Numerical Formula Recognition from Tables

Qingping Yang[1,2], Yixuan Cao[1,2], Hongwei Li[3], Ping Luo[1,2,4]
[1]Key Lab of Intelligent Information Processing of Chinese Academy of Sciences, Institute of Computing Technology, CAS, Beijing 100190, China
[2]University of Chinese Academy of Sciences, Beijing 100049, China
[3]Research Department, P.A.I. Ltd., Beijing 100025, China
[4]Peng Cheng Laboratory, Shenzhen 518066, China

ABSTRACT

Claims over the numerical relationships among some measures are commonly expressed in tabular forms, and widely exist in the published documents on the Web. This paper introduces the problem of numerical formula recognition from tables, namely recognizing all numerical formulas inside a given table. It can well support many interesting downstream applications, such as numerical error correction in tables, formula recommendation in tables. Here, we emphasize that table is a kind of language that adopts a different linguistic paradigm from natural language. It uses *visual grammar* like *visual layout* and *visual settings* (e.g., indentation, font style) to express the grammatical relationships among the table cells. Understanding tables and recognizing formulas require decoding the visual grammar while simultaneously understanding the textual information. Another challenge is that formulas are complicated in terms of diverse math functions and variable-length of arguments. To address these challenges, we convert this task into a uniform framework, extracting relations of table cell pairs in a table. A two-channel neural network model TaFor is proposed to embed both the textual and visual features for a table cell. Our framework achieves the formula-level F1-score = 0.90 on a real-world dataset of 190,179 tables while a retrieval-based method achieves F1-score = 0.72. We also perform extensive experiments to demonstrate the effectiveness of each component in our model, and conduct a case study to discuss the limits of the proposed model. With our published data this study also aims to attract the community's interest in deep semantic understanding over tables.

CCS CONCEPTS

• **Information systems** → **Information extraction**; *Data extraction and integration*; • **Applied computing** → *Business intelligence.*

KEYWORDS

numerical formulas, tabular presentation, table understanding

ACM Reference Format:
Qingping Yang[1,2], Yixuan Cao[1,2], Hongwei Li[3], Ping Luo[1,2,4]. 2021. Numerical Formula Recognition from Tables. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3447548.3467425

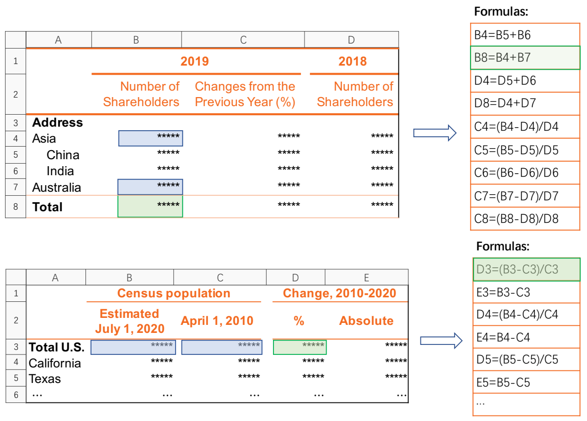
1 INTRODUCTION

Claims over the numerical relationships among some objective measures widely exist in the published documents on the Web. These numerical relationships are expressed not only in verbal descriptions, but also more frequently in tabular forms. For example, various financial documents (e.g. IPO prospectus, bond prospectus, corporate annual report etc.) contain some verbal descriptions over the finance indicators of the corporates. Such verbal descriptions often appear in sentences around tables and describe only a small portion of the data presented in tables. That is to say, formulas are more common in tables.

Figure 1 gives two typical tables with their expressed formulas. One is from a company's annual report, while the other is from the U.S. demographics. For the first table, with the basic understanding of tabular forms and some common sense about geography humans can easily recognize its corresponding formulas. For example, the second formula on its right is `B8=B4+B7`, expressing that the total number of shareholders in 2019 equals to the sum of those in Asia and Australia. For the second table, the first formula on its right is `D8=(B3-C3)/C3`, expressing that the change rate of the total U.S. Census population from 2010 to 2020 equals to the value inside `D8`. Since formulas are quite common in tables, in this study we introduce the problem of numerical formula recognition from tables, namely recognizing all numerical formulas (e.g., the right side in Figure 1) inside a given table (e.g., the left side in Figure 1). To the best of our knowledge, although there are extensive studies dedicated to table understanding [35], this is the first attempt that tackles this problem.

This task opens a wide perspective of interesting downstream applications. A straightforward but meaningful application is *error correction in tables.* Even in published financial documents, which have been reviewed many times, numerical errors caused by formulas are still inevitable. In 2012, since the formulas within the copied cells were not adjusted accordingly, JP Morgan severely underestimated the downside of its synthetic credit portfolio, which led to the bank suffering approximately $6.5 billion in losses and fines[1]. Furthermore, such errors may cause greater disasters in government and academia. In 2013, academic critics had found 3 major errors in the paper "Growth in a Time of Debt" that had been published in 2010 and led to unjustified adoption of austerity policies for countries with various levels of public debt. One of these errors

[1]https://www.bloomberg.com/news/articles/2013-09-19/jpmorgan-chase-agrees-to-pay-920-million-for-london-whale-loss



	A	B	C	D
1		2019		2018
2		Number of Shareholders	Changes from the Previous Year (%)	Number of Shareholders
3	Address			
4	Asia	*****	*****	*****
5	China	*****	*****	*****
6	India	*****	*****	*****
7	Australia	*****	*****	*****
8	Total	*****	*****	*****

Formulas:

B4=B5+B6
B8=B4+B7
D4=D5+D6
D8=D4+D7
C4=(B4-D4)/D4
C5=(B5-D5)/D5
C6=(B6-D6)/D6
C7=(B7-D7)/D7
C8=(B8-D8)/D8

	A	B	C	D	E
1		Census population		Change, 2010-2020	
2		Estimated July 1, 2020	April 1, 2010	%	Absolute
3	Total U.S.	*****	*****	*****	*****
4	California	*****	*****	*****	*****
5	Texas	*****	*****	*****	*****
6	...	...	...	...	...

Formulas:

D3=(B3-C3)/C3
E3=B3-C3
D4=(B4-C4)/C4
E4=B4-C4
D5=(B5-C5)/C5
E5=B5-C5
...

Figure 1: Two example tables with their recognized formulas. The indexes of rows and columns in gray boxes are added to identify different cells. Best viewed in color.

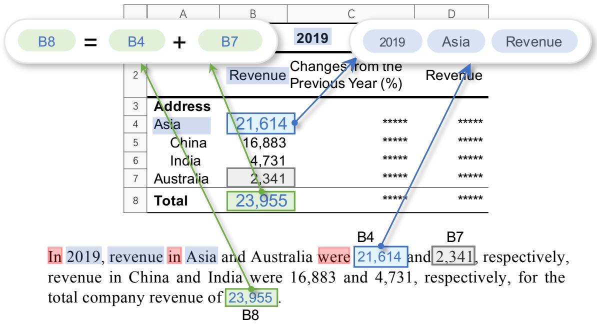
is that an average formula excluded five countries from the list in a table[2]. As we can see, these errors may cause severe consequences, thus they should be thoroughly removed before official publishing. An automatic solution for this cross-check is to first recognize each formula from a table, and then bring the involved values into the formula to verify whether the left and right sides of the formula are equal or not.

Another interesting application is *formula recommendation in tables*. In the process of table editing, after users have filled in the table headers and the overall table layout is developed, we can automatically suggest the formulas among the table cells in real-time. These recommended formulas can guide the user on how the remaining empty cells can be filled, prevent users from making mistakes due to negligence, and improve the effectiveness and efficiency for the table editing process.

There are also extensive studies [21, 26] to show that values in tables are error-prone and at least one error caused by a formula was found in more than 95% of spreadsheets [25]. Thus, all the values and formulas (if some existing) are not reliable to our task. It drives us to only leverage the text and visual appearance of table headers and table layout structure, which are more reliable features for this task. With these reliable features our solution can eliminate the impact of value and formula errors and support a wider range of applications. To emphasize this, all the numerical values in the example tables used in this paper are replaced with asterisks.

