Semantic Matching over Matrix-Style Tables in Richly Formatted Documents

Hongwei Li[1,2⋆], Qingping Yang[1,2⋆], Yixuan Cao[1,2], Ganbin Zhou[3], and Ping Luo[1,2]

[1] Key Lab of Intelligent Information Processing of Chinese Academy of Sciences (CAS), Institute of Computing Technology, CAS, Beijing 100190, China
{lihongwei, luop}@ict.ac.cn
[2] University of Chinese Academy of Sciences, Beijing 100049, China
[3] Search Product Center, WeChat Search Application Department, Tencent, China

Abstract. Table is an efficient way to represent a huge number of facts in a compact manner. As practitioners in the vertical domain share lots of common prior knowledge, they tend to represent facts more concisely using *matrix-style tables*. However, such tables are originally intended for human reading, but not machine-readable due to their complex structures including row header, column header, metadata, external context, and even hierarchies in headers. In order to improve the efficiency of practitioners in mining and utilizing these matrix-style tables, in this study we introduce a challenging task to discover *fact-overlapping relations* between matrix-style tables. This relation focuses on fine-grained local semantics instead of overall relatedness in conventional tasks. We propose an attention-based model for this task. Experiments reveal that our model is more capable of discovering the local relatedness, and outperforms four baseline methods. We also conduct an ablation study and case study to investigate our model in detail.

Keywords: matrix-style tables · semantic matching · fact-overlapping relations · richly formatted documents

1 Introduction

Tables, as a compact representation of data, are widely used on the Web and in vertical domains. Mining the relationships between tables has its value in a range of applications, such as table retrieval [22], knowledge base construction, entity disambiguation, and intelligent reading [1, 2, 10, 12]. Taking intelligent reading [2, 12] as an example, professional documents in vertical domains are usually hundreds of pages long with dozens of tables. These tables are related to each other so that they can provide coherent evidence to support the arguments in the document. However, the related tables might scatter over hundreds of pages, and tracking these linked content requires flipping back and forth in the current reading experiences. It is even more cumbersome for reading on a digital

⋆ equal contribution

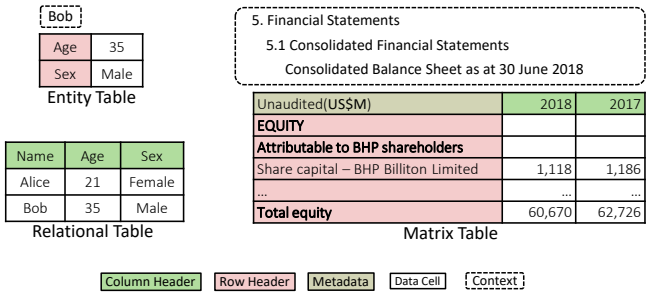

Fig. 1. Examples of entity, relational and matrix tables.

device than using a physical document. Hence, relating tables and displaying them dynamically based on the readers' goal will greatly enhance the reading efficiency.

Distinguished by how the information in the table is organized, there are *entity* tables, *relational* tables, and *matrix-style tables*[1](exemplified in Fig. 1). Matrix tables have a more concise layout than other kinds of tables. Practitioners in vertical domains usually put plenty of data and facts into matrix tables in richly formatted documents [28]. For the public disclosure documents from the financial area, the proportion of matrix tables is as high as 90% based on our empirical study. Therefore, in this paper, we study the relationships over matrix tables within a richly formatted document.

As shown in Fig. 1, a matrix table usually consists of five *components*: *context*, *metadata*, *column headers*, *row headers*, and *data cells* (see the legend for each component in Fig. 1). Here, the *metadata*, *column headers*, *row headers* and *data cells* are the areas inside a table [9, 23]. And the *context* is the outside-table context, including all the text along the path from the root to this table leaf node in the tree of logical document hierarchy [17]. Since there exist approaches to identify the table components [16, 20, 23], in this study, we assume that all these table components are extracted in some predecessor steps.

Additionally, each data cell in a matrix table refers to a *fact* whose complete semantics is scattered in multiple table components. For example, in Fig. 2, the fact expressed by the data cell in the dashed box is shown at the bottom. It is a complex composition of several cells from context, metadata, column and row headers. Thus, a matrix table T conveys a set of facts. Their values lie in data cells, while their semantics are presented in the table context, metadata, row and column headers succinctly.

In this paper, we study the semantic matching problem over matrix tables within a document. Our goal is to determine whether two tables have *fact-overlapping relations*. Specifically, if two tables have some facts in common, they have fact-overlapping relations. For example, the two tables from document *BHP Annual Report 2018* are shown in Fig. 3, where the left one is the *consolidated*

[1] *matrix tables* for short in the following of this paper.

