Tabular Data Concept Type Detection Using Star-Transformers

Yiwei Zhou*
yiwei1.zhou@gmail.com

Siffi Singh
Amazon Alexa
United Kingdom
siffis@amazon.com

Christos Christodoulopoulos
Amazon Alexa
United Kingdom
chrchrs@amazon.com

ABSTRACT

Tabular data is an invaluable information resource for search, information extraction and question answering about the world. It is critical to understand the semantic concept types for table columns in order to fully exploit the information in tabular data. In this paper, we focus on learning-based approaches for column concept type detection without relying on any metadata or queries to existing knowledge bases. We propose a model that employs both statistical and semantic features of table columns, and use Star-Transformers to gather and scatter information across the whole table to boost the performance on individual columns. We apply distant supervision to construct a tabular dataset with columns annotated with DBpedia classes. Our experiment results show that our model achieves 93.57 accuracy on the dataset, exceeding that of the state-of-the-art baselines.

CCS CONCEPTS

• **Computing methodologies** → **Natural language processing**; *Machine learning*; • **Information systems** → *Information integration.*

KEYWORDS

Tabular data, concept type detection, neural networks, column classification

ACM Reference Format:
Yiwei Zhou, Siffi Singh, and Christos Christodoulopoulos. 2021. Tabular Data Concept Type Detection Using Star-Transformers. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21), November 1–5, 2021, Virtual Event, QLD, Australia.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3459637.3482197

1 INTRODUCTION

Tabular data is one of the most common data formats, which widely exists in HTML documents, databases, as well as business reports. Tables present information in a compact and clean manner, with cells in the same column belonging to the same concept type, such as `Movie`, `Person`, `Award`, `Software`, etc., and columns in the same table belonging to related concept types. In order to search, extract and aggregate information from tabular data, concept type detection is needed to understand the semantic meaning of table columns.

Most early concept type detection approaches [9, 11, 14, 18, 21] rely on querying knowledge bases (KB) in real-time in order to associate each cell with an existing entity first. The concept type of a column is therefore dependent on the concept types of linked entities. These lookup-based approaches are heavily influenced by the completeness of the KB, and are difficult to generalise to unseen entities or entity references. Large number of queries to the KB also introduce extra operational cost.

To overcome the limitations of lookup-based approaches, recently some studies [4, 6, 8, 13, 15, 20] have explored the use of learning-based approaches for column classification. While these works have achieved significant progress, there are still some challenges to be tackled. (1) Each cell value in a table is a phrase or short string, rather than complete and natural sentences. The ordering of rows and columns has little influence on the overall semantic meaning of the table, which needs to be considered when modelling the tabular data. (2) Since there does not exist a large scale tabular dataset annotated with concept types, learning-based approaches need to be trained on datasets collected through distant supervision. On one hand, it is essential to reduce the noise in the collected dataset; on the other hand, the proposed solution needs to be robust to missing labels, which is inevitable for distant supervision datasets. (3) Predicting the concept type for a single column can be challenging, as the information contained is limited. Since a table is a collection of columns sharing related semantic meanings, it is beneficial to make the prediction based on the whole table rather than individual columns. In this paper, we focus on the aforementioned challenges, and the main contributions are as follows:

- We employ order-invariant statistical and semantic features to generate representation vectors for table columns, and propose a novel network architecture based on Star-Transformer [7] to enable the information exchange between columns. The resulting model outperforms existing approaches across all the evaluation metrics in the concept type detection task.
- We construct a concept type detection dataset by leveraging hyperlinks in Wikipedia tables, columns in this dataset are labelled with DBpedia classes.

2 TABLE COLUMN CONCEPT DETECTION

2.1 Problem Definition

One motivation for this task is to expand the scope of concept type detection models to different kinds of tabular data coming from diverse resources, without assuming the existence of information beyond the table content, such as informative table headers, table captions, hyperlinks to entities, etc. Thus, following [8, 20], we define the concept type detection problem for table columns as:

Definition 2.1 (Concept type detection problem)**.** Given a horizontal table T of M data rows and N data columns, classify the

*Work done while at Amazon.