In this study, we also emphasize that table is a kind of language that adopts a different linguistic paradigm from natural language. In a sentence, content words (e.g., nouns, most verbs) contribute to the meaning of the sentence and functional words (e.g., preposition, pronouns) express grammatical relationships among content words. Analogously, the words in table cells contribute to the meaning, while *visual layout* (namely table structure) and *visual settings* (e.g., indentation, font style, row height, column width, even the visibility of table lines) express the grammatical relationships among the table cells. Take Figure 2 as an example, for the value in the blue box inside the table, its meaning is determined by words: "2019",

[2]https://en.wikipedia.org/wiki/Growth_in_a_Time_of_Debt



	A	B	C	D
1		2019		
2		Revenue	Changes from the Previous Year (%)	Revenue
3	Address			
4	Asia	21,614	*****	*****
5	China	16,883	*****	*****
6	India	4,731	*****	*****
7	Australia	2,341	*****	*****
8	Total	23,955	*****	*****

In 2019, revenue in Asia and Australia were 21,614 and 2,341, respectively, revenue in China and India were 16,883 and 4,731, respectively, for the total company revenue of 23,955.

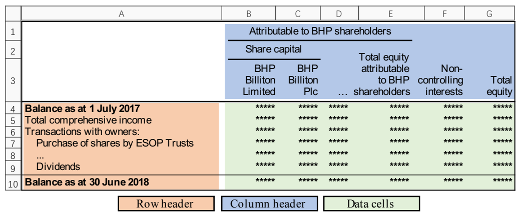
Figure 2: Table is a two dimensional language. Best viewed in color.

"Revenue", and "Asia". The same meaning can also be expressed in natural language with similar words, where these words are connected by functional words like "in", "were", shown in red in Figure 2. However, in tables they are connected by visual layout (e.g., in the same row, column). Moreover, tables have hierarchy among headers, like the syntactic tree of a sentence. It is usually used to express relations like part of the whole. For example, we can infer B8=B4+B7 from the table in Figure 2. However, B8=B4+B5+B6+B7 does not hold since the table expresses that "China" and "India" with the visual setting of indentation belong to "Asia" without the indentation. Therefore, Understanding a table and recognizing its formulas is similar to natural language understanding tasks, but imposes more challenges to crack its *visual grammar* [20].

Text understanding is already a challenging task, but table understanding requires taking visual information into consideration. This poses the first challenge to recognize formulas in tables.

Another challenge is the complexity of formulas. There are many kinds of math functions, like *Subtraction*, *Division*, *Sum*, *Average*. For functions like *Sum* and *Average*, the number of arguments is variable. Moreover, the candidate space is huge (any table cell can be any argument in a formula). In Section 2, we describe these challenges in detail and present some observations on how the text and visual settings affect the table meaning.

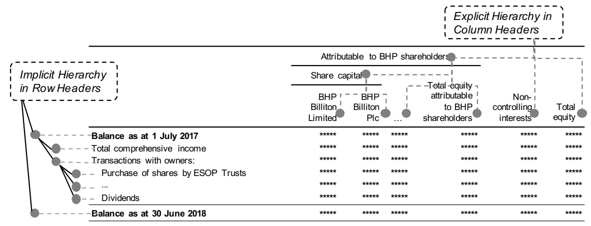
To address these challenges, we convert this task into a uniform framework, extracting relations of table cell pairs in a table. The overall process has two steps: 1) result cell detection: classify whether a cell is a result cell (the left side of a formula); 2) cell pair classification: for each result cell extract its arguments in the right side of a formula. Specifically, a two-channel neural network model is proposed to embed both the textual and visual features for a table cell. It is elaborately designed so that the implicit and explicit *header hierarchy* can be considered. This multi-modality embedding is detailed in Section 3.3. Extensive experiments show that our method outperforms the baseline methods by significant margins from $F1 = 0.72$ to 0.90 on a real-world dataset of 190,179 tables with their labeled formulas. Ablation studies demonstrate the effectiveness of each component in our model, and most importantly that visual features can improve the performance of formula recognition significantly. Furthermore, we provide some case studies to demonstrate the generalization ability and limits of our model. We hope our study can push forward the research on in-depth understanding of table semantics.



	A	B	C	D	E	F	G
1		Attributable to BHP shareholders					
2		Share capital			Total equity		
3		BHP Billiton Limited	BHP Billiton Plc	…	attributable to BHP shareholders	Non-controlling interests	Total equity
4	**Balance as at 1 July 2017**	*****	*****	*****	*****	*****	*****
5	Total comprehensive income	*****	*****	*****	*****	*****	*****
6	Transactions with owners:	*****	*****	*****	*****	*****	*****
7	Purchase of shares by ESOP Trusts	*****	*****	*****	*****	*****	*****
8	...	*****	*****	*****	*****	*****	*****
9	Dividends	*****	*****	*****	*****	*****	*****
10	**Balance as at 30 June 2018**	*****	*****	*****	*****	*****	*****

🟧 Row header	🟦 Column header	🟩 Data cells

(a) Table Areas


Explicit Hierarchy in Column Headers
Implicit Hierarchy in Row Headers
Attributable to BHP shareholders
Share capital
BHP Billiton Limited
BHP Billiton Plc
…
Total equity attributable to BHP shareholders
Non-controlling interests
Total equity
Balance as at 1 July 2017
Total comprehensive income
Transactions with owners:
Purchase of shares by ESOP Trusts
...
Dividends
Balance as at 30 June 2018

(b) Table Hierarchy

Figure 3: A matrix table [22] with its table areas and table header hierarchy. Best viewed in color.

2 FORMULAS IN TABLES

In this section, we describe the complexity of tables and formulas and illustrate the challenges of recognizing formulas in detail by exemplifying some formulas in tables.

2.1 Preliminaries on Tables

Following Eberius et al. [8], tables can be classified into three categories: *Relational, Entity*, and *Matrix*. A relational table is similar to a table in the relational database. It describes a series of similar entities with the same attributes. The name of attributes is usually placed at the beginning of each column which is also called the column header. An entity table describes the information of only one entity and each row is an attribute-value pair. An example is the info-box in Wikipedia. Each row header in an entity table corresponds to an attribute. The name of attributes is usually placed at the beginning of each row which is also called the row header. A Matrix table has both row headers and column headers. It is often used in vertical domains to represent data in a more compact and concise form for human reading. Since formulas appear much more frequently in matrix tables than the other two table types, we pay more attention on matrix tables in this study.

In contrast with the first two table types, a value in matrix tables depends on two or more attributes. The interweaving of rows and columns makes matrix tables usually appear as a two-dimensional structure. Figure 3(a) is a matrix table with its three areas from [22], i.e., row headers, column headers, and data cells. We refer to the intersection of each row and column as a *grid cell* and multiple grid cells can form a *merged cell*. For example, in Figure 3(a) the two grid cells of B2 and C2 are merged into a new cell.

Headers with hierarchy is the most salient feature of matrix tables. Normal headers can only represent the value in a data cell that depends on one or two attributes (i.e., its row header and column header). If some values depend on more than two attributes, headers will be grouped, divided, and rearranged so that they produce explicit or implicit hierarchy to represent richer content. Namely, a header hierarchy is usually in the form of a tree structure of headers. It endows tables, a two-dimensional structure, with the ability to express multi-dimensional semantics.

As shown in Figure 3(b), there are two kinds of header hierarchy: explicit and implicit. The explicit hierarchy is represented by merged cells and visual patterns in column headers. The implicit hierarchy usually exists in row headers, and is implied by the combination of text semantics and visual patterns. Based on the internal table structure with the information on merged cells, the explicit hierarchy is easy to be recognized while detecting the implicit hierarchy is difficult.

Such structures of hierarchical headers are frequently used to reduce redundant expressions inside a table for human reading but poses a challenge for machine understanding [22]. To understand the complete meaning of a data cell, it is necessary to combine the semantics of all its ancestor nodes in the tree of hierarchy. Furthermore, the representation of formulas, especially summation formulas, is closely related to the implicit header hierarchy.

2.2 Challenges of Formula Recognition

The challenges of formulas recognition come from two aspects: table representation complexity and formula complexity.

2.2.1 *Table Representation Complexity.*

Diverse representation of tables is the major challenge for recognizing formulas. The semantics of a formula is entailed in the table headers with hierarchy. To clearly illustrate this challenge, we exemplify a bunch of formulas in the three tables in Figure 4. With these examples we have the following observations.