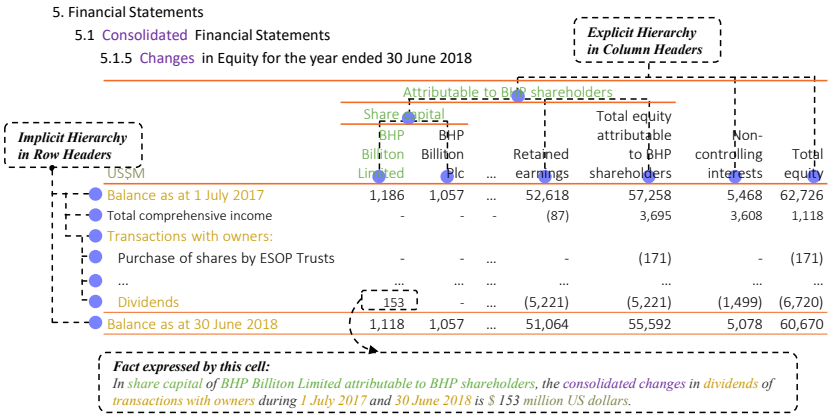

Fig. 2. A table is a set of facts. The fact of a data cell is shown at the bottom, which involves many cells in multiple components, shown in different colors. The row and column headers have hierarchical structures, shown in dotted lines.

balance sheet, and the right one is the *consolidated statement of changes in equity*. They are semantically matched as the facts of data cells marked in boxes exist in both tables.

Matching the two tables in Fig. 3 has its practical value. In financial area, people read disclosure documents with different purposes. For investors, they read documents to learn about the operation of the company. When looking at the retained earnings in the left table, readers want to investigate how it changed during the year, which is detailed in the right table. For financial practitioners, they need to ensure the correctness and consistency of the disclosure of the company's financial position. Therefore, they have to cross-check the numbers in the left table with the numbers in the right table. Both groups of people want to link these two tables together to facilitate their reading process and to avoid jumping back and forth to find relating tables in a document of hundreds of pages. To this end, we propose to equip readers with a kind of *table-linking* functionality. When clicking on a table, a list of tables having fact-overlapping relations with that table is shown to the user as a sidebar on the right.

Simply linking two tables with at least one data cell with the same value cannot solve this problem, since the same value might refer to different facts and some mistaken data cells with different values might refer to the same fact [4]. Hence, we need to model the semantics of the facts inside matrix tables. The challenges of this task are summarized in three aspects.

First, the same fact can be expressed differently in terms of the table layout and its surface form. For example, for the common fact with value "1,118" in Fig. 3, the date of the fact is contained in its column header and context in the left table, while it is contained in its row header in the right one.

Second, the fact-overlapping relations focus on the semantic similarity at the fact level locally. For example, the two tables shown in Fig. 3 are very different

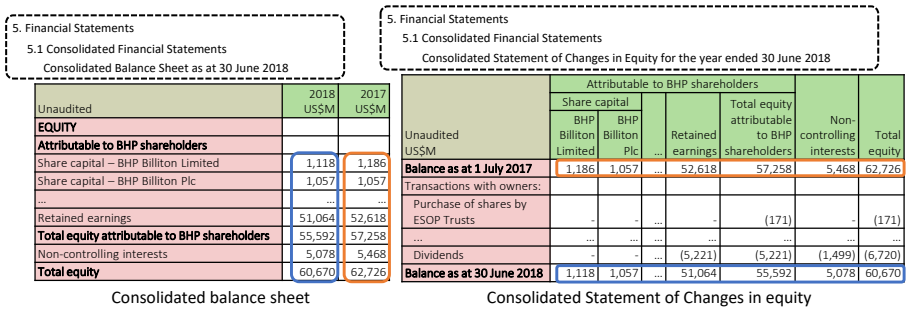

Fig. 3. Two semantically matched tables. The overlapping facts are shown in the boxes.

on the whole, but they have overlapping facts. In extreme cases, there could be only one overlapping fact in two semantically matching tables. This requires to retain the detailed local information when modeling the table. However, previous studies in semantic matching usually consider global relatedness in the sense that two objects have similar meanings on the whole, including matching between sentences [15, 18, 26], documents [29], images [30] and tables [1, 10, 22].

Third, understanding the exact fact of each data cell is challenging since matrix tables have complex structures so that the meaning of facts scatters over multiple components as we mentioned before. Moreover, row and column headers might have *implicit* and *explicit* hierarchies. Take Fig. 2 as an example. The implicit hierarchy of row headers has three levels, conveyed by their visual cures (e.g. font styles, indentation) and text semantics without a unified standard [5]. Its explicit hierarchy of column headers also has three levels and is presented by the internal table structure with merged cells. With such hierarchies, the integrates of a fact might involve several cells in column and row headers. In Fig. 2, the fact of the data cell in the dashed box involves four cells in the row headers (in orange) and three cells in the column headers (in green).

