ConTabulizer – cells contextualization for tabular data

Ben Shapira **Roi Cohen**

Tel-Aviv University

{benshapira, roi1}@mail.tau.ac.il

https://github.com/bensha6757/ConTabuarization_-_cells_contextualization_for_tabular_data

Abstract

Tabular data underpins numerous high-impact applications of machine learning from fraud detection to genomics and healthcare. Classical approaches to solving tabular problems, such as gradient boosting and random forests, are widely used by practitioners. However, recent deep learning methods have achieved a degree of performance competitive with popular techniques. We devise a hybrid deep learning approach to solving tabular data problems which contain a relatively large amount of text.

Our method utilizes NLP common approaches for enriching a cell inside a table with relevant context that might contain useful information for solving a broad range of downstream tasks. We introduce a new architecture that contextualizes representations via template generation and attention blocks designed to imitate the process a human goes through when facing a table.

1 Introduction

Recent years have witnessed the burgeoning of deep neural networks for understanding tabular data. Although the incredible success of those models in a wide range of AI domains such as Vision and Language, such models have been facing a lot of challenges when it comes to tabular data. That kind of data is composed of a two-dimensional array of entries, each entry consists of some information that may be represented through a broad range of data types – free-form text, either discrete or continuous numbers, categorical classes, dates, etc. Consequently, some challenges arise. Firstly, it is unnecessarily clear how to introduce a table as an input for such models. Second, those tables might consist of shallow information, namely a category or a number, without any context around it. Third, the use of some common mechanisms which are widely used in the AI world – NLP, Vision, demand context for better embedding.
In this work, we're addressing those challenges, proposing a novel deep learning architecture for encoding tables, which is devoted to facing those challenges in a different way than those which has been already tried.
First, we propose the idea of an "entry template generator". This component, given the entry content and the right context, which is the row ID, column label, and table name, generates a contextual sentence for each entry, which is devoted to appending some additional information about the cell. For example, given a table of NBA game statistics, instead of encoding the cell which contains the number of points the player "Joel Embiid" scored, we would like to generate a sentence as: "Joel Embiid scored 34 points on the Sixers vs Celtics game", such that if we use a pretrained natural language encoder, it would be able to encode effectively that cell content and understand it.
second, as noted, we take advantage of the advancement that had been occurring throughout recent years within the world of language models and use a language encoder for getting an informative representation for each cell, by encoding its corresponding template.
Third, we propose a new architecture that exploit the attention mechanism, which took over the NLP world as well as the vision world, to get contextualized-over-other-cells representations to each cell.

Those three components build together an architecture that stands as an encoder for tabular data.
Additionally, we pretrained our architecture over a large corpus of tables, in a masking task, for the purpose of creating a model which is already pretrained and may easily be useful for various downstream tasks over tables.

2 Prior work

Classical models widely adopted approaches for supervised and semi-supervised learning on tabular datasets eschew neural models due to their black-box nature and high compute requirements. When one has reasonable expectations of linear relationships, a variety of modeling approaches are available. In more complex settings, non-parametric tree-based models are used. Commonly used tools such as XGBoost, CatBoost, and LightGBM provide several benefits such as interpretability, the ability to handle a variety of feature types including null values, as well as performance in both high and low data regimes.