Observation 1. *Textual information on the header hierarchy is the key to understanding the meaning of tables.* Here, the textual information in table headers is closely related to both common sense and domain knowledge. First, the rich common sense and worldly facts entailed in the text are essential for table understanding [32]. For example, there are some formulas in Figure 4(a):

$$\text{Figure 4(a):}\quad \texttt{B8} = \texttt{B4+B5+B6+B7}, \tag{1}$$

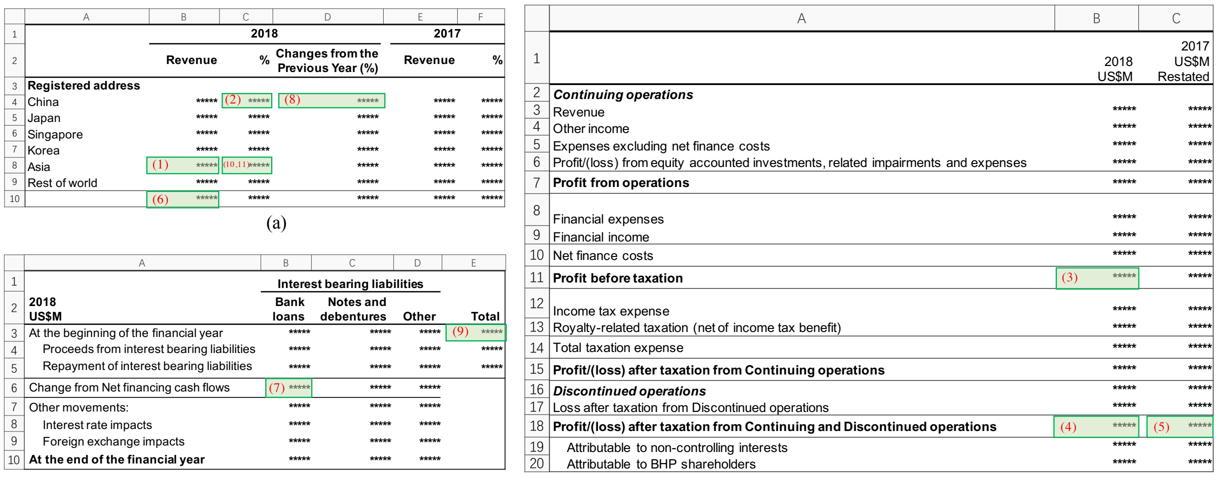
$$\texttt{C4} = \texttt{B4/B10}. \tag{2}$$

Our common sense tells us that the row headers in cells of A4 through A7 refer to some countries, belonging to cell A8 "Asia". Thus, Formula (1) exists in the table. Also, another common sense of text "%" in cell C2 indicates that there may be division formulas in the column C, e.g. Formula (2). Second, the text in tables usually contains domain knowledge to provide clues for formulas. For example, to recognize the formula

$$\text{Figure 4(c):}\quad \texttt{B11} = \texttt{B7+B10}, \tag{3}$$

it is necessary to know the specific finance knowledge such as what "profit before taxation" and "net finance costs" are. Without a financial background, we are unable to understand the table in Figure 4(c), let alone identify the formula. Moreover, domain knowledge is used extensively in the tables from vertical domains.

Observation 2. *The visual appearances serve as auxiliary information for representing formulas.* There are diverse visual objects inside tables. For example, in Figure 4(c), bold text in cell A18 and indentation in cells A19 and A20 give a strong signal that they have a hierarchical structure and there are two summation formulas



	A	B	C	D	E	F
1		2018			2017	
2		Revenue	%	Changes from the Previous Year (%)	Revenue	%
3	**Registered address**					
4	China	*****	(2) *****	(8) *****	*****	*****
5	Japan	*****	*****	*****	*****	*****
6	Singapore	*****	*****	*****	*****	*****
7	Korea	*****	*****	*****	*****	*****
8	Asia	(1) *****	(10,11) *****	*****	*****	*****
9	Rest of world	*****	*****	*****	*****	*****
10		(6) *****	*****	*****	*****	*****

(a)

	A	B	C	D	E
1		Interest bearing liabilities			
2	**2018 US$M**	Bank loans	Notes and debentures	Other	Total
3	At the beginning of the financial year	*****	*****	*****	(9) *****
4	Proceeds from interest bearing liabilities	*****	*****	*****	*****
5	Repayment of interest bearing liabilities	*****	*****	*****	*****
6	Change from Net financing cash flows	(7) *****	*****	*****	
7	Other movements:	*****	*****	*****	
8	Interest rate impacts	*****	*****	*****	
9	Foreign exchange impacts	*****	*****	*****	
10	**At the end of the financial year**	*****	*****	*****	

(b)

	A	B	C
1		2018 US$M	2017 US$M Restated
2	***Continuing operations***		
3	Revenue	*****	*****
4	Other income	*****	*****
5	Expenses excluding net finance costs	*****	*****
6	Profit/(loss) from equity accounted investments, related impairments and expenses	*****	*****
7	**Profit from operations**	*****	*****
8	Financial expenses	*****	*****
9	Financial income	*****	*****
10	Net finance costs	*****	*****
11	**Profit before taxation**	(3) *****	*****
12	Income tax expense	*****	*****
13	Royalty-related taxation (net of income tax benefit)	*****	*****
14	Total taxation expense	*****	*****
15	**Profit/(loss) after taxation from Continuing operations**	*****	*****
16	***Discontinued operations***	*****	*****
17	Loss after taxation from Discontinued operations	*****	*****
18	**Profit/(loss) after taxation from Continuing and Discontinued operations**	(4) *****	(5) *****
19	Attributable to non-controlling interests	*****	*****
20	Attributable to BHP shareholders	*****	*****

(c)

Figure 4: Example of formulas in three tables. The cells in green are used as examples to show their formulas, and the red numbers indicate their reference indexes to the corresponding formulas. Best viewed in color.

(4) and (5). In Figure 4(a), even though the cell `A10` is empty, the visibility of the upper table line implies the existence of Formula (6).

$$\text{Figure 4(c):} \quad \texttt{B18} = \texttt{B19+B20}, \tag{4}$$

$$\texttt{C18} = \texttt{C19+C20}, \tag{5}$$

$$\text{Figure 4(a):} \quad \texttt{B10} = \texttt{B8+B9}. \tag{6}$$

However, visual patterns are not always reliable as their usage is quite flexible and has no standard rules. Sometimes the text is bold just for content emphasis (e.g., cell `A10` in Figure 4(b)). Also, the indentation of headers can only imply that there may be a formula here, but the complete content of the formula is unknown. For example in Figure 4(b), even though there is indentation in cell `A4`, `A5`, we still have to consider the textual semantics of cell `A3` and `A6` to get the formula

$$\text{Figure 4(b):} \quad \texttt{B6} = \texttt{B4+B5}, \tag{7}$$

instead of `B3 = B4+B5`.

Observation 3. *Horizontal formulas are common in tables.* Although the cases mentioned before involve the cells in different rows, there are also some formulas where the cells are in the same row, which is called *horizontal formulas*. For example,

$$\text{Figure 4(a):} \quad \texttt{D4} = \texttt{(B4-E4)/E4}, \tag{8}$$

$$\text{Figure 4(b):} \quad \texttt{E3} = \texttt{B3+C3+D3}, \tag{9}$$

are both horizontal formulas.

Observation 4. *Multiple formulas might appear in the same table cell.* Because the financial indicators displayed in the same table have strong correlation, some indicators can be obtained through different calculations over different sets of table cells. For example,

$$\text{Figure 4(a):} \quad \texttt{C8} = \texttt{C4+C5+C6+C7}, \tag{10}$$

$$\texttt{C8} = \texttt{B8/B10}. \tag{11}$$

Both of the above two formulas are located in cell `C8`.

2.2.2 *Formula Complexity.*

With all the examples of formulas in Section 2.2.1, here we give the formal definition of formula as follows.

Definition 2.1 (Formula). A formula can be defined as:

$$r = f(e_1, \cdots, e_i, \cdots, e_n), \tag{12}$$

where r is the result symbol; $f \in \mathcal{F}$ is the function type of the formula, $\mathcal{F}$ is a predefined function type set; e_i is the i-th argument of the formula, n is the number of arguments. For example, a division formula $r = e_1/e_2$ contains two arguments. It can also be expressed as $r = f_d(e_1, e_2)$, where f_d refers to the division function.

Namely, a formula refers to an expression with an equal sign, the left-hand side, and the right-hand side. The left-hand side of a formula corresponds to the result table cell where this formula is located. The right-hand side of a formula shows the concrete expression to calculate the result. In our task, the result symbol and arguments in the function might refer to a cell in the table.