To preserve the meanings of table facts while tolerating the diverse expressions in tables, we propose an attention-based method for fact-overlapping matching between matrix tables. It employs a deep neural network that consists of two components: table embedding network, and symmetric matching network. The input of this neural network is a pair of tables. First, the embedding network calculates the embedding of each table by considering its 4 components, namely context, metadata, (hierarchical) column headers, (hierarchical) row headers. Then, a symmetric matching network entangles these two embeddings to discover all their local fact-level relatedness and predicts whether these two tables are semantically matched or not. Experiments reveal that our attention-based method is more suitable for this task than four baseline methods with 0.75, 0.77, 0.67 and 0.29 absolute improvement on F_1 respectively. Moreover, our attention-based method has a certain ability to explain why and where two tables are semantically matched, which is illustrated in Section 5.3. We also

conduct an ablation study to check the importance of each table component and some case studies to investigate our model in detail.

2 Related Work

We introduce the work on semantic matching of two tables as follows. Sarma et al. [22] proposed a table retrieval task that given a query table, retrieved the most similar tables. They took each table as a set of attributes (i.e. column headers in relational table) and linked the attributes of two tables to form a weighted bipartite graph. Then, they used the max-weight matching as matching scores of two tables. We consider this method as a baseline in our experiments. Fetahu et al. [10] proposed a deep learning-based method to recognize two types of table relations: equivalent and subPartOf. But they only focus on the relational tables with column headers, and their method is not suitable for matrix tables. Besides the matching of tables as a whole, some studies on schema matching [3,19] concentrated on the correspondence of columns between two tables [11]. Zhang and Chakrabarti [32] computed semantic matching between columns of web relational tables. If two tables describe different attributes of the same set of entities, they can be used to augment the attributes of entities. The existing studies on the relationships between tables mainly focus on entity and relational tables, since these two types of tables with simple structures are prevalent on the Web (accounting for 98.3% of tables on Web according to Web Data Commons [14]).

Other studies matched tables to other things like query or knowledge base [7, 24, 33]. Zhang and Balog [33] represent text query and table as a set of vectors of words respectively and compute their similarity. We adopt this method as a baseline in the experiments. Additionally, matching between a table and a knowledge base establishes the mapping between the entities described by them in order to understand the table data [21, 34].

There are also some related studies about table classification. Tables have various layouts, and there are many standards to categorize them. For practical purposes, Ahmadov et al. [1] categorized tables to five classes: relational, entity, matrix, layout, and others. In this paper, we adopt this taxonomy and focus on matrix tables as it is prevalent in vertical domains. Also, Wang et al. [27] proposed the taxonomy with three table classes: 1-dimensional tables, 2-dimensional tables, and complex tables. Considering the layout and structure of tables, Crestan and Pantel [6] proposed a more fine-grained classification, which classifies tables into two broad categories: *relational knowledge* and *layout*. Furthermore, based on structural characteristics, Lautert et al. [13] divided relational knowledge tables into concise, nested, multi-valued, and split tables.

3 Network Architecture

As we have discussed in Section 1, understanding the exact fact of each data cell is challenging. Therefore, we model whole semantics of tables to determine fact-overlapping relations end to end. Our Semantic Matching (SM) method

employs an attention-based deep neural network that consists of two components: table embedding network and symmetric matching network. The input of this neural network is a pair of tables. First, the embedding network calculates the embedding of each table, which preserves its local semantic information. Then, these two table embeddings are fed into the symmetric matching network that predicts whether these two tables are semantically matched or not. Our symmetric matching network leverages an attention encoder to discover the local semantic matching on the fine-grained level between tables, and leverages another attention encoder to aggregate the local matched semantics for classification. We adopt supervised learning to train this deep neural network model with cross-entropy as the loss function.

3.1 Table Embedding

Since the local semantics of tables are essential to the task of determining fact-overlapping relations, we encode one table to a sequence of cell embeddings instead of a fixed-length embedding. We encode four components of a table to form its embedding sequence: row headers, column headers, metadata, and context, as described in Section 1. Each component contains some cells (we call each heading in a header cell and also each title in the table context as a "cell" for convenience). We first encode each cell into a vector, add component embeddings, and then serialize vectors of four components into a sequence of vectors as the embedding of the table. Such embeddings preserve the local information in the table.

The first step is cell embedding. For row headers or column headers with explicit hierarchy (like the column headers in Fig. 2), we extend the text of a leaf cell by joining the texts of cells on the path from root to leaf with a special token "&". For example, the extended text of the cell in the second column header in Fig. 2 is

Attributable to BHP shareholders & *Share capital* & *BHP Billiton Plc*

As you can see, each joined text contains the complete local and hierarchical information.

As each leaf node corresponds to a column or row in the table, and we have extended its text to incorporate the hierarchy, we only use the leaf cells for table embedding. A transformer [25] encodes the (extended) text of each leaf cell into a text embedding e^t. Moreover, to distinguish cells among different components, we introduce component embeddings e^s for each component (like the segment embedding in [8]). Finally, cell embedding is

$$e = e^t + e^s \in \mathbb{R}^{d_m},$$

where d_m is the dimension of embedding.