CIKM '21, November 1–5, 2021, Virtual Event, QLD, Australia

ACM ISBN 978-1-4503-8446-9/21/11.
https://doi.org/10.1145/3459637.3482197

concept type for all the columns: $Y = (y_0, y_1, \cdots, y_{N-1})$, where $y_n \in \{0, \cdots, K-1\}$, K is the total number of concept types under consideration.

For simplicity, we use $t_{m,n}$, where $m \in \{0, \cdots, M-1\}$ and $n \in \{0, \cdots, N-1\}$ to represent a cell in the table; $c_n = (t_{0,n}, t_{1,n}, \cdots, t_{M-1,n})$ to represent a column in the table.

2.2 Model Description

Feature engineering [4, 8, 13, 15, 20] has been extensively applied in *learning-based* column classification approaches. Previous experiments [8] show that these extracted statistical features add extra value on top of semantic embedding features of cell value tokens. Following [8], we use three different ways to generate a representation vector for each column: global statistics ($\boldsymbol{s}_n$), character-level distributions ($\boldsymbol{v}_n$) and semantic embeddings ($\boldsymbol{e}_n$). The 27 global statistic features include number of values, maximum value length, mean number of alphabetic characters in cells, fraction of cells with numeric characters, column entropy, etc. Character-level distribution features aim at describing distributions of 96 ASCII-printable characters in a column. First, the appearance frequency of each character in each cell of a column is calculated; cell-level frequencies are then aggregated through 10 aggregation functions (any, all, mean, variance, min, max, median, sum, kurtosis and skewness) to generate 960 column-level character distribution features. [8] provides additional details about those two types of features.

For semantic embedding features, we first generate the cell-level embedding by taking the mean of word embeddings in the cell, then we use the average of cell-level embeddings as the embedding features for a column. Note that we only use order-invariant features to represent a column, as the semantic meaning of a table column is not dependent on the ordering of cells.

In summary, the representation vector for each column is $\boldsymbol{c}_n = (\boldsymbol{s}_n \oplus \boldsymbol{v}_n \oplus \boldsymbol{e}_n)$, where $\oplus$ denotes the concatenation operation.

Most previous works [4, 8, 13, 15] treat columns in the same table separately. This greatly limits their models' abilities to incorporate signals from other columns in the same table when making a prediction. Column concept type prediction on individual columns can be difficult because the information provided within each column is limited; however, columns in the same table are usually of related concept types, which can help disambiguate each other. Similar to table rows, table columns are very weakly order-dependent. We would like the model to allow direct information flow between any two columns in the same table, and minimise the influence of their relative positions. Considering the aforementioned requirements, we apply Star-Transformer [7] on columns' representation vectors to enable information sharing among columns and capture cross-column dependencies. Star-Transformer has proven to achieve significant improvements against standard Transformer [17], and is good at capturing both local composition and long-range dependency with reduced complexity [7].

The overall architecture of the proposed model can be found in Figure 1. The Star-Transformer, as shown in Figure 2, reflects the relationship between columns in a table. It consists of one relay node ($\boldsymbol{s}$) and N satellite nodes ($\boldsymbol{h}_n$), with each satellite node representing one column and the relay node acting as a virtual hub to gather and scatter information from and to all the columns. Parameters

