Web Table Column Type Detection Using Deep Learning and Probability Graph Model

Tong Guo, Derong Shen(✉), Tiezheng Nie, and Yue Kou

School of Computer Science and Engineering, Northeastern University, Shenyang 110004, China
Shirley_GT@163.com, {shenderong, nietiezheng, kouyue}@cse.neu.edu.cn

Abstract. The rich knowledge contains on the web plays an important role in the researches and practical applications including web search, multi-question answering, and knowledge base construction. How to correctly detect the semantic types of all the data columns is critical to understand the web table. The traditional methods have the following limitations: (1) Most of them rely on dictionary lookup and regular expression matching, and are generally not robust to dirty data; (2) They only consider character data besides numeric data which accounts for a large proportion; (3) Some models take the characteristics of a single column and do not consider the special organizational structure of the table. In this paper, a column type detection method combining deep learning and probability graph model is proposed, taking the semantic features of a single column and the interaction between multiple columns into account to improve the prediction accuracy. Experimental results show that our method has higher accuracy compared with the state-of-the-art approaches.

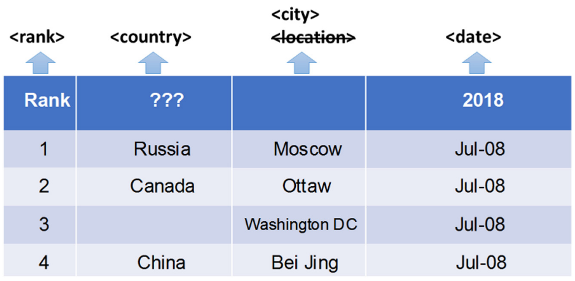
Keywords: Web tables · Column type detection · Deep learning · Probability graph model

1 Introduction

Web tables are web content displayed in the form of tables, which can provide a large amount of high-quality data and have a wide range of topics. The effective use of massive amounts of high-quality table data is of great significance. There are currently many practical applications that utilize the rich semi-structured data resources of web tables, including question answering, table search, table expansion, knowledge base (KB) completion, semantic retrieval and the creation of linked open data (LOD) [1] and so on. The usefulness of web table data depends largely on the semantic understanding. One way to gain the semantics of a table is to match it with a KB, and the process is also called "table interpretion" [2]. The correct detection of the semantic types of table columns is vital for the task of table interpretation.

There have been many works for column type detection. Traditional methods based on search and ontology matching, which find the corresponding entities of the cells contained in the column and use the types corresponding to these entities as the result. These works are only applicable to the named-entity columns (NE-columns) and can't deal with the large amount of dirty data in the real-world data set. Another category of prior method uses a probabilistic model which is still based on the ontology matching of single cells, so it has the same disadvantages. Most of the research in recent years is based on feature engineering. A variety of features in web tables and knowledge bases are extracted to build supervised or unsupervised models. However, most of the features contains metadata like external web page information and column header information which are manually extracted, time-consuming and labor-intensive. At the same time, the accuracy is not high due to the neglect of the semantic association between the cells.

For example, Fig. 1 shows an example of one web table with unknown column types. The tags on the top of headers are the correct results after semantic type detection. The third header is missing, so the header description message cannot be used to determine the type of data in the column; the city "Ottawa" in the second row of the third column is incorrectly spelled "Ottaw". If only use the data of single column, it is possible to get the incorrect semantic type <location> instead of the correct semantic type <city> .



Rank	???		2018
1	Russia	Moscow	Jul-08
2	Canada	Ottaw	Jul-08
3		Washington DC	Jul-08
4	China	Bei Jing	Jul-08

Fig. 1. An instance of Web Table Column Type Detection

In view of the above reasons, this paper first synthesizes the contextual semantics of cells in a single column and uses the deep learning model to build a single-column classification model to pre-classify the semantic type, then uses the relationship between columns to comprehensively detect all the columns' types, hence further improve the accuracy of column type detection. Our contributions can be summarized as follows:

1. A method for detecting column types of web tables combining deep learning with probability graph model is proposed. This method can detect both of character data columns and numeric data ones at the same time without any metadata.
2. A single-column type detection model based on hybrid neural network is proposed. The deep learning model of BiGRU + Attention is used to further obtain deep level semantic relationship for improving prediction performance.
3. A multi-column type detection model based on the probability graph model is proposed, which comprehensively considers the co-occurrence relationship of various semantic types in the knowledge base and the real data set, and better utilizes the implicit semantic relationship between columns.

