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

Sven Langenecker\textsuperscript{1,2}, Christoph Sturm\textsuperscript{1}, Christian Schalles\textsuperscript{1} and Carsten Binnig\textsuperscript{2,3}

\textsuperscript{1}DHBW Mosbach, Lohrtalweg 10, 74821 Mosbach, Germany
\textsuperscript{2}TU Darmstadt, Karolinenplatz 5, 64289 Darmstadt, Germany
\textsuperscript{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-numeric 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
Basketball Player Statistics

<table>
<thead>
<tr>
<th>Player Name</th>
<th>Field Position</th>
<th>Points per Game</th>
<th>Assists per Game</th>
<th>Rebounds per Game</th>
</tr>
</thead>
<tbody>
<tr>
<td>LeBron James</td>
<td>SF/PF</td>
<td>31.3</td>
<td>7.5</td>
<td>8.2</td>
</tr>
<tr>
<td>Myles Turner</td>
<td>PF/C</td>
<td>15.4</td>
<td>2.1</td>
<td>9.8</td>
</tr>
</tbody>
</table>

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 BERT1[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 features2, 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

---

1Note 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.

2The 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.

<table>
<thead>
<tr>
<th>Model</th>
<th>support weighted F1-Score</th>
<th>macro F1-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>numeric</td>
<td>non-numeric</td>
</tr>
<tr>
<td>Sherlock[6]</td>
<td>0.609</td>
<td>0.856</td>
</tr>
<tr>
<td>Sato[7]</td>
<td>0.703</td>
<td>0.961</td>
</tr>
<tr>
<td>Dosolo[8]</td>
<td>0.313</td>
<td>0.822</td>
</tr>
<tr>
<td>Doduo[8]</td>
<td>0.623</td>
<td>0.98</td>
</tr>
<tr>
<td>GPT-3 (fine-tuned)[13]</td>
<td>0.446</td>
<td>0.872</td>
</tr>
<tr>
<td><strong>Pythagoras</strong></td>
<td><strong>0.829</strong></td>
<td><strong>0.996</strong></td>
</tr>
</tbody>
</table>

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.

**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.

---

3For 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 HMKW cluster project 3AI, hessian.AI, and DFKI Darmstadt for their support.

References


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].

<table>
<thead>
<tr>
<th>Feature</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>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)</td>
<td>150</td>
</tr>
<tr>
<td>Number of values</td>
<td>1</td>
</tr>
<tr>
<td>Column entropy</td>
<td>1</td>
</tr>
<tr>
<td>Fraction of values with unique content</td>
<td>1</td>
</tr>
<tr>
<td>Fraction of values with numerical characters</td>
<td>1</td>
</tr>
<tr>
<td>Fraction of values with alphabetical characters</td>
<td>1</td>
</tr>
<tr>
<td>Mean and std. of the number of numerical characters in cell-values</td>
<td>2</td>
</tr>
<tr>
<td>Mean and std. of the number of alphabetical characters in cell-values</td>
<td>2</td>
</tr>
<tr>
<td>Mean and std. of the number special characters in cell-values</td>
<td>2</td>
</tr>
<tr>
<td>Mean and std. of the number of words in values</td>
<td>2</td>
</tr>
<tr>
<td>Percentage, count, only/has-Boolean of the None values</td>
<td>4</td>
</tr>
<tr>
<td>Stats, sum, min, max, median, mode, kurtosis, skewness, any/all-Boolean of length of values</td>
<td>10</td>
</tr>
<tr>
<td>Column values statistics (min, max, mean, median, 8’ quantile, mode, skewness, kurtosis of all column values)</td>
<td>15</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>192</strong></td>
</tr>
</tbody>
</table>