Statements: Universal Information Extraction from Tables with Large Language Models for ESG KPIs

Lokesh Mishra[1], Sohayl Dhibi[1], Yusik Kim[2], Cesar Berrospi Ramis[1], Shubham Gupta[2], Michele Dolfi[1], Peter Staar[1],

[1]IBM Research Zurich, Säumerstrasse 4, Rüschlikon, Switzerland,
[2]IBM Research Paris-Saclay, 2 Rue d'Arsonval, Orsay, France

[mis, ceb, dol, taa]@zurich.ibm.com

[sohayl.dhibi, yusik.kim, shubham.gupta1]@ibm.com

Abstract

Environment, Social, and Governance (ESG) KPIs assess an organization's performance on issues such as climate change, greenhouse gas emissions, water consumption, waste management, human rights, diversity, and policies. ESG reports convey this valuable quantitative information through tables.Unfortunately, extracting this information is difficult due to high variability in the table structure as well as content. We propose `Statements`, a novel domain agnostic data-structure for extracting quantitative facts and related information. We propose translating tables to `statements` as a new supervised deep-learning universal information extraction task. We introduce SemTabNet – a dataset of over 100K annotated tables. Investigating a family of T5-based Statement Extraction Models, our best model generates `statements` which are 82% similar to the ground-truth (compared to baseline of 21%). We demonstrate the advantages of `statements` by applying our model to over 2700 tables from ESG reports. The homogeneous nature of `statements` permits exploratory data analysis on expansive information found in large collections of ESG reports.

1 Introduction

It is invaluable to assess mankind's impact on climate. Climate change related information is often published in so-called "Environment, Social, and Governance (ESG)" reports. Corporations report valuable quantitative data regarding their efforts to improve their impact on environment, working conditions, and company culture in these ESG reports (Bingler et al., 2022; Schimanski et al., 2024).

Like most technical documents, ESG reports present their key information in tables, making table understanding and information extraction (IE) an important problem (Mishra et al., 2024). This problem becomes further complicated due to the large variety and diversity of tabular representations used in these reports. Despite efforts to standardize these reports, this diversity makes the task of extracting information from these documents extremely challenging (see Appendix Fig. 5 for an example table).

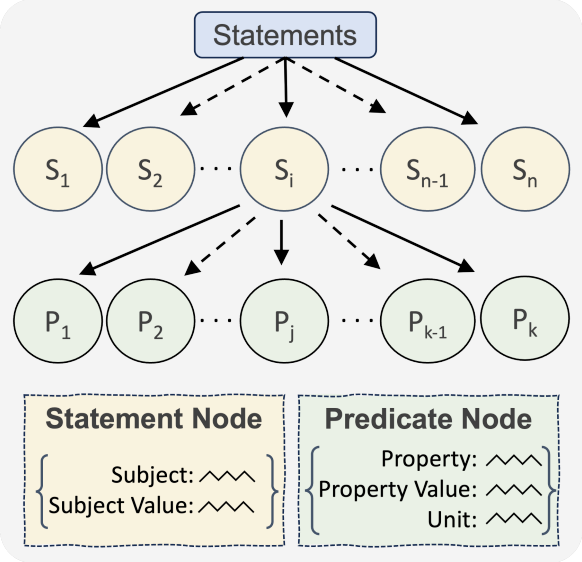

Figure 1: The knowledge model of `Statements` represented as a tree. From the root node, individual statements emerge as branches. Associated with each individual statement node are the leaf predicate nodes.

Large Language Models (LLMs) have turned out to be excellent tools for IE, due to their ability to parse, understand, and reason over textual data (OpenAI et al., 2023; Touvron et al., 2023). This, in combination with their in-context learning ability, makes them excellent for IE from text (Brown et al., 2020). This approach breaks down when applying the same techniques on tables (Zhu et al., 2021).

In this paper, we present a general approach for universal IE from tables. Universal IE involves named entity recognition and relationship extraction among other tasks. To this end, we propose a new tree-like data structure, called '`Statement`', which can combine multiple (named) entities and

Proceedings of the 1st Workshop on Natural Language Processing Meets Climate Change (ClimateNLP 2024), pages 193–214

