Automatic Consistency Checking of Table and Text in Financial Documents

Syed Musharraf Ali[1,3], Tobias Deußer[2,3], Sebastian Houben[1], Lars Hillebrand[2,3], Tim Metzler[1], and Rafet Sifa[3]

[1]Hochschule Bonn-Rhein-Sieg, Sankt Augustin, Germany
[2]University of Bonn, Bonn, Germany
[3]Fraunhofer IAIS, Sankt Augustin, Germany

Abstract

A company's financial documents use tables along with text to organize the data containing key performance indicators (KPIs) (such as profit and loss) and a financial quantity linked to them. The KPI's linked quantity in a table might not be equal to the similarly described KPI's quantity in a text. Auditors take substantial time to manually audit these financial mistakes and this process is called consistency checking. As compared to existing work, this paper attempts to automate this task with the help of transformer-based models. Furthermore, for consistency checking it is essential for the table's KPIs embeddings to encode the semantic knowledge of the KPIs and the structural knowledge of the table. Therefore, this paper proposes a pipeline that uses a tabular model to get the table's KPIs embeddings. The pipeline takes input table and text KPIs, generates their embeddings, and then checks whether these KPIs are identical. The pipeline is evaluated on the financial documents in the German language and a comparative analysis of the cell embeddings' quality from the three tabular models is also presented. From the evaluation results, the experiment that used the English-translated text and table KPIs and Tabbie[1] model to generate table KPIs' embeddings achieved an accuracy of 72.81% on the consistency checking task, outperforming the benchmark, and other tabular models.

[1]TABBIE: Pretrained Representations of Tabular Data by Iida et al.

1 Introduction

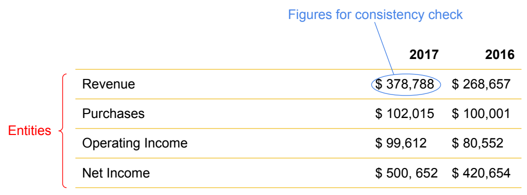
Various documents such as web pages, spreadsheets, and PDFs use tables to represent the data in a compact format. Because of this wide usage of tables, several models specifically tailored to processing them have been developed. These models are mostly trained using tables and are used for tabular applications such as table augmentation and table search. Similar to BERT [9], most of the state-of-the-art (SOTA) models developed for tabular data are based on the transformer architecture [26] and they are pre-trained only on tables (or the combination of table and text data). Some of these models are Tuta [27], Turl [6], TaBERT [29], TaPas [16], and Tabbie [19]. These SOTA models have improved results in several table processing tasks such as table type classification, column type identification and table augmentation [10].

Tables are widely used in financial documents to structure the facts and figures and to manage important financial indicators in a compact format. Most corporations are legally bound to create and publish financial documents to give stakeholders and investors a more accurate picture of the company's situation. These financial documents usually contain complete annual financial reports, cash flow statements, business plans, IPO prospectus, or an income statement. The information in these documents is arranged in the form of free-form text and tables.

Audit firms do consistency checking on these financial documents by manually going over the whole document. As part of this, they check their validity by comparing whether the figures (mostly

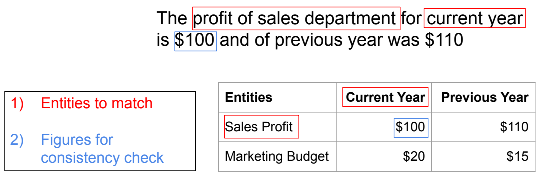
https://doi.org/10.7557/18.6816

numerical) of some entity in a text are the same as those figures presented by a similar entity in a corresponding table. In Figure 1, an example of a table from a financial document is given. For the task of consistency checking first, the entities in the table are matched with similar entities in the text, and then corresponding figures are checked for consistency. The task of consistency checking is difficult and time-consuming because the entities in the table and text can differ either syntactically or semantically. Figure 2 shows an example, in which the entities *"profit of sales department"* and *"Sales profit"* are semantically same but they are syntactically different. For consistency checking, first, they should be matched while considering the year, then the figures associated with these entities is checked for consistency.


Figures for consistency check
Entities

	2017	2016
Revenue	$ 378,788	$ 268,657
Purchases	$ 102,015	$ 100,001
Operating Income	$ 99,612	$ 80,552
Net Income	$ 500, 652	$ 420,654

Figure 1: Example of a table from a financial document. Entities are marked in red and numerical figures are in a blue circle.