The second step is table embedding. We do not compress the table into a fixed-length embedding as it will lose the information of individual cells. Instead, the table embedding is a sequence of cell embeddings. In each component, we

stack the leaf cells by order. Then, we stack these four components into an embedding sequence

$$E_T = [e_{c_1}, ..., e_{c_{n_c}}, e_{r_1}, ..., e_{r_{n_r}}, e_{x_1}, ..., e_{x_{n_x}}, e_{m_1}, ..., e_{m_{n_m}}] \in \mathbb{R}^{n \times d_m},$$

where e_{c_i} is the embedding of the i-th cell in column headers (r for row headers, x for context and m for metadata), $n = n_c + n_r + n_x + n_m$ is the number of leaf cells in the four components. Note that we only encode the four components in a table and ignore data cells, since data cells, usually containing digits, are not semantically expressive without the other four table components.

3.2 Symmetric Matching Network

The above table encoding preserves the local semantic information of the table to the maximum extent so that we can explicitly carry out the local semantic interaction between tables in the symmetric matching network. The symmetric matching network takes as inputs two table embeddings, E_{T_1} and E_{T_2}, and outputs the probability that they are matched. First, we entangle the embeddings of these two tables, to get E'_{T_1} and E'_{T_2}, where each cell of T_1 is informed of (by attention on) every cell in T_2 and vice versa. This allows the local semantic matching between two tables. Then, we aggregate E'_{T_1} and E'_{T_2} respectively to get two vectors e_{T_1} and e_{T_2}, which gathers the local matched semantics. Finally, the prediction layer outputs the probabilities. Both entangling and prediction layers are symmetric with regard to T_1 and T_2.

The building block of our network is an attention encoder. In brief, an attention encoder $Q' = \mathit{Attention}(Q, V)$ uses Q, V as input and gets a new embedding Q', where Q and $Q' \in \mathbb{R}^{l \times d_m}$ and $V \in \mathbb{R}^{m \times d_m}$. This attention encoder $\mathit{Attention}(Q, V)$ will be detailed in Section 3.3.

In entangling layer, taking E_{T_1} and E_{T_2} as Q and V respectively we get E'_{T_1}:

$$E'_{T_1} = \mathit{Attention}_1(E_{T_1}, E_{T_2}). \tag{1}$$

The attention encoder allows each cell in table T_1 to interact with every cell in table T_2. This is essential to discover the local semantic matching on the fine-grained level between tables. We will show the effectiveness of this encoder in the case study. Symmetrically, we get $E'_{T_2} = \mathit{Attention}_1(E_{T_2}, E_{T_1})$.

In the aggregation layer, a learnable special embedding vector $e_{[TAB]}$ is used to aggregate the information of a table:

$$e^*_{T_1} = \mathit{Attention}_2(e_{[TAB]}, E'_{T_1}). \tag{2}$$

Here, $e^*_{T_1}$ is an aggregated vector representing table T_1 after attention on table T_2. Using the attention allows this aggregation to focus on the local matched semantics flexibly instead of rough semantics such as *mean* or *max*. We also show the effectiveness of this encoder for capturing local matched semantics in the case study. Similarly, we get $e^*_{T_2} = \mathit{Attention}_2(e_{[TAB]}, E'_{T_2})$.

In the prediction layer, we concatenate $e^*_{T_1}$ and $e^*_{T_2}$ in both orders and compute e_O as follows:

$$e_O = \text{ElementwiseMax}(FFN(\text{concat}(e^*_{T_1}, e^*_{T_2})), FFN(\text{concat}(e^*_{T_2}, e^*_{T_1}))) \quad (3)$$

where FFN is a feed-forward network, and the element-wise maximum of the two vectors ensures that this matching network is symmetric. Finally, e_o is used for classification. The symmetric property of our model ensures that no matter the input is (E_{T_1}, E_{T_2}) or (E_{T_2}, E_{T_1}), the result will be the same.

3.3 Attention Encoder

The attention encoder is the building block of the symmetric matching network. It takes as input Q and V, and outputs Q':

$$Q' = Attention(Q, V) \quad (4)$$

where Q and $Q' \in \mathbb{R}^{l \times d_m}$, $V \in \mathbb{R}^{m \times d_m}$. The attention encoder consists of b Layers ($Layer_1, ..., Layer_b$) in sequence:

$$Q_i = Layer_i(Q_{i-1}, V) \quad (i = 1, \cdots, b) \quad (5)$$

where $Q_0 = Q$ as the initial input and $Q' = Q_b$ as the final output. Each layer consists of Multi-head Attention Layer (MAL) and Feed-Forward Network (FFN). There is Residual Connection (RC) and Layer Normalization (LN) following both MAL and FFN. The detailed structure of $Layer_i$ is as follows:

$$Z_{i,1} = MAL_i(Q_{i-1}, V) \quad (6)$$
$$Z_{i,2} = LN_{i,1}(Q_{i-1} + Z_{i,1}) \quad (7)$$
$$Z_{i,3} = FFN_i(Z_{i,2}) \quad (8)$$
$$Q_i = LN_{i,2}(Z_{i,2} + Z_{i,3}) \quad (9)$$