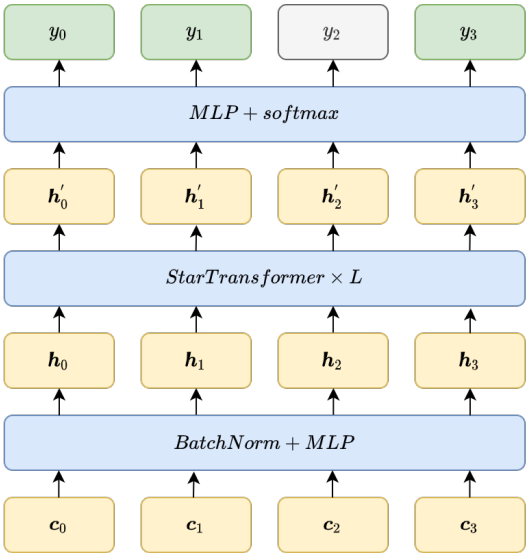

Figure 1: Model architecture. We take the columns $c_0, c_1, c_2, ..$ as inputs, and apply batch normalisation to multi-layer perceptron (MLP), which is followed by L layers of Star Transformer. We then use a MLP with softmax function to classify over a set of target concept types. Columns with missing labels are ignored (here y_2).

in Star-Transformer are updated through multiple steps. For each step t, state (representation vector) for a column is updated from its neighbouring columns' states, the relay node and its own previous state:

$$\boldsymbol{h}_n^t = MultiAtt(\boldsymbol{h}_n^{t-1}, [\boldsymbol{h}_{n-1}^{t-1}; \boldsymbol{h}_n^{t-1}; \boldsymbol{h}_{n+1}^{t-1}; \boldsymbol{h}_n; \boldsymbol{s}^{t-1}]),$$

where $MultiAtt$ represents Multi-head Attention calculation as defined in [17]. After that, the relay node summarises information of all the columns and its own previous state:

$$\boldsymbol{s}^t = MultiAtt(\boldsymbol{s}^{t-1}, [\boldsymbol{s}^{t-1}; \boldsymbol{H}^t]),$$

where $\boldsymbol{H}^t = [\boldsymbol{h}_0^t; \cdots; \boldsymbol{h}_{N-1}^t]$.

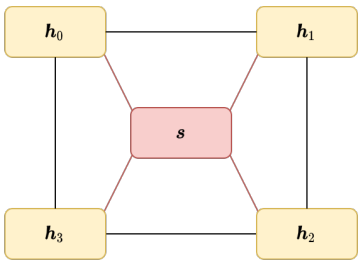

Figure 2: Star-Transformer. h_n represents the state for each column (satellite node), s represents the state for the relay node.

3 DATASET CONSTRUCTION

We created a tabular dataset for *open domain* from Wikipedia tables [1], with columns annotated with *DBpedia classes* through distant supervision. Different from other knowledge bases, DBpedia classes follow clear hierarchical structure: there is no semantic overlap between classes in different branches, and it is straightforward to infer the common ancestors of any pair of classes. Besides, DBpedia is actively being updated and enriched, thus our solution can be easily extended to new entities and classes. We use level 2 and level 3 DBpedia classes[1] (children and grandchildren of `http://dbpedia.org/ontology/Thing`) as target concept types. Many cells in Wikipedia tables [1] contain hyperlinks to corresponding Wikipedia entities, whose class information can be inferred from the DBpedia knowledge base. We use Algorithm 1 to aggregate DBpedia class information for hyperlinked Wikipedia entities in the same column, in order to generate the column-level concept type label. In Algorithm 1, we prioritise to use more specific (level 3) DBpedia classes if possible, and will fallback to less specific (level 2) DBpedia classes only if the hyperlinked Wikipedia entities cannot agree on a more specific DBpedia classes. Note that the hyperlinks in tables are only used to generated distant supervision labels for training, they are not needed for inference. We do not use the dataset in [8] because that dataset only contains individual columns and it is impossible to revert them back to tables; and their labels are a mix of DBpedia classes and DBpedia properties with overlapping semantic meanings, e.g., `Ranking` and `Rank`.

