mathematics

MDPI

Article

SNMatch: An Unsupervised Method for Column Semantic-Type Detection Based on Siamese Network

Tiezheng Nie, Hanyu Mao *, Aolin Liu, Xuliang Wang, Derong Shen and Yue Kou

School of Computer Science and Engineering, Northeastern University, Shenyang 110169, China; nietiezheng@mail.neu.edu.cn (T.N.); 2210713@stu.neu.edu.cn (A.L.); xuliang.wang@uwaterloo.ca (X.W.); shendr@mail.neu.edu.cn (D.S.); kouyue@mail.neu.edu.cn (Y.K.)

* Correspondence: 2010676@stu.neu.edu.cn

Abstract: Column semantic-type detection is a crucial task for data integration and schema matching, particularly when dealing with large volumes of unlabeled tabular data. Existing methods often rely on supervised learning models, which require extensive labeled data. In this paper, we propose SNMatch, an unsupervised approach based on a Siamese network for detecting column semantic types without labeled training data. The novelty of SNMatch lies in its ability to generate the semantic embeddings of columns by considering both format and semantic features and clustering them into semantic types. Unlike traditional methods, which typically rely on keyword matching or supervised classification, SNMatch leverages unsupervised learning to tackle the challenges of column semantic detection in massive datasets with limited labeled examples. We demonstrate that SNMatch significantly outperforms current state-of-the-art techniques in terms of clustering accuracy, especially in handling complex and nested semantic types. Extensive experiments on the MACST and VizNet-Manyeyes datasets validate its effectiveness, achieving superior performance in column semantic-type detection compared to methods such as TF-IDF, FastText, and BERT. The proposed method shows great promise for practical applications in data integration, data cleaning, and automated schema mapping, particularly in scenarios where labeled data are scarce or unavailable. Furthermore, our work builds upon recent advances in neural network-based embeddings and unsupervised learning, contributing to the growing body of research in automatic schema matching and tabular data understanding.

Keywords: data integration; tabular data; column matching; unsupervised learning; Siamese network

MSC: 68M99

Academic Editor: Chengjie Sun

Received: 7 January 2025
Revised: 5 February 2025
Accepted: 10 February 2025
Published: 13 February 2025

Citation: Nie, T.; Mao, H.; Liu, A.; Wang, X.; Shen, D.; Kou, Y. SNMatch: An Unsupervised Method for Column Semantic-Type Detection Based on Siamese Network. *Mathematics* **2025**, *13*, 607. https://doi.org/10.3390/math13040607

1. Introduction

As an important task of data preparation and data integration, schema matching [1] relies on aligning columns of structured tabular data based on the semantics of columns. However, missing semantic types of table columns is one of the crucial problems for tabular data [2]. This problem significantly increases the difficulty of schema matching in data integration. To make use of unlabeled tabular data, it is required to cluster columns of different tables into groups by identifying their semantic types, and each group can be mapped to an attribute of the potential global schema. Therefore, distinguishing the semantic type corresponding to the column in tabular data becomes an important task [3], which is called column semantic-type detection.

The basic method of column semantic-type detection is based on knowledge matching, combined with keyword extraction and regular expression technology [3]. By using a large knowledge base to match the cell content with the entity, entities are mapped to the semantic type of the column. Some research works tried to apply statistical machine learning models to column semantic-type detection [4–6]. In recent years, scholars have begun to apply neural networks and pre-trained models to column semantic-type detection and have achieved good results [7–11]. Existing learning-based methods transform the column semantic-type detection into a multi-classification task. However, clustering methods are more suitable for detecting the semantic type of columns since classification methods always require supervised learning models in which labeled data are hard to obtain from massive tabular data.

This paper proposes an unsupervised clustering method for column semantic-type detection based on Siamese network, SNMatch. In this method, the neural network-based column text-embedding method is adopted, which clusters column texts through semantic types. By making full use of structured data, Siamese networks [12] and cell pair semantic type consistency judgment tasks are used to adjust the weight. In order to comprehensively consider the format and semantic characteristics of the cell text, this paper also proposes a unique encoder structure, which uses a character-level input bidirectional Gate Recurrent Unit (GRU) and self-attention mechanism networ, and introduces the semantic feature compensation of the cell through the FastText [13] pre-trained word vector. Finally, the training data of the neural network are based on the non-negative PU risk estimator proposed by Kiryo et al. [14].

The contribution of this paper is mainly shown as follows:

(1) An unsupervised schema matching method called SNMatch is proposed for column semantic-type matching on unlabeled tabular data.
(2) A cell text encoder and a column text-embedding method for column text clustering by semantic type are proposed, which consider cell format features and semantic features.
(3) We combine PU learning technology into the model of column semantic-type detection.
(4) We show that SNMatch achieves better performance than existing methods on column semantic-type detection tasks without training data.

The rest of this paper is organized as follows: Section 2 discusses existing related work. Section 3 gives the data model and formal problem definition. Section 4 shows the model proposed in this paper in detail and introduces the data and evaluation methods used in this paper, and the experimental results are given in Section 5. Finally, this paper is concluded in Section 6.

2. Related Works

2.1. Column Semantic-Type Detection

Knowledge-based methods were first proposed for column semantic-type detection. Venetis et al. [15] first used a large knowledge base to perform column semantic-type detection tasks and determined the semantic type of the column with a majority vote. Deng et al. [16] used MapReduce to extend the knowledge base matching method to a distributed environment and used fuzzy matching instead of exact matching to alleviate the problem of a knowledge gap. They also pointed out the deficiencies of the majority vote and proposed a similarity function as a substitute. An et al. [3] not only use the cell content but also use each independent token in the cell as a query, combining with predefined regular expressions to obtain semantic types from the knowledge base. Then, other works use statistical machine learning methods to solve column semantic-type detection. The most representative of such methods is the work of Limaye et al. [4]. Bhagavatula et al. [6] focused on calibrating corresponding entities for cell values. They use a dense Markov

network to capture the relationship between a cell value and all other cell values in the same column and the same line.