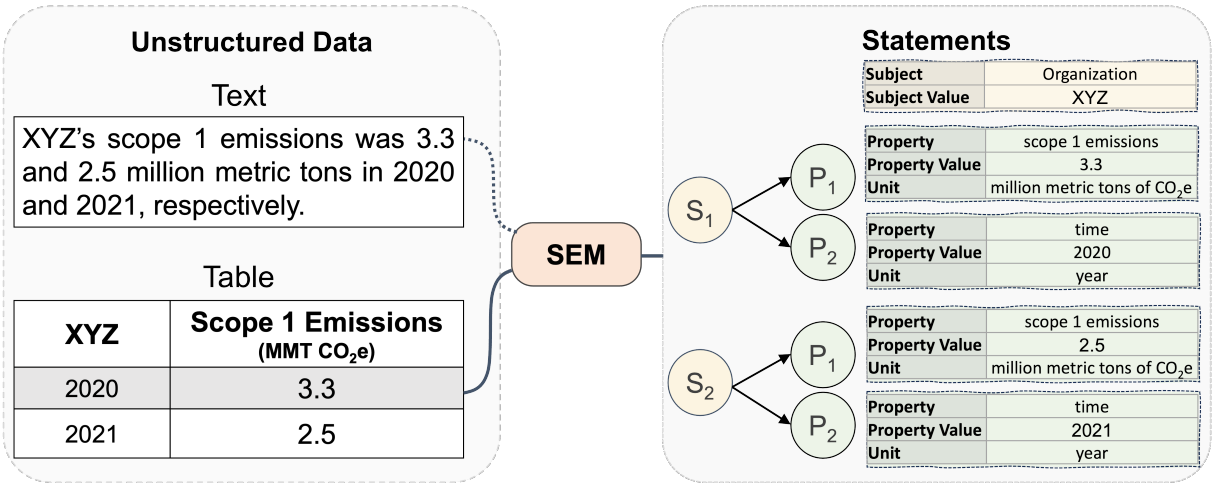

Figure 2: A diagram explaining the framework introduced in this paper. We fine-tune LLMs on the task of 'Statement Extraction' leading to a family of "**S**tatement **E**xtraction **M**odels" (SEM). Quantitative facts are extracted from heterogenous unstructured data (only tables in this paper) and stored as Statements.

(n-ary) relations (Fig. 1). It allows us to represent information in a homogeneous domain agnostic fashion. A `statement` tree can contain content from different subjects, allowing for universal IE approach to tables across multiple domains. With the introduction of `statements`, the IE problem from tables becomes a *translation problem* which we call *'statement extraction'* – translating the original table into a set of statements. ESG reports, to this day, are manually analyzed by consultancy firms and professional organisations (Henisz et al., 2019). With our proposed statement extraction, this process can now be fully automated.

To evaluate our model generated statements, we propose a novel application of the well-established Tree Edit Distance (Pawlik and Augsten, 2016). We propose Tree Similarity Score (t_s) for measuring the similarity between two trees. As baseline, we experiment with in-context learning using state-of-the-art LLMs like Mistral (Jiang et al., 2023), Mixtral (Jiang et al., 2024), Llama2 (Touvron et al., 2023), and Falcon (Almazrouei et al., 2023). These models show an average t_s varying from 0% to 21%. On the other hand, our best-performing fine-tuned T5 based model shows a t_s of 82%.Our main contributions are:

- We introduce a new knowledge model called `Statement` for mapping complex, irregular, and heterogeneous information to a uniform domain agnositc structure.
- We present a new supervised deep learning universal IE task called *'statement extraction'*. The fine-tuned models show significant improvement over baseline experiments providing competitive benchmarks for the community.
- We contribute to the field of table understanding, by providing "SemTabNet" a dataset containing over 100K annotated ESG tables. All cells in these tables are annotated to reflect their semantic relationship with other cells.
- We propose Tree Similarity Score, which in a single number quantifies the quality of entities and relationships extraction in the statement.

We begin, in Sect. 2 discussing related works. In Sect. 3 we explain the concept of '`Statements`' and present the SemTabNet dataset in Sect. 4. In sect. 5, we discuss the various experiments we performed and their results. We end the paper with an application of our model on ESG reports.

2 Related works

Fang et al. (2024) group the applications of deep learning methods to tables or tabular data into four broad categories. (1) Tree based methods such as gradient-boosted decision trees (Borisov et al., 2022) for predictions on tabular data. (2) Attention-based methods which includes developing models that learn tabular representations such as TAPAS (Herzig et al., 2020), TABERT (Yin et al., 2020), and/or fine-tuning models for downstream tasks on tabular data like fact-checking (Wenhu Chen and Wang, 2020, TABFACt), question-answering (Liu et al., 2021; Mishra et al., 2024), semantic parsing (Yu et al., 2020). (3) Regularization methods which attempts to modify model sensitivity to tabular fea-

tures (Kadra et al., 2021). (4) Data transformation methods which aim at converting heterogeneous tabular inputs to homogeneous data, like an image (Sun et al., 2019) or feature engineering (Liu et al., 2020).