Deep Tabular Models - While classical methods are still the industry favorite, some recent work brings deep learning to the tabular domain. For example, TabNet uses neural networks to mimic decision trees by placing importance on only a few features at each layer. The attention layers in that model do not use the regular dot-product self-attention common in transformer-based models, rather there is a type of sparse layer that allows only certain features to pass through. Yoon et al. propose VIME, which employs MLPs in a technique for pre-training based on denoising. TABERT, a more elaborate neural approach inspired by the large language transformer model BERT, is trained on semi-structured test data to perform language-specific tasks. Several other studies utilize tabular data, but their problem settings are outside of our scope.
Transformer models for more general tabular data include TabTransformer, which uses a transformer encoder to learn contextual embeddings only on categorical features. The continuous features are concatenated to the embedded features and fed to an MLP. The main issue with this model is that continuous data do not go through the self-attention block. That means any information about correlations between categorical and continuous features is lost. In our model, we address this issue by neutralizing the difference between continuous features and categorical features by generating a template sentence for them, then projecting the generated sentence to a higher dimensional embedding space and passing them both through the transformer blocks. In addition, we propose a new type of attention to explicitly allow data points to attend to each other to get better representations. Axial Attention in Multidimensional Transformers - Ho et al. are the first to propose row and column attention in the context of localized attention in 2-dimensional inputs (like images) in their Axial Transformer. This is where for a given pixel, the attention is computed only on the pixels that are on the same row and column, rather than using all the pixels in the image. The MSA Transformer extends this work to protein sequences and applies both column and row attention across similar rows (tied row attention). TABBIE is an adaptation of axial attention that applies self-attention to rows and columns separately, then averages the representations and passes them as input to the next layer.
In all these works, different features from the same data point communicate with each other and with the same feature from a whole batch of data. Our approach, intersample attention, is hierarchical in its nature; first, features of a given data point interact with each other, then data points interact with each other using entire rows/samples. In a similar vein, Graph Attention Networks (GAT) seek to compute attention over neighbors on a graph, thereby learning which neighbor's information is most relevant to a given node's prediction. One way to view our intersample attention is as a GAT on a complete graph where all tabular rows are connected to all other rows. Yang et al. explore hierarchical attention for the task of document classification where attention is computed between words in a given sentence and then between the sentences, but they did not attempt to compute the attention between entire documents themselves.
Self-Supervised Learning Self-supervision via a 'pretext task' on unlabeled data coupled with finetuning on labeled data is widely used for

improving model performance in language and computer vision. Some of the tasks previously used for self-supervision on tabular data include masking, denoising, and replaced token detection. Masking (or Masked Language Modeling(MLM)) is when individual features are masked, and the model's objective is to impute their value. Denoising injects various types of noise into the data, and the objective here is to recover the original values. Replaced Token Detection (RTD) inserts random values into a given feature vector and seeks to detect the location of these replacements. Contrastive pre-training, where the distance between two views of the same point is minimized while maximizing the distance between two different points is another pretext task that applies to tabular data. In this paper, we adopt contrastive learning for tabular data. We couple this strategy with denoising to perform pre-training on a plethora of datasets with varied volumes of labeled data, and we show that our method outperforms traditional boosting methods.

2 Data

We used a few datasets to pre-train the model.

2.1 Pre-training Data

Most of the datasets were taken from Kaggle.com, link to the dataset can be found at the end of each dataset description.

Churn Modelling (size: 10,000) This data set contains details of a bank's customers, and the target variable is a binary variable reflecting the fact whether the customer left the bank (closed his account), or he continues to be a customer. https://www.kaggle.com/datasets/shrutimechlearn/churn-modelling

Netflix Movies and TV Shows (size: 8,807) Netflix is one of the most popular media and video streaming platforms. They have over 8000 movies or tv shows available on their platform, as of mid-2021, they have over 200M Subscribers globally. This tabular dataset consists of listings of all the movies and tv shows available on Netflix, along with details such as - cast, directors, ratings, release year, duration, etc. https://www.kaggle.com/datasets/shivamb/netflix-shows

California Housing Prices (size: 20,640) The data pertains to the houses found in a given California district and some summary stats about them based on the 1990 census data. https://www.kaggle.com/datasets/camnugent/california-housing-prices

Petfinder Adoption Prediction (size: 14,993) PetFinder.my has been Malaysia's leading animal welfare platform since 2008, with a database of more than 150,000 animals. PetFinder dataset contains information on dogs and cats ready to be adopted and includes information relevant to the adoption. https://www.kaggle.com/competitions/petfinder-adoption-prediction/data