Recently, neural networks have been used for column semantic-type detection. Chen et al. [7] combined transfer learning technology to build a training set from the knowledge base and trained a classifier based on a convolution neural network (CNN). Hulsebos et al. [8] regarded column semantic-type detection as a multi-classification task, manually extracting column characters, words, cell text features, and using a fully connected feedforward neural network to complete the training, from features to semantic types. Zhang et al. [9] used the CRF model to further capture the relevance of the semantic types of columns appearing in a table. They also further improved the performance of the model by introducing table intents. Xie et al. [10] trained a CNN semantic-type classifier suitable for Chinese tables and used the content of the "info box" searched in the Baidu Encyclopedia table as the input for the classifier. In the realm of table understanding, transformer-based models have gained prominence. Yang et al. proposed TableFormer [17], a robust transformer model for table-text encoding that incorporates tabular structural biases through learnable attention mechanisms. This approach ensures invariance to row and column orders and enhances table-text understanding, outperforming strong baselines on multiple datasets. Furthermore, Maji et al. introduced DCoM [18], a collection of multi-input NLP-based deep neural networks designed to detect semantic data types by feeding raw column values as text inputs. Trained on a substantial dataset, DCoM outperforms contemporary methods, demonstrating the effectiveness of leveraging raw data inputs for semantic-type detection.

2.2. Semantic Features

Until recent years, pre-trained models were widely used in the task of column semantic-type detection. These works introduced the concept of "embedding" into structured data. Some studies directly used the word-embedding algorithm of unstructured text to train the embedding of words in structured data [19]. However, pre-trained word embedding in general text is not completely suitable for expressing the relationship between words in structured text. In this process, the work of Fernandez et al. [20] extracted the entity–attribute–attribute ternary relationship from structured and unstructured datasets and used the Siamese network to optimize the correlation judgment task. Cappuzzo et al. [21] used graphs to abstract the relationship between tuples, columns, and cells in structured data. They used the random walk algorithm to complete the vectorized representation of the nodes in the graph and generated a universal structured data cell embedding. In [22], a multi-task learning framework based on pre-trained language models takes the entire table as input and predicts column types/relations using a single model. And the transformer models are used in latest works [23,24] to encode the cell content of columns. Different from the above existing methods, the unsupervised method proposed in this paper can handle a large number of out-of-vocabulary (OOV) words and make full use of the format features and semantic features in the cell text.

3. Problem Definition

In this paper, we regard tabular data as relational tables. Let the set of relational tables be T, and $t_i \in T(1 \leq i \leq |T|)$ is the i^{th} relational table in the set. Consider a relational table as a collection of columns, and $C_j \in t_i(1 \leq j \leq |t_i|)$ is the j^{th} column in the relation table t_i. Think of the column as a collection of cells, and $C_k \in C_j\left(1 \leq k \leq \left|C_j\right|\right)$ is the k^{th} cell in the column C_j. The j^{th} column of the i^{th} relational table in T is abbreviated as T_i^j, and the k^{th} cell in this column is abbreviated as $T_i^{j,k}$.

Definition 1. *Let* $\{G_1, G_2, \ldots, G_n\}$ *be n semantic-type clusters, and the task of column semantic-type detection is equal to the mapping columns of table T into clusters:*

$$\left\{ T_i^j \middle| 1 \le i \le |T|,\ 1 \le j \le |t_i| \right\} \rightarrow \{G_1, G_2, \ldots, G_n\}^{\sum_{i=1}^{|T|} |t_i|} \tag{1}$$

In this paper, the mapping operation in Equation (1) is divided into two steps: the extracting features of column data and clustering columns. As shown in Figure 1, two tables map their columns into five potential clusters. The first step is to extract the features of column data by encoding columns, and the second step is to cluster columns with a model.

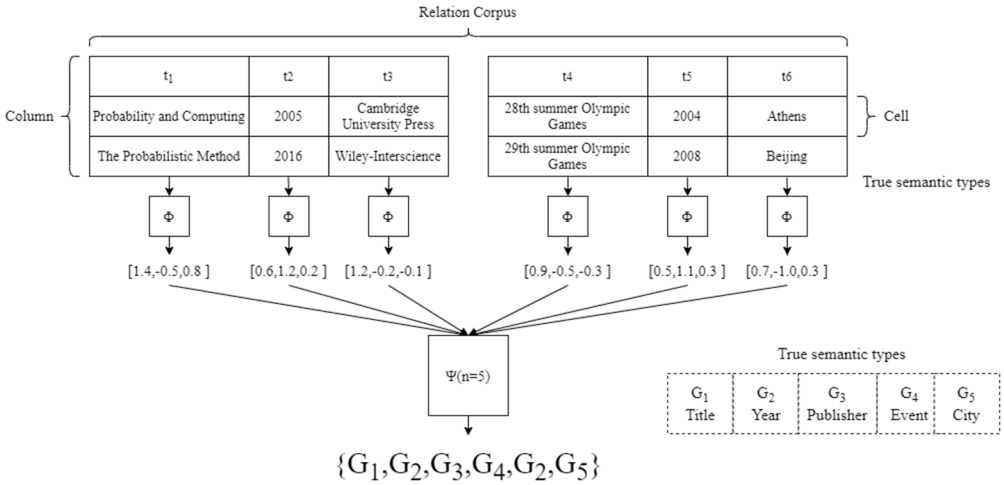

Figure 1. Two steps of column semantic−type detection.

Definition 2. *Let* Φ *be the feature extractor of the column data, which executes the mapping of the column text to the feature vector with a fixed length of d, where* $\mathbb{R}$ *is the real number field:*

$$\Phi \ : \ T_i^j \rightarrow \mathbb{R}^d \tag{2}$$

Definition 3. *Let* Ψ *be the clustering algorithm to complete the mapping of the column feature vector to the cluster label, where* $V_i^j \in \mathbb{R}^d$ *represents the result of column* T_i^j *being mapped by* Φ*:*

$$\Psi \ : \ \left\{ V_i^j \middle| 1 \le i \le |T|,\ 1 \le j \le |t_i| \right\} \rightarrow \{G_1, G_2, \ldots, G_n\}^{\sum_{i=1}^{|T|} |t_i|} \tag{3}$$

Therefore, the SNMatch proposed in this paper focuses on proposing an unsupervised method for constructing the column data feature extractor Φ. The distance between any two column feature vectors generated by Φ should be able to reflect the relevance of the column in the semantic type. And with using the clustering algorithm Ψ, the columns of tabular data are clustered into $\{G_1, G_2, \ldots, G_n\}$ by the feature vector of the columns.

