyen, T.T., Nguyen, Q.V.H., Weidlich, M., Aberer, K.: Result selection and summarization for web table search. In: 2015 IEEE 31st International Conference on Data Engineering, pp. 231–242 (2015). <https://doi.org/10.1109/ICDE.2015.7113287>
18. Nishida, K., Sadamitsu, K., Higashinaka, R., Matsuo, Y.: Understanding the semantic structures of tables with a hybrid deep neural network architecture. In: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4–9, 2017, San Francisco, California, USA, pp. 168–174 (2017)
19. Pham, M., Alse, S., Knoblock, C.A., Szekely, P.: Semantic labeling: a domain-independent approach. In: *International Semantic Web Conference*, pp. 446–462. Springer (2016)
20. Ramnandan, S.K., Mittal, A., Knoblock, C.A., Szekely, P.: Assigning semantic labels to data sources. In: *European Semantic Web Conference*, pp. 403–417. Springer (2015)
21. Ritze, D., Lehmberg, O., Bizer, C.: Matching html tables to dbpedia. In: *Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics, WIMS '15*, pp. 10:1–10:6. ACM, New York, NY, USA (2015). <https://doi.org/10.1145/2797115.2797118>
22. Schroff, F., Kalenichenko, D., Philbin, J.: Facenet: A unified embedding for face recognition and clustering. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 815–823 (2015). <https://doi.org/10.1109/CVPR.2015.7298682>
23. Sekhavat, Y.A., Paolo, F.D., Barbosa, D., Meriardo, P.: Knowledge base augmentation using tabular data. In: LDOW, CEUR Workshop Proceedings, vol. **1184**. CEUR-WS.org (2014)

24. Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., Lecun, Y.: Overfeat: Integrated recognition, localization and detection using convolutional networks. In: International Conference on Learning Representations (ICLR2014), CBLS, April 2014 (2014)
25. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: International Conference on Learning Representations (2015)
26. Stonebraker, M., Bruckner, D., Ilyas, I.F., Beskales, G., Cherniack, M., Zdonik, S.B., Pagan, A., Xu, S.: Data curation at scale: the data tamer system. In: CIDR (2013)
27. Taigman, Y., Yang, M., Ranzato, M., Wolf, L.: Deepface: Closing the gap to human-level performance in face verification. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1701–1708 (2014). <https://doi.org/10.1109/CVPR.2014.220>
28. Venetis, P., Halevy, A., Madhavan, J., Paşca, M., Shen, W., Wu, F., Miao, G., Wu, C.: Recovering semantics of tables on the web. *Proc. VLDB Endow.* **4**(9), 528–538 (2011). <https://doi.org/10.14778/2002938.2002939>
29. Wang, J., Wang, H., Wang, Z., Zhu, K.Q.: Understanding tables on the web. In: Atzeni, P., Cheung, D., Ram, S. (eds.) *Conceptual Modeling*, pp. 141–155. Springer, Berlin (2012)
30. Wikipedia contributors: Inverse transform sampling—Wikipedia, the free encyclopedia (2018). [Online; accessed 3 July 2018]
31. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) *Computer Vision—ECCV 2014*, pp. 818–833. Springer, Cham (2014)
32. Zhang, M., Chakrabarti, K.: Infogather+: semantic matching and annotation of numeric and time-varying attributes in web tables. In: *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pp. 145–156. ACM (2013)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Phuc Nguyen received a master's degree in computer science from UIT, Vietnam National University, HCMC, Vietnam, 2016. His research background is about data extraction and integration. He is currently a Ph.D. student at the Graduate University for Advanced Studies, Japan and a research assistant at the National Institute of Informatics, Japan.

Khai Nguyen's research background is ontology, linked data, data mining, ML, and NLP. He received a Ph.D. degree in Informatics from The Graduate University for Advanced Studies (Japan) in 2016. From 2017 to early 2019, he worked at the National Institute of Informatics (Japan) as a researcher and conducted research in ontology integration. He is currently a solution architect in an AI-technology company.

Ryutaro Ichise received his Ph.D. degree in computer science from Tokyo Institute of Technology, Tokyo, Japan, in 2000. From 2001 to 2002, he was a visiting scholar at Stanford University. He is currently an associate professor in the Principles of Informatics Research Division at the National Institute of Informatics in Japan. His research interests include Semantic Web, machine learning, and data mining.

Hideaki Takeda, Dr. Eng. is a professor of National Institute of Informatics (NII) and a professor of Graduate University for Advanced Studies (SOKENDAI). He received Dr. Eng. degrees from the University of Tokyo, Japan in 1991. He has been the Sumitomo endowed professor in the University of Tokyo between 2005 and 2010. His research interests are in Semantic Web, Web Informatics, Knowledge-based Systems and Scholarly Communication.