Clearly, formulas are complicated in terms of diverse math functions, and even for the same function like `SUM` the number of arguments can not be fixed in advance. Also, we need to consider the order of the arguments for some functions such as division, while this is not always necessary for some other functions such as `SUM`, `AVG`, `MIN`, and `MAX`, where the commutative property holds. Therefore, we need a concise, effective, and machine-friendly framework to express all kinds of formulas.

3 SOLUTION

To address the challenges introduced in Section 2, we propose our solution here. The formula recognition task converted into a relation extraction task between two cells, by first detect result cells and then classify cell pairs. To do the classification, a table cell encoding model TaFor is proposed which considers both textual and visual information.

3.1 Problem Conversion

Instead of directly decoding the whole formula as a sequence, we convert a formula into several relations between result and element cells (we call them triplets) in the training phase and extract these relations to compose a formula in the inference phase.

Table 1: Examples of formulas with their triplets.

Name	In Definition 2.1	Computation Rule	Triplets	Label Group
Division (d)	$r = f_d(e_1, e_2)$	$r = e_1/e_2$	$(r, f_d^1, e_1), (r, f_d^2, e_2)$	L(d)=$\{none, f_d^1, f_d^2\}$
Growth Rate (gr)	$r = f_{gr}(e_1, e_2)$	$r = (e_1 - e_2)/e_2$	$(r, f_{gr}^{new}, e_1), (r, f_{gr}^{old}, e_2)$	L(gr)=$\{none, f_{gr}^{new}, f_{gr}^{old}\}$
Average (avg)	$r = f_{avg}(\cdots)$	$r = (e_1 + \cdots + e_n)/n$	$(r, f_{avg}, e_1), \cdots, (r, f_{avg}, e_n)$	L(avg)=$\{none, f_{avg}\}$
Addition and subtraction ($\pm$)	$r = f_{\pm}(\cdots)$	$r = e_1 - e_2 \cdots$	$(r, f_{\pm}^+, e_1), (r, f_{\pm}^-, e_2), \cdots$	L($\pm$)=$\{none, f_{\pm}^+, f_{\pm}^-\}$

Definition 3.1 (Triplet). A triplet consists of three parts: result cell r, relation type f^i, and element cell e:

$$(r, f^i, e) \tag{1}$$

indicating that there is a relationship f^i between cells r and e.

Clearly, each formula $r = f(e_1, \cdots, e_i, \cdots, e_n)$ in Definition 2.1 can be represented as a set of triplets as follows:

$$\{(r, f^1, e_1), \cdots, (r, f^i, e_i), \cdots, (r, f^n, e_n)\}, \tag{2}$$

where the triplet (r, f^i, e_i) means that e_i is the i-th argument in the formula with the result cell r and function f. So, a formula with n-argument function f is transformed into n triplets. In practice, if the commutative property holds for some functions (e.g. *Average*), we do not need to distinguish the position of different arguments. For example, $r = f_{avg}(e_1, e_2)$ is converted to $\{(r, f_{avg}, e_1), (r, f_{avg}, e_2)\}$, where there is only one relation type f_{avg} as we do not have to specify the position of arguments e_1 and e_2. Another example is that $r = e_1 + e_2 - e_3$ is converted to $\{(r, f_{\pm}^+, e_1), (r, f_{\pm}^+, e_2), (r, f_{\pm}^-, e_3)\}$, where we only have to specify the operator $f_{\pm}^+$ or $f_{\pm}^-$ and ignore the position of arguments in the formula. We refer to such functions as *unordered functions*. Table 1 gives some example functions with their corresponding triplets.

In the training phase, we train our model based on the union set of triplets converted from all of the annotated formulas. In the inference phase, we perform our conversion method in an inverse manner to convert the predicted triplets to predicted formulas. In detail, we group the predicted triplet set by the result cell r and the function type f of the triplet. Each group can form a candidate formula. In general, if there are more than two triplets in a group with the same relationship f^i, we keep the one with the highest confidence (not adopted in unordered functions). For instance, if there are two formulas

$$a = Average(b, c, d), \ b = Division(c, d).$$

In the training phase, they are divided into five triplets:

$$\{(a, f_{avg}, b), (a, f_{avg}, c), (a, f_{avg}, d), (b, f_d^1, c), (b, f_d^2, d)\}$$

In the inference phase, if our model predicts perfectly these five triplets, we can aggregate them according to their result cell and function type into two groups:

$$\{(a, f_{avg}, b), (a, f_{avg}, c), (a, f_{avg}, d)\}, \{(b, f_d^1, c), (b, f_d^2, d)\}.$$

Then, we can easily obtain the two original formulas. As we need adequate samples to train and evaluate our model, we only consider the functions that appear in more or nearly 1% of the tables for the experiments (detailed in Table 6). Therefore, in this study, we only consider the four types of formulas shown in Table 1.

3.2 Solution Framework

This section describes the overall framework of our solution. Directly generating all candidate cell pairs in an n by n table will produce $O(n^4)$ pairs. To reduce the number of candidates, we transform the whole task into the following two sequential sub-tasks:

1) **Result Cell Detection**. For each table cell this task detects whether it is the left-hand side (result) of a formula. It is formulated as a binary classification task for table cells. For example, the table cells in green in Figure 4 are *positive* samples for this task.

2) **Cell Pair Classification**. For each positive result cell detected by the first step, the second step extracts its relationships with other cells. It is formulated as a classification problem for cell pairs. For each cell pair (r, e), we do multi-label classification to select one label from each label group in Table 1. So, we can predict multiple formulas with different function types in a single cell.

By decomposition, the number of candidate pairs reduces to $O(Fn^2)$ where F is the number of result cells. We introduce how to do the classification in the next section.

3.3 Two-channel Model

To do cell classification and cell pair classification, we propose a two-channel model to encode the visual and textual information of each data cell. A data cell is represented by its column and row headers, since the semantics of a data cell does not lie in its value but mostly in the headers. Due to the hierarchy within table headers, especially the implicit hierarchy, we also encode the context information into each column and row header. So, a data cell looks not only at the headers in the same column and row, but the whole header region.

The architecture is shown in Figure 5. First, we encode each header. For a row header cell r_i, its text information is encoded into a vector t_i^r, its visual information is encoded into v_i^r, and we represent it as $h_i^r = [t_i^r; v_i^r]$. Then, we connect all row headers by a LSTM. Similar for column headers. Finally, a data cell $c_{i,j}$ is represented as $[h_i^r, h_j^c]$. Note that the table structure, row headers, and column headers have been recognized as the input to this framework. Next, we introduce how to encode the textual information and the visual information, and how to combine them.

3.3.1 *Text Module*.

The text module is illustrated in Figure 5(a). We use row headers to illustrate the text module. The computation for column headers are similar.

The explicit hierarchy like merged cells (Figure 3(b)) is the input to our model. To explicitly capture this information, for each leaf node of the hierarchy, we extend its text by concatenating the text of its ancestor nodes with a special token "&" from root to leaf. For example, in Figure 3(b), the extended sequence of the second column is "Attributable to BHP shareholder & share capital & BHP Billiton Limited", the extended sequence of the third column is "Attributable to BHP shareholder & share capital & BHP Billiton Plc". Then, the sequence of each row are fed into an embedding layer and a Bi-LSTM network to output a hidden vector to represent

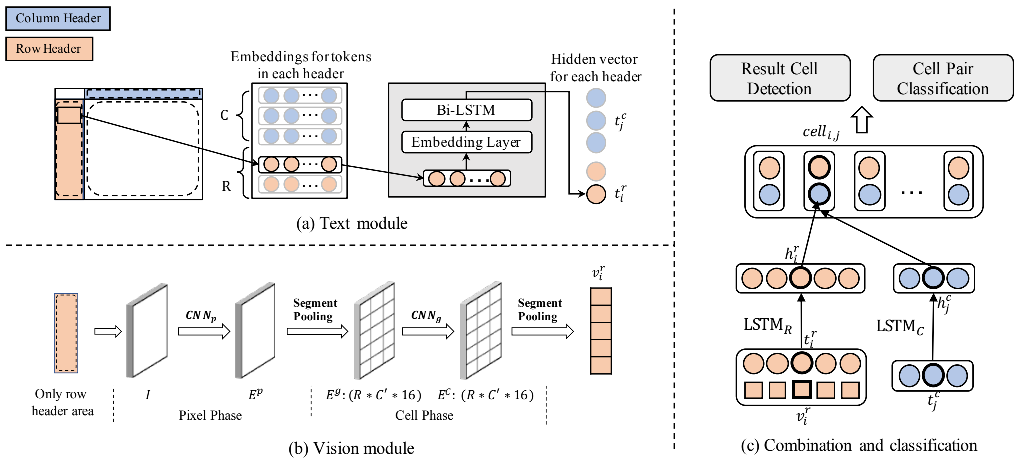


Figure 5: Model architecture. Orange indicates a row header and blue indicates a column header. Best viewed in color.

the textual information of that row:

$$t_i^r = \text{Bi-LSTM}(\text{Embedding}(w_{i,1}, w_{i,2}, ...)), \tag{3}$$

where t_i^r is the textual hidden vector of the i-th row, $(w_{i,1}, w_{i,2})$ is the word sequence of the i-th row, Bi-LSTM$(\cdot)$ returns a single vector to represent the sequence. Similarly, the textual information of the j-th column is encoded in t_j^c.