4. Methodology

4.1. The Architecture of SNMatch

In order to handle the tabular data without column labels, this paper proposes an unsupervised method, SNMatch, to vectorize column text by constructing a Siamese network and cluster columns into semantic types. Figure 2 shows the architecture of SNMatch. The network weight is optimized by the binary classification task of the cell pair

semantic-type consistency judgment task. After training, the cell text encoder can be used to generate column text feature vectors in the Column Encoder. Then, a general clustering algorithm can be used to complete the clustering of column feature vectors where each cluster of columns maps to a semantic type.

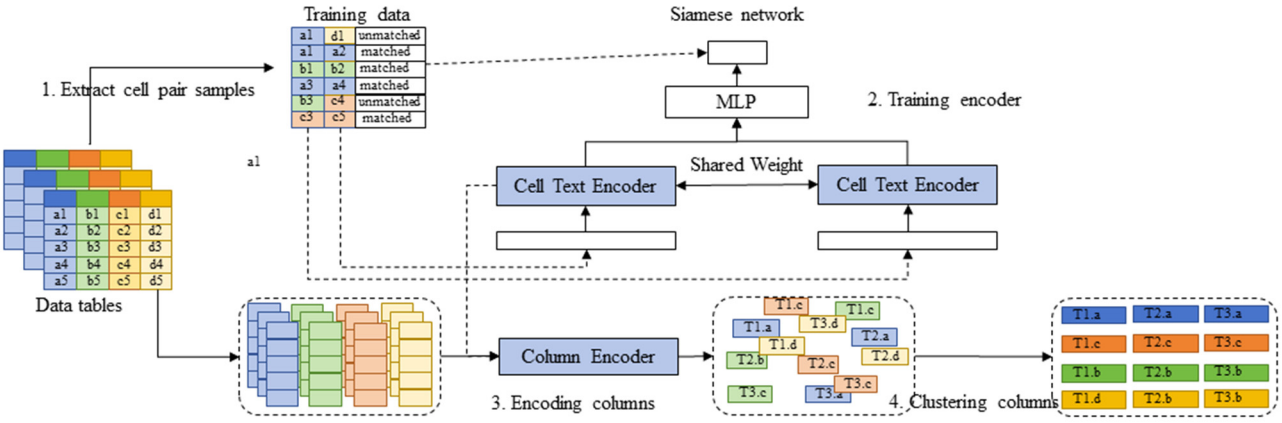

Figure 2. The architecture of SNMatch.

4.2. Training Data Generation

The first step of SNMatch is to extract cell pairs as the training data by an unsupervised method, since there are no labeled data in massive web tables. To generate labeled training data, we use the structured features of tables and the non-negative PU risk estimator. In relational data, all cells in the same column have the same semantic type. We generate training data based on this insight.

For example, as shown in Table 1, there are five semantic types in the relational dataset: name, post, email, phone number and available. Since the semantic types of all the cells in a column are the same, two cell values arbitrarily taken from the same column can be labeled with "matched" as a positive sample of training data. Such as cell pair <"Nick Bayley", "Arthur Fritch"> will be labeled "matched" since they are from the same column, "name". For cells from different columns, the semantic types they belong to may be the same or inconsistent. Therefore, taking a cell value from each of the two columns will form an unlabeled sample. Such as the two cell values "Patrick Rutherford" and "Team Leader" are extracted from the "name" column and the "post" column, respectively, an unlabeled sample <"Patrick Rutherford", "Team Leader", "?"> is generated.

Table 1. Example of table data.

Name	Post	Email	Phone Number	Available
Nick Bayley	HR Assistant	nickbaley@fmb.com	123-45678	Mon.–Fri.
Arthur Fritch	Program Manager	arthur@fmb.com	135-79864	Mon.–Sat.
Patrick Rutherford	Team Leader	prutherford@fmb.com	246-80975	Mon.–Sat.

Now, we can extract positive samples and unlabeled samples extracted from T, with which the training loss can be calculated through the non-negative PU risk estimator:

$$\hat{L}_{PU}(\sigma) = \pi_p \hat{L}_p^1(\sigma) + \max\left\{0, \hat{L}_p^0(\sigma) - \pi_p \hat{L}_p^0(\sigma)\right\} \tag{4}$$

The essence of this method is to use the distribution of a known unlabeled sample and the distribution of a positive sample to estimate the loss caused by the unknown negative sample:

$$\pi_n L_n^0(\sigma) = L_u^0(\sigma) - \pi_p L_p^0(\sigma) \tag{5}$$

Calculate loss $L_u^0(\sigma)$ by treating unlabeled samples as negative samples. The unlabeled samples contain positive samples and negative samples, so the part $\pi_p L_p^0(\sigma)$ calculated with the positive sample as the negative sample is subtracted, and the remaining part is the estimated loss $\pi_n L_n^0(\sigma)$ of the negative samples in the training set. Here, $L_p^0(\sigma)$ can be estimated using known positive samples. The non-negative PU risk estimator treats all unlabeled samples as negative samples to calculate the loss. Therefore, when extracting PU training data from T, we can directly set the unlabeled sample label to "unmatched", that is, <"Patrick Rutherford", "Team Leader", "?"> is actually recorded as <"Patrick Rutherford", "Team Leader", "unmatched">. In this way, training data are extracted from T with an unsupervised method.

4.3. The Siamese Network

In SNMatch, the Siamese network is designed to construct the data feature extractor Φ, and the matching semantic type of a cell pair is formulated to complete the training of the encoder. The structure of the Siamese network is shown in the upper right of Figure 2, in which the cell text encoders partly share weights. The encoders encode the input cell text into a fixed-length vector. After the two feature vectors are spliced, they are input into the upper part of the Multi-Layer Perception (MLP), and whether they belong to the same semantic type of the binary classification task result is obtained. After training with generated samples in Section 4.2, the cell text encoder in the Siamese network can be used to encode cell text. Let ψ represent the cell text encoder, and ψ can map the cell T_i^j to a feature vector $T_i^{j,k} \in \mathbb{R}^d$:

$$\psi : T_i^{j,k} \rightarrow \mathbb{R}^d \tag{6}$$

Then, we use the average of all cell feature vectors in a column as the semantic embedding of the column:

$$\Phi\left(T_i^j\right) = \frac{1}{\left|T_i^j\right|} \sum_{k=1}^{|T_i^j|} \psi\left(T_i^{j,k}\right) \tag{7}$$

The advantage of using cells instead of columns as distinguishing objects of semantic types is that the relationship between values and semantic types is preserved. The encoder can achieve more effective and more significant features for distinguishing the semantic types of cells.