4. Through experiments, the proposed method performs better than other column type detection algorithms in applications.

2 Related Work

Column type detection of web tables constitutes table semantic interpretation tasks like attribute annotations and foreign key discovery [2]. Its research results can also provide labeled table data for tasks such as question answering [3], knowledge base completion [4] and table expansion [5]. The current methods can be divided into three categories: ontology-based methods, probabilistic methods, and featured-based methods.

The traditional methods take ontology matching as the core. [6] and [7] focus on instance-level matching. They find the matched candidate entity classes for each cell in the target column, then chooses the type with the highest frequency as the result. [8] uses word vectors to represent entities in cells and knowledge bases and then uses majority voting algorithm. However, these methods are limited to the low coverage of knowledge base and ignores the great differences between data from different sources, such as different naming and abbreviations.

Another kind of studies use probabilistic model. [9] used a probabilistic graphical model to express different degrees of matching relationships and annotate the table with a series of related random variables according to a suitable joint density distribution. Based on this work, [10] used a more lightweight graphical model. [11] weakened the assumption that there is a knowledge base correspondence between columns and entity sets, but strengthened the relationship with Wikipedia documents. [12] proposed a maximum likelihood inference model. This kind of methods ignore possible associations between cells.

In recent years, some researchers based on feature engineering have been presented. [13] used Kolmogorov-Smirnov test and TF-IDF to describe these types, [14] used more features, including Mann-Whitney test for numerical data and Jaccard similarity for text data to train logistic regression and random forest models. These methods rely on some external data such as the table title, so the detect results depend on the size of the table and the number of entity columns.

Compared with the existing work, this paper has the following advantages: (1) More consideration is given to the semantic information and don't rely on external data and metadata since this kind of data is often missing; (2) In order to solve the problems of dirty data, this article combines word embedding and character embedding to learn the distribution information of short sequences in words to a certain extent; (3) By selecting the appropriate deep learning model to better extract the deep semantic information of the text; (4) Based on the single-column classification model, our model combined with the probability graph model to extract the contextual characteristics and relationship characteristics between columns make the classification results more accurate.

3 Problem Statement and Definition

We first give the relevant definitions, and then elaborate on the problems.

Definition 1. Web table. The web table x consists of m rows and n columns, $\{C_1, C_2, \ldots, C_n\}$ represents the set of all columns, where each column is composed of a column header C_i^h and a group of cells, marked as $C_i = \{v_{i1}, v_{i2}, \ldots, v_{im}\}$, the data in the cell exists in the form of characters or numbers. Column headers and cells are allowed to be null.

Definition 2. KB. Given a knowledge base, define it as a six-tuple form: $KB = \{E, T, R, B, G, F\}$, where E, T, R represent a group of instances, a collection of types and the relationship between two entities; B is a collection of $<e, T>$ binary tuples, used to indicate that instance e belongs to a certain type T; G is a classification graph, representing the hierarchical structure between types, each directed edge is from a more general type to a more specific type, there is a subclass-of relationship between the parent class and the subclass; F is a collection of fact triples, representing the relationship fact or attribute fact between two entities.

Problem 1. Given a KB, for each table in the web table corpus (WTC), find the semantic type $Y_i \in T$ in KB for each column C_i in the table that best capture C_i's content.