3.3.2 *Vision Module*.

The vision module is illustrated in Figure 5(b). While the row header area contains rich visual information that implies the hierarchy of a table, using visual patterns to imply hierarchy appears rarely in the column headers. Therefore, we only consider the visual appearances of the row header area in this module. The vision module has two phases: *Pixel Phase* and *Cell Phase*.

Pixel Phase. In this phase, an image I is generated based on the row header area of the input table T. The size of the image I is equal to the original size of the table row header area in documents. The image contains 4 binary channels. For the first channel, a pixel has a value of 1 if it is covered by a table line or any word box, otherwise is 0. The other three channels encode the bold, italic, and light font style, where a pixel has a value of 1 if it is covered by a word with the corresponding font style. Here, the first channel reflects implicitly the table layout, indentation, and other potential visual settings. The other three channels about font style are auxiliary. We scale the longer side of the input image to 256 while keeping its aspect ratio. Then a multi-layer CNN encoder for pixels (CNN_p) is used for capturing the visual feature of each pixel with its context. It is an encoder-decoder neural network that produce E^p with the same width and height as the input tensor.

Cell Phase. Then, we get the hidden vector e_i^g of grid cell i by aggregating vectors of pixels in i. All the vectors of grid cells form a tensor E^g. The CNN_g encoder is another multi-layer CNN to capture contextual information at the cell-level, which takes E^g as input and produces E^c without changing the size.

In some tables, the row header area may have multiple columns and merged cells. The representation of the i-th row is obtained by aggregating the hidden vectors of grid cells in this row. Finally, the visual information of the i-th row is encoded in v_i^r.

3.3.3 *Feature Combination and Cell Representation*.

The header of i-th row now is represented by $\tilde{h}_i = [t_i^r; v_i^r]$ (concatenation of textual and visual vector), combining the textual and visual information. To further integrate these two information, and enhance the context information, these header hidden vectors are fed into LSTM networks. In summary,

$$(h_1, ..., h_R) = \text{LSTM}_R\left(\tilde{h_1}, ..., \tilde{h_R}\right) \tag{4}$$

Similar for column headers.

Finally, we obtain the representation of $cell_{i,j}$ by concatenating the hidden vectors of the headers in the same row and column:

$$h_{i,j} = [h_i^r; h_j^c]. \tag{5}$$

3.4 Joint Training

With the representation of each data cell, we introduce how to do result cell detection and cell pair classification.

Result Cell Detection. In this task, the feature $h_{i,j}$ is fed into two fully-connected layers with a ReLU activation in between. Finally, a softmax non-linear layer predicts the probability of whether a cell is a result cell. The cross-entropy loss of result cell classification $\mathcal{L}_{rc}$ is defined as:

$$\mathcal{L}_{rc} = -\sum_c y_c \log\left(p(c)\right) + (1 - y_c)\log\left(1 - p(c)\right) \tag{6}$$

where $p(c)$ is the predicted probability that cell c is a result cell, y_c equal to 1 if cell c is a result cell in ground truth.

Cell Pair Classification. For two cells in a candidate cell pair, we concatenate the features of them and feed it into several independent classifiers. Each classifier contains two fully-connected layers with a ReLU activation in between, followed by a softmax layer. The cross-entropy loss of cell pair classification is defined as:

$$\mathcal{L}_{cp} = -\sum_{f_t \in \mathcal{F}} \sum_{s \in \mathbb{P}} \sum_{k \in L(f_t)} \mathbb{1}_{[k = y_s(f_t)]} \log\left(p_k(s)\right) \tag{7}$$

where $\mathcal{F}$ is the predefined formula type set, $\mathbb{P}$ is the candidate set of cell pairs, $y_s(f_t)$ indicates the label of candidate cell pair s for the function f_t, $L(\cdot)$ indicates the label set of f_t, $p_k(s)$ indicates the probability that candidate cell pair s belongs to the label k.

Finally, our overall optimization objective is to jointly minimize these two losses: $\mathcal{L} = \mathcal{L}_{rc} + \alpha\mathcal{L}_{cp}$ where α is a trade-off parameter.

4 EXPERIMENTS

Two datasets are adopted in this study. DECO-F is a small dataset with 1,264 tables we derived from DECO [19]. As this dataset is small, we put the discussion in Appendix B.2. In this section, we focus the result on a larger dataset FinFormulas we collected.

4.1 Dataset

We collected a dataset, namely **FinFormulas**, consisting of 190,179 tables from various types of 4,746 Chinese financial documents (in PDF format). The documents in finance are table-intensive and computation-intensive. All the documents used are the annual reports and IPO prospectuses in Chinese financial markets, crawled from CNINFO[3], a website for the China Securities Regulatory Commission. For each PDF file, we use PDFlux[4] to recognize all the tables and their inner table structures. To annotate formulas in tables, we recruited ten in-house annotators for this labeling task. Each annotator has at least 3-year experience in auditing so that she is capable to recognize all the formulas in financial tables. For each table, the first two annotators independently annotate it. If there are any conflicts between these two labeling results, another annotator proof-reads and corrects the results to get the final ground truth for each table. This rigorous process ensures the high quality of the labeled data. We follow [3] and use *micro F1-score* to measure inter-annotator agreement of our dataset. The final micro F1-score is 94.83%, indicating there is a high degree of agreement between the two independent labeling results.

The resultant corpus contains 190,179 tables in 4,746 financial documents, with a total of 1,442,227 formulas. On average each document contains 40.07 tables and each table contains 7.58 formulas. As we mentioned in Section 3.1, we only consider four types of formulas as we need enough samples to train the model. These four types of formulas account for 98.22% of all formulas in the corpus. More statistics can be found in Table 5 in Appendix A.

4.2 Experimental Setup

4.2.1 Candidate Generation. For a result cell (i, j), we only consider to make candidate pair with cells in the same row or column $(i, \cdot), (\cdot, j)$, or with cells in the adjacent rows or columns $(i \pm 1, \cdot), (\cdot, j \pm 1)$. These candidates cover over 99% of annotated formulas. Details about candidate generation and training setting for reproducibility are discussed in Appendix A.

4.2.2 Metrics. We evaluate the performance of our framework on the two tasks. For the cell detection task, We report the F1-score. For the cell pair classification task, we report the F1-score on the pair level and formula level which computed by comparing the predicted formula set $\mathbb{F}'$ and the ground truth formula set $\mathbb{F}$. We primarily report F1 on formula level unless otherwise specified.

4.2.3 Methods Compared. We compare our solution against two retrieval-based baselines:

• **Header Hard Matching (HHM)**. Given a test table, we extract its header set (contains row and column headers) and find the table in the training set with the same header set. Then the annotated formula set of the found table is suggested as the answer for the test table. If no table is found, the answer for the test table is empty.

• **Header Soft Matching (HSM)**. For each table, we concatenate the text in all headers with a special token to obtain a header sequence according to the order of rows and columns. We define the similarity of two tables to be the similarity between their header sequences depended on the edit distance[5], i.e.,

$$Similarity(T_1, T_2) = \frac{|s_1| + |s_2| - distance(s_1, s_2)}{|s_1| + |s_2|}, \quad (8)$$

where s_1, s_2 are the header sequences of table T_1 and T_2, $distance(s_1, s_2)$ calculates the edit distance between s_1 and s_2, $|\cdot|$ indicates the length of a string. Given a test table, we use their similarity to rank the tables in the training set that have the same number of rows and columns with the test table. The annotated formula set of the top-1 table is suggested as the answer.