4.4. Cell Text Encoder

In the Siamese network, the cell text encoder $\phi\backslash$phi maps the cell text into a fixed-length vector that reflects the semantic characteristics of the column. While pre-trained word-embedding methods are commonly used for vectorizing cell text, they have limitations in representing structured data: (1) handling out-of-vocabulary (OOV) words is challenging; (2) semantic connections between words in structured data are often inadequately captured; and (3) structural information of tables is ignored.

To address these issues, the cell text encoder proposed in this paper incorporates both format and semantic features. The encoder structure is shown in Figure 3, where the input cell text is processed into two types of inputs: (1) a character sequence encoded as integer values, including space characters for format information; and (2) a word sequence embedded using the pre-trained FastText model, which captures semantic features.

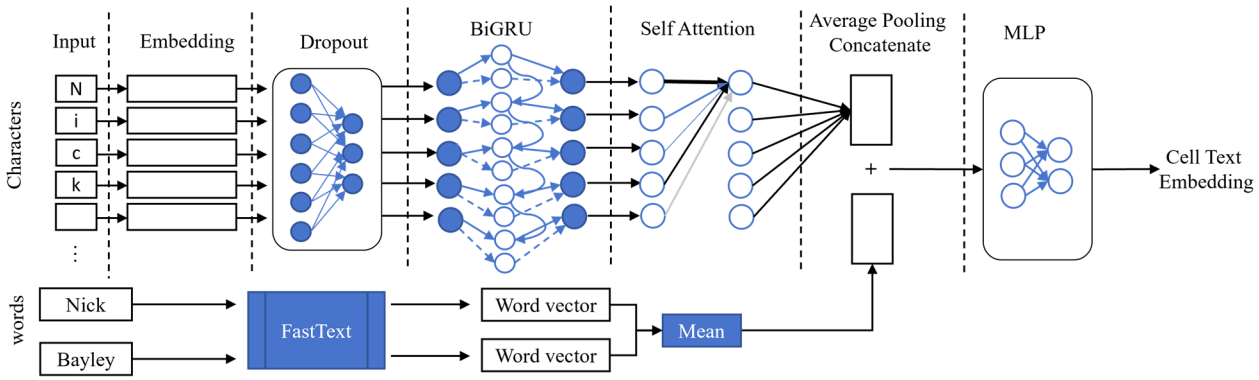

Figure 3. The structure of the cell text encoder.

Quantitative Validation of Hierarchical Type Detection

To evaluate the encoder's effectiveness in detecting hierarchical semantic types, we analyze its performance on categories such as "Location," "City," and "Hometown". These types exhibit a nested hierarchy, where "City" is a subset of "Location". Our experimental results demonstrate that the encoder can distinguish higher-level categories (e.g., "Location") with high accuracy, while the distinction between nested types (e.g., "City" vs. "Hometown") is more challenging due to overlapping semantic features. The macro and micro F1 scores in Table 3 confirm the encoder's ability to capture hierarchical relationships.

Mathematical Formalization of Hierarchical Relationships

The hierarchical relationships among semantic types are represented as a tree structure $H = (N, E)$, where N denotes the nodes (semantic types), and E denotes the parent–child relationships. The encoder considers these relationships during clustering by incorporating a hierarchical distance metric:

$$D_H(C_i, C_j) = \alpha \cdot D_{semantic}(C_i, C_j) + \beta \cdot D_{jierarchical}(C_i, C_j)$$

where $D_{semantic}$ measures the semantic similarity between columns C_i and C_j, and $D_{jierarchical}$ quantifies their relative positions in the hierarchy. The weights α and β control the contribution of semantic and hierarchical features, respectively.

Performance Impact Analysis

The encoder's ability to capture hierarchical relationships was further analyzed by comparing its performance on nested semantic categories with and without hierarchical features. Ablation studies show that removing hierarchical features reduces clustering accuracy, particularly for deeply nested types. For instance, distinguishing between "City" and "Hometown" becomes significantly harder without hierarchical features, as illustrated in Table 6. This highlights the importance of incorporating hierarchical relationships into the encoder design.

By combining character-level format features, semantic embeddings, and hierarchical relationships, the proposed encoder achieves robust performance across a wide range of semantic types, including nested categories.

4.5. Clustering Column Semantic Types

The final step is clustering the column semantic types by column vectors. A clustering algorithm, such as K-means++, is used to complete the clustering of column feature vectors, where each cluster of columns maps to a semantic type. For large-scale implementations, the asymptotic complexity of the clustering algorithm is $O(knd)$, where k represents the number of clusters, n is the number of columns, and d is the dimensionality of the feature vectors. Given that the number of columns and their dimensionality can grow considerably,

we propose a memory optimization framework based on mini-batch K-means to reduce memory consumption during clustering. This technique processes data in small batches, significantly improving scalability without compromising the clustering quality. Moreover, to ensure the robustness of the model, the neural architecture consists of a Siamese network with a BiGRU-based cell encoder, which optimizes feature extraction while maintaining manageable model size. Training convergence is monitored through the evaluation of loss reduction over epochs. Empirical results indicate that the model converges within 50 epochs with a decrease in loss of at least 20%, ensuring fast training times and efficient convergence under typical dataset sizes.

Clustering Parameter Optimization Methodology

To optimize the clustering process, we conducted experiments to tune the number of clusters k, the initialization method, and the distance metric used in the K-means++ algorithm. The optimal number of clusters k was determined by using the **elbow method** on the dataset, where we measured the within-cluster sum of squares (WCSS) for different values of k and selected the point where the rate of decrease slowed down. Additionally, we tested various distance metrics such as Euclidean, cosine, and Manhattan distance, and found that the Euclidean distance provided the best performance in clustering column embeddings.

Comparative Analysis of Clustering Algorithms

While K-means++ was the primary clustering algorithm used, we also compared its performance with other popular clustering algorithms, including **DBSCAN** and **Agglomerative Clustering**. DBSCAN, which is density-based, was particularly effective in handling outliers in smaller datasets, while Agglomerative Clustering showed potential in capturing hierarchical relationships within data. However, K-means++ outperformed both algorithms in terms of clustering accuracy, as indicated by the silhouette scores in Table 4. This suggests that for our task, K-means++ is the most appropriate clustering method.