Algorithm 1: Generate column concept type label through distant supervision.

```
Data: Level 2 and level 3 DBpedia classes for hyperlinked
      Wikipedia entities in a column.
Result: Concept type label for the column.
Calculate frequency (freq_3) of the most common level 3
 DBpedia class (cls_3);
Calculate frequency (freq_2) of the most common level 2
 DBpedia class (cls_2);
if freq_3 ≥ 3 then
    return cls_3;
else
    if freq_2 ≥ 3 then
        return cls_2;
    else
        return None
    end
end
```

We split the collected tables into training, validation and test datasets. We introduce an additional `Other` class to cover concept types that appear in fewer than 100 tables in the training dataset. Unseen concept types in the validation and test datasets are also mapped to the `Other` class. This results in 43 target concept types. Detailed statistics about the training/validation/test datasets can be found in Table 1. According to Wikipedia's editorial policy, editors are encouraged to adopt homogeneous headers on tables that describe similar concepts [2]. To avoid information leakage and increase the model's generalisation capability to tabular data coming from any resource, we did not use headers of the collected tables.

[1] https://wiki.dbpedia.org/services-resources/ontology

Table 1: Dataset Statistics

Dataset	# Tables	# Columns	# Labelled Columns
Training	172,293	1,091,223	301,801
Validation	21,537	136,659	37,835
Test	21,537	135,715	37,739

4 EXPERIMENTS

4.1 Baselines

The proposed model is compared with the following baselines.

Sherlock [8]. The original Sherlock[2] model is a multi-input neural network. Specifically, authors consider features of four categories: global statistics, character-level distributions, word embeddings and paragraph vectors. They train a sub-network for each feature category except global statistics, then concatenate output weights of the three sub-networks with global statistics to form the input layer of the primary network.

Sherlock+. If the Star-Transformer layer is removed from our model, it is similar to Sherlock, thus we use *Sherlock+* to denote this model. Comparison between Sherlock+ and our model can demonstrate the effectiveness of Star-Transformer [7] layer. Note that *Sherlock+* is different from original Sherlock model as follows. (1) Instead of using the original multi-input architecture, we concatenate all the features directly as the input for the model. This allows direct interactions among different feature categories. (2) We remove paragraph vectors from the feature set, as it requires to train a separate table column-specific paragraph vector [5] model. Paragraph vectors aim at capturing semantic meanings of table columns, which can be covered by word embeddings if properly trained. (3) In the original Sherlock model, column embedding features contain pre-calculated mean, mode, median and variance of cell-level embeddings. To simplify the model and train end-to-end with respect to word embedding matrix, only the mean operation in Sherlock+ is considered.

TaBERT [19]. TaBERT[3] is a pre-trained language model that jointly learns representations for natural language sentences and tables. TaBERT represents each cell by the column name, column datatype and cell value, which is followed by using the Transformer model to generate row-level vectors. To allow for information flow across cells of different rows, TaBERT uses a vertical self-attention mechanism. TaBERT introduces two pre-training tasks, Masked Column Prediction and Cell Value Recovery to generate powerful contextualised representations. TaBERT has been tested only on semantic parsing tasks. We fine-tune TaBERT for the table column concept type detection task by introducing an additional classification layer.

TURL [6]. Different from TaBERT, which first models table rows and then table columns, TURL[4] employs a tabular structure-aware

[2] https://github.com/mitmedialab/sherlock-project
[3] https://github.com/facebookresearch/TaBERT
[4] https://github.com/sunlab-osu/TURL

encoder. This is achieved through a visibility matrix, which acts as an attention mask so that words in each cell can only aggregate information from other structurally related words during the *MultiAtt* calculation. Pre-training tasks of TURL are Masked Language Model and Masked Entity Recovery. We directly fine-tune TURL's concept type detection model on our dataset using its released code.