The profit of sales department for current year is $100 and of previous year was $110

1) Entities to match
2) Figures for consistency check

Entities	Current Year	Previous Year
Sales Profit	$100	$110
Marketing Budget	$20	$15

Figure 2: Example of a text and a corresponding table with marked entities and figures. Entities are marked by a red box and numerical figures are marked by a blue box.

The consistency checking task for one company takes approximately 3 months[2]. Automating this task will save valuable manual labor of financial auditors as they will only have to validate those instances where the values don't match. This paper contributes one step in automating the consistency checking task by decomposing it into three components. The first and second step is to obtain the KPIs' embeddings from the text and the corresponding table within the same document. The last step is to match the identified table and text embeddings and then the figures associated with them can be validated. This paper mainly focuses on the methods for extracting table cell embeddings and the quality of the cell embeddings is checked by their performance on the consistency checking task. For embeddings of the text KPIs, a BERT model trained on German language data (i.e. bert-base-german-dbmdz-cased[3] or bert-base-multilingual-cased[4]) is used and for consistency checking, support vector machine is used as a binary classifier.

The rest of the paper is structured as follows: Section 2 defines previous work related to the topic. Thereafter, Section 3 introduces the consistency checking pipeline, the data generation process, the models, and the process of getting embeddings from inferring the models. Section 4 provides details and results about different experiments and finally, Section 5 summarizes all the results and gives the concluding remarks.

2 Related Work

To the best of our knowledge. contemporary work regarding the automatization of checking or processing financial documents includes a pipeline proposed by Zisman and Athanasopoulou [33] to correct inconsistencies in similar instances of the financial document in standard Extensible Markup Language. They are generated by multiple actors that are involved in the same transaction. D'Atri [12] proposed a logic-based approach using OntoDLP[5] to validate financial documents written in eXtensible Business Reporting Language format and most recently Hillebrand et al. [17] did automated consistency checking using BERT and a contrastive autoencoder.

Furthermore, Sifa et al. [24] developed a tool that uses machine learning techniques to automatically check whether the financial statements of a

[2]https://holtzmanpartners.com/2021/04/21/how-to-prepare-for-an-audit-your-ultimate-guide/

[3]https://huggingface.co/dbmdz/bert-base-german-cased

[4]https://huggingface.co/bert-base-multilingual-cased

[5]an ontology representation and reasoning language based on Disjunctive Logic Programming

certain corporation are according to specific legal requirements. Ramamurthy et al. [23] then made improvements to this tool by modifying the mechanism of the recommendations of legal requirements and by obtaining textual representations from a BERT encoder. Zhu et al. [32] introduced a cost-sensitive capsule network to identify abnormal instances from normal instances in audit data. Hillebrand et al. [18] introduced an automated system called KPI-BERT to automatically extract KPIs from financial documents in German language and also linked these KPIs with a proper relationship to their respective numerical value. Similarly, Deußer et al. [8] did this for a dataset in the English language. Cao et al. [2] introduced an iterative relation extractor (IRE) model that is the main component of a numerical cross-checking (NCC) system. IRE extract formulas from the text of numerical description and then these formulas are used for NCC. Biesner et al. [1] developed a framework based on state-of-the-art deep learning techniques to anonymize sensitive information in financial documents in the German language so that the documents can be further used in other applications without any restriction. As compared to the approaches that do consistency checking, this paper's approach automates the consistency checking task using different transformer-based tabular models.

Previous work related to understanding the structure of the table includes Table2Vec introduced by Zhang et al. [31]. Table2Vec provides static embeddings of the different elements of a serialized table. Gentile et al. [14] proposed a blocking approach using table embeddings obtained via a neural language model. Chen et al. [3] introduced an approach called ColNet that provides column embeddings. It utilizes a knowledge base, word embeddings, and a machine learning model to annotate the type of columns and a CNN[6] to learn correlations between cells. Ghasemi-Gol et al. [15] presented an approach that provides cell embeddings and does table cell-type classification in two steps. Sun et al. [25] introduced a hybrid probabilistic approach for table understanding, this approach uses table cell embeddings from Ghasemi-Gol et al. [15]. Nishida et al. [22] proposed an architecture called TabNet that can provide cell embeddings. It uses RNN[7] and CNN to perform table-type classification. All of the above approaches use or provide embeddings that are static and not contextual.

