Pythagoras: Semantic Type Detection of Numerical Data Using Graph Neural Networks

Sven Langenecker[1,2], Christoph Sturm[1], Christian Schalles[1] and Carsten Binnig[2,3]

[1]DHBW Mosbach, Lohrtalweg 10, 74821 Mosbach, Germany
[2]TU Darmstadt, Karolinenplatz 5, 64289 Darmstadt, Germany
[3]DFKI Darmstadt, Hochschulstrasse 10, 64289 Darmstadt, Germany

Abstract

Detecting semantic types of table columns is a crucial task to enable dataset discovery in data lakes. However, prior semantic type detection approaches have primarily focused on non-numeric data despite the fact that numeric data play an essential role in many enterprise data lakes. Therefore, typically, existing models are rather inadequate when applied to data lakes that contain a high proportion of numerical data. In this paper, we introduce *Pythagoras*, our new learned semantic type detection approach specially designed to support numerical data along with non-numerical data. *Pythagoras* uses a graph neural network based on a new graph representation of tables to predict the semantic types for numerical data with high accuracy. In our initial experiments, we thus achieve F1-Scores of 0.829 (support-weighted) and 0.790 (macro), respectively, exceeding the state-of-the-art performance significantly.

Keywords

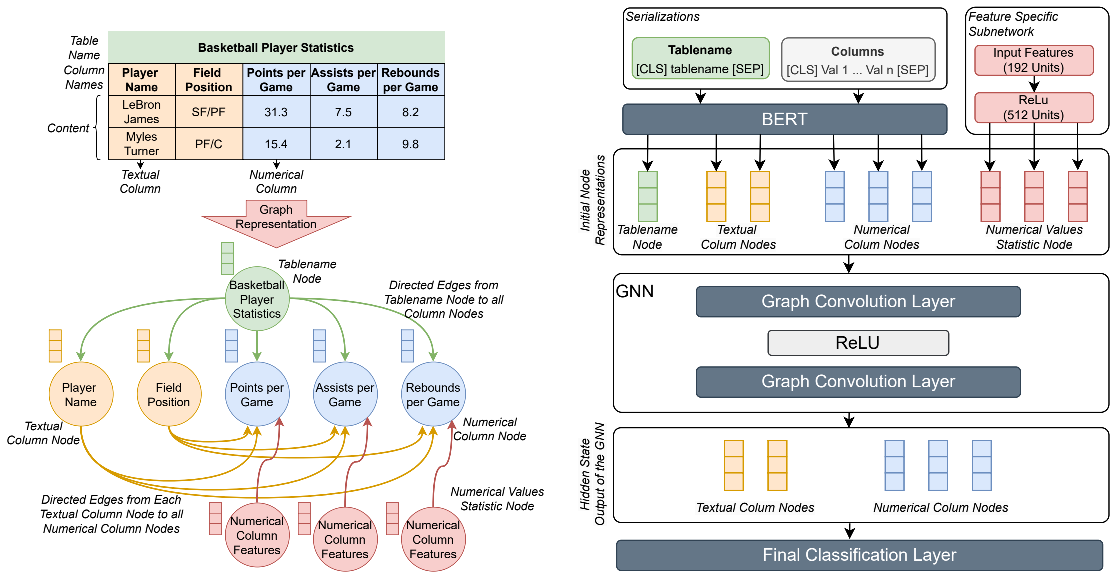
Semantic Type Detection, Data Discovery in Data Lakes, Tabular data, Work in Progress

1. Introduction

Dataset discovery of numerical data is important in enterprise data lakes. Enterprise data lakes serve as invaluable repositories of diverse data types, enabling organizations to store and manage vast amounts of information [1]. In these data lakes, numerical data plays a dominant role, making up a much larger proportion compared to non-numerical data [2] and providing insights into various business domains, including finance, manufacturing, healthcare, and marketing. Such data often contain critical information such as sales figures, production metrics, customer demographics, and financial records. Therefore, it is essential to automatically detect the correct semantic type of table columns with numerical data enabling data scientists to find required data for downstream analysis and thus address the dataset discovery problem in data lakes [3, 4, 5].

Existing approaches are mainly designed for non-numerical data. In order to provide the task of semantic type detection, many solutions using deep learning techniques have been proposed in the past [6, 7, 8, 9, 10]. Unfortunately, all these existing approaches have primarily focused on detecting the semantic type of non-numerical data table columns, leaving a critical

LWDA'23: Lernen, Wissen, Daten, Analysen. October 09–11, 2023, Marburg, Germany
sven.langenecker@mosbach.dhbw.de (S. Langenecker); christoph.sturm@mosbach.dhbw.de (C. Sturm); christian.schalles@mosbach.dhbw.de (C. Schalles); carsten.binnigcs.tu-darmstadt.de (C. Binnig)
0009-0002-2809-5331 (S. Langenecker); 0009-0008-5706-3041 (C. Sturm); 0009-0005-7036-3012 (C. Schalles); 0000-0002-2744-7836 (C. Binnig)



(a) Graph Representation (b) Model architecture

Figure 1: (a) Shows the conversion of a table into a graph representation. The key aspect of the graph is that it provides all the necessary contextual information through its structure (nodes and directed edges), resulting in improved predictions of the semantic types of numerical columns. (b) Shows the complete model architecture of the neuronal network.

need for innovative approaches that effectively handle the detection of semantic types for numerical table columns [2].