MAL consists of multiple attention operations. It concatenates the results of these operations to represent a richer attention. The definition of MAL_i is as follows:

$$MAL_i(Q_{i-1}, V) = \text{Concat}(H_{i,1}, \cdots, H_{i,h})W^O_i \quad (10)$$
$$H_{i,j} = W_{i,j}(VW^{V_1}_{i,j}) \quad (11)$$
$$W_{i,j} = \text{Softmax}\left(\frac{(Q_{i-1}W^Q_{i,j})(VW^{V_2}_{i,j})^T}{\sqrt{d_v}}\right) \quad (12)$$

where $H_{i,j}$ is the j-th head of the multi-head attention in the i-th layer, and h is the number of heads. The projections are parameter matrices $W^Q_{i,j}, W^{V_1}_{i,j}, W^{V_2}_{i,j} \in \mathbb{R}^{d_m \times d_v}$ and $W^O_i \in \mathbb{R}^{(h \cdot d_v) \times d_m}$. And we let $d_v = d_m/h$.

The attention weights can reveal the relations among elements in two inputs, thus give an interpretation of the prediction result. To analyze the effectiveness

of our methods in Section 5.3, we define the *summary weights* of the attention encoder as:

$$SW = \sum_{j=1,\cdots,h} W_{b,j} \tag{13}$$

where SW is a vector, and its k-th dimension $SW_k \in [0, h]$ indicates the importance of the k-th cell in V to Q. The greater the value of SW_k, the more important the k-th cell in V is to Q. Here, b is the index of the last layer, indicating that we only display the weights of the last attention layer.

4 Experimental Setup

4.1 Dataset

We downloaded public annual reports from CNINFO[2]. They are all long documents with on average 71.12 pages and 100.8 tables per document. The average numbers of cells in row header, column header, context and metadata per table are 7.91, 4.22, 1.84 and 0.18 respectively. We annotated the matching relations among tables in each document, which took three financial practitioners more than a month to complete. In a document, for each table, annotators find out all tables that match with this table, and other table pairs that do not match are set to negative samples. They do not annotate which parts of table facts are overlapping as it is laborious. There are 358,111 table pairs in total in these documents. And the ratio of matching and not matching pairs is 1:66. All the table pairs are randomly divided into a training and test dataset by 9:1.

4.2 Baselines

There are no previous studies that can be directly applied to our task. Thus, we adapt some of them to build the following four baseline methods.

Term-based Schema Matching (TSM). We represent each table by bag-of-words of its four components[3]. The matching score of two tables is the Jaccard index of their token sets. Two tables are matched if the matching score is greater or equal than a threshold.

Embedding-based Schema Matching (ESM). Using similar methods in TSM, we represent each table as a set of tokens in its four table components. Then, we build a bipartite graph where an edge links each pair of tokens from the two sets. The edge weight is computed as the cosine similarity between the embeddings of the two tokens. Finally, the max-weight matching score of the bipartite is used as the similarity between the two tables [22]. We regard two tables as a positive sample if their similarity is more than a threshold. In detail, we use the method in [31] to train the token embeddings.

Learning-based Schema Matching (LSM). We perform a learning-based baseline. In this baseline, we convert each table to a binary vector B whose

[2] `http://www.cninfo.com.cn`. A financial information disclosure website.

[3] We tokenize each cell using Jieba, a popular Chinese word segmentation toolkit.

length is equal to vocabulary size. If a word w_i appears in the table and its index in the vocabulary is i, we set B_i to 1, otherwise 0. For a pair of tables, we concatenate the two table vectors as $B_p = [B_1; B_2]$. Then B_p is regarded as the input feature, the relation between the two tables is regarded as the label, which are put into a logistic regression classifier for training.

Semantic Matching without Attention Encoder (SM$^-$). The above three methods use the bag-of-words model. In this baseline, we first use the *table embedding* component (detailed in Section 3.1) to preserve both the term sequence in each table cell and the hierarchies in row and column headers. Then, the *symmetric matching network* is replaced with the following component [33]: four similarity scores (*Early, Late-max, Late-sum, Late-avg*) between two sets of vectors are computed, and used as the features for classification of matching or not.

4.3 Experimental Settings

We use the *Precision*, *Recall*, and F_1 to evaluate the above methods. Each feed-forward network is a fully connected layer with two linear transformations ($d_m \times 2d_m$ and $2d_m \times d_m$) and ReLU in between. We apply Adam for optimization. 10% of the training data are reserved as the validation set to select the best hyper-parameters. As a result, the number of blocks b in each cross encoder is 3 (over 1, 3, 5); the dimension of embeddings d_m is 128 (over 128, 256, 512); the number of heads h in each multi-head attention is 4 (over 4, 8); and the dimension of each head d_v is 32 (over 32, 64); and the learning rate is 10^{-4} (over 10^{-4}, 10^{-3}, 10^{-2}).

5 Experimental Results

5.1 Results of Different Methods

Table 1. Table matching results on the test set.