Tabular model approaches pre-trained on the surrounding text and tabular data include TaPas introduced by Herzig et al. [16]. It is based on BERT and trained by weak supervision. Yang et al. [28] introduced a model that is an upgraded version of TaPas. The model is invariant to the position of rows and columns in the table and thus produces more efficient results in the downstream tasks. Eisenschlos et al. [13] introduced a model called MATE[8] that can process larger tables as compared to TaPas. Yin et al. [29] presented a generalized approach called TaBERT. It is a pre-training approach for understanding the joint representation of tabular data and natural language utterances. Yu et al. [30] introduced a framework called GRAPPA which is initialized using RoBERTa architecture [21]. They also created synchronous context-free grammar (SCFG) from already available text-to-SQL[9] datasets. SCFG transforms natural language sentences into SQL queries. Deng et al. [7] introduced a framework for text-to-SQL transformation that was trained using a table corpus. Liu et al. [20] proposed a model called Tapex which is pre-trained to act like an SQL query executor. However, it is an encoder + decoder model so it does not explicitly provide any type of embeddings.

Tabular model approaches pre-trained on tabular data only include a framework called Turl introduced by Deng et al. [6]. This framework learns a deep contextualized representation of relational tables using unsupervised methods. Later, the pre-trained model can be fine-tuned further on task-specific datasets. Du et al. [11] proposed a neural network architecture called TabularNet that provides an in-depth understanding of tabular data by utilizing both spatial and relational information simultaneously. Wang et al. [27] proposed an approach called Tuta. It is aimed at tables of different kinds of structures such as relational tables, spreadsheet tables, and hierarchical tables. Iida et al. [19] introduced an approach called Tabbie which uses two independent transformers to provide cell embeddings by independently encoding rows and columns. Cheng et al. [4] proposed a model called

[6]CNN: Convolutional neural network

[7]RNN: Recurrent neural network

[8]Multi-view attention for table transformer efficiency

[9]SQL: Structured Query Language

Fortap which uses Tuta as its base and it uses spreadsheet formulas for pre-training.

3 Methodology

To automate the consistency checking task, the proposed pipeline divides it into three steps to make it modular. The steps are:

1. Getting key performance indicators (KPI) embeddings from text
2. Getting KPI embeddings from table
3. Doing consistency checking

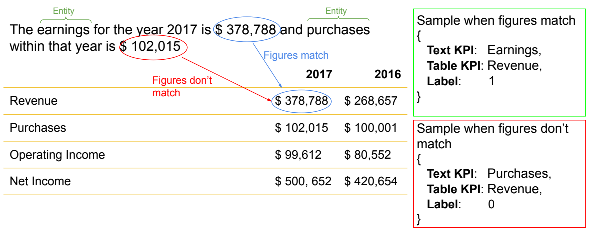
In the first step, the KPIs embeddings from the text of the financial document are extracted using BERT [9].

In the second step, the embeddings of the KPIs from the tables of financial documents are obtained. In a table, the KPIs are present in each cell of the first column. For extracting the KPI embedding from the table, three different transformer-based models that are pre-trained using tabular data and/or their metadata only are used. This work does not consider models that require surrounding text with the table as the cell embeddings should be generated while only considering the tabular data. Therefore, this work considers only Tabbie [19], Turl [6], and Tuta [27] as models for generating KPIs embeddings.

Finally, for performing consistency checking support vector machine for classification (SVC) from scikit-learn is used as a binary classifier to identify whether the sample is consistent or not. SVC is used because it defines clear boundaries for the classification of two classes. For consistency checking SVC takes input a concatenation of text KPI embeddings, table KPI embeddings, and embeddings of numbers related to text KPI. Based on such inputs, the SVC outputs a label that indicates whether the KPIs are similar or not.