Another class of problem which is similar to the data transformation approach is (generative) information extraction (IE) which involves adopting LLMs to generate structural information from an information source. Recent studies have found that LLMs can also perform universal IE (Kardas et al., 2020; Paolini et al., 2020; Wang et al., 2022a, 2023).

In a universal IE task, a model is trained to generate desirable structured information y, given a pre-defined schema s, and information source x (Lu et al., 2022). Using pre-trained language models, Wang et al. (2022b) perform IE in two steps: argument extraction and predicate extraction. Based on this, they introduced a text-based open IE benchmark. Wang et al. (2021) presented DeepEx for extracting structured triplets from text based data. Wang et al. (2022a) demonstrate that pre-training models on task-agnostic corpus lead to performance improvement on tasks like IE, entity recognition, etc. However, these approaches are limited to textual data.

Bai et al. (2024) have shown that LLMs can perform IE on tabular data when prompted with a table and a relevant extraction schema. Their approach is based on a human-in-the-loop in-context learning. A domain-expert is necessary for producing robust extraction schema, which instructs the model to generate structured records from a table. This strongly limits the adaptability of their approach to different domains. Although limited to text, (Lu et al., 2022) also propose a schema-driven universal IE system. They use a structure extraction language which generates structural schema prompt which guides the model in its IE tasks.

As we show, the `statements` data structure removes several limitations of previous universal IE approaches and is applicable to 'wild' heterogenous information sources.

3 Definition of `Statements`

The `statements` data structure aims to homogenize data coming from complex, irregular, heterogeneous information source (text or tables). At its core, the `statements` data structure is a tree structure (fig. 1). From the root of the tree, we have 'subject'-nodes, which contain information regarding the 'subject' and the 'subject-value' keys. From each subject-node, there are one or more predicate nodes, which define the 'property', 'property-value', and 'unit' keys. Each predicate node carries an atomic piece of quantitative information.

The `statement` knowledge model can be applied to both text and tables. In Fig. 2, we show the same `statements` structure which could be obtained from a text or a corresponding table. As such, the `statements` structure is not bound only to tables, however, it shows its usefulness particularly when normalising information from heterogeneous tables. The details of how we create trees are presented with examples in appendix C.

The tree structure of `statements` allows us to quantify, with a single number, the transformation of information from a table. This is accomplished by computing the Tree Similarity Score (based on the Tree Editing Distance (TED) Pawlik and Augsten (2016); Schwarz et al. (2017)) between predicted and ground-truth `statements`. TED is defined as the minimum-cost sequence of node operations that transform one tree into another. Like the Levenshtein distances on strings (Levenshtein, 1966), TED involves three kinds of operations: node insertions, deletions, and renaming. The cost of each operation can be freely defined, which makes this metric both flexible and powerful. Two trees are exactly same when their tree similarity score is 100%. To ensure high quality statement extraction, we setup robust TED costs such that minor differences can lead to poor tree similarity scores. In appendix C.2, we demonstrate tree similarity score with some examples.

It is also instructive to look at the edit types which converted the predicted statements into ground-truth statements. For this, we measure the ratio of edit type to the total number of edits. We find that the ratio of insertions and ratio of deletions carries the information about the structural similarity of two trees. If the model predicted too few nodes, the ratio of insertions will be high. Correspondingly, if the statements from the model's prediction has too many nodes, the deletion ratio dominates. If two trees are structurally similar, then the ratio of both insertion and deletions is low. In this case, the edits are dominated by renaming.

While tree-based metrics are sensitive to both entity and relationship extraction, we also would like to understand the ability of a Statement Extrac-

tion Model (SEM) to extract entities alone [1]. For this, we concatenate all the predicate nodes in a statement. We create sets of values corresponding to: subject, subject value, property, property value, unit. We count true positives when an entity is found in both the sets from model prediction and ground truth. True negatives are counted when an entity is present only in the ground truth set and false positives when the entity is present only in the predicted set. Based on these, we measure the standard accuracy, recall and F1 measures.

4 SemTabNet: `Statements` Data

There are many large data sets of annotated tables which suffer from two major limitations: (1) they focus on understanding table structure only i.e. demarcating table headers from table content, and (2) contain little diversity in shape, size, and complexity of the table. Tables found in ESG reports are of high complexity with little common underlying pattern. In this work, we advance deep learning on table understanding by annotating the content of the table and annotating complex tables.

We used the Deep Search toolkit [2] to collect over 10K ESG reports from over 2000 corporations. Deep Search crawled these PDF reports, converted them into machine readable format, and provided this data along with the metadata of each report in json format.