Wine Reviews (size: 84,123) Classify the variety of wines based on tasting descriptions from sommeliers and numeric features like price and categorical features like country of origin. https://www.kaggle.com/zynicide/wine-reviews

Store Chains Selling Kaggle Merchandise (size: 26,297) There are two (fictitious) independent store chains selling Kaggle merchandise that want to become the official outlet for all things Kaggle. The dataset stores the details of sales of Kaggle Products by Stores in multiple countries. https://www.kaggle.com/code/aestheteaman01/tabular-playground-2022-the-storytelling/notebook?scriptVersionId=84320932

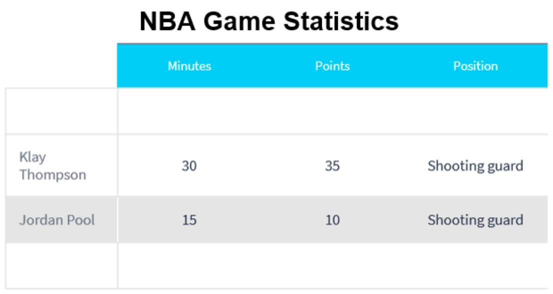
Adult Data Set (size: 32,561) An individual's annual income results from various factors. Intuitively, it is influenced by the individual's education level, age, gender, occupation, etc. https://archive.ics.uci.edu/ml/datasets/adult

IMDB Movie Ranking (size: 1,000) Predict whether a movie falls within the Drama category based on text features like its name, description, actors/directors, and numerical features like its release year, runtime, etc. https://www.kaggle.com/PromptCloudHQ/imdb-data

Real / Fake Job Posting Prediction (size: 14,304) Predict whether online job postings are real, or fake based on their text and additional tabular features like the amount of salary offered and degree of education required. https://www.kaggle.com/shivamb/real-or-fake-fake-jobposting-prediction

4 ConTabulizer Model

Our model was built to resemble the process humans go through when processing the information of a table they see for the first time. We will go with a leading example (Figure 2):



NBA Game Statistics

	Minutes	Points	Position
Klay Thompson	30	35	Shooting guard
Jordan Pool	15	10	Shooting guard

Figure 1: NBA Game Statistics table contains information on two players – Klay Thompson and Jordan Pool.

First, when a human examines a cell in the table, he tries to gain initial context that can help him understand the cell's content. What is the subject of this table? What column is my cell under? What is the line "headline"?

Considering the above, to supply our model with this context, we decided to train a model that will be able to generate a natural language sentence, given the column name, row name, cell content, and table name.

In our leading example - given "Jordan Pool", "Points", "10", and "NBA Game Statistics" - we would like the model to generate the following template: "Jordan Pool has scored 10 points".

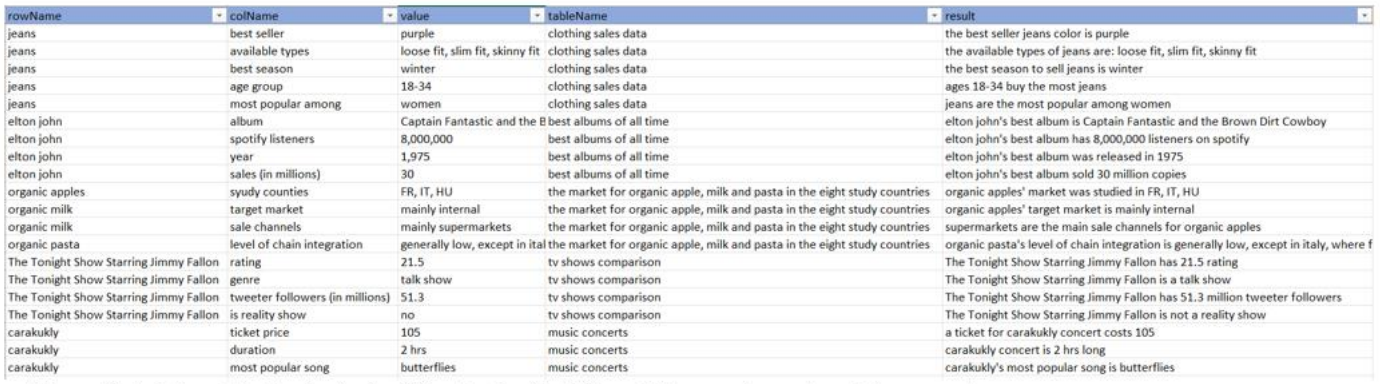
As pre-trained language models have a good understanding of natural language, this task is familiar to them. We manually tagged a small number of examples (~200; see Figure 2.), as this task requires good natural language understanding and the templates the model should generate are mostly straightforward and can be inferred from the inputs relatively easily. Given its great language understanding learned from pretraining on millions of examples, we chose to fine-tune a T5 model for this task of template generation.