3.1 Data Generation Process

During the consistency checking procedure, the trained model computes embeddings of each KPI in the text and matches it with the computed embeddings of the KPIs in the table, if the KPIs match then the numerical consistency of the two KPIs can be checked. To train such a model, first, the data extracted from financial documents is passed through a named entity recognition and relation extraction model to identify KPIs, numerical values, and to which year (current year or previous year) these numerical values belong [18]. Then training samples consisting of KPIs from both text and table along with a label indicating whether their numerical values match or not are constructed using only those financial documents that have tables consisting of "balance", "profit and loss account" and "fixed assets" information. Because these tables record information that is relevant to tracking the progress of a company or a similar entity. The process and pseudo examples of dataset samples are shown in Figure 3.


Entity
Entity
The earnings for the year 2017 is $ 378,788 and purchases within that year is $ 102,015
Figures match
Figures don't match

	2017	2016
Revenue	$ 378,788	$ 268,657
Purchases	$ 102,015	$ 100,001
Operating Income	$ 99,612	$ 80,552
Net Income	$ 500, 652	$ 420,654

Sample when figures match
{
Text KPI: Earnings,
Table KPI: Revenue,
Label: 1
}

Sample when figures don't match
{
Text KPI: Purchases,
Table KPI: Revenue,
Label: 0
}

Figure 3: Process of generating dataset samples for consistency checking task. The green box shows a positive sample and the red box shows a negative sample.

3.2 Process for Getting Embeddings from Text Models

To get embeddings of the text KPI entities *bert-base-german-dbmdz-cased*[10] is used because the financial documents considered in this work are in the German language. From BERT's ablation studies, token representation obtained by concatenating embeddings from the top four layers of BERT gives the best performance Therefore the embeddings of KPI from the top four layers of BERT are extracted and summed to combine the four layers' embeddings. Furthermore, BERT's tokenizer tokenizes the word into subwords so currently the embeddings of KPI obtained from BERT are also divided into subwords. The complete word embeddings were obtained by taking the mean of subwords embeddings. If the KPI entity is made up of

[10]https://huggingface.co/dbmdz/bert-base-german-cased

more than one word then the mean of all word embeddings is taken to get one embedding vector of 756 elements for the whole KPI. The sequences of operations for combining the embeddings of the last four layers of BERT (i.e. sum, mean, and mean) are found by experimenting with all the possible combinations. As an example, assume a KPI of one word "Auditing". Its embeddings will be obtained by first tokenizing it into

$$[Au,\ \#\#diti,\ \#\#ng]$$

BERT will process this tokenized KPI and provide embeddings of $\mathbb{R}^{1\times3\times768}$ (not considering the [cls] and [sep] token). After stacking the embedding from the top four layers (embeddings of $\mathbb{R}^{4\times1\times3\times768}$) summing operation will be applied to obtain embeddings of $\mathbb{R}^{1\times3\times768}$. Then mean operation will be applied to get $\mathbb{R}^{1\times768}$ size embeddings. Also if the KPI was of three words then the embeddings would be of $\mathbb{R}^{3\times768}$ and again the mean operation will be applied to convert it into $\mathbb{R}^{1\times768}$.

3.3 Process for Getting Embeddings from Table Models

Tabbie is a pre-training methodology designed specifically for tables and it allows easy access to representations of different parts of a table which are rows, columns, and cells. Tabbie's architecture is based on two different types of transformers that encode columns and rows independently. The training objective used by Tabbie is based on ELECTRA's objective function [5] and it is much simpler as compared to the masked language modeling objective. The authors have provided two different models of Tabbie which are FREQ and MIX models and they are based on different pre-training strategies. To get table cell embeddings from Tabbie, each of the tables is transformed into a structure that is understandable by Tabbie, and then it is provided to the model as input. Moreover, Tabbie provides two types of cell embeddings one from row transformer and the other from column transformer so to get the overall cell embeddings, an average operator is applied to these two embeddings.

Tuta is also a transformer architecture-based method and it is used for the understanding of generally structured tables such as tables from a spreadsheet, relational tables, and web tables. The locations of the cells of the table are understood by Tuta using a structure called Bi-dimensional coordinate tree. Tuta captures hierarchical information of table using tree-based positional embeddings and to capture spatial information Tuta combines tree-based coordinates with rectangular coordinates. For cells to locally aggregate their structural information, Tuta implements a mechanism called a structure-aware attention mechanism. Tuta is trained on 3 different pre-training tasks to provide representations of different parts of the table. These tasks are Masked Language Model for getting token-level representation, Cell-Level Cloze for cell-level representation, and Table-Context-Retrieval for table representation. To get cell embeddings from Tuta, its cell-type classification model with implicit positional embeddings and tree-based attention of size two is selected because ablation studies suggest that for the task of cell-type classification this variant has better performance than the other model variants. In addition, to get the embeddings each table is transformed into a specific structure that is a list of lists (each list represents information of a single row)[11], and then further meta information is appended with the transformed structure so that it is suitable to be processed by modified Tuta tokenizer and Tuta model.