The model proposed in this paper is mainly divided into two parts: single-column pre-classification model and multi-column type detection model. The overall framework is shown in Fig. 2. Firstly, the columns in the web table are divided into character and numeric columns, character column type detection model and numeric column type detection model are constructed respectively; The linear chain conditional random field in the graph model is used to construct the undirected graph model, where the state features are represented by the classification probability obtained by the single-column classification model, and the transition features are represented by the matrix obtained from the existing latent semantic relationship in the corpus and KB; Finally, the final classification result is obtained through calculation.

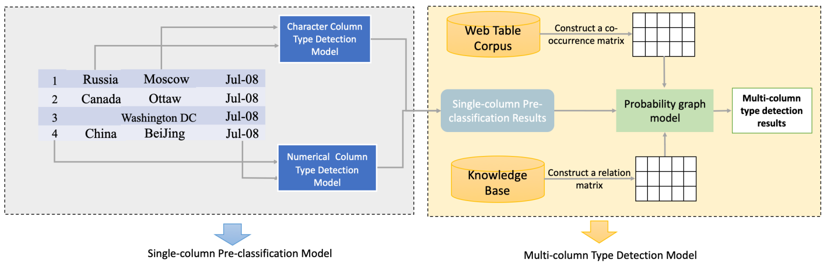

Character Column Type Detection Model
1 Russia Moscow Jul-08
2 Canada Ottaw Jul-08
3 Washington DC Jul-08
4 China BeiJing Jul-08
Numerical Column Type Detection Model
Single-column Pre-classification Model
Web Table Corpus
Construct a co-occurrence matrix
Single-column Pre-classification Results
Probability graph model
Multi-column type detection results
Knowledge Base
Construct a relation matrix
Multi-column Type Detection Model

Fig. 2. The overall framework

4 Single-Column Pre-classification Model

We first propose a semantic classification model for single columns, which detects the semantic type based on the content of cells contained in the target column. This task can be formally defined as a multi-classification problem, assuming that a set of K predefined KB types which do not intersect with each other is given as $\{P_1, P_2, \ldots, P_k\}, P_i \in T$. Taking all cells in the column as input, we assign that each category P_i has an actual value score S_{P_i} through the classification model, so that the correct type of the target column has the highest score. We separate the columns into numeric and character ones and then build two classification models.

4.1 Character Column Type Detection Model

Embedding Layer. Existing research shows that there is inter-cell correlation in the web table [15]. In order to better extract the local features and context features of the data in the column, we use a sliding window of size H to randomly collect adjacent H cells on the target column and combine all the word sequences in these cells. The initial text is formed together, then M initial texts are obtained after collecting M times. Because there is no uniform standard for the structure of the network table, we give a fixed size value N as the specified text length.

Considering the diversity and irregularity of the sources of web tables and the low frequency of most entity words, large-scale data sets will inevitably have a large number of OOV (Out-Of-Vocabulary) words, so this paper uses a vector that combines character embedding and word embedding as input to the subsequent neural network model(see Fig. 3). Compared with the latest research such as the Sherlock model [16], which uses character statistics as a feature method, character-level representation is more flexible in handling spelling errors and rare word problems [17].

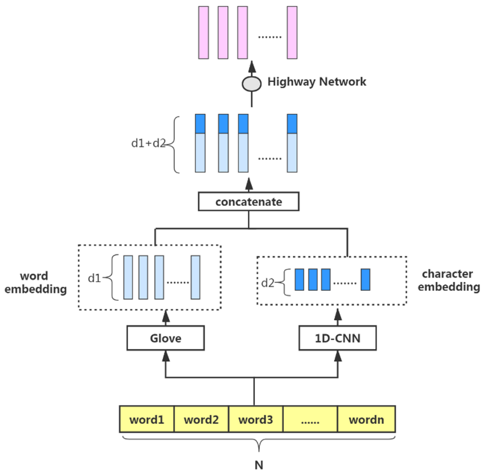


Fig. 3. Embedding model combined word embedding with character embedding.