5. Experiments

5.1. Datasets

The experimental datasets used in this paper include the MACST and the Manyeyes part of the VizNet dataset [25]. The MACST is a new relational table with column semantic type. We extract MACST from the Web Table Corpora dataset [26] by manual annotation, called the Manually Annotated Column Semantic Type (MACST). The MACST contains 362 selected tables with relabelled columns and includes 58 column semantic types. In the VizNet-Manyeyes dataset, the processed 2821 tables contain 78 column semantic types. Both datasets delete all the columns whose titles are not in the standard column semantic-type set, and keep the tables whose scale is still larger than three rows and three columns. In this process, one semantic type no longer appears, and 77 semantic types remain. Table 2 shows the statistics of two datasets used in this paper and their statistics.

Table 2. Statistics of datasets.

Name	Table	Columns	Cells	Semantic Types	SD of Semantic-Type Count
MACST	362	1191	50,311	58	25.08
VizNet-Manyeyes	2821	9469	1,333,693	77	184.74

5.2. Baselines

In experiments, we take the original column headings of the table as the ground-truth of the semantic type. In order to evaluate the performance of SNMatch, we selected the

following multi-classification methods for comparison: LDA, BTM, TF-IDF, pre-trained FastText, and pre-trained BERT. LDA, BTM, and TF-IDF are methods based on the bag-of-words model, which have certain requirements for the length of the text. This paper merges the text of all cells in a column as the "document" in the algorithm for execution. For pre-trained FastText, this paper inputs all words in a column into the model to achieve the corresponding word vectors and then uses the average value as the embedding of the column text. For pre-trained BERT, this paper merges all cell texts in a column as input and uses the global average pooling of the output layer as the embedding of the column text.

5.3. Experiment Metrics

The evaluation metrics used in this paper include macro precision, macro recall, macro F1 score, and micro F1 score. Assuming that there are n candidate classes for a multi-classification problem, TP_i refers to the number of samples in the i^{th} category that are divided into the i^{th} category, FN_i refers to the number of samples in the i^{th} category but not classified into the i^{th} category, FP_i refers to the number of samples that are not in the i^{th} category but are classified into the i^{th} category.

Macro precision, macro recall, and macro F1 score are the averages of precision, recall, and F1 scores of all candidate classes:

$$macro\ precision = \frac{1}{n}\sum_{i=1}^{n} \frac{TP_i}{TP_i + FN_i} \tag{8}$$

$$macro\ recall = \frac{1}{n}\sum_{i=1}^{n} \frac{TP_i}{TP_i + FP_i} \tag{9}$$

$$macro\ f_1\ score = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{1 + \frac{FN_i + FP_i}{2 \times TP_i}} \tag{10}$$

To calculate the micro-evaluation index, the sum of the sample numbers of TP, FN, and FP of all candidate classes is taken as TP, FN and FP in the two-class evaluation index formula. From the overall point of view, every time an FP sample is generated in a category, an FN sample will inevitably be generated in the category corresponding to the sample label, and vice versa. Therefore, the sum of the number of FN and FP samples of all candidate classes results in the same result:

$$macro\ f_1\ score = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{1 + \frac{FN_i + FP_i}{2 \times TP_i}} = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} TP_i + FP_i} = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} TP_i + FN_i} \tag{11}$$

5.4. Experiment Results

5.4.1. Comparison with the Baselines

To evaluate the performance of SNMatch, we conducted experiments on the MACST and VizNet-Manyeyes datasets and compared the results with several baseline methods, including LDA, BTM, TF-IDF, FastText, and BERT. The results are summarized in Table 3. SNMatch outperforms the baselines on most metrics, demonstrating its effectiveness in column semantic-type detection, particularly in handling unlabeled tabular data.

Table 3. Performance comparison of methods.

Method	Macro Precision	Macro Recall	Macro F1 Score	Micro F1 Score
MACST				
LDA	0.194	0.212	0.162	0.216
BTM	0.255	0.271	0.213	0.317
TF-IDF	0.343	0.125	0.127	0.120
FastText	0.343	0.254	0.207	0.307
BERT	0.357	0.312	0.267	0.317
SNMatch	0.361	0.357	0.290	0.377
VizNet-Manyeyes				
LDA	0.212	0.225	0.186	0.450
BTM	0.263	0.260	0.212	0.444
FastText	0.257	0.170	0.126	0.264
BERT	0.289	0.216	0.164	0.264
SNMatch	0.292	0.287	0.220	0.380

In addition, methods have various abilities to capture different semantic-type features. Figure 4 shows the average performance of each semantic type on the MACST dataset compared with the BERT model. The X-axis is arranged according to the number of occurrences of different semantic types in the corpus from most to least. It can be seen that whether the features of semantic types can be effectively learned is not completely dominated by the amount of corpus of this semantics type but is also related to the characteristics of the model and the semantic type itself. In most semantic types, the performance of SNMath is better than the comparison method.