Turl is the framework for pre-training/fine-tuning relation tables. It has a structure-aware transformer encoder with a visibility matrix that can capture the structural information of the table. Moreover, in addition to the masked language model objective that helps in capturing semantic information, Turl is also trained using an objective called Masked entity recovery, which helps in capturing factual information. To get cell embeddings from Turl, its column type annotation (CT) model is used because to label the column, CT generates cell embeddings to get whole column embeddings. Also, the CT model that does not require additional entity embedding is used because the generated data does not have this information. Additionally, the financial tables are transformed into a form that is understandable by CT and are passed with fake meta information to Turl to extract the cell embeddings.

[11]python implementation

4 Evaluation

To evaluate the proposed consistency checking methodology and compare the cell embeddings from the three selected tabular models, a dataset composed of consistent and inconsistent samples is generated using the process defined in Subsection 3.1. 80% of the dataset is split into training and 20% into testing. So overall training data has 6044 data points and test data has 1512 data points and both have half consistent and inconsistent samples. For evaluation, the conventional accuracy metric is used because the dataset has a similar number of consistent and inconsistent samples and all the experiments were run 10 times by using 10 different seeds[12]. Additionally, for each tabular model, special experiments are also done in which the tables and text KPIs are translated into the English language using the *opus-mt* model provided by the EasyNMT library. The purpose of these experiments is to check how much the consistency checking accuracy is affected when the selected tabular models are used in the language on which they were originally pre-trained.

Initially, to evaluate whether cell embeddings from tabular models help in the consistency checking task, a benchmark is created in which both text and table KPI embeddings are generated using the *bert-base-german-dbmdz-cased* model. After evaluating the benchmark model on the test dataset, an accuracy of **0.7226** $\pm$ **0.0126** is achieved.

In the first set of experiments, the table KPI embeddings are extracted from Tabbie. Moreover, Tabbie's both "Freq" and "Mix" models are used in the experiments. Also, Tabbie by default uses *bert-base-uncased* as its base model, to check whether Tabbie performs better if the base BERT model was different, the default Tabbie base model is replaced by *bert-base-multilingual-cased.* For the text KPI embeddings, *bert-base-german-dbmdz-cased* and *bert-base-multilingual-cased* versions of BERT models are used in the experiments. Finally, experiments were also done by translating the KPIs from text and table into English, in these experiments, Tabbie's base model and text KPIs' BERT model is *bert-base-uncased.* After a grid search of all the possible combinations of extracting table and text KPIs embeddings and also considering the translation mechanism, the best-performing combination has an accuracy of **0.7281** $\pm$ **0.0122** and in it, the KPIs are translated into English, and Tabbie model type is **FREQ**.

In the second set of experiments, the Tuta-implicit model with default setting is used to extract table KPI embeddings, and text KPI embeddings are obtained using *bert-base-german-dbmdz-cased* and *bert-base-multilingual-cased.* For experiments in which the KPIs are translated into the English language, *bert-base-uncased* is used to get text KPI embeddings. After considering all the possible combinations of table and text models in the experiments and also considering the translation mechanism, the best performing combination is found to be using English-translated KPIs from both text and table and it has an accuracy of **0.7248** $\pm$ **0.0150**.

In the final set of experiments, the Turl CT model as described in Subsection 3.3 is used to extract table KPI embeddings and for text KPI embeddings similar versions of the BERT model as in the experiments of Tabbie and Tuta are used. Experiments are also performed in which English-translated table and text KPIs are used and in them *bert-base-uncased* is used to get text KPI embeddings. After performing all of the possible experiments, the best performing combination uses English-translated KPIs, uses *bert-base-uncased* as its text KPI embeddings model, and has an accuracy of **0.7011** $\pm$ **0.0110**.