For each initial text of length N, we use Glove and 1D-CNN (one-dimensional convolutional neural network) to get word embedding and character embedding about it, and then connect the two vertically to generate a matrix, and then pass it through Highway-NN (High Speed Neural Network) to get the embedding result. Only a small part of the input will be affected by Eq. (2), which is a single-layer feedforward neural network, and the rest is allowed to pass through the untransformed network.

$$z = t \odot g(W_H y + b_H) + (1 - t) \odot y \tag{1}$$

$$z = g(Wy + b) \tag{2}$$

Model Building Layer. Existing studies have used simple feed-forward NN [16], CNN [15] and BiRNN [19] to capture the features of text. However, these methods still can't fully learn the context information of the text and extract the deep-level information in the table content. Therefore, we propose a column type detection method based on BiGRU-Attention hybrid neural network model, as shown in Fig. 4. The model is divided into three parts: input layer, hidden layer and output layer.

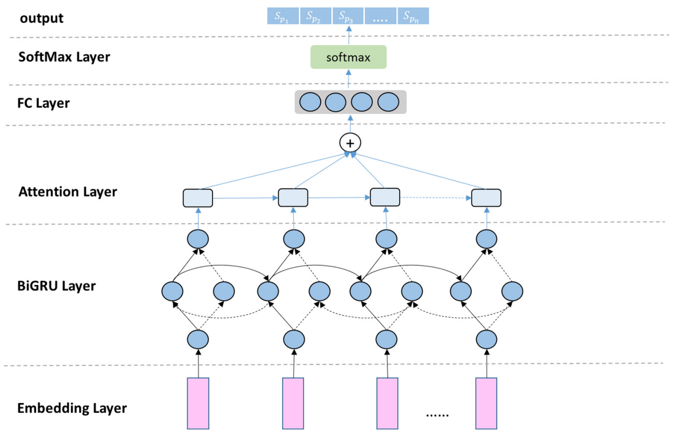


Fig. 4. BiGRU + Attention network model.

In embedding layer, through the word embedding and character embedding fusion model introduced above, a fixed-size matrix is obtained as the vectorized representation of the text of the target column, and then the text features are extracted through the hidden layer. BiGRU layer contains two unidirectional, opposite GRUs, respectively (t−1) the output of the hidden layer state forward $\overrightarrow{h_t}$ and the output of the reverse hidden layer state $\overleftarrow{h_t}$, it also contains a neural network model composed of GRUs determined by the output of the two.

$$\begin{cases} \overrightarrow{h_t} = GRU\left(x_t, \overrightarrow{h_{t-1}}\right) \\ \overleftarrow{h_t} = GRU\left(x_t, \overleftarrow{h_{t-1}}\right) \\ h_t = w_t \overrightarrow{h_t} + v_t \overleftarrow{h_t} + b_t \end{cases} \tag{3}$$

We then put the output vector of the BiGRU layer into the Attention layer, which can further highlight the key information of the text to improve the quality of the feature extraction of the data layer of the hidden layer. The attention mechanism calculates the probability weight of the word vector, performs automatic weighted transformation on the data, and highlights the important words. The weight coefficient is specifically calculated, where u_w represents a randomly initialized attention matrix, h_t is the output vector of the BiGRU in the previous layer, and w_w is the weight coefficient.

$$\begin{gathered} \alpha_t = \frac{\exp\left(u_t^T u_w\right)}{\sum_t \exp\left(u_t^T u_w\right)} \\ u_t = \tan h(w_w h_t + b_w) \\ s_t = \sum_{i=1}^{n} \alpha_t h_t \end{gathered} \tag{4}$$

Next we further stacked with a fully connected (FC) layer that learns the nonlinear relationship between input and output. Finally, the SoftMax function is used to perform the corresponding calculation on the input of the output layer to perform text classification. Get the output vector C_i of the target column $\overrightarrow{C_i} = \left\{S_{p_1}, S_{p_2}, \ldots, S_{p_k}\right\}$, K is the number of types in the predefined KB, each bit of C_i represents the probability that the column belongs to a certain semantic type.