Model	*Pre.*	*Rec.*	F_1
TSM	0.1016	0.0841	0.0920
ESM [22]	0.0668	0.0682	0.0675
LSM	0.0944	0.7628	0.1680
SM$^-$ [33]	0.6920	0.4559	0.5496
SM	**0.8685**	**0.8090**	**0.8376**

Table 1 shows the results of table matching of each method on the test dataset. The F_1 score of TSM and ESM method peaks at threshold 0.5, 0.8099 in the training set respectively. Their corresponding F_1 on the test set are 9.2% and 6.75%. The learning-based method LSM also has only achieved 16.8% F_1.

The poor performance of these three methods indicates that similarity on terms is not adequate for this problem. SM^- method has great improvement comparing with TSM, ESM, and LSM methods. Because it takes into account both the term sequence in each table cell and the hierarchies in row and column headers. This proves the effectiveness of our table embedding network. However, as the SM^- method does not focus on the local semantics, its performance is still poor in this task. SM performs best among these models. The absolute F_1 improvement is 74.56%, 77.01%, 66.96% and 28.80% compared with TSM, ESM, LSM and SM^-. Compared with SM^-, our SM model takes more attention on local semantics by our symmetric matching network. Supported by the fact of a larger improvement in recall than precision, we argue that SM is more capable of discovering overlapping facts at the local level.

5.2 Ablation Study

Table 2. Ablation study on components, layout and symmetric model design.

Ablation	*Pre.*	*Rec.*	F_1
SM	0.8685	0.8090	0.8376
– Column headers	0.8391	0.7971	0.8176
– Metadata	0.8273	0.8076	0.8173
– Context	0.6574	0.6548	0.6561
– Row headers	0.6540	0.5455	0.5948
– Headers	0.6553	0.7088	0.6810
– Hierarchy	0.8499	0.7984	0.8234
– Component embedding	0.8399	0.8090	0.8242
– Symmetric	0.8630	0.7721	0.8150

To analyze the effectiveness of each component of tables for our model, we ablate each component of tables in the SM method respectively. As shown in Table 2, we list components by F_1 score after ablation in descending order. When we ablate column headers, F_1 only decreases by 2%. By analyzing the dataset, we find most of the column headers are about the time, like 2016, 2017. As one document usually describes financial position within the same period of time, removing column headers has less impact on semantic matching over tables. Metadata also only has little impact on performance, since the metadata appears less frequently in the dataset and usually describes the unit of data cells. However, as removing column headers and metadata mainly deteriorates the precision, we think they provide information to filter out false-positive samples, but less information for matching. When we ablate context, F_1 decreases by 18.15%, this means that it is important information for this task. Also, ablating row headers will decrease F_1 by 24.28% and severely hurt the recall (to 54.55%). So, row headers are very important in discovering matching information.

The result of ablating both column and row headers are shown in the "-Headers" row, which uses only metadata and context (MX). We denote the model ablating only row headers (retaining column headers, metadata, and context) as CMX. An interesting phenomenon is that although with more information (column headers), CMX performs poorer than MX (59.48% vs. 68.10% on F_1). And the difference comes from recall (54.55% vs. 70.88%). We argue that this is because the CMX model overfits the training set on the column information. As discussed above, row headers contain vital information for matching. Thus, both CMX and MX, without row headers, do not have enough information for matching. As the column contains little information for matching, it might become noise for the CMX model. To examine this, we report the result on the training set: the recall of the CMX is higher than (86.11% and 80.56%). It means that the CMX tries to use column headers to improve its recall which did not generalize well. And that results in overfitting and worse results on the test set.

We also study our modeling on the table layout. The first is the hierarchy of headers. For each cell on the tree, we directly use its text (not extended) for text embedding. All cells including non-leaf nodes are used. The result is shown in the "-Hierarchy" row. The F_1 score drops by 1.42% after removing hierarchical information. The second is the component type, which is shown in the "-Component embedding" row. We observed that the recall of the model did not decrease, but the precision drops by 2.86%. The lacking of hierarchical information and component type decreases the effectiveness of the model, which indirectly proves that the table layout affects the fact-overlapping relations. Therefore, it is necessary to keep the layout information of the table in table embedding.

We also test the symmetric design in the model by replacing symmetric E_O with an asymmetric one using

$$e_O = FFN(\text{concat}(e^*_{T_1}, e^*_{T_2})) \tag{14}$$

instead of Formula (3) in SM. Thus, changing the order of two tables may lead to different results in this model. We select the one with the highest confidence as the result of a table pair. The result is shown in the "-Symmetric" row. The symmetric design outperforms the asymmetric model by 2.26% on F_1.

5.3 Case Study

We show two cases in Fig. 4 to show that the SM can correctly attend on cells related to their overlapping facts. The SM model correctly predicts that both cases have fact-overlapping relations. We show the table pair A and B in each case and visualize the summary weights defined in Formula (13). In table A, we show the summary weight on each cell in $Attention_2$ for $e_{[TAB]}$. In table B, we show the weight on each cell in $Attention_1$ for a specific cell in table A. Cells with darker shading have larger weights.