After enriching the cell content with context coming from the relevant headlines, we want to create a contextual embedding to be transferred to the next stage of our cell context enrichment journey.

Now that the cell value becomes an NL sentence holding context from the table, we can utilize BERT power and natural language understanding ability - which can provide rich semantics for language representation – to encode the sentence and serve as a contextual embedding layer.

Now our human continues to examine the table and absorb information from it. After looking at some headlines to help him gain some context, he starts examining the cell's neighbors. Which neighbor cells have a greater influence on the current cell than others? What can we learn about the current cell from its neighbors?

We decided to adopt one of the most successful NLP methods – the self-attention mechanism.



rowName	colName	value	tableName	result
jeans	best seller	purple	clothing sales data	the best seller jeans color is purple
jeans	available types	loose fit, slim fit, skinny fit	clothing sales data	the available types of jeans are: loose fit, slim fit, skinny fit
jeans	best season	winter	clothing sales data	the best season to sell jeans is winter
jeans	age group	18-34	clothing sales data	ages 18-34 buy the most jeans
jeans	most popular among	women	clothing sales data	jeans are the most popular among women
elton john	album	Captain Fantastic and the B	best albums of all time	elton john's best album is Captain Fantastic and the Brown Dirt Cowboy
elton john	spotify listeners	8,000,000	best albums of all time	elton john's best album has 8,000,000 listeners on spotify
elton john	year	1,975	best albums of all time	elton john's best album was released in 1975
elton john	sales (in millions)	30	best albums of all time	elton john's best album sold 30 million copies
organic apples	syudy counties	FR, IT, HU	the market for organic apple, milk and pasta in the eight study countries	organic apples' market was studied in FR, IT, HU
organic milk	target market	mainly internal	the market for organic apple, milk and pasta in the eight study countries	organic apples' target market is mainly internal
organic milk	sale channels	mainly supermarkets	the market for organic apple, milk and pasta in the eight study countries	supermarkets are the main sale channels for organic apples
organic pasta	level of chain integration	generally low, except in ital	the market for organic apple, milk and pasta in the eight study countries	organic pasta's level of chain integration is generally low, except in italy, where f
The Tonight Show Starring Jimmy Fallon	rating	21.5	tv shows comparison	The Tonight Show Starring Jimmy Fallon has 21.5 rating
The Tonight Show Starring Jimmy Fallon	genre	talk show	tv shows comparison	The Tonight Show Starring Jimmy Fallon is a talk show
The Tonight Show Starring Jimmy Fallon	tweeter followers (in millions)	51.3	tv shows comparison	The Tonight Show Starring Jimmy Fallon has 51.3 million tweeter followers
The Tonight Show Starring Jimmy Fallon	is reality show	no	tv shows comparison	The Tonight Show Starring Jimmy Fallon is not a reality show
carakukly	ticket price	105	music concerts	a ticket for carakukly concert costs 105
carakukly	duration	2 hrs	music concerts	carakukly concert is 2 hrs long
carakukly	most popular song	butterflies	music concerts	carakukly's most popular song is butterflies

Figure 2: training data for the task of fine-tuning the T5 model to create our template generator.