We compiled a list of important keywords which capture many important concepts in ESG reports (see appendix A). Next, we select only those tables which have some relevance with the keywords. For this we used the following conditions: the ROUGE-L precision (longest common sub-sequence) score between raw data and keywords must be greater than 0.75 and there must be quantitative information in the table.

We need a strategy for understanding the content of a table and extracting statements from it. After manually observing hundreds of table, we decided a two step approach to prepare our ground-truth data. First, we classify all the cells in a table based on the semantic meaning of their content into 16 categories which helps us in constructing statements. For each table, this step creates a 'labels-table' with the same shape and structure as the original, but the cells of this labels-table only contain category labels (see fig. 3). Secondly, we create a program which reads both the labels-table and the original table and extracts statements in a rule-based approach. The algorithm is described in appendix E. The 16 labels are:

- Property, Property Value
- Sub-property
- Subject, Subject Value
- Unit, Unit Value
- Time, Time Value
- Key, Key Value
- Header 1, Header 2, Header 3
- Empty, Rubbish

During annotation, all cells are mapped to one of the above labels. For cells which contain information pertaining to more than one label, we pick the label which is higher in our ordered list of labels. So "Revenue (US$)", is labelled as `property`. The 'property' and 'sub-property' cells always have associated 'property value' cell(s). The 'header' cells never have an associated value and often divide the table into smaller sections. Empty cells are labelled 'empty'. When a table contain unnecessary parts due to faulty table recovery or non-quantitative information. We label such cells as 'rubbish'. When a property/property value pair carries supplementary information, those cells are annotated as 'key'/'key values'.

Additionally, we observed that most tables can be reasonably classified into three baskets: simple, complex, and qualitative. There are simple tables whose structure cannot be further subdivided into any smaller table. There are complex tables whose structure can be further divided into multiple smaller tables. Finally, there are qualitative tables (like table of contents) which contain little valuable information for our endeavour.

We collected about 2,800 tables and found $\sim 20\%$ had simple layout, $\sim 20\%$ had complex layout (composed of multiple simpler tables arranged hierarchically), and $\sim 60\%$ were qualitative. We discarded all qualitative tables from any further analysis. To ensure that our data is not biased towards either simple or complex tables, we manually annotated all the cells of 569 simple tables and 538 complex tables. In total, we annotated 1,107 tables (84,890 individual cells) giving rise to 42,982 statements.

Due to the nature of our strategy, one can extract statements from tables either directly in a zero shot manner (direct SE) or by predicting cell labels and

[1] Here, 'entity' refers to the values of attributes in a statement. For example, 'scope 1 emissions' is an entity from the statement shown in fig. 2.

[2] Available via: https://ds4sd.github.io.