In the first case, Table A is a detailed sheet about "Surplus reserve" with only one row "Statutory surplus reserve". Meanwhile, "Surplus reserve" is one of

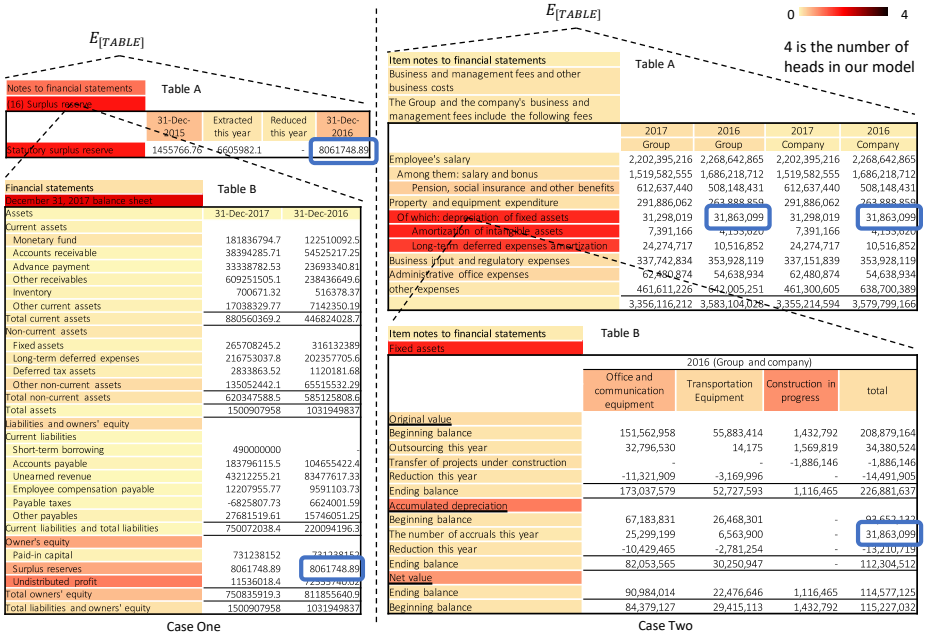

Fig. 4. The illustration of the interpretability of SM method by two cases. The selected data cells have the same fact in each case.

the row headers in Table B. Weights in Table A indicate that "(16) Surplus reserve" and "Statutory surplus reserve" in Table A are important for classification. So, in table B, we show the weight of each cell corresponding to "(16) Surplus reserve". It demonstrates that "December 31, 2017 balance sheet", "Owner's equity", "Surplus reserves" and "Undistributed profit" are important. That means our model attends to the correct cells.

In the second case, Table B details the "Fixed assets" that contains the changes in depreciation of fixed assets. Meanwhile, "Of which: depreciation of fixed assets" is one of the row headers in Table A. Weights in Table A indicate that "Of which: depreciation of fixed assets", "Amortization of intangible assets" and "Long-term deferred expenses amortization" in Table A is important for classification. Then, in Table B, we show the importance of each cell corresponding to "Of which: depreciation of fixed assets". It demonstrates that the existences of "Fixed assets" and "Accumulated depreciation" in Table B cause the importance of "Of which: depreciation of fixed assets" in Table A for classification.

6 Conclusion

Automatically mining the relations among tables can support multiple applications. To this end, we propose an attention-based method to solve the problem of

semantic matching over matrix tables. Our method consists of a table embedding that preserves local semantic information of tables, and a symmetric matching network that can discover the local semantic matching on the fine-grained level between tables and aggregate the local matched semantics for classification. Experiments reveal that the method works well in the local semantic similarity task despite the diverse expressions of facts and complex layout of tables. Meanwhile, through case studies, we demonstrate its ability to explain why two tables are semantically matched.

7 Acknowledgements

This work was supported by the National Key Research and Development Program of China under Grant No. 2017YFB1002104, the National Natural Science Foundation of China under Grant No. U1811461, and the Innovation Program of Institute of Computing Technology, CAS.