4.2 Training Details

We use 300 dimensional Glove [12] word embeddings pre-trained on 42 billion uncased tokens from Common Crawl to initialise the word embedding matrix. We apply a cross entropy loss of the softmax outputs with respect to the true concept type. It is inevitable that labels for some table columns in the training dataset are missing, either because of missing hyperlinks or missing DBpedia classes for hyperlinked Wikipedia entities. We ignore columns with missing labels in our training; hence y_2 (representing a column with missing label) in Figure 1 is in a different color with the rest. We additionally leverage label smoothing [16] to regularise model training. We empirically set the maximum number of tokens per cell to 10 and the maximum number of rows to 100, in order to improve the model efficiency without impacting the performance. All the parameters were optimised using AdamW [10]. For Sherlock, TaBERT and TURL, we adopt their original regularisation techniques and hyperparameters. Both implementations of TaBERT and TURL set limits on the maximum number of rows, as their memory usage will increase quadratically. Here we experimentally set the limit to 30 for both, the largest number possible without causing GPU out of memory errors.

4.3 Result Analysis

Table 2: Performance Comparison of Different Models

Model	Accuracy	Macro			Weighted		
		P	R	F1	P	R	F1
TaBERT	88.84	69.66	56.59	59.78	88.34	88.84	88.29
Sherlock	90.56	72.20	63.82	66.41	89.89	90.56	90.02
Sherlock+	91.11	74.33	58.66	63.64	90.12	91.11	90.33
TURL	93.39	78.24	72.42	74.34	93.16	93.39	93.16
Our Model	**93.57**	**80.45**	**73.71**	**75.80**	**93.31**	**93.57**	**93.31**

Table 2 displays the performance (accuracy, macro precision/recall/F1, precision/recall/F1 weighted by number of instances) of our model and baselines on the test dataset. TaBERT does not achieve as good performance as the rest models. This is because TaBERT aims at semantic parsing over tables, its column embeddings can help to understand alignments between input texts and column schema, but do not correlate well with the column concept type detection task. TURL achieves better performance than Sherlock and Sherlock+, which demonstrates effectiveness of its structure-aware Transformer by regulating table cells to only attend to relevant contexts, as well as the table-specific pretraining tasks. The changes we introduced to Sherlock was helpful in increasing the model's representation power and generalisation capability. As Sherlock+ outperforms the original Sherlock model in five out of seven of the evaluation metrics (accuracy, macro precision, weighted precision/recall/F1). Our model based on Star-Transformer achieves the best performance across all the evaluation metrics. The significant gap in macro-averaged evaluation metrics illustrates that our model is especially good at boosting the performance for less-populated concept types by leveraging signals from neighbouring columns. The comparison between our model and Sherlock+ demonstrates the importance of using Star-Transformer layers to capture both local and global semantic compositions and dependencies in table columns. Because of our model's relatively simple structure, it can consume large tables with minor extra cost. Additionally, unlike [6, 8], our model does not require table-specific pre-training, which further reduces its maintenance cost.

5 RELATED WORK

Lookup-based concept type detection approaches [9, 11, 14, 18, 21] depend on lexical matching between cell values and entity names. Most of them [9, 11, 14, 21] make joint decisions for linking cells to KB entities, classifying the column concept types, as well as deciding the relationship between column pairs. Through iterative similarity calculation [14] or probabilistic graphic models [9, 11, 21], they try to arrive at the overall optimal solution.

Learning-based Concept Type Detection for tabular data is not a well-defined topic. One related topic is *Semantic Labelling* [13, 15], in which columns are annotated by a pair of values consisting of a class and one of its properties. Another related topic is *Semantic Type Detection* [4, 8, 20], in which column headers are used as targets, which could include both class names (`Artist`, `Company`, `City`, etc.) and property names (`Affiliate`, `BirthDate`, etc.). Our work is different from them as we focus on detecting column's concept type, i.e., the DBpedia class one column belongs to. We do not include properties into our targets because they should be based on column pairs, rather than individual columns. Note that our model aims at predicting concept types for all the columns in a table, rather than only for one subject column [3]. As far as we know, we are the first work to introduce information exchange between columns using Star-Transformers. Besides, our model does not make any assumptions on the availability of table metadata. It can be trained end-to-end without any table-specific pre-training, and it can consume tables generated by distant supervision that are only partially annotated [20].