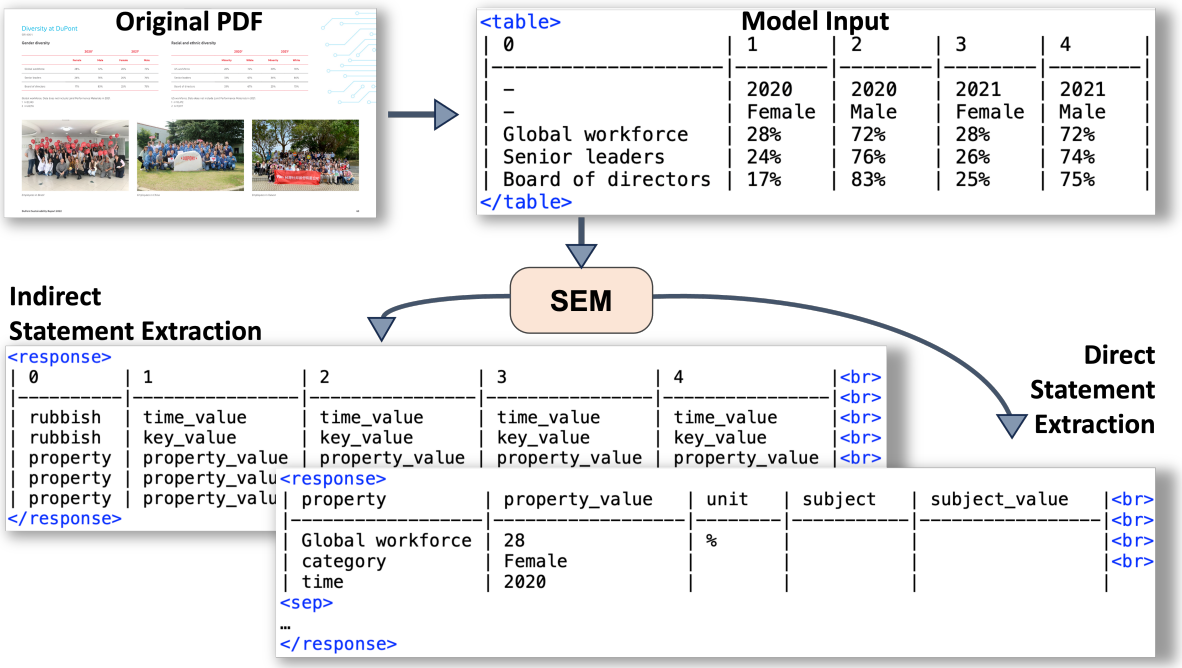

Figure 3: Input and output for the task of "Statement Extraction". *Top Left:* Page from an ESG report containing tables. *Top Right:* One of the table, from the same page, prepared as markdown for model input. *Bottom Left:* Model output for the task of indirect statement extraction. *Bottom Right:* Model output for the task of direct statement extraction.

then using the rule-based approach to construct statements (indirect SE) (see Fig. 3. We have experimented with both approaches.

We further augmented the annotated tables to create a large training data. We shuffle the rows and columns of tables corresponding to property-values to create new augmented tables, while keeping their contents the same. While this is straightforward for simple tables, special care was taken for complex tables such that only rows/columns which belonged together within a category were shuffled. The maximum number of augmented tables emerging from the shuffling operations was limited to 130, leading to over 120K tables. To promote further research and development, we open source this large dataset of semantic cell annotations as SemTabNet[3]. Table 1 shows the data counts in SemTabNet.

Table 1: Counts of data in SemTabNet. Tasks are explained in section 5.[4].

TASK	TRAIN	TEST	VAL
SE DIRECT	103,455	11,682	5,445
SE INDIRECT 1D	72,580	8,489	3,821
SE INDIRECT 2D	93,153	22,839	4,903

5 Experiments & Results

Fig 3 presents Statement Extraction as a supervised deep learning task. Due to the nature of how tables are annotated (see section 4), it is possible to train models for statement extraction statements both directly and indirectly. We consider the following three seq2seq experiments: (1) *SE Direct*: the model is presented with an input table as markdown in a prompt. The model generates the tabular representation of the resulting statements as markdown. (2) *SE Indirect 1D*: In this experiment, the model input is the individual table cell contents. For a table with n cells, we predict n labels sequentially (hence, 1D) and then use this information to construct statements. Individual cell labels predicted by the model are stitched together to form the labels table, which is then used to construct the predicted statement by using our rule-based al-

[3]Links for code and data, respectively:
https://github.com/DS4SD/SemTabNet
https://huggingface.co/datasets/ds4sd/SemTabNet

[4]The counts differ slightly due to the manner in which the final data was harmonized. The SE Indirect 1D data consists of the 84 890 original cells annotated from 1 107 tables. The test/train split of tables for SE Indirect 1D was prepared by stratifying across all cell labels. This split was augmented (as described in text) to prepare data for SE Indirect 2D. The test/train split and augmentation for SE Direct was done independently.

gorithm. (3) *SE Indirect 2D*: As opposed to SE Indirect 1D, in this experiment, we predict the cell labels of all cells in a table simultaneously. The entire table, as markdown, is input to the model (hence 2D) and the model generates the labels table, as markdown. Using the rule-based algorithm, the predicted labels table is converted into predicted statements.

We use six special tokens, which allow us to control and parse model output.

- Input table start token: `<table>`
- Input table stop token: `</table>`
- Output start token: `<response>`
- Output stop token: `</response>`
- Newline token: `<br>`
- Separate list item token: `<sep>`

This allows us to parse the predicted statements from a LLM. Once successfully parsed, the output statements can be trivially converted from one representation to another. This is crucial because we compare model predicted statements with ground truth by converting statements into a tree structure. These tokens are added to the tokenizer vocabulary before fine-tuning any model.

Since the nature of these tasks naturally fits the paradigm of sequence-to-sequence models, we fine-tune T5 models (Raffel et al., 2020). T5 models are encoder-decoder transformer architecture models which are suitable for many sequence-to-sequence tasks. In our experiments, we train T5 variants (Small, Base, Large, and 3B) to create a family of Statement Extraction Models (SEM).

In our training data for tables, the input token count is less than 512 for 50% of the data, and it is less than 1024 for 90% of the data. Thus, except where mentioned, we train T5 models (small, base, large) with context windows of 512 and 1024, and T5-3b with context window of 512. All models are fine-tuned in a distributed data parallel (DDP) manner simultaneously across 4 GPU devices (Nvidia A100-40GB for T5-Small, T5-Base, T5-Large and NVIDIA A100-80GB for T5-3B). Additionally, the largest possible batch size was used for all models. The batch size is impacted by factors like model size, GPU memory, and context window. In turn it affects the number of epochs we can fine-tune in a reasonable time.