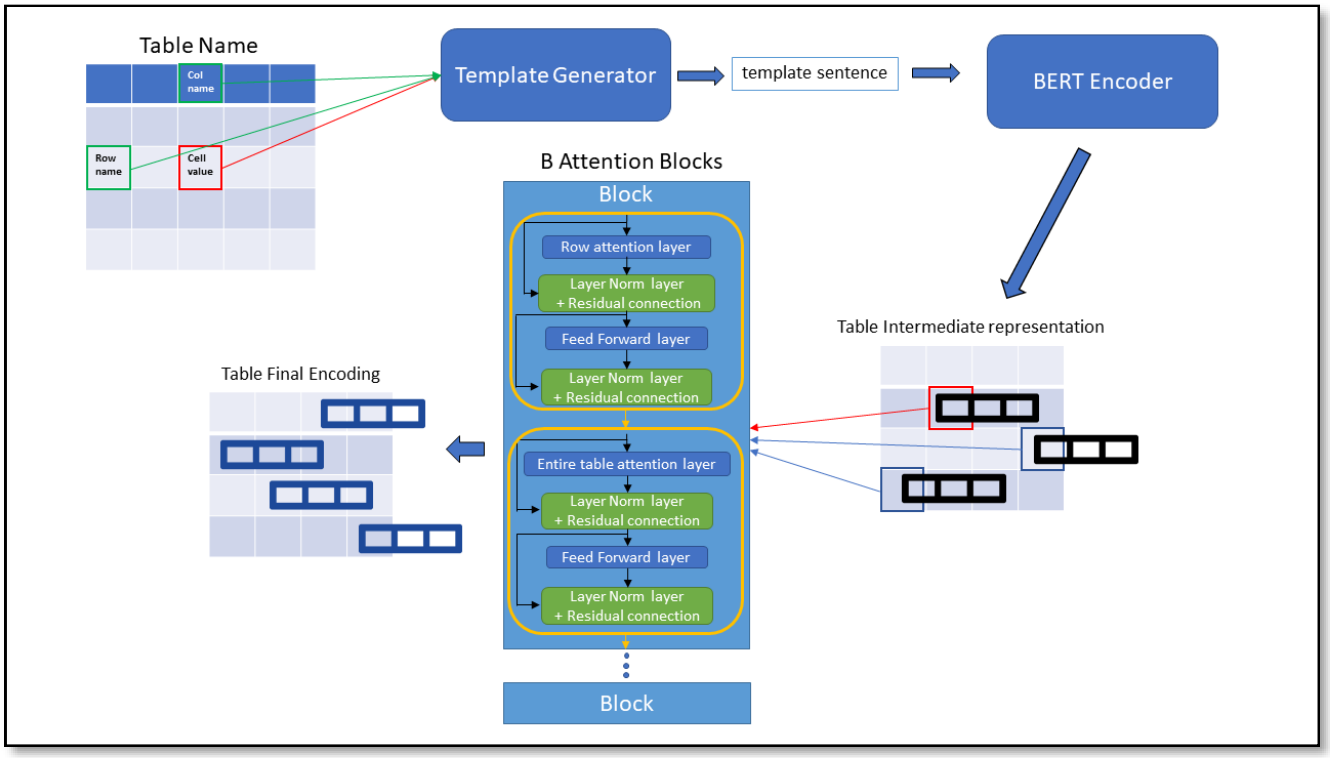


Table Name
Col name
Row name
Cell value
Template Generator
template sentence
BERT Encoder
B Attention Blocks
Block
Row attention layer
Layer Norm layer + Residual connection
Feed Forward layer
Layer Norm layer + Residual connection
Entire table attention layer
Layer Norm layer + Residual connection
Feed Forward layer
Layer Norm layer + Residual connection
Block
Table Intermediate representation
Table Final Encoding

Figure 3: ConTabulizer model architecture

We applied both row-level attention and entire table attention between the cells of the table.