Towards a new learned semantic type detection model for numerical data. In this paper, we introduce our new vision of a semantic type detection model called *Pythagoras*, which can not only predict the semantic type of non-numerical table columns with high accuracy but also of numerical table columns. To achieve this, the main idea of the new model architecture is to use graph neural networks (GNNs) together with a novel graph representation of tables and their columns. This graph representation includes directed edges to provide necessary context information (e.g. neighboring non-numerical columns) for predicting the semantic type of numerical columns using GNNs message passing mechanism. The graph representation and the new model architecture are the main contributions of this paper. Moreover, as a second contribution, we show initial highly promising results comparing Pythagoras against five existing state-of-the-art models on the *SportsTables* corpus [2]. The results of this experiment demonstrate that we outperform all existing semantic type detection models on numerical data.

2. Overview of *Pythagoras*

In the following, we will introduce our new semantic type detection model *Pythagoras* and discuss the main design aspects that will lead to better predictions on numerical table columns.

Figure 1a demonstrates how we convert a table and its columns into a graph representation using an example. We can see that the table is transformed into a graph containing four different node types. The green node represents the table name. The orange and blue nodes are responsible for the representation of the textual and numerical columns. In addition, there is another node in the graph for each numerical column, which contains 192 selected statistical features (see Table 2 in the Appendix) of the numerical column values (red node). Because detecting semantic types of numerical columns is generally harder than for textual columns, using only the numerical column values to specify the type is too limited [10]. Hence, we designed the graph structure with directed edges to inject necessary context information into the numerical column representation and thus enrich it for better predictions. Looking at a *Numerical Column Node* we can see that three directed edge types go towards the node. With that, the node will embed information from its connected neighbors into its own representation during a GNN layer iteration based on the message passing paradigm [11]. Specifically, the green edge provides information about the table name, the yellow edges convey information from each textual column within the table and the red edge facilitates the transmission of the additional statistical features. As a consequence, the GNN layers transforms the representation of the *Numerical Column Nodes*, leading to enhanced information content for accurate semantic type prediction. For instance, when faced with a numerical column with values in the range of 60-100, where the semantic type could be ambiguous (e.g., *basketball.player.weight* or *humidity*), the embedding of information from a neighboring textual column containing basketball player names allows for a more precise identification of the semantic type as *basketball.player.weight*.

In Figure 1b we can see the whole model architecture of *Pythagoras* which encodes the table structure as a graph. The upper part of the architecture illustrates how we generate the initial embedding vector representations of the nodes in the graph. For encoding table names as well as cell values (textual and numerical), we use the pre-trained transformer-based language model BERT[1] [12]. In addition, to embed the features of the *Numerical Values Statistic Nodes*, we train a feature specific subnetwork similar to the approach in [6]. This subnetwork embeds the extracted features from the numerical column to an output of fixed length using one hidden layer with a rectifier linear unit (ReLu) activation function. For each numerical column, we extract 192 features[2], including for example mean and median of all values, as well as statistical metrics regarding the occurrence of individual digits. The initial node representations, along with the discussed graph structure, serve as the input for the GNN model. As GNN, we use a graph convolutional neural network [11]. After traversing the GNN layers, we extract the hidden states of *Textual* as well as *Numerical Column Nodes* from the last convolutional layer. These hidden states are then passed as inputs to a final classification layer to perform the semantic type classification task.

3. Initial Experimental Results

In this section, we present initial experimental results applying *Pythagoras* on the *SportsTables* corpus. We compare the performance of our approach against five state-of-the-art models.

Baseline models. As state-of-the-art models we consider *Sherlock* [6], *Sato* [7], *Dosolo* [8], *Doduo* [8] and *GPT-3* [13, 14]. While *Sherlock* and *Dosolo* are models that utilize only the values of a single column for the prediction, *Sato* and *Doduo* are successors of them that adopt a

[1]Note that *Pythagoras* is independent of how to generate these initial embeddings, and there may exist alternative language models or embedding methods that could potentially yield even better results in this context.

[2]The complete feature list can be found in Table 2 in the Appendix

Table 1
Experimental results of our new semantic type detection model Pythagoras in comparison to several state-of-the-art models on *SportsTables* corpus.