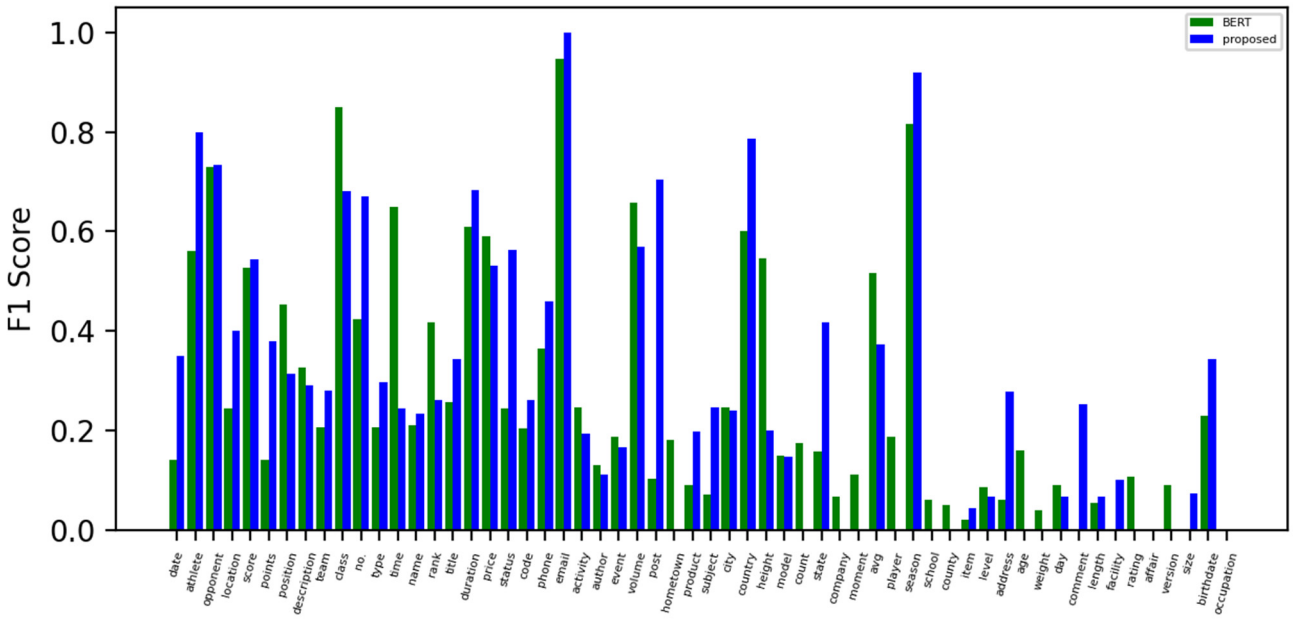

Figure 4. Performance comparison of methods on different semantic types (MACST).

From the experimental results on semantic types, the semantic types that are easier to distinguish with the method in this paper include email, season, athlete, country, and post, with F1 scores of 1.0, 0.92, 0.8, 0.77, and 0.71. Semantic types that are more difficult to distinguish mainly include two types, one is numerical values, such as rating, weight, and age. The other part is that there are other semantic types that have a hierarchical relationship with each other in the collection. For example, in the MACST dataset, there are

semantic types with hierarchical relationships between "Location", "City", "Hometown", and "County" at the same time. The column texts of these semantic types are very similar, resulting in almost indistinguishable clustering.

Statistical Significance Testing

To ensure the reliability of the comparative results, we performed statistical significance tests using paired *t*-tests on the macro and micro F1 scores across all methods. The *p*-values indicate that SNMatch's improvements over the baselines are statistically significant ($p < 0.05$) for both datasets, confirming the robustness of the observed performance gains.

Confidence Interval Calculations

For all performance metrics (macro precision, macro recall, macro F1 score, and micro F1 score), we calculated 95% confidence intervals (CIs) to provide a clearer understanding of result variability. For example, the macro F1 score of SNMatch on the MACST dataset is 0.290 ± 0.015 (95% CI), while the micro F1 score is 0.377 ± 0.012. These intervals demonstrate that the performance of SNMatch is consistently better than the baselines within the calculated range.

Error Analysis and Failure Mode Categorization

To identify potential weaknesses in SNMatch, we conducted a comprehensive error analysis by categorizing failure modes into three main types:

Semantic Overlap: Errors caused by columns with overlapping semantic features, such as "City" and "Hometown".

Format Ambiguity: Errors arising from columns with similar format patterns but different semantic meanings, such as "phone number" and "postal code".

Sparse Data: Errors in columns with insufficient or highly imbalanced data, where rare semantic types are underrepresented.

Figure 5 illustrates the distribution of these failure modes across the datasets. The analysis reveals that most errors are due to semantic overlap, highlighting the need for more sophisticated feature extraction mechanisms for fine-grained semantic distinctions.

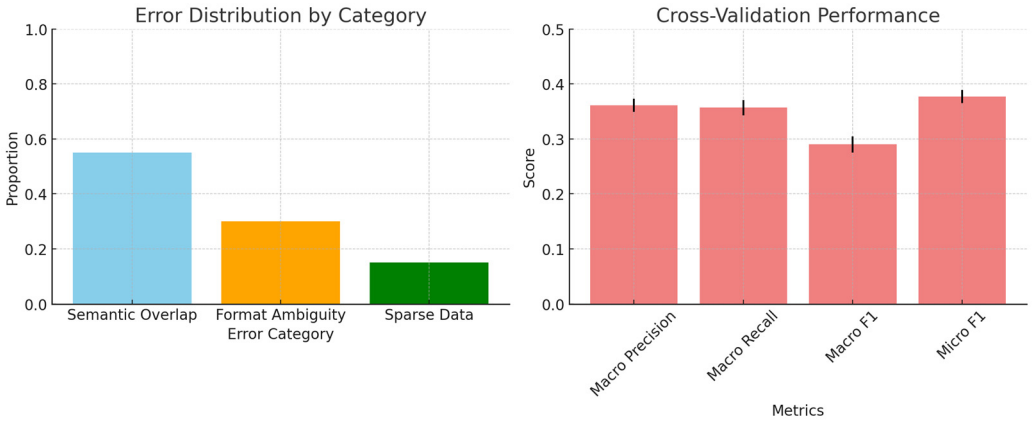

Figure 5. Error analysis and cross-validation performance diagram.

Cross-Validation Protocols

As shown in Figure 5, to verify the robustness of SNMatch, we implemented k-fold cross-validation ($k = 5$) on both datasets. The results, averaged across folds, showed minimal variance, with standard deviations of less than 0.01 for all metrics. This indicates that SNMatch's performance is stable and generalizable across different data splits.

5.4.2. Siamese Network

The performance of the Siamese network is reflected in the accuracy of determining whether cell pairs are matched in semantic type. In order to ensure that the distribution

of training data and test data are as similar as possible, the number of positive samples and negative samples drawn is the same. Table 4 shows the performance of the Siamese network on matching the cell pair semantic type.

Table 4. Performance of the Siamese network on matching semantic type of cell pairs.

Dataset	Precision	Recall	F1 Score
MACST	0.84	0.63	0.72
VizNet-Manyeyes	0.66	0.69	0.68

5.4.3. Efficiency of Featured Vectors

In order to evaluate whether the cell encoder proposed in this paper can learn the features of cell text better than existing classifiers, this paper uses the MatchAttribute (MA) test proposed by Cappuzzo et al. In the MA test, four cell texts extracted from the same column and one cell text extracted from another column with a different semantic type are combined into a set of data. Then, compare which method can better distinguish the characteristics of the cell text and pick out the cell texts of different semantic types.