Going back to our example, Since Jordan Pool's minutes influence his points, we apply row attention. Since Klay Thompson is playing in the same position as Jordan Pool, Thompson's minutes influence Pool's points and this can be captured within the entire table's attention.

At first, the cell attends to its row neighbors, to understand better its connection to the whole record (the strongest connection) and then the cell attends to the entire table to learn weaker, but important, connections.

4.1 Model Architecture

ConTabulizer is inspired by the State-of-the-Art NLP approaches and tries to convert the table to natural language and make the most of the SOTA language models and proven language processing techniques. We will now describe the various component in our architecture

(a graphical overview of ConTabulizer is presented in Figure 4).

Template Generator: fine-tuned T5 model (220 million parameters), that was trained independently on 166 manually tagged examples.

The input-output structure is of the form - <u>input</u>: row name # col name # cell value # table name -> <u>output</u>: template sentence generated.

The generated sentence is then transferred to the BERT encoder (110 million parameters) to create embeddings for the table cells.

Attention Blocks: the heart of the model architecture. Composed of B identical blocks, each block is divided into a row attention sub-block and an entire table attention sub-block, each of these sub-blocks consists of an attention layer, followed by a fully connected feed-forward layer with a GELU non-linearity. Each layer has a residual skip connection and layer normalization.

The attention is a "self-attention" that attends to individual features within each data sample, in this layer – the cell attends to its row neighbors as a first step and then attends to the entire table cells.

4.2 Pre-Training

For pre-training, we used a self-supervised pre-training technique, known from the NLP realm called the Masking technique. We invented a made-up task of randomly choosing cells in the table and putting a mask on them (we randomly put masks on a number of cells equals to the number of rows in the table, expecting roughly 1 mask per row). The task is then to predict the value of the masked cells.

To complete the task, we need a decoder to use our table encoder model (the ConTabulizer) and decode the masked cells and predict the cell values. We wanted to utilize the fact that T5 was trained on the masking task as well (except that in T5's pre-training, parts of sentences were masked, however in our case full cells are masked), so we tried a new approach of splitting up T5's decoder from the encoder and using its decoder for our pre-training masking prediction task. (See experiments analysis)

Thus, the pre-training process is as follows:

1. Mask a few cells in the input table
2. Encode the table with the ConTabulizer
3. Decode the masked cells using T5's decoder

5 Experiments

We conducted a few experiments we will now present

5.1 Experimental Setup

We used one Nvidia A-100 GPU for our experiments since our model requires a relatively large number of resources and memory.

Due to memory limitations, we had to consider each table as a collection of table crops. After some tries, we concluded that the maximal crop size we can use given the resources is 5 rows per crop. We also limited the number of columns per crop.

5.2 Tuning the ConTabulizer architecture hyperparameters

- Template generator model
- Embedder's encoder model
- Decoder model
- Number of transformer blocks
- Representation of hidden dimension
- Number of attention heads
- Additional architecture improvements

In addition, some training hyper-parameters such as dropout and size of the table crop (will be explained in the Experimental Setup section).

5.3 Experiment Analysis

The optimal collection of hyperparameters considering the validation loss was:

- Template generator model – we tried various sizes of the T5 model, and as expected the larger the model gets, the better the results are. The difference between the base model to the larger ones wasn't big enough, but when comparing it to T5-Small, we did notice a significant performance gap. Thus, we chose T5-Base as our template generator considering model size issues.
- Embedder's encoder model – the same rationale led us to choose distilBert-Base which is 2 times smaller than BERT-Base.
- Decoder model – we contemplated building a new decoder or not, but after recalling that T5 was trained on the same pre-training task, we decided to split up the T5 decoder from its encoder and use T5-Small as our decoder.
- Number of transformer blocks, Representation hidden dimension, and Number of attention heads – those were determined after a basic hyperparameters search, while the optimal combination was: 4 transformer blocks, 8 attention heads, and a representation dimension of 512. Every enlargement of those parameters wasn't

necessarily harmful, though unnecessarily significant as well. The sizes we chose were the smallest sizes that kept the performance acceptable.