Model	support weighted F1-Score			macro F1-Score		
	numeric	non-numeric	overall	numeric	non-numeric	overall
Sherlock[6]	0.609	0.856	0.641	0.555	0.767	0.57
Sato[7]	0.703	0.961	0.736	0.650	0.903	0.668
Dosolo[8]	0.313	0.822	0.379	0.245	0.782	0.285
Doduo[8]	0.623	0.98	0.67	0.567	0.933	0.594
GPT-3 (fine-tuned)[13]	0.446	0.872	0.501	0.404	0.760	0.423
Pythagoras	**0.829**	**0.996**	**0.851**	**0.790**	**0.97**	**0.803**

context-based approach similar to our model. Despite their similarities to our model, Sato and Doduo do not specifically address the prediction of semantic types for numerical-based columns and do not offer a well-defined approach for injecting contextual information into the prediction process. Furthermore, to have another benchmark, we developed in our experiments a fine-tuned *GPT-3* model for the task of semantic type detection. We chose fine-tuning over prompt designs for higher model quality and the capacity to train on a larger number of examples[3].

Experiment setup. In our experiments, we use the *SportsTables* dataset, due to its high proportion of numerical-based columns. To perform the experiments, we split the corpus into 60/20/20 for train, validation, and test set. After training the models, the checkpoint with the best accuracy on the validation set is used for evaluation on the test set. We report end results as an average of five runs with different random seeds using the evaluation metrics support-weighted and macro F1-Score as in previous studies [6, 7, 8, 15]. To implement *Pythagoras* we used Python together with PyTorch [16], DGL [17] and the Transformers library [18].

Results of study. The experimental results are shown in Table 1. For each model, we list the F1-Scores overall data types to show the total performance, but also the separate average F1-Scores for only numerical and non-numerical data types, respectively. The results show that our model *Pythagoras* outperforms all existing models in detecting the semantic type of numerical columns. To the best performing existing model *Sato* with F1-Scores of 0.703/0.650 (support-weighted/macro F1-Score) we can achieve an improvement of +0.126/+0.140. Furthermore, a notable observation across all existing models is the substantial performance discrepancy between the prediction on non-numerical and numerical columns. On non-numerical data, the accuracy of the models is generally high, whereas their performance on numerical data tends to be poorer. In contrast, our model exhibits a different behavior, as we are able to achieve a more balanced accuracy for both numerical and non-numerical data. In summary, the results demonstrate that our model, in conjunction with the graph representation of tables, leads to a significantly improved performance.

The road ahead. To establish the generalizability of our approach, additional experiments on diverse datasets are crucial. These experiments will validate the effectiveness of our approach across different data domains and assess its robustness. Furthermore, conducting an ablation study is essential to examine the impact of various design choices in our architecture.

[3]For fine-tuning the *GPT-3* model, we use OpenAIs API described in https://platform.openai.com/docs/guides/fine-tuning (visited on 09/04/2023)

Acknowledgements. This research and development project was funded by DHBW Mosbach. We also want to thank the NHR Program, the BMBF project KompAKI (grant number 02L19C150), the HMWK cluster project 3AI, hessian.AI, and DFKI Darmstadt for their support.

References

[1] J. Dixon, Data Lakes Revisited, https://jamesdixon.wordpress.com/2014/09/25/data-lakes-revisited/, 2014.

[2] S. Langenecker, C. Sturm, C. Schalles, C. Binnig, Sportstables: A new corpus for semantic type detection, in: B. König-Ries, S. Scherzinger, W. Lehner, G. Vossen (Eds.), Datenbanksysteme für Business, Technologie und Web (BTW 2023), 20. Fachtagung des GI-Fachbereichs „Datenbanken und Informationssysteme" (DBIS), 06.-10, März 2023, Dresden, Germany, Proceedings, volume P-331 of *LNI*, Gesellschaft für Informatik e.V., 2023, pp. 995–1008. URL: https://doi.org/10.18420/BTW2023-68. doi:10.18420/BTW2023-68.

[3] G. Fan, J. Wang, Y. Li, R. J. Miller, Table discovery in data lakes: State-of-the-art and future directions, in: Companion of the 2023 International Conference on Management of Data, SIGMOD '23, Association for Computing Machinery, New York, NY, USA, 2023, p. 69–75. URL: https://doi.org/10.1145/3555041.3589409. doi:10.1145/3555041.3589409.

[4] F. Nargesian, E. Zhu, R. J. Miller, K. Q. Pu, P. C. Arocena, Data lake management: Challenges and opportunities, Proc. VLDB Endow. 12 (2019) 1986–1989. URL: https://doi.org/10.14778/3352063.3352116. doi:10.14778/3352063.3352116.