6 CONCLUSION

Detecting the concept types for table columns is crucial for extracting and integrating information from tabular data. In this work, we propose a column concept type detection model which represents a table column with both statistical and semantic features, and employs Star-Transformer layers to allow information exchange among columns within the same table. We evaluate our model on Wikipedia tables with columns annotated by DBpedia classes through distant supervision. Experimental results demonstrate its performance outperform all the baseline models across all the valuation metrics. A future direction of the task could be applying structured prediction techniques that are robust to missing labels, in order to capture correlations between column labels within the same table.

REFERENCES

[1] Chandra Sekhar Bhagavatula, Thanapon Noraset, and Doug Downey. 2015. Tabel: Entity linking in web tables. In *ISWC*.

[2] Matteo Cannaviccio, Lorenzo Ariemma, Denilson Barbosa, and Paolo Merialdo. 2018. Leveraging wikipedia table schemas for knowledge graph augmentation. In *WebDB*.

[3] Jiaoyan Chen, Ernesto Jiménez-Ruiz, Ian Horrocks, and Charles Sutton. 2019. Learning semantic annotations for tabular data. In *IJCAI*.

[4] Zhiyu Chen, Haiyan Jia, Jeff Heflin, and Brian D Davison. 2018. Generating schema labels through dataset content analysis. In *WWW*.

[5] Andrew M Dai, Christopher Olah, and Quoc V Le. 2015. Document embedding with paragraph vectors. In *NIPS Deep Learning Workshop*.

[6] Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. 2021. Turl: Table understanding through representation learning. In *VLDB*.

[7] Qipeng Guo, Xipeng Qiu, Pengfei Liu, Yunfan Shao, Xiangyang Xue, and Zheng Zhang. 2019. Star-transformer. In *NAACL*.

[8] Madelon Hulsebos, Kevin Hu, Michiel Bakker, Emanuel Zgraggen, Arvind Satyanarayan, Tim Kraska, Çagatay Demiralp, and César Hidalgo. 2019. Sherlock: A deep learning approach to semantic data type detection. In *KDD*.

[9] Girija Limaye, Sunita Sarawagi, and Soumen Chakrabarti. 2010. Annotating and searching web tables using entities, types and relationships. In *VLDB*.

[10] Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *ICLR*.

[11] Varish Mulwad, Tim Finin, and Anupam Joshi. 2013. Semantic message passing for generating linked data from tables. In *ISWC*.

[12] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

[13] Minh Pham, Suresh Alse, Craig A Knoblock, and Pedro Szekely. 2016. Semantic labeling: a domain-independent approach. In *ISWC*.

[14] Dominique Ritze and Christian Bizer. 2017. Matching web tables to dbpedia-a feature utility study. In *EDBT*.

[15] Natalia Rümmele, Yuriy Tyshetskiy, and Alex Collins. 2018. Evaluating approaches for supervised semantic labeling. In *WWW Linked Data on the Web Workshop*.

[16] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *CVPR*.

[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

[18] Petros Venetis, Alon Y Halevy, Jayant Madhavan, Marius Pasca, Warren Shen, Fei Wu, and Gengxin Miao. 2011. Recovering semantics of tables on the web. In *VLDB*.

[19] Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. Tabert: Pretraining for joint understanding of textual and tabular data. In *ACL*.

[20] Dan Zhang, Yoshihiko Suhara, Jinfeng Li, Madelon Hulsebos, Çağatay Demiralp, and Wang-Chiew Tan. 2020. Sato: Contextual semantic type detection in tables. In *VLDB*.

[21] Ziqi Zhang. 2017. Effective and efficient semantic table interpretation using tableminer+. *Semantic Web* 8, 6 (2017), 921–957.