From all the results, it can be concluded that those experiments that use English-translated KPIs perform better than those in the German language and this makes sense because models used here are originally pre-trained on the English language corpus. Additionally, Tabbie performs better than the benchmark and the rest of the models this may be because Tabbie is originally designed to work only with tables and thus produces better results when only tables are provided[13].

[12]Seeds from 1 to 10

[13]We apologize that we can't publish the code and dataset because it is the property of the customer.

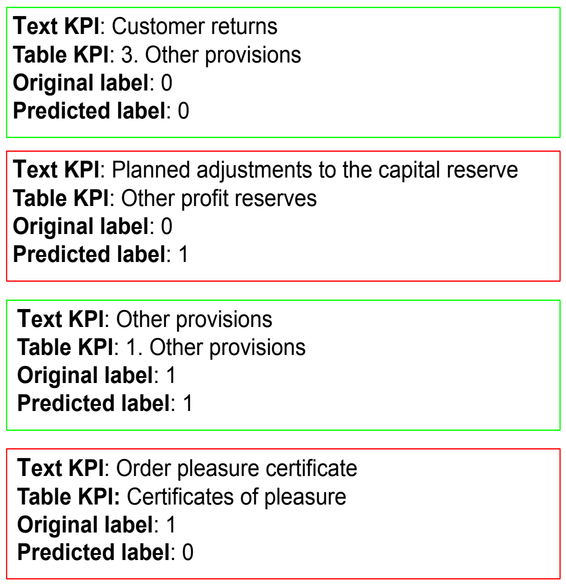
5 Conclusion and Future Work

This paper proposes a pipeline to automate the task of consistency checking within a financial document. This pipeline is evaluated on a dataset of financial documents in the German language. The main component of the pipeline is the model that provides KPIs embedding from the table and three different models that are Tabbie, Turl, and Tuta are selected to get these embeddings. These models and the benchmark are evaluated on the consistency checking task using the proposed pipeline to check the practical usage of our approach and the quality of the cell embeddings obtained from these models. The best results from the experiments on the three models along with the benchmark are provided in Table 1. From the results, it can be seen that cell embeddings obtained from the Tabbie-FREQ version produced the highest accuracy in the task of consistency checking, surpassing the benchmark by 0.0055 points. This high performance of the Tabbie model can be due to its architecture and the way it is pre-trained which specifically allows it to process the information in the tables, the examples of classified samples are given in Figure 4. Moreover, the cell embeddings from Tuta also surpass the benchmark by 0.0022 points. However, the cell embeddings from Turl do not surpass the benchmark, this may be because Turl also needs extra information while processing tables and since this was not available therefore fake information was provided to it. Also, all of those experiments that produced the highest accuracy used English-translated text and table KPIs, and all used *bert-base-uncased* to embed the KPIs from the text.

However, all of the models used here are evaluated on a dataset consisting of a similar number of consistent and inconsistent examples. In future work, the accuracy of the consistency checking task can be evaluated by creating an imbalanced dataset that would have more inconsistent examples than consistent examples (i-e real-world scenarios). Furthermore, all of the table-related models that are used in this project were pre-trained on an English corpus, hence, to get a more contextually rich representation and better results, these models can be pre-trained on a German table corpus.

	Table cell embeddings model	**Text embeddings model**	**Accuracy**	**F1-score Match labels**
Benchmark	BERT-Y	BERT-Y	0.7226± 0.0126	0.7092± 0.0170
1	Tabbie-FREQ	BERT-X	**0.7281± 0.0122**	**0.7139± 0.0144**
2	Tuta	BERT-X	0.7248± 0.0150	0.7095± 0.0196
3	Turl	BERT-X	0.7011± 0.0110	0.6788± 0.0182

Table 1: Best results from all models along with the benchmark. Here BERT-X is *bert-base-uncased* and BERT-Y is *bert-base-german-dbmdz-cased*. All tabular models use English-translated table and text KPIs. F1-score is of match labels.



Text KPI: Customer returns
Table KPI: 3. Other provisions
Original label: 0
Predicted label: 0

Text KPI: Planned adjustments to the capital reserve
Table KPI: Other profit reserves
Original label: 0
Predicted label: 1

Text KPI: Other provisions
Table KPI: 1. Other provisions
Original label: 1
Predicted label: 1

Text KPI: Order pleasure certificate
Table KPI: Certificates of pleasure
Original label: 1
Predicted label: 0

Figure 4: Examples of samples classified by Tabbie FREQ model. The samples in the green box are correctly classified and the ones in the red box are wrongly classified.