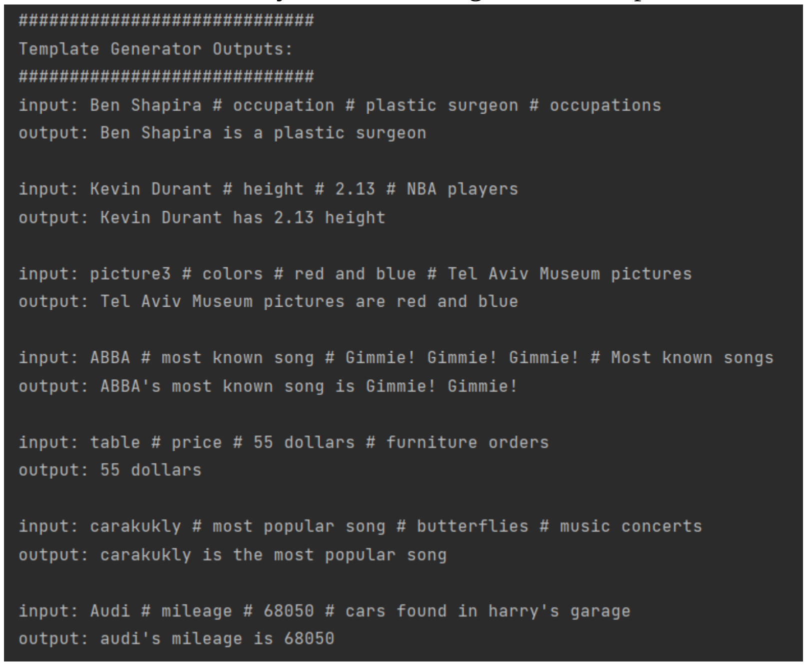
- Additional architecture improvements – we tried adding to the model architecture another stage of enriching the cell content with relevant context that can be found on the internet (Wikipedia) by retrieving the cell content or the column/row name from Wikipedia. We used BM25 to retrieve a few sentences where the term occurs. Essentially, it wasn't a big success and even committed harmful misses that misled the model, so we decided to omit it.

5.4 Template Generator Experiment

We tested the template generator component apart from the model to analyze its performance.

The input to the model is of the structure: row name # column name # cell content # table name.

Let's analyze some of the generated templates:



```
##############################
Template Generator Outputs:
##############################
input: Ben Shapira # occupation # plastic surgeon # occupations
output: Ben Shapira is a plastic surgeon

input: Kevin Durant # height # 2.13 # NBA players
output: Kevin Durant has 2.13 height

input: picture3 # colors # red and blue # Tel Aviv Museum pictures
output: Tel Aviv Museum pictures are red and blue

input: ABBA # most known song # Gimmie! Gimmie! Gimmie! # Most known songs
output: ABBA's most known song is Gimmie! Gimmie!

input: table # price # 55 dollars # furniture orders
output: 55 dollars

input: carakukly # most popular song # butterflies # music concerts
output: carakukly is the most popular song

input: Audi # mileage # 68050 # cars found in harry's garage
output: audi's mileage is 68050
```

As we can see, the model is able to understand the structure of the input very well, as it identifies that the row name is the subject of the sentence, the column name is the relation and cell value is the most important part of the input. It is also able to understand that the table name is there just for context and doesn't have to be incorporated in the sentence.

Sometimes, the model fails to understand the relations between the input parts and generates sentences that don't explain this relation. However, mostly it generates syntactically correct English sentences and most importantly, includes the cell content in the sentence, transferring to the network the most important information about the cell (see examples 3 and 5).

5.4 Pre-trained model evaluation

Since our model is very large (383 million parameters) we had to train it on an expensive hardware resource that was very limited for us (Nvidia A-100 GPU). One epoch of training took us 24 hours (and sometimes stopped in the middle due to an out-of-memory exception or full capacity usage of the hardware resource).