References

1. Ahmadov, A., Thiele, M., Eberius, J., Lehner, W., Wrembel, R.: Towards a hybrid imputation approach using web tables. In: BDC. pp. 21–30 (Dec 2015)
2. Badam, S.K., Liu, Z., Elmqvist, N.: Elastic documents: Coupling text and tables through contextual visualizations for enhanced document reading. TVCG **25**(1), 661–671 (2019)
3. Bernstein, P.A., Madhavan, J., Rahm, E.: Generic schema matching, ten years later. Proceedings of the VLDB Endowment **4**(11), 695–701 (2011)
4. Cao, Y., Li, H., Luo, P., Yao, J.: Towards automatic numerical cross-checking: Extracting formulas from text. In: WWW. pp. 1795–1804 (2018)
5. Chen, X., Chiticariu, L., Danilevsky, M., Evfimievski, A., Sen, P.: A rectangle mining method for understanding the semantics of financial tables. In: ICDAR. pp. 268–273 (nov 2017)
6. Crestan, E., Pantel, P.: Web-scale table census and classification. In: WSDM. pp. 545–554 (2011)
7. Deng, L.: Table2Vec: Neural Word and Entity Embeddings for Table Population and Retrieval. Master's thesis, University of Stavanger, Norway (2018)
8. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. In: NAACL-HLT. pp. 4171–4186 (2019)
9. Fang, J., Mitra, P., Tang, Z., Giles, C.L.: Table header detection and classification. In: AAAI. pp. 599–605 (2012)
10. Fetahu, B., Anand, A., Koutraki, M.: Tablenet: An approach for determining fine-grained relations for wikipedia tables. In: WWW. pp. 2736–2742 (2019)
11. Ju, F., Lu, M., Ooi, B.C., Tan, W.C., Zhang, M.: A hybrid machine-crowdsourcing system for matching web tables. In: ICDE. pp. 976–987 (2014)
12. Kim, D.H., Hoque, E., Kim, J., Agrawala, M.: Facilitating document reading by linking text and tables. In: UIST. pp. 423–434 (2018)
13. Lautert, L.R., Scheidt, M.M., Dorneles, C.F.: Web table taxonomy and formalization. ACM SIGMOD Record **42**(3), 28–33 (2013)

14. Lehmberg, O., Ritze, D., Meusel, R., Bizer, C.: A large public corpus of web tables containing time and context metadata. In: WWW. pp. 75–76 (2016)
15. Liu, P., Qiu, X., Chen, J., Huang, X.: Deep fusion lstms for text semantic matching. In: ACL. pp. 1034–1043 (2016)
16. Liu, Y., Bai, K., Mitra, P., Giles, C.L.: Tableseer: automatic table metadata extraction and searching in digital libraries. In: Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries. pp. 91–100. ACM (2007)
17. Mao, S., Rosenfeld, A., Kanungo, T.: Document structure analysis algorithms: a literature survey. In: Document Recognition and Retrieval X. vol. 5010, pp. 197–207 (01 2003)
18. Pang, L., Lan, Y., Guo, J., Xu, J., Wan, S., Cheng, X.: Text matching as image recognition. In: AAAI. pp. 2793–2799 (2016)
19. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. the VLDB Journal **10**(4), 334–350 (dec 2001)
20. Rastan, R., Paik, H.Y., Shepherd, J.: Texus: A unified framework for extracting and understanding tables in pdf documents. Information Processing & Management **56**(3), 895–918 (2019)
21. Ritze, D., Lehmberg, O., Bizer, C.: Matching html tables to dbpedia. In: WIMS. pp. 10:1–10:6 (2015)
22. Sarma, A.D., Fang, L., Gupta, N., Halevy, A.Y., Lee, H., Wu, F., Xin, R., Yu, C.: Finding related tables. In: SIGMOD. pp. 817–828 (2012)
23. Seth, S., Nagy, G.: Segmenting tables via indexing of value cells by table headers. In: ICDAR. pp. 887–891 (2013)
24. Sun, Y., Yan, Z., Tang, D., Duan, N., Qin, B.: Content-based table retrieval for web queries. Neurocomputing **349**, 183–189 (2018)
25. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: NIPS. pp. 5998–6008 (2017)
26. Wan, S., Lan, Y., Guo, J., Xu, J., Pang, L., Cheng, X.: A deep architecture for semantic matching with multiple positional sentence representations. In: AAAI. pp. 2835–2841 (2016)
27. Wang, H.L., Wu, S.H., Wang, I., Sung, C.L., Hsu, W.L., Shih, W.K.: Semantic search on internet tabular information extraction for answering queries. In: CIKM. pp. 243–249 (2000)
28. Wu, S., Hsiao, L., Cheng, X., Hancock, B., Rekatsinas, T., Levis, P., Ré, C.: Fonduer: Knowledge base construction from richly formatted data. In: SIGMOD. pp. 1301–1316 (2018)
29. Wu, Z., Zhu, H., Li, G., Cui, Z., Huang, H., Li, J., Chen, E., Xu, G.: An efficient wikipedia semantic matching approach to text document classification. Information Sciences **393**(C), 15–28 (jul 2017)
30. Yu, W., Sun, X., Yang, K., Rui, Y., Yao, H.: Hierarchical semantic image matching using cnn feature pyramid. Computer Vision and Image Understanding **169**, 40–51 (2018)
31. Zhang, L., Zhang, S., Balog, K.: Table2vec: Neural word and entity embeddings for table population and retrieval. In: SIGIR. pp. 1029–1032 (2019)
32. Zhang, M., Chakrabarti, K.: Infogather+: semantic matching and annotation of numeric and time-varying attributes in web tables. In: SIGMOD. pp. 145–156 (2013)
33. Zhang, S., Balog, K.: Ad hoc table retrieval using semantic similarity. In: WWW. pp. 1553–1562 (2018)
34. Zhang, Z.: Towards efficient and effective semantic table interpretation. In: ISWC. pp. 487–502 (2014)