6 Acknowledgement

We are grateful to the anonymous reviewer for pointing out the flaws in our approach and helping us in improving them.

References

[1] D. Biesner, R. Ramamurthy, R. Stenzel, M. Lübbering, L. Hillebrand, A. Ladi, M. Pielka, R. Stenzel, R. Loitz, C. Bauckhage, and R. Sifa. Anonymization of German financial documents using neural network-based language models with contextual word representations. *Int. J. Data Sci. Anal.*, 2021. doi: 10.1007/s41060-021-00285-x.

[2] Y. Cao, H. Li, P. Luo, and J. Yao. Towards automatic numerical cross-checking: Extracting formulas from text. In *Proc. WWW*, 2018. doi: 10.1145/3178876.3186166.

[3] J. Chen, E. Jiménez-Ruiz, I. Horrocks, and C. Sutton. Colnet: Embedding the semantics of web tables for column type prediction. In *Proc. AAAI*, 2019. doi: 10.1609/aaai.v33i01.330129.

[4] Z. Cheng, H. Dong, R. Jia, P. Wu, S. Han, F. Cheng, and D. Zhang. Fortap: Using formulas for numerical-reasoning-aware table pretraining. In *Proc. ACL*, 2022. doi: 10.18653/v1/2022.acl-long.82.

[5] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning. Electra: Pre-training text encoders as discriminators rather than generators. In *Proc. ICLR*, 2020. doi: 10.48550/ARXIV.2003.10555.

[6] X. Deng, H. Sun, A. Lees, Y. Wu, and C. Yu. Turl: Table understanding through representation learning. *Proc. VLDB Endow.*, 2020. doi: 10.5555/3430915.3442430.

[7] X. Deng, A. Hassan, C. Meek, O. Polozov, H. Sun, and M. Richardson. Structure-grounded pretraining for text-to-sql. In *Proc. NAACL-HLT*, 2021. doi: 10.18653/v1/2021.naacl-main.105.

[8] T. Deußer, S. M. Ali, L. Hillebrand, D. Nurchalifah, B. Jacob, C. Bauckhage, and R. Sifa. KPI-EDGAR: A novel dataset and accompanying metric for relation extraction from financial documents. In *Proc. ICMLA*, 2022. doi: 10.48550/arXiv.2210.09163.

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proc. NAACL-HLT*, 2019. doi: 10.18653/v1/n19-1423.

[10] H. Dong, Z. Cheng, X. He, M. Zhou, A. Zhou, F. Zhou, A. Liu, S. Han, and D. Zhang. Table pre-training: A survey on model architectures, pretraining objectives, and downstream tasks. *CoRR*, 2022. doi: 10.48550/ARXIV.2201.09745.

[11] L. Du, F. Gao, X. Chen, R. Jia, J. Wang, J. Zhang, S. Han, and D. Zhang. TabularNet: A neural network architecture for understanding semantic structures of tabular data. In *Proc. KDD*, 2021. doi: 10.1145/3447548.3467228.

[12] G. D'Atri. Logic-based consistency checking of XRBL instances. *IJACT*, 2014.

[13] J. Eisenschlos, M. Gor, T. Mueller, and W. Cohen. MATE: multi-view attention for table transformer efficiency. In *Proc. EMNLP*, 2021. doi: 10.18653/v1/2021.emnlp-main.600.

[14] A. L. Gentile, P. Ristoski, S. Eckel, D. Ritze, and H. Paulheim. Entity matching on web tables: a table embeddings approach for blocking. In *Proc. EDBT*, 2017. doi: 10.5441/002/edbt.2017.57.

[15] M. Ghasemi-Gol, J. Pujara, and P. Szekely. Learning cell embeddings for understanding table layouts. *Knowl. Inf. Syst.*, 2021. doi: 10.1007/s10115-020-01508-6.

[16] J. Herzig, P. K. Nowak, T. Mueller, F. Piccinno, and J. Eisenschlos. TaPas: Weakly supervised table parsing via pre-training. In *Proc. ACL*, 2020. doi: 10.18653/v1/2020.acl-main.398.