4.2 Numerical Column Type Detection Model

Most of the cells in the numeric column only have a single numeric value, and generally do not contain too many context features, so we use a simple classification method based on statistical features. For example, statistics such as the mean, variance, median, mode, maximum, minimum, peak, skewness, and standard deviation of values.

In addition to numbers, the numeric column also has the possibility of characters. Taking Fig. 1 as an example, the second column has a cell of “Jul-08”. These text messages greatly influence the judgment of column types, so we also extract features from these text messages. Statistics such as the frequency of occurrence of each letter, the mean and variance of the character length and the proportion of cells with characters. After extracting these values, they are modeled using classic machine learning algorithms such as random forest, the vector representation of probability $\left\{S_{p_1}, S_{p_2}, \ldots, S_{p_k}\right\}$ is also obtained.

5 Multi-column Type Detection Model

Through the single-column classification model proposed in Sect. 4, we can get a preliminary judgment on the semantic type. In order to make the result more accurate, we also need to consider the relationship between the semantic types of adjacent columns, and use the local context features between the columns to detect the column type.

5.1 CRF Model

In view of the above situation, this paper proposes a multi-column type detection based on a probability graph model to model the correlation between structural link variables to perform joint prediction. Considering that the interaction between the two columns does not have directionality, we use linear-CRF (linear chain conditional random field) on the entire network table, an undirected graph model that directly models the conditional probability.

$$score(C, y) = \sum_{k,i}^{n} \lambda_k \phi_{single}(C_i, y_i) + \sum_{l,i}^{n} \mu_l \phi_{multi}(y_{i-2} y_{i-1}, y_i) \tag{5}$$

$$P(y|C) = \frac{e^{score(C,y)}}{\sum_{y \in Y_c} e^{score(C,y)}} \tag{6}$$

C represents the sequence of the input table columns $\{C_1, C_2, \ldots, C_n\}$, Y represents the type sequence corresponding to each column of the output $\{Y_1, Y_2, \ldots, Y_n\}$, $Y_i \in T$ in KB, λ_k and μ_l are weight coefficients. $\mathrm{P}(y|C)$ obtained after normalizing $score(C, y)$ represents the conditional probability that the predicted output sequence is Y when the input sequence C is given. The model defines the multi-column type detection problem as maximizing the joint conditional probability when the content in the table column is given. When the probability $\mathrm{P}(y|C)$ reach the maximum, the corresponding Y is the semantic type sequence of the table columns. Figure 5 shows an instance.

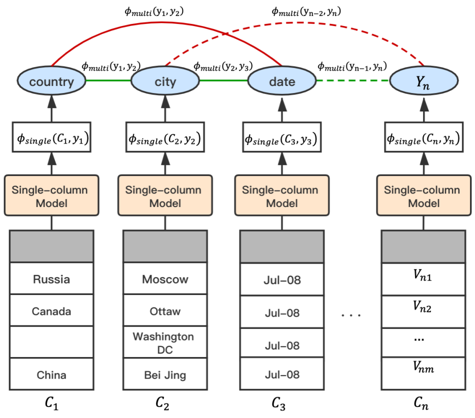


Fig. 5. Multi-column type detection model based on CRF.

5.2 Potential Functions

We define two potential functions for this model:

Status Feature Function $\phi_{single}(C_i, y_i)$**.** We use it to measure the influence of the feature obtained based on the column value on the semantic type detection of a column when a column is given, here we use the type probability value corresponding to the column $\{S_{p_1}, S_{p_2}, \ldots, S_{p_k}\}$ to represent. This probability can be obtained by the single-column model proposed in Sect. 4. The character sequence corresponds to the hybrid network model in 4.1, and the numerical sequence corresponds to the random forest model in 4.2.

Transfer Feature Function $\phi_{multi}(y_{i-2}, y_{i-1}, y_i)$. It depends on the status of the current column y_i, $y_{i-2}$and y_{i-1}, which is used to indicate the influence of the existing relationship between the columns on its semantic type detection. When the semantic relationship is closer, more likely to occur at the same time, $\phi_{multi}(y_{i-2}, y_{i-1}, y_i)$ has a higher value. In order to quantify this relation, we define a matrix Q of size K × K. This paper will calculate this value from two aspects.