Moreover, we run our solution with two setups:

• **TaFor** (for **Ta**ble **For**mulas Recognition)**.** The framework in Section 3 with our two-channel model and two classification tasks.

• **HHM+TaFor.** Given a test table, we first perform HHM for it. If there is no table in the training set with the same table headers, we feed it into our TaFor framework and get its predicted answer.

Finally, we conduct ablation experiments over each key component in order to explore their relative importance.

Table 2: Evaluation results.

	±	*d*	*gr*	*avg*	overall
HHM	42.57	46.29	48.78	46.37	44.08
HSM	68.00	78.97	74.45	67.12	72.05
TaFor	**90.15**	91.66	85.87	87.38	90.65
HHM + TaFor	90.02	**93.58**	**92.19**	**89.18**	**91.31**

4.3 Performance Evaluation.

4.3.1 Method Performances. Table 2 summarizes our results on the test set compared with baselines.

For two baselines, HSM achieves higher performance with $F1 = 72.05\%$ than HHM. For HHM, there are only about 28% tables in the test set can hit a result table in the training set. When we recalculate the F1-score of HHM on the subset of these 28% test tables, the overall F1-score reaches 99.3%. This result justifies the rationality of the design of using the feature of the corresponding row and column header to represent a data cell, since HHM only takes the text in table headers to hit a result table.

Compared with the two baselines, our TaFor achieves an overall $F1 = 90.65\%$ across all four formula types, which outperforms best baseline HSM (72.05%) by a large margin. The ensemble of HHM and TaFor further improves the performance to 91.31%.

An experiment is conducted to analyze the performance of TaFor on tables that are not similar to any table in training dataset and its result is shown in Figure 6. We define the similarity of a test table T to the training dataset $\mathbb{T}$ based on Equation 8 as:

$$similarity(T, \mathbb{T}) = \max_{T' \in \mathbb{T}} \left(similarity(T, T')\right) \quad (9)$$

Then, the test tables are split into 10 groups by their similarity to the training dataset. In each group, we count the number of

[3]http://www.cninfo.com.cn/

[4]http://pdflux.com/

[5]The similarity is calculated by `Levenshtein.ratio` in python

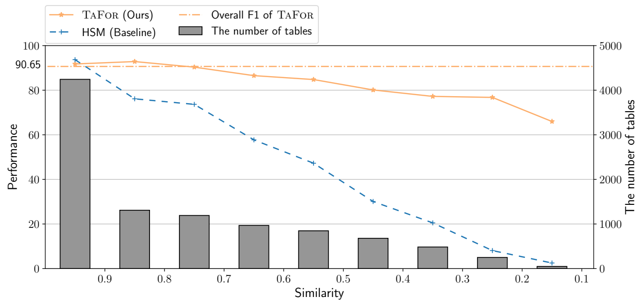


Figure 6: The generalization ability of TaFor.

Table 3: Evaluation results on different groups of tables.

F1 (# of tables)		# of columns		
		0-5	6-10	>11
# of rows	1-10	90.99 (5476)	93.77 (1916)	79.82 (55)
	11-20	86.71 (1048)	89.83 (507)	75.48 (23)
	21-30	91.47 (324)	83.96 (169)	90.13 (10)
	31-50	92.85 (218)	86.02 (80)	68.32 (6)
	51-80	89.10 (52)	65.42 (23)	– (0)
	>80	94.60 (31)	78.85 (17)	– (0)

Table 4: Ablation results.

	Result cell	Pair	Formula level				
	detection	level	±	*d*	*gr*	*avg*	overall
TaFor	96.12	95.17	90.15	91.66	85.87	87.38	90.65
−text	61.43	65.42	64.24	0	0	46.40	48.78
−vision	94.42	93.93	87.86	90.89	83.69	83.59	88.77

tables and test the performance of TaFor and HSM. The overall F1-scores on formula level are shown in Figure 6. Note that the group of similarity [0, 0.1] is not shown in the figure as there are no tables in this group. Although the performance of TaFor decreases slightly as the similarity decreases, it is much better than the baseline HSM. We can observe that even in the group of similarity [0.2, 0.3], TaFor achieves an overall F1-score of nearly 80% which significantly outperforms HSM (nearly 10%).

We also analyze the performance of TaFor in terms of table size. We split tables in the test dataset into several groups and mark the number of tables of each group in Table 3. Along with the number of rows and columns increases, the F1-score tends to decrease.

4.3.2 Ablation Study. We conduct ablation studies on the information representation and location. The results are listed in Table 4. The experiments starting with "−" indicate that we ablate some features from TaFor to evaluate their contributions.

• **Ablation on textual information.** When we ablate the text information of inputs ("−text"), i.e., remove t_i^r and t_j^c from cell representation, the overall F1-score on formula level drops dramatically from 90.65% to 48.79%, which demonstrates the textual information is actually the key to understanding the tables. In this experiment, the performances of formula *d* and *gr* are even zero because their existence strongly depends on the textual information. For example, in Figure 4(a) *d* formulas from C4 to C9 and *gr* formulas from D4 to D10 cannot be recognized without the text in C2 and D2.

• **Ablation on visual appearances.** Comparing with TaFor, we find that by ablating the visual feature (i.e., v_i^r), the overall F1-score decreases by 2% (90.65% $\longrightarrow$ 88.77%), which meets our expectation that the visual appearances serve as auxiliary information for representing formulas.

In summary, the textual information and visual appearances are complementary to recognize formulas.

4.4 Limitations and Future Work

We analyze a bunch of failure cases to identify the limitations of our solution, which might lead more interesting future studies. We find that although our solution has achieved a high formula-level F1 of 90% there is still room for improvement on the model robustness to the variations of table headers. Some cases are shown in Appendix A.4. We find natural language understanding is important for this task, for example, named entity recognition can help handle unseen text. We also observe that the common sense and domain knowledge are vital to fully recognize the formulas in tables. Our model does not encode the prior knowledge. Thus, it is still hard to transfer the model obtained in one domain to another domain, leading to other interesting future work. Hence, formula recognition over tables also provide a good scenario to develop models which combine deep learning and symbolic prior knowledge.

5 RELATED WORK

We summarize our related work into the following aspects.

Table Detection and Structure Recognition. Tables lose their structure information when they are exported into PDFs. Research on table detection and structure recognition started from the last century. In 1993, Itonori [15] proposed a rule-based method to detect table position and recover table cells. Some methods [4, 7] based on generative or statistical machine learning methods were developed to avoid manually proposing rules. More recently, Schreiber et al. [28] proposed a deep learning architecture for detecting table location and identifying the row and column positions. The graph neural network is also used in [27]. Our work is based on the result of table detection and structure recognition.

Table Understanding. Much of the previous table-related research focused on understanding tables at different levels. The tasks of table type classification [9, 24], table cell classification [11] or region detection in tables [9, 29] could be assumed as the preliminary tasks of table understanding. Further, [5, 6] have explored the ways to identify the hierarchy of table headers. Recent work [14] proposed a system BriQ linking quantity mentions in texts and tables, which dealt with the aggregation formulas in tables. However, this method is value-dependent and performs certain calculations among the specific numerical values. Our solution recognizes formulas by understanding deeper semantics of tables entailed in the textual information and visual appearances of table headers, which opens a deeper perspective for the research of table understanding.

Besides, there are many interesting downstream studies based on tables, such as relation extraction between tables [10, 31], entity linking [2, 34], table retrieval [23, 33]. Most of these studies are based on the Web tables with simple internal table structure. Meanwhile, the matrix tables with complex hierarchy exist widely on the

Web. We hope our study will push forward the related downstream applications on the tables in vertical domains.

Most research on table understanding utilizes deep neural networks. Recent studies [11, 12] embedded a table into the vector space to classify tables or table cells. TabNet [24] is a neural network architecture for table type classification. It conducted RNN and CNN on text to represent a table. However, the CNN in TabNet only captures textual correlations between cells but omits the visual appearance of the table. Our experiments demonstrate that the visual and textual features are complementary to each other and can be jointly conducted to achieve better performance.