[5] A. Khatiwada, G. Fan, R. Shraga, Z. Chen, W. Gatterbauer, R. J. Miller, M. Riedewald, Santos: Relationship-based semantic table union search, Proc. ACM Manag. Data 1 (2023). URL: https://doi.org/10.1145/3588689. doi:10.1145/3588689.

[6] M. Hulsebos, K. Hu, M. Bakker, E. Zgraggen, A. Satyanarayan, T. Kraska, c. Demiralp, C. Hidalgo, Sherlock: A deep learning approach to semantic data type detection, in: SIGKDD, KDD '19, ACM, New York, NY, USA, 2019, p. 1500–1508. URL: https://doi.org/10.1145/3292500.3330993. doi:10.1145/3292500.3330993.

[7] D. Zhang, M. Hulsebos, Y. Suhara, c. Demiralp, J. Li, W.-C. Tan, Sato: Contextual semantic type detection in tables, in: VLDB, volume 13, VLDB Endowment, 2020, p. 1835–1848. URL: https://doi.org/10.14778/3407790.3407793. doi:10.14778/3407790.3407793.

[8] Y. Suhara, J. Li, Y. Li, D. Zhang, c. Demiralp, C. Chen, W.-C. Tan, Annotating columns with pre-trained language models, in: SIGMOD, ACM, New York, NY, USA, 2022, pp. 1493–1503.

[9] X. Deng, H. Sun, A. Lees, Y. Wu, C. Yu, TURL: Table Understanding through Representation Learning, in: VLDB, volume 14, VLDB Endowment, 2021, pp. 307–319. URL: https://github.com/sunlab-osu/TURL. doi:10.14778/3430915.3430921. arXiv:2006.14806v2.

[10] S. Langenecker, C. Sturm, C. Schalles, C. Binnig, Steered training data generation for learned semantic type detection, Proc. ACM Manag. Data 1 (2023) 201:1–201:25. URL: https://doi.org/10.1145/3589786. doi:10.1145/3589786.

[11] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France,

April 24-26, 2017, Conference Track Proceedings, OpenReview.net, 2017. URL: https://openreview.net/forum?id=SJU4ayYgl.

[12] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Association for Computational Linguistics, Minneapolis, Minnesota, 2019, pp. 4171–4186. URL: https://aclanthology.org/N19-1423. doi:10.18653/v1/N19-1423.

[13] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei, Language models are few-shot learners, in: Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS'20, Curran Associates Inc., Red Hook, NY, USA, 2020.

[14] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. E. Miller, M. Simens, A. Askell, P. Welinder, P. F. Christiano, J. Leike, R. J. Lowe, Training language models to follow instructions with human feedback, ArXiv abs/2203.02155 (2022).

[15] X. Deng, H. Sun, A. Lees, Y. Wu, C. Yu, Turl: Table understanding through representation learning, Proc. VLDB Endow. 14 (2020) 307–319. URL: https://doi.org/10.14778/3430915.3430921. doi:10.14778/3430915.3430921.

[16] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, Pytorch: An imperative style, high-performance deep learning library., in: H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. B. Fox, R. Garnett (Eds.), NeurIPS, 2019, pp. 8024–8035. URL: http://dblp.uni-trier.de/db/conf/nips/nips2019.html#PaszkeGMLBCKLGA19.

[17] M. Wang, D. Zheng, Z. Ye, Q. Gan, M. Li, X. Song, J. Zhou, C. Ma, L. Yu, Y. Gai, T. Xiao, T. He, G. Karypis, J. Li, Z. Zhang, Deep graph library: A graph-centric, highly-performant package for graph neural networks, arXiv preprint arXiv:1909.01315 (2019).

[18] Z. Wang, H. Dong, R. Jia, J. Li, Z. Fu, S. Han, D. Zhang, Tuta: Tree-based transformers for generally structured table pre-training, Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining (2020).

A. List of Features

Table 2
Extracted features of a numerical column to build the representation of the *Numerical Values Statistic Node*. Except for the last feature listed, all features are also included in the model of [6].

Feature	#
Character-level distribution (any, all, mean, variance, min, max, median, sum, kurtosis, skewness of the digits 0-9 and the symbols comma, point, plus, minus, blank)	150
Number of values	1
Column entropy	1
Fraction of values with unique content	1
Fraction of values with numerical characters	1
Fraction of values with alphabetical characters	1
Mean and std. of the number of numerical characters in cell-values	2
Mean and std. of the number of alphabetical characters in cell-values	2
Mean and std. of the number special characters in cell-values	2
Mean and std. of the number of words in values	2
Percentage, count, only/has-Boolean of the None values	4
Stats, sum, min, max, median, mode, kurtosis, skewness, any/all-Boolean of length of values	10
Column values statistics (min, max, mean, median, 8*quantile, mode, skewness, kurtosis of all column values)	15
Total	**192**