First we consider to quantify the co-occurrence relationship extracted from the data set: give a marked web table, take $\{Y_1, Y_2, Y_3\}$ = {country, language, currency} as an example, co-occurrence type pairs are <country,language> , <language, currency > and <country, currency > . Traverse all tables to get matrix P_{corr}. In order to increase the accuracy and versatility of the algorithm, we use the semantic relationship marked in KB to further calculate P_{rela} (see Algorithm 1). Then Q_{ij} is the probability value obtained after normalizing the $P_{corr}(Y_i, Y_j)$ and $P_{rela}(Y_i, Y_j)$.

```
Algorithm 1: ComputeRelation
Input: KB, $P = \{P_1, P_2, \ldots, P_k\}$, $P_i \in T$
Output: matrix $P_{rela}$
1.  for each $P_i$ in P
2.      { Properties} ⟵ search for $P_i.(is\ rdfs{:}domain\ of)$   /* Query the type's attribute */
3.          for each property in { Properties }
4.                  {classes} ⟵ search for property. $(rdfs{:}range)$
5.                  if $\exists P_j \in P$ in {classes}/* classes contains predefined semantic types */
                        $P_{rela}(P_i, P_j) = 1$
6.      { Properties} ⟵ search for $P_i.(is\ rdfs{:}\mathrm{range}\ of)$
7.          for each property in {Properties }
8.                  { classes} ⟵ search for property. $(rdfs{:}domain)$
9.                  if $\exists P_l \in P$ in {classes}
                        $P_{relation}(P_i, P_l) = 1$
10. return $P_{rela}$
```

6 Experiments

6.1 Experiment Dataset

Two real-world data sets T2Dv2 and Limaye derived from the Gold Standard are used as the web table corpus in the experiment. The data set is json files obtained from various types of web pages. These tabular data manually marked by previous researchers which shows diversity in size, existence and sparsity of relationships. The specific information is shown in Table 1. The KB used in this article is DBPedia which is at the core of the LOD project.

Table 1. Statistics of Web Tables datasets.

Name	Tables	Avg. cells	Rows	Columns	Types	Coverage of entity
T2Dv2	779	124	84	5	65	30.5%
Limaye	428	23	34	4	421	25.8%

6.2 Comparative Experiment

In order to verify the feasibility of the model proposed in this paper, compare it with the following models: Lookup-Vote [5], T2K Match [18], ColNet [15], and Sherlock [16]. T2K Match is a collective matching method which first selects a series of candidate instances from the KB and then performs attribute matching based on the repeated pattern. ColNet integrates KB reasoning and lookup with machine learning and trains CNN model. Sherlock is a multi-layer neural network model.

6.3 Evaluation Metrics

In order to evaluate the performance of the model, we use the precision and recall rates obtained by comparing the real and predicted values and F1 score value as evaluation indicators to measure the effect of different models.

$$F1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{7}$$

6.4 Experimental Results

Overall Performance: We run the proposed model on T2Dv2 and Limaye repeatedly, and then compared the classification results with other models. The specific data under the three measurement indicators are shown in Table 2 and Table 3. It can be seen that in these methods, the overall classification effects of Lookup-Vote and T2K Match are the most unsatisfactory, because these two methods based on ontology matching rely heavily on whether the cell in the web table can find the corresponding entity in the KB. It can be seen from Table 1 that the entity coverage of the cells in the web table is very low, and the difference in performance on different data sets is significant.

Table 2. Results of models on T2Dv2.

Methods	Precision	Recall	F1 score
Lookup-Vote	0.862	0.821	0.841
T2K Match	0.624	0.727	0.671
ColNet	0.765	0.811	0.787
Sherlock	0.850	0.834	0.842
Proposed	0.903	0.871	0.887

Table 3. Results of models on Limaye.