The output of the model, as we designed it, is basically a string containing a list of "[extra_id_#] <extra_id_prediction> ... ". After one epoch of training, the model still seems a bit random but started to understand the pattern of how the output should look syntactically. So, for example, after one epoch the model is outputting "[]extraa,2" while the F1 score remains 0. After another epoch, the model is starting to semantically understand the task, outputting real masked cell values, which are starting to make sense. For example, for a datapoint whose label is: "[extra_id_0] KaggleRama [extra_id_1] Norway [extra_id_2] 2/26/2015"

The model outputs: "[extra_id_0] KaggleRama [extra_id_1] Finland [".

The model's improvement is expressed by the F1 score, which is 0.3 for the second epoch.

6 Future Work

One immediate thing is to pretrain our model for a few more epochs in addition to testing it on a much wider range of downstream tasks. Since our model is very large (383 million parameters) we had to train it on an expensive hardware resource that was very limited for us (Nvidia A-100 GPU). One epoch of training took us 24 hours (and sometimes stopped in the middle due to an out-of-memory exception or full capacity usage of the hardware resource). These hard conditions made it difficult for us to pre-train it and test it on downstream tasks.

As analyzed in previous sections, to improve the performance of our model, the Template Generator can be trained on a larger dataset. Hence, tagging a lot more examples will improve the ConTabulizer performance significantly.

Our model performs best on tables containing textual fields, as the template generator can generate meaningful natural language sentences, which later will be encoded using BERT. Thus, training the template generator to deal with other types of values (such as numeric values) and adjusting the architecture might help encode such tables.

Another thing is to try that model within the world of question answering / semantic parsing, namely, use its encoding for question answering (maybe together with the question encoding etc.).

7 Lessons learned

Handling tabular data is indeed a task that consists of many challenges, especially since each table is completely different (as it is coming out from a different distribution) – namely, generalization might not be trivial.

When building an architecture from scratch and appending number of different components that aren't familiar with each other, requires a lot of pretraining in order to adjust them to communicate with each other well enough.

Another lesson is that large pretraining might take time since it never ends in just one run. The third lesson is that cleaning the data may help a lot sometimes, so it's better to identify "noise" in advance.

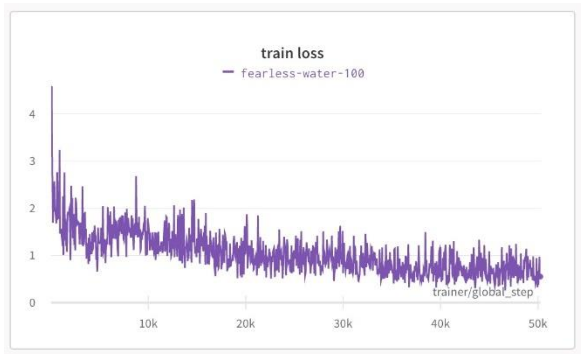


Figure 4 – learning process of our chosen architecture over 3 epochs of pre-training.

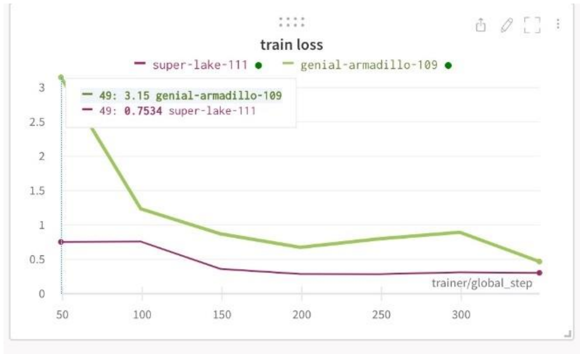


Figure 5 – learning process over one downstream dataset. The green curve is a model initialized with random weights and the purple curve is initialized with our pre-trained weights.