For all tasks, we stop the fine-tuning process either after 500,000 steps or after 7 days. We use the AdamW optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. All models are trained with a maximum learning rate of 5×10^{-4}. There is a warm-up phase of 1000 steps in which the learning rate increases linearly from 10^{-10} to 5×10^{-4}. After another 1000 steps, the learning rate is exponentially decayed until it reaches its lowest value of 10^{-6}, where it remains until the end of the training.

Table 2 presents the key results of our experiments. For each table, we evaluate the statements predicted by the model (directly or indirectly) against the ground truth statements. For each task and each model therein, we present the averaged tree similarity score (t_s) (measuring entity & relationship extraction) and the averaged F1 score (measuring entity extraction). Also present are the averaged ratios of tree edit types, which helps us understand t_s. For all reported values, assuming a normal distribution, the standard error of the mean is below 5×10^{-5} and the 99% confidence interval for all values is about $\sim 0.1\%$.

Baseline Experiments: For baseline experiments, several state of the art LLMs were tested for their in-context learning ability. In the prompt, we show the model an example of direct statement extraction (1-shot), followed by a test table.

The models produce statements in markdown format, which are evaluated against ground truth statements. The average tree similarity score across 1100 annotated tables varies from 0% for Falcon40b to 20% for Mixtral (8×7b models). For entity extraction, Llama2-13b performed the best with an average F1 score of 38. Not all outputs generated by the model were in correct markdown format. Minor changes in the prompt were found to create vast differences in the quality of extracted statements. In appendix D, we show examples of the prompt and the model output for some cases.

Statement Extraction Indirect 1D: All models trained on this task have context window of 512. Their performance tends to scale with model size. These models can learn to extract entities, but relationship extraction is difficult. For SEM-T5-small, the ratio of insertion is $\approx 98\%$ which means that the predicted statements does not have enough nodes.

Statement Extraction Indirect 2D: All models trained on this task perform well on entity extraction with average F1 scores of over 95%. The highest performing model is the SEM-T5-3b (512) with an average tree similarity score of 81.76%.

Statement Extraction Direct:Based on tree similarity score, most models show poor performance in direct SE. The best performing model is SEM-

Table 2: Results of experiments performed for Statement Extraction (bold indicates the best in each experiment). The comparison between the ground truth and the model-predicted statements is encapsulated by the Tree Similarity Score (t_s). t_s measures if two trees are similar (100% being an exact match). For each statement, the precision, recall and F1 score (reported) of entity extraction extraction was also measured. For all reported values, the 99% confidence interval, assuming a Gaussian distribution, is $\sim 0.1\%$. The standard error of the mean in all cases is below 0.005%.

		Context	Invalid	Ratio Tree Edits [%]			Average [%]	
Task	Model	Length	Output [%]	Insert	Delete	Rename	F1	t_s
	Falcon-40b	2048	12.59	45.69	28.77	25.54	17.94	0.15
Baseline	Llama-2-13b	4096	17.93	79.95	5.78	14.27	**37.94**	5.29
In-Context	Llama-2-70b	4096	24.82	89.65	2.56	7.79	3.18	6.31
1-shot	Mistral-7b	8192	21.92	53.37	18.76	27.87	16.92	11.57
	Mixtral-8x7b	8192	19.20	56.39	18.06	25.56	6.51	**21.07**
	SEM-T5-small	512	00.00	98.13	00.00	1.87	62.32	00.86
Indirect 1D	SEM-T5-base	512	00.00	83.95	01.68	14.37	83.46	09.21
	SEM-T5-large	512	00.00	34.68	12.03	53.30	**94.67**	**55.68**
	SEM-T5-3b	512	00.00	36.70	23.24	40.05	90.49	22.24
	SEM-T5-small	512	64.62	17.34	13.36	69.30	97.06	75.15
	SEM-T5-base	512	57.85	15.53	21.60	62.86	96.85	73.87
	SEM-T5-large	512	61.81	09.58	22.80	67.62	97.55	80.83
Indirect 2D	SEM-T5-3b	512	50.88	08.00	28.40	63.59	**97.38**	**81.76**
	SEM-T5-small	1024	58.37	18.53	18.71	62.75	95.85	68.45
	SEM-T5-base	1024	46.39	17.80	16.04	66.16	96.15	69.27
	SEM-T5-large	1024	53.33	08.20	17.00	74.79	97.53	79.89
	SEM-T5-small	512	00.00	98.14	00.04	01.82	60.65	00.62
	SEM-T5-base	512	00.00	97.86	00.06	02.09	68.62	04.46
	SEM-T5-large	512	00.00	98.18	00.02	01.80	67.41	04.23
Direct	SEM-T5-3b	512	00.00	97.98	0.01	02.01	70.06	03.47
	SEM-T5-small	1024	00.00	92.93	00.14	06.93	70.35	02.98
	SEM-T5-base	1024	00.00	88.42	00.22	11.35	**76.99**	**11.11**
	SEM-T5-large	1024	00.00	89.34	00.21	10.45	76.59	06.06