Spreadsheet-related Tasks. Spreadsheets are common in the enterprise. To improve the quality of spreadsheets, most studies about spreadsheets focus on error detection [16]. Singh et al. [30] performed a simple neural network to find errors where an author has placed a number in a cell that should be a formula. Barowy et al. [1] proposed ExceLint based on information-theoretic static analysis which finds missing formulas and fixes the current formulas based on the existing formulas. These studies are based on spreadsheets where formulas are accessible. However, such existing formulas only appear in spreadsheets and are also unreliable. Our solution can predict new, not existing formulas based on semantic, which can be used for formula recommendation.

6 CONCLUSION

In this study, we provide a uniform schema of formulas and convert this task into a problem of triplet extraction. Specifically, we divide this task into two sequential sub-tasks: result cell detection and cell pair classification. Then, an end-to-end model TaFor is developed to capture the joint correlation of textual information and visual appearances of tables. Extensive experiments show that our TaFor achieves an overall F1-score 0.9024 on formula level with an increase of at least 0.18 F1-score compared with two retrieval-base baselines. Ablation studies demonstrate that the textual information is actually the key to understanding tables, and the visual features are complementary to the textual information. They can be jointly conducted to achieve better performance.

We need to point out that the high performance of our model heavily relies on a large scale of annotated data with great labeling burden. In the future, we will explore how to inject common sense and domain knowledge to decrease the labeling costs while obtaining high performances. Also, this study aims to attract the interests of the community in deep semantic understanding for tables and the development of related downstream applications.

7 ACKNOWLEDGMENTS

The research 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 Nos. 62076231 and U1811461.

REFERENCES

[1] Daniel W. Barowy, Emery D. Berger, and Benjamin Zorn. 2018. ExceLint: automatically finding spreadsheet formula errors. In *ACM on Programming Languages.*

[2] Chandra Sekhar Bhagavatula, Thanapon Noraset, and Doug Downey. 2015. TabEL: entity linking in web tables. In *International Semantic Web Conference.*

[3] Alex Brandsen, Suzan Verberne, Milco Wansleeben, and Karsten Lambers. 2020. Creating a Dataset for Named Entity Recognition in the Archaeology Domain. In *LREC*. 4573–4577.

[4] Francesca Cesarini, Simone Marinai, L Sarti, and Giovanni Soda. 2002. Trainable table location in document images. In *ICPR.*

[5] Xilun Chen, Laura Chiticariu, Marina Danilevsky, Alexandre Evfimievski, and Prithviraj Sen. 2017. A Rectangle Mining Method for Understanding the Semantics of Financial Tables. In *ICDAR.*

[6] Zhe Chen and Michael Cafarella. 2014. Integrating spreadsheet data via accurate and low-effort extraction. In *KDD.*

[7] Ana Costa e Silva. 2009. Learning rich hidden markov models in document analysis: Table location. In *ICDAR.*

[8] Julian Eberius, Katrin Braunschweig, Markus Hentsch, Maik Thiele, Ahmad Ahmadov, and Wolfgang Lehner. 2015. Building the dresden web table corpus: A classification approach. In *International Symposium on Big Data Computing.*

[9] Jing Fang, Prasenjit Mitra, Zhi Tang, and C Lee Giles. 2012. Table Header Detection and Classification.. In *AAAI.*

[10] Besnik Fetahu, Avishek Anand, and Maria Koutraki. 2019. TableNet: An Approach for Determining Fine-grained Relations for Wikipedia Tables. In *WWW.*

[11] Majid Ghasemi-gol, Jay Pujara, and Pedro Szekely. 2019. Tabular Cell Classification Using Pre-Trained Cell Embeddings. In *ICDM.*

[12] Majid Ghasemi-Gol and Pedro Szekely. 2018. TabVec: Table Vectors for Classification of Web Tables. *arXiv* (2018).

[13] Felienne Hermans and Emerson Murphy-Hill. 2015. Enron's spreadsheets and related emails: A dataset and analysis. In *ICSE.*

[14] Yusra Ibrahim, Mirek Riedewald, Gerhard Weikum, and Demetrios Zeinalipour-Yazti. 2019. Bridging quantities in tables and text. In *ICDE.*

[15] Katsuhiko Itonori. 1993. Table structure recognition based on textblock arrangement and ruled line position. In *ICDAR.*

[16] Dietmar Jannach, Thomas Schmitz, Birgit Hofer, and Franz Wotawa. 2014. Avoiding, finding and fixing spreadsheet errors - A survey of automated approaches for spreadsheet QA. *Journal of Systems and Software* (2014).

[17] Diederik P Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR.*

[18] Bryan Klimt and Yiming Yang. 2004. Introducing the Enron corpus.. In *CEAS.*

[19] Elvis Koci, Maik Thiele, Josephine Rehak, Oscar Romero, and Wolfgang Lehner. 2019. DECO: A dataset of annotated spreadsheets for layout and table recognition. In *ICDAR.*

[20] Christian Leborg. 2006. *Visual Grammar: A Design Handbook (Visual Design Book for Designers, Book on Visual Communication).* Princeton Architectural Press.

[21] Da Li, Huiyan Wang, Chang Xu, Fengmin Shi, Xiaoxing Ma, and Jian Lu. 2019. WARDER: Refining cell clustering for effective spreadsheet defect detection via validity properties. In *International Conference on Software Quality, Reliability and Security.*

[22] Hongwei Li, Qingping Yang, Yixuan Cao, Jiaquan Yao, and Ping Luo. 2020. Cracking Tabular Presentation Diversity for Automatic Cross-Checking over Numerical Facts. In *KDD.*

[23] Thanh Tam Nguyen, Quoc Viet Hung Nguyen, Matthias Weidlich, and Karl Aberer. 2015. Result selection and summarization for web table search. In *ICDE.*

[24] Kyosuke Nishida, Kugatsu Sadamitsu, Ryuichiro Higashinaka, and Yoshihiro Matsuo. 2017. Understanding the semantic structures of tables with a hybrid deep neural network architecture. In *AAAI.*

[25] Ray Panko. 2015. What we don't know about spreadsheet errors today: The facts, why we don't believe them, and what we need to do. In *The European Spreadsheet Risks Interest Group.*

[26] Stephen G Powell, Kenneth R Baker, and Barry Lawson. 2008. A critical review of the literature on spreadsheet errors. *Decision Support Systems* (2008).

[27] Shah Rukh Qasim, Hassan Mahmood, and Faisal Shafait. 2019. Rethinking table recognition using graph neural networks. In *ICDAR.*

[28] Sebastian Schreiber, Stefan Agne, Ivo Wolf, Andreas Dengel, and Sheraz Ahmed. 2017. Deepdesrt: Deep learning for detection and structure recognition of tables in document images. In *ICDAR.*

[29] C Sharad and George Nagy. 2013. Segmenting Tables via Indexing of Value Cells by Table Headers. In *ICDAR.*

[30] Rishabh Singh, Benjamin Livshits, and Ben Zorn. 2017. *Melford: Using Neural Networks to Find Spreadsheet Errors.* Technical Report.

[31] Hong Wang, Anqi Liu, Jing Wang, Brian D. Ziebart, Clement T. Yu, and Warren Shen. 2015. Context Retrieval for Web Tables. In *International Conference on The Theory of Information Retrieval.*

[32] Jingjing Wang, Haixun Wang, Zhongyuan Wang, and Kenny Q Zhu. 2012. Understanding tables on the web. In *International Conference on Conceptual Modeling.*

[33] Shuo Zhang and Krisztian Balog. 2018. Ad Hoc Table Retrieval using Semantic Similarity. In *WWW.*

[34] Shuo Zhang and Krisztian Balog. 2019. Auto-completion for Data Cells in Relational Tables. In *CIKM.*

[35] Shuo Zhang and Krisztian Balog. 2020. Web Table Extraction, Retrieval, and Augmentation: A Survey. *ACM Transactions on Intelligent Systems and Technology* (2020).

Table 6: Distribution of formula type and effectiveness of candidate generation strategy in FinFormulas.

formula type	#	%	generation strategies applied	upper bound of recall
±	893k	61.95	**S1**	99.98
d	483k	33.52	**S1**, **S2**	97.50
gr	26k	1.80	**S1**, **S2**	99.70
avg	14k	0.95	**S1**	99.96
other	25k	1.78	–	–

A DETAILS ON FINFORMULAS

A.1 Statistic

The statistics about FinFormulas is summarized in Table 5. On average each document contains 40.07 tables and each table contains 7.58 formulas. There are 75.43% of formulas whose elements are all in the same line (row or column), and 99.14% of formulas whose elements are included in the same line or adjacent lines. This motivates the candidate generation in Appendix A.2. The distributions of tables about the number of rows and columns are shown in Figure 7. Most tables contain 1-20 rows and 2-10 columns. Some huge tables have more than 80 rows, 15 columns.