For the encoder φ proposed in this paper, the trained φ is used to encode each cell text into a feature vector, and the Euclidean distance is calculated with the idea of clustering. This method selects the feature vector farthest from the other four, and this feature vector corresponds to cell texts of different semantic types. Finally, the accuracy is calculated: the ratio of the correctly classified group to all groups.

For classification algorithms, this paper uses two algorithms as comparison methods: multi-class SVM and a simple neural network with one hidden layer. The cell texts are used as the training data, and the corresponding column semantic type is used as the training label. For data preprocessing, the auto-encoder is constructed by removed FastText semantic feature input from our cell text encoder. In the processing of the results, because the classification algorithm cannot reflect the hierarchical relationship like the clustering algorithm, the Euclidean distance cannot be used to calculate the division of a group of five elements. Therefore, this paper adopts the method of finding the maximum set and regards the number of elements in the maximum set as the correct division, and the rest is regarded as the division error. For example, for the classification result (11,22,11,11,36): If all five elements are classified correctly, the first four elements should be classified into the same category, and the fifth elements should be classified into another category. In the results returned by the classifier, the second element is classified into a different category, which should be regarded as a classification error. The classification result of the fifth element should be different from the first four elements, so the classification is correct. Therefore, four classification results are correct, and one classification result is incorrect. Finally, we calculate the ratio of the number of correctly classified cells to the total number of cells as the accuracy.

The results of the comparative experiments on the VizNet-Manyeyes dataset are shown in Table 5. The experimental results show that our encoder φ can pick out the different one from a set of data more accurately than the classification algorithm, which proves that φ can better learn the characteristics of the cell text. At the same time, it can be noted that the division effect of the neural network and multi-class SVM is indeed improved on the basis of the auto-encoder. But if the performance of the neural network and the multi-class SVM is to be further improved, it will take a lot of data and time to train, but this is obviously not comparable to the encoder φ, which requires only a small overhead.

Table 5. Accuracy comparison of SNMatch and classifier.

Method	Accuracy
Encoder φ + clustering	0.93
Auto-encoder + clustering	0.72
Auto-encoder + one layer neural network	0.82
Auto-encoder + multi-class SVM	0.75

5.4.4. Ablation Experiments

In order to evaluate the effectiveness of the introduced pre-trained model FastText in distinguishing the semantic type of the cell, this paper sets up an ablation experiment to test the performance of the extractor without introducing the FastText pre-trained word vector.

The results of the comparison are shown in Table 6. On the two datasets, the introduction of FastText semantic features leads to a certain improvement in the performance of the model. The improvement of introducing FastText pre-trained word vectors on the MACST dataset is less obvious, while the introduction of FastText pre-trained word vectors on the VizNet-Manyeyes dataset has greatly improved all evaluation indicators. This is caused by deleting columns with more OOV words in VizNet-Manyeyes. On the one hand, when there are less OOV words, the semantic features of word vectors provided by FastText are more accurate. On the other hand, OOV words are mostly columns with obvious format features such as email and phone. The deletion of these columns makes the model's ability to capture format features unable to reflect, so the performance is not particularly outstanding.

Table 6. Efficiency evaluation on pre-trained model FastText.

Method	Macro Precision	Macro Recall	Macro F1 Score	Micro F1 Score
MACST				
without FastText	0.314	0.302	0.240	0.340
with FastText	0.361	0.357	0.290	0.377
VizNet-Manyeyes				
without FastText	0.282	0.214	0.090	0.220
with FastText	0.292	0.287	0.220	0.380

6. Conclusions

This paper proposes a novel model, SNMatch, for detecting the column semantic type of tabular data based on a Siamese network. This model makes full use of both the structural information and semantic information of the tabular data and transforms the column semantic-type detection into a column cluttering task. In our model, the cell text of columns is utilized to generate training samples with an unsupervised method based on the PU method, and the Siamese network is designed to train the cell text encoder, which use both character sequences and word semantics from FastText as input. Finally, the columns encoder maps columns to feature vectors, which are clustered into groups of corresponding semantic types. Through experiments, this paper proves the effectiveness of the proposed method SNMatch and its superiority on small data volume and multiple OOV word datasets.

Technical Limitations and Future Research Directions

While the proposed SNMatch model demonstrates significant improvements in column semantic-type detection, it is important to note some technical limitations:

Data Scalability: The current approach may face challenges when applied to very large datasets due to the computational complexity of clustering. Future work could explore more scalable clustering techniques or implement parallelization to improve performance on large-scale data.

Out-of-Vocabulary (OOV) Words: While FastText embeddings help mitigate OOV issues, some rare or specialized semantic types may still pose a challenge. Future research could explore integrating more sophisticated pre-trained models, such as domain-specific embeddings or large language models (LLMs), to better handle these cases.

Hierarchical Semantic Relationships: Although the current model captures basic semantic types, deeper hierarchical relationships between types (e.g., "Location" vs. "City") are not fully exploited. Future research could enhance the model by incorporating hierarchical clustering techniques or multi-level embedding structures to better capture these relationships.

In the future, we aim to extend the SNMatch model to address these challenges by incorporating more advanced clustering algorithms and exploring the use of transfer learning for the better handling of rare and complex semantic types.

Author Contributions: Conceptualization, T.N.; Methodology, X.W.; Software, X.W.; Validation, T.N., H.M. and X.W.; Formal analysis, T.N., H.M., D.S. and Y.K.; Resources, T.N., D.S. and Y.K.; Data curation, H.M.; Writing–original draft, H.M., A.L. and X.W.; Writing–review & editing, T.N.; Project administration, T.N.; Funding acquisition, T.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (grant numbers 62072086, 62172082, and 62072084), and the Special Funds for Basic Scientific Research of Central Universities (grant number N2116008).