Methods	Precision	Recall	F1 score
Lookup-Vote	0.732	0.660	0.694
T2K Match	0.560	0.408	0.472
ColNet	0.763	0.820	0.791
Sherlock	0.810	0.859	0.833
Proposed	0.864	0.841	0.852

ColNet and Sherlock, two feature-based methods, achieved better results by introducing convolutional neural networks and feedforward neural networks, but only considered the characteristics of single-column data. Our method takes the semantic characteristics of the single column and the influence of the relationship between the columns into account, so the overall performance on both data sets is significantly better than several other methods.

Relationship Between Columns Impact: In order to further verify the influence of the relationship between columns on the overall classification results, the experiment compared the results of the model proposed by Sherlock and the model proposed in this paper. In order to comprehensively measure the classification effect of the model, we separately calculate the evaluation indicators of Micro-F1, Macro-F1 and weighted-F1.

As shown in Fig. 6, the fusion of the multi-column detection model proposed in this paper can improve the classification accuracy, and the fusion of the multi-column detection model on the single-column model proposed in this paper can also improve the classification accuracy. It shows that the relationship between columns can affect the final classification effect.

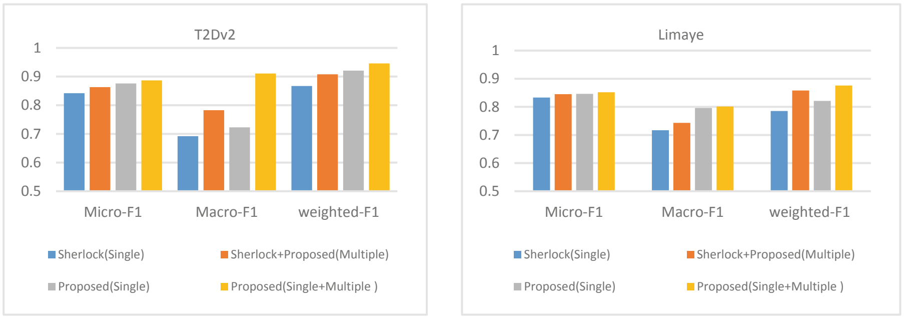


Fig. 6. Comparison of F1 scores of our algorithm and other algorithms on T2Dv2 and Limaye.

Sliding Window Size Impact: When embedding the text data in the character column, we use the sliding window method to combine adjacent cells together to form a fixed-size text. We set 5 different sizes to determine the effect of the sliding window size on the result. As can be seen from Fig. 7, the larger the sliding window, the higher the accuracy of the classification. This situation shows that combining multiple cells together for embedding is more conducive to model building local features. However, when the sliding window value is too large, considering the small size of some web tables, random data may be introduced to reduce the accuracy.

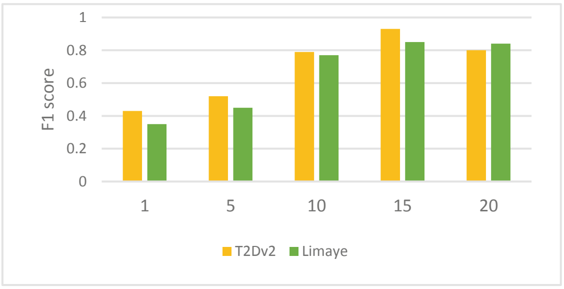


Fig. 7. The effect of sliding window size on character data column type detection

7 Conclusion and Outlook

In view of the problem of semantic type detection of data columns in web tables, this paper proposes a method that combines deep learning and probability graph model. This method uses deep learning model to fully extract the semantic relationship features that exist in single column. The probability graph model is used to take the relationship between columns into account. Experimental results show that the method we proposed can achieve satisfactory results on two real-world data sets, and is significantly better than other comparison methods in accuracy. Next, on the basis of this article, we will conduct a deeper study on how the relationship between the rows and columns of the table affects the classification results. Furthermore, it is an important direction in the future to develop an iterative process during the classification.