Table 5: Statistic of FinFormulas and DECO-F datasets

	FinFormulas	DECO-F
# of documents	4,746	1,165
# of tables	190,179	1,263
# of formuals	1,442,227	4,014
# of formulas with elements in the same line	1,087,904	3,944
# of formulas with elements in the same or adjacent line	1,429,808	3984
Average number of rows in tables	10.77	31.91
Average number of columns in tables	4.86	12.15

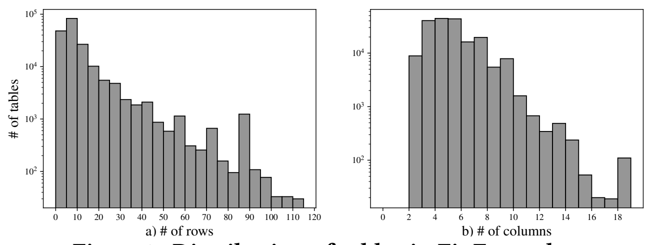


Figure 7: Distribution of tables in FinFormulas.

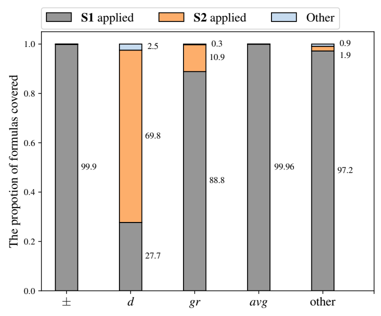


Figure 8: The effectiveness of generation methods in each formula type. Best viewed in color.

A.2 Cell Pair Candidate Generation

For cell pair classification task, we need to consider how to generate the candidate pairs for each cell (i, j). Exhaustively enumerating all the possible pairs leads to a quadratic space (in table size). So we generate candidates for result cell (i, j) by two strategies. Strategy **S1**: make pair with cells in the same row or column $(i, \cdot), (\cdot, j)$, which covers over 75% of formulas. Strategy **S2**: make pair with cells in the adjacent rows or columns $(i \pm 1, \cdot), (\cdot, j \pm 1)$, which covers over 99% of formulas combined with **S1**. We apply different strategies for different types of formulas so that more than 60% of candidates are removed while over 99% of positive pairs are reserved. This is detailed in Table 6.

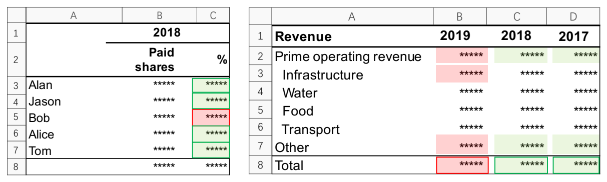
A.3 Training Settings

We use a 128-dimsional character-based embedding layer for text module. We only keep 1000 most frequent tokens in our vocabulary and the set the rest as `UNKONW` token. Meanwhile, we replace the text of date with a special token `DATE`. All text in non-header cells are discarded. The dimensions of Bi-LSTM, LSTM hidden, fully-connected layer before the softmax layer are 128. The CNN_g encoder contains 4 convolutional layers, with kernel size $(3 * 3 * 16)$. The number of tables in each mini-batch is 84. The balance factor α in loss is set to 1. Adam [17] with learning rate 0.003 is used for optimization. We perform 5-fold cross-validation to evaluate our model performance and report metrics averaged over 5 folds.

A.4 Case Study

We show some bad cases of our model prediction in Figure 9. Note that since the original tables are big, the tables in Figure 9 are simplified to save the space.

In Figure 9(a), our method loses the formula `C5=B5/B8` while successfully recognizing the other four formulas in column `C`, namely `C3=B3/B8`, `C4=B4/B8`, `C6=B6/B8`, `C7=B7/B8`. The differences between the error formula and the four correct formulas are their row headers with different people names. Even though they belong to the same entity type, i.e., people, their word-level representations might quite different. Applying entity extraction over table cells in an initial step might be a good solution to alleviate this issue.



	A	B	C
1		2018	
2		Paid shares	%
3	Alan	*****	*****
4	Jason	*****	*****
5	Bob	*****	*****
6	Alice	*****	*****
7	Tom	*****	*****
8		*****	*****

(a)

	A	B	C	D
1	Revenue	2019	2018	2017
2	Prime operating revenue	*****	*****	*****
3	Infrastructure	*****	*****	*****
4	Water	*****	*****	*****
5	Food	*****	*****	*****
6	Transport	*****	*****	*****
7	Other	*****	*****	*****
8	Total	*****	*****	*****

(b)

Figure 9: Two bad cases. The bordered rectangle indicates that this is a result cell, the borderless rectangle indicates that this is an argument of a formula, the red box indicates an false formula, and the green box indicates an true formula. Best viewed in color.

Table 7: Performance on DECO-F.

	Result cell detection	Pair level	Formula level on ±
TaFor	65.23	79.34	56.42
−text	5.31	1.86	2.32
−vision	58.82	70.26	51.12

In Figure 9(b), our method successfully recognizes the formulas in C8 and D8, namely C8=C2+C7, D8=D2+D7. However, for cell B8 the predicted formula is B8=B2+B3+B7. In other words our model makes a wrong prediction that a triplet of (B8, $f_{\pm}^{+}$, B3) exists, while it makes the correct predictions that the triplets of (C8, *none*, C3) and (D8, *none*, D3) hold. In our method, we replace the text of date, e.g. cells B2, C2, D2, with a special token DATE in advance. Thus, readers might think that the embeddings of the following 3 pairs, i.e., (B8, B3), (C8, C3), (D8, D3), should be the same. However, this is not true since our model connects all the column headers into a sequence and deliberately captures its sequential information to detect the *horizontal formulas* (Observation 3 in Section 2.2.1) via LSTM in Equation 4.

B DETAILS ON DECO-F

B.1 Pre-processing and Statistics

To test our model on different domains, we generate a dataset **DECO-F** from DECO [19]. DECO is a dataset of spreadsheet files, annotated on the basis of layout and contents. It has 1,165 files, extracted from the Enron email archive [18]. Here, Enron is an energy, commodities, and services company, so this dataset is not focused on finance as FinFormulas dataset. This dataset marked the borders and headers of tables inside each worksheet. We extract all the annotated tables and filter the numerical formulas from the existing Excel formulas in these tables and use them as ground truth to form our DECO-F dataset.

We use openpyxl to parse the Excel file in DECO corpus. For each table of the annotated results of DECO, we extract and parse the Excel formulas in its table region as the ground truth, use the *Header* cells and *GroupHeader* cells in DECO as its row headers and column header, respectively. The text in table headers are stored, in which we convert the text of datetime into "YY/mm/dd" format string. We read the row height for each row, the column width for each column, the visibility of the table line, font size, font style to build the visual appearances of the table.

Some statistics about DECO-F are shown in Table 5. The results show there are 4,014 formulas in 1,263 tables, which have an average of 31.95 rows and 12.15 columns. Comparing with FinFormulas, tables in DECO-F in general have a larger table size and contains fewer formulas. We publish this dataset in https://github.com/qingping95/DECO-F.

B.2 Experiment

As the tables that contain formula *d*, *gr*, and *Avg* are few (<30) to train our model, we only report the performance on ± formulas in Table 7. In DECO-F, the model can extract over 50% of formulas correctly. Similar to FinFormulas corpus, we find the model depends more on text, but the vision can help improve the result. Compared with FinFormulas corpus, the performance drops sharply for two main reasons. First, the number of tables containing formulas in DECO-F is too small (257) to support an end-to-end model to obtain a satisfactory recognition capability. Second, the erroneous formulas in DECO affect the model performance. According to an investigation [13] on the spreadsheets of Enron corpus, 24% of Enron spreadsheets with at least one formula contain an Excel error. On the other hand, Koci et al. [19] point out that more than 20% "derived" cells do not contain Excel formulas, where the derived cells represent aggregations of data in DECO corpus. The poor performance on DECO-F means that, as indicated in Appendix A.4, we may have to study how to transfer common sense, world knowledge, and natural language understanding results to improve the performance of table formula recognition on small dataset.