Data Availability Statement: The original contributions presented in this study are included in this article; further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Doan, A.; Halevy, A.; Ives, Z. 5-Schema Matching and Mapping. In *Principles of Data Integration*; Doan, A., Halevy, A., Ives, Z., Eds.; Morgan Kaufmann: Burlington, MA, USA, 2012; pp. 121–160.
2. Wang, R.; Li, Y.; Wang, J. Sudowoodo: Contrastive Self-supervised Learning for Multi-purpose Data Integration and Preparation. In Proceedings of the 2023 IEEE 39th International Conference on Data Engineering, Anaheim, CA, USA, 3–7 April 2023; pp. 1502–1515.
3. An, X.; You, S.; Guo, Z.; Lu, Z.; Zheng, B.; Shi, S.; Song, Y. Column concept determination based on multiple evidences. *Concurr. Comput. Pract. Exp.* **2021**, *33*, e5457. [CrossRef]
4. Limaye, G.; Sarawagi, S.; Chakrabarti, S. Annotating and searching web tables using entities, types and relationships. *Proc. VLDB Endow.* **2010**, *3*, 1338–1347. [CrossRef]
5. Goel, A.; Knoblock, C.A.; Lerman, K. Exploiting structure within data for accurate labeling using conditional random fields. In Proceedings of the International Conference on Artificial Intelligence (ICAI). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), Las Vegas, NV, USA, 16–19 July 2012; pp. 1–9.
6. Bhagavatula, C.S.; Noraset, T.; Downey, D. TabEL: Entity linking in web tables. In *International Semantic Web Conference*; Springer: Cham, Switzerland, 2015; pp. 425–441.
7. Chen, J.; Jiménez-Ruiz, E.; Horrocks, I.; Sutton, C. Colnet: Embedding the semantics of web tables for column type prediction. *Proc. AAAI Conf. Artif. Intell.* **2019**, *33*, 29–36. [CrossRef]
8. Hulsebos, M.; Hu, K.; Bakker, M.; Zgraggen, E.; Satyanarayan, A.; Kraska, T.; Demiralp, C.; Hidalgo, C. Sherlock: A deep learning approach to semantic data type detection. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019; pp. 1500–1508.
9. Zhang, D.; Suhara, Y.; Li, J.; Hulsebos, M.; Demiralp, C.; Tan, W.-C. Sato: Contextual semantic type detection in tables. *Proc. VLDB Endow.* **2019**, *13*, 1835–1848. [CrossRef]

10. Xie, J.; Cao, C.; Liu, Y.; Cao, Y.; Li, B.; Tan, J. Column Concept Determination for Chinese Web Tables via Convolutional Neural Network. In *International Conference on Computational Science*; Springer: Cham, Switzerland, 2018; pp. 533–544.
11. Wang, D.; Shiralkar, P.; Lockard, C.; Huang, B.; Dong, X.L.; Jiang, M. TCN: Table Convolutional Network for Web Table Interpretation. In Proceedings of the Web Conference 2021, New York, NY, USA, 18 May 2021; pp. 4020–4032.
12. Bromley, J.; Guyon, I.; LeCun, Y.; Säckinger, E.; Shah, R. Signature verification using a "siamese" time delay neural network. *Adv. Neural Inf. Process. Syst.* **1993**, *6*, 737–744. [CrossRef]
13. Bojanowski, P.; Grave, E.; Joulin, A.; Mikolov, T. Enriching word vectors with subword information. *Trans. Assoc. Comput. Linguist.* **2017**, *5*, 135–146. [CrossRef]
14. Kiryo, R.; Niu, G.; du Plessis, M.C.; Sugiyama, M. Positive-unlabeled learning with non-negative risk estimator. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 1674–1684.
15. Venetis, P.; Halevy, A.Y.; Madhavan, J.; Pasca, M.; Shen, W.; Wu, F.; Miao, G. Recovering semantics of tables on the web. *Proc. VLDB Endow.* **2011**, *4*, 528–538. [CrossRef]
16. Deng, D.; Jiang, Y.; Li, G.; Li, J.; Yu, C. Scalable Column Concept Determination for Web Tables Using Large Knowledge Bases. *Proc. VLDB Endow.* **2013**, *6*, 1606–1617. [CrossRef]
17. Yang, J.; Gupta, A.; Upadhyay, S.; He, L.; Goel, R.; Paul, S. TableFormer: Robust Transformer Modeling for Table-Text Encoding. *arXiv* **2022**, arXiv:2203.00274.
18. Maji, S.; Rout, S.S.; Choudhary, S. DCoM: A Deep Column Mapper for Semantic Data Type Detection. *arXiv* **2024**, arXiv:2106.12871.
19. Bordawekar, R.; Bandyopadhyay, B.; Shmueli, O. Cognitive database: A step towards endowing relational databases with artificial intelligence capabilities [DB/OL]. *arXiv* **2017**, arXiv:1712.07199.
20. Fernandez, R.C.; Madden, S. Termite: A system for tunneling through heterogeneous data. In Proceedings of the Second International Workshop on Exploiting Artificial Intelligence Techniques for Data Management, Amsterdam, The Netherlands, 5 July 2019; pp. 1–8.
21. Cappuzzo, R.; Papotti, P.; Thirumuruganathan, S. Creating embeddings of heterogeneous relational datasets for data integration tasks. In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, Portland, OR, USA, 14–19 June 2020; pp. 1335–1349.
22. Suhara, Y.; Li, J.; Li, Y.; Zhang, D.; Demiralp, Ç.; Chen, C.; Tan, W.C. Annotating Columns with Pre-trained Language Models. In Proceedings of the 2022 International Conference on Management of Data, New York, NY, USA, 22–27 June 2023; pp. 1493–1503.
23. Sun, Y.; Xin, H.; Chen, L. RECA: Related Tables Enhanced Column Semantic Type Annotation Framework. *Proc. VLDB Endow.* **2023**, *16*, 1319–1331. [CrossRef]
24. Deng, X.; Sun, H.; Lees, A.; Wu, Y.; Yu, C. TURL: Table Understanding through Representation Learning. *SIGMOD Rec.* **2022**, *51*, 33–40. [CrossRef]
25. Hu, K.; Gaikwad, S.; Hulsebos, M.; Bakker, M.A.; Zgraggen, E.; Hidalgo, C.; Kraska, T.; Li, G.; Satyanarayan, A.; Demiralp, Ç. Viznet: Towards a large-scale visualization learning and benchmarking repository. In Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, Glasgow, UK, 4–9 May 2019; pp. 1–12.
26. 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, Montreal, QC, Canada, 11–15 April 2016; pp. 75–76.