Acknowledgment. This work is supported by the National Key R&D Program of China (2018YFB1003404) and the National Natural Science Foundation of China (61672142).

References

1. Shuo, Z., Krisztian, B.: Web table extraction, retrieval, and augmentation: a survey. ACM. Trans. Intell. Syst. Technol. **11**, 2, Article 13, 35 (2020)
2. Michael, C., Hongrae, L.: Ten Years of Web Tables. PVLDB, **11**(12), 2140–2149 (2018). http://doi.org/10.14778/3229863.3240492
3. Sun, H.: Table cell search for question answering. In: Proceedings of the 25th International Conference on WWW, pp. 771–782 (2016). https://doi.org/10.1145/2872427.2883080
4. Ritze, D., Lehmberg, O.: Profiling the potential of web tables for augmenting cross-domain knowledge bases. In: Proceedings of the 25th International Conference on World Wide Web, pp. 251–261 (2016). https://doi.org/10.1145/2872427.2883017
5. Yoones, A., Paolo, M.: Knowledge base augmentation using tabular data. In: Prof. of WWW 2014 (2014)
6. Zwicklbauer, S., Einsiedler, C., Seifert, C.: Towards disambiguating web tables. In: International Semantic Web Conference, pp. 205–208 (2013)
7. Zhang, Z.: Effective and efficient semantic table interpretation using tableminer + . Semantic Web, **8**(6), 921–957 (2017). https://doi.org/10.3233/sw-160242
8. Efthymiou, V., Hassanzadeh, O., Rodriguez-Muro, M., Christophides, V.: Matching web tables with knowledge base entities: from entity lookups to entity embeddings. In: d'Amato, C., Fernandez, M., Tamma, V., Lecue, F., Cudré-Mauroux, P., Sequeda, J., Lange, C., Heflin, J. (eds.) ISWC 2017. LNCS, vol. 10587, pp. 260–277. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68288-4_16
9. Limaye, G., Chakrabarti, S.: Annotating and searching web tables using entities, types and relationships. Proc. VLDB Endowment **3**(1–2), 1338–1347 (2010)
10. Mulwad, V.: Using linked data to interpret tables. Proc. First Int. Workshop Consum. Linked Data (2010). https://doi.org/10.13016/M2NS0M24R
11. Bhagavatula, C., Noraset, T., Downey, D.: Tabel: entity linking in web tables. In: International Semantic Web Conference, pp. 425–441 (2015)
12. Venetis, P., Halevy, A., Wu, C.: Recovering semantics of tables on the web. In: Proc. VLDB, pp. 528–538 (2011). https://doi.org/10.14778/2002938.2002939
13. Krishnamurthy, S., Pedro, S.: Assigning semantic labels to data sources. In European Semantic Web Conference. Springer, pp. 403–417(2015)
14. Minh, P., Suresh, A., and Pedro, S.: Semantic labeling: a domain-independent approach. In International Semantic Web Conference. Springer, pp. 446–462(2016)
15. Jiaoyan, C., Ernesto, J.: ColNet: embedding the semantics of web tables for column type prediction. AAAI (2018). https://doi.org/10.1609/aaai.v33i01.330129
16. Hulsebos, M., K. Z. Hu.: Sherlock: a deep learning approach to semantic data type detection. In: KDD, pp. 1500–1508 (2019). https://doi.org/10.1145/329250
17. Quoc, L., Tomas, M.: Distributed representations of sentences and documents. In: International Conference on Machine Learning. pp. 1188–1196 (2014)

18. Ritze, D., Lehmberg, O., Bizer, C.: Matching html tables to dbpedia. In: Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics, p. 10. ACM (2015). https://doi.org/10.1145/2797115.2797118
19. Xu, B., Yan, S., Yang, D.: BiRNN-DKT: transfer bi-directional LSTM RNN for knowledge tracing. In: Ni, W., Wang, X., Song, W., Li, Y. (eds.) WISA 2019. LNCS, vol. 11817, pp. 22–27. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30952-7_3