T5-base with a context window of 1024. It gets an average F1 score of 76.99% and an average tree similarity score of only 11%. To understand, why these models performs so poorly on direct SE, we look at the ratio of tree edits.

We note that the ratio of deletions for all models in this task is close to 0. On the other hand, the ratio of insertions for all models is high (from 88% to 98%). This suggests that the statement trees produced by these models is missing vast number of nodes compared to the ground truth. In fact, perusing the model output shows that while the output is of high quality, it contains significantly less nodes than ground truth statements.

Discussion: SE Indirect 1D shows good performance on entity extraction, but performs poorly for both entity and relationship extraction. In this task, the model only sees the content of one cell at a time which makes it easy to extract entities. However, this does not allow the model to develop a strong capability to learn tabular relationships. On the other hand, SE Direct, gives poor performance on both entity extraction and relationship extraction. Direct SE expects the models to unravel a dense table into `statements`, for which they must produce many output tokens. For example, the average number of output tokens in the test data for SE direct is 5773 ± 51, which is significantly larger than the number of tokens for SE indirect 2D (346 ± 1). Thus, direct SE is a very challenging task and might require different strategies to be executed successfully.

SE Indirect 2D, avoids the disadvantages of both the tasks. In this case, the model sees the entire input table (has the chance to learn tabular relationships) and is only tasked with producing a labels