[17] L. Hillebrand, T. Deußer, T. Dilmaghani, B. Kliem, R. Loitz, C. Bauckhage, and R. Sifa. Towards automating numerical consistency checks in financial reports. In *Proc. BigData*, 2022. doi: 10.48550/arXiv.2211.06112.

[18] L. Hillebrand, T. Deußer, T. Dilmaghani, B. Kliem, R. Loitz, C. Bauckhage, and R. Sifa.

KPI-BERT: A joint named entity recognition and relation extraction model for financial reports. In *Proc. ICPR*, 2022. doi: 10.1109/ICPR56361.2022.9956191.

[19] H. Iida, D. Thai, V. Manjunatha, and M. Iyyer. TABBIE: pretrained representations of tabular data. In *Proc. NAACL-HLT*, 2021. doi: 10.18653/v1/2021.naacl-main.270.

[20] Q. Liu, B. Chen, J. Guo, M. Ziyadi, Z. Lin, W. Chen, and J.-G. Lou. TAPEX: table pre-training via learning a neural SQL executor. In *Proc. ICLR*, 2022. doi: 10.48550/arXiv.2107.07653.

[21] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *CoRR*, 2019. doi: 10.48550/arXiv.1907.11692.

[22] K. Nishida, K. Sadamitsu, R. Higashinaka, and Y. Matsuo. Understanding the semantic structures of tables with a hybrid deep neural network architecture. In *Proc. AAAI*, 2017. doi: 10.1609/aaai.v31i1.10484.

[23] R. Ramamurthy, M. Pielka, R. Stenzel, C. Bauckhage, R. Sifa, T. D. Khameneh, U. Warning, B. Kliem, and R. Loitz. ALiBERT: improved automated list inspection (ALI) with BERT. In *Proc. DocEng*, 2021. doi: 10.1145/3469096.3474928.

[24] R. Sifa, A. Ladi, M. Pielka, R. Ramamurthy, L. Hillebrand, B. Kirsch, D. Biesner, R. Stenzel, T. Bell, M. Lübbering, et al. Towards automated auditing with machine learning. In *Proc. DocEng*, 2019. doi: 10.1145/3342558.3345421.

[25] K. Sun, H. Rayudu, and J. Pujara. A hybrid probabilistic approach for table understanding. In *Proc. AAAI*, 2021. doi: 10.1609/aaai.v35i5.16562.

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Proc. NIPS*, 2017. doi: 10.48550/arXiv.1706.03762.

[27] Z. Wang, H. Dong, R. Jia, J. Li, Z. Fu, S. Han, and D. Zhang. TUTA: tree-based transformers for generally structured table pre-training. In *Proc. KDD*, 2021. doi: 10.1145/3447548.3467434.

[28] J. Yang, A. Gupta, S. Upadhyay, L. He, R. Goel, and S. Paul. TableFormer: robust transformer modeling for table-text encoding. In *Proc. ACL*, 2022. doi: 10.18653/v1/2022.acl-long.40.

[29] P. Yin, G. Neubig, W.-t. Yih, and S. Riedel. TaBERT: pretraining for joint understanding of textual and tabular data. In *Proc. ACL*, 2020. doi: 10.18653/v1/2020.acl-main.745.

[30] T. Yu, C.-S. Wu, X. V. Lin, Y. C. Tan, X. Yang, D. Radev, C. Xiong, et al. GraPPa: grammar-augmented pre-training for table semantic parsing. In *Proc. ICLR*, 2021. doi: 10.48550/arXiv.2009.13845.

[31] L. Zhang, S. Zhang, and K. Balog. Table2Vec: neural word and entity embeddings for table population and retrieval. In *Proc. SIGIR*, 2019. doi: 10.1145/3331184.3331333.

[32] F. Zhu, D. Ning, Y. Wang, and S. Liu. A Novel Cost-sensitive Capsule Network for Audit Fraud Detection. In *Proc. IUCC*, 2021. doi: 10.1109/IUCC-CIT-DSCI-SmartCNS55181.2021.00091.

[33] A. Zisman and A. Athanasopoulou. Consistency Management of Financial XML Documents. In *Proc. CAiSE*, 2001. doi: 10.1007/3-540-45341-5_15.