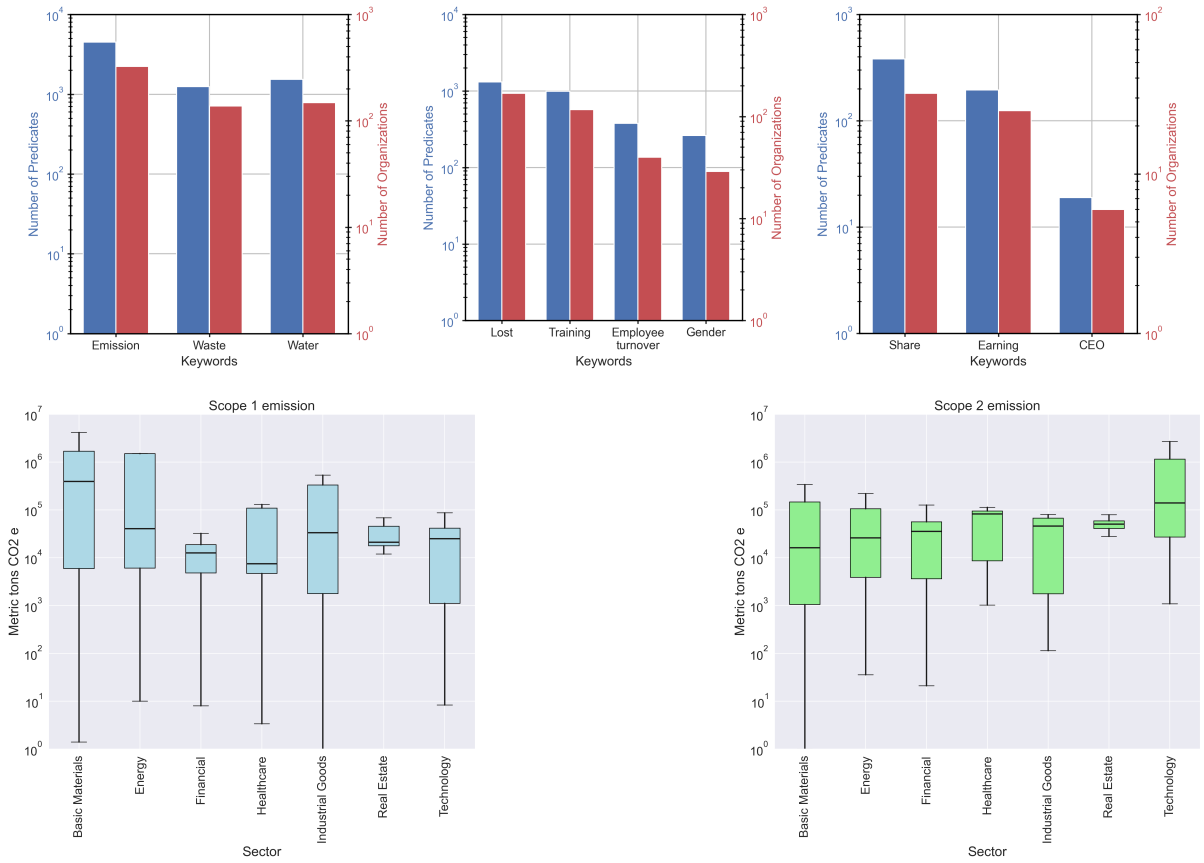

Figure 4: Exploratory data analysis of `statements` from over 2700 Tables published in ESG reports in 2022. *Top:* We searched about 50,000 predicates using keywords (shown on the x-axis) related to environment (left), social (middle), and governance (right). The plot shows the distribution of predicates and the number of organizations from this search. *Bottom:* Box plot for extracted Scope 1 and Scope 2 emission values grouped by business sectors from over 300 companies across multiple years. Only sectors with more than 20 data points are included.

table (can finish generation in a reasonable number of tokens). Our experiments clearly demonstrate that statement extraction via the Indirect 2D approach gives better results. This is an unexpected finding of our study, and we hope it motivates other researchers to improve zero-shot statement extraction capability.

6 Application to ESG results

Due to their homogeneous structure, statements enable large-scale exploratory data analysis and data science. To demonstrate the advantage of statements over traditional tabular data science, we applied SEM-T5-large (512 SE Indirect 2D) over 2700 tables published in over 1000 ESG reports in 2022. This lead to 14,766 statements containing over 100k predicates. This dataset containing ESG related KPIs is invaluable to researchers, policymakers, and analysts.

We filter this large dataset to contain only those predicates with quantitative property values. This subset contains 47 901 predicates from 601 corporate ESG reports. We search the properties in this dataset for some keywords representative of ESG KPIs. Fig. 4 (top) shows the distribution of the number of predicates and the number of distinct organizations which matched our simple keyword search. For example, using 'emission' as a keyword, we obtain over 4000 hits with results coming from over 300 distinct corporations.

Fig. 4 (bottoms) shows the total scope 1 emissions (left) and total scope 2 emission (right). Each box shows the distribution of emission from multiple corporations across sectors ($\sim$ 20 in Healthcare to $\sim$ 100 in Technology and Industrial Goods) containing data from several years. The data reported in the original report contained emissions in different units, which were harmonized for creating this plot.

Since we only took a small subset of 1000 reports for this analysis, our data is incomplete and is only representative. The `statements` dataset al-

lows one to study how emissions from individual companies or across sectors have evolved over time. This dataset can also serve as a starting point for many other downstream applications like question-answering, fact checking, table retrieval, etc.

7 Conclusion & Future Works

We have presented a novel approach to map complex, irregular, and heterogeneous information to a uniform structure, `Statements`. We presented Statement Extraction which is a new supervised deep-learning information extraction task. We contribute the field of table understanding by open-sourcing SemTabNet consisting of 100K ESG tables wherein all cells.

Investigating three variations of the statement extraction task, we found that using a model to generate table annotations and then construct `statements` produces best results. This approach has the advantage, that it produces homogeneous structured data with reduced hallucinations. `Statements` are an advantageous vehicle for quantitative factual information. They enable down-stream tasks like data science over a large collection of documents. We extracted over 100K facts (predicates) from only 1000 ESG reports.

This work can be easily extended to include domains other than ESG. It can also be extended towards multi-modality by including text data. We leave for future exploration, the use of statements in downstream tasks like QA or document summarization.

Limitations

Although, the ideas and the techniques we describe in this paper are domain agnostic, we limit the scope of this paper to the domain of corporate Environment, Social, and Governance (ESG) reports. This choice is motivated by two observations. First, corporations report valuable quantitative data regarding their efforts to improve their carbon emissions, working conditions, and company culture in ESG reports. These reports contain valuable information regarding the environmental impact of businesses, and the urgency of climate change motivates us to target this domain. Secondly, there is a large variety and diversity of tabular representations used in these reports. Despite efforts to standardize these reports, this diversity makes the task of extracting information from these documents extremely challenging, motivating our choice.

The scope of this work is limited to declarative, explicit knowledge only. All other kinds of knowledge such as cultural, implicit, conceptual, tacit, procedural, conditional, etc. are ignored. We focus on information which one colloquially refers to as 'hard facts'. Additionally, we limit the scope of this work to quantitative statements i.e. statements whose property values are numerical quantities. We implement this restriction in the notion that we avoid qualitative statements i.e. statements which are not quantitative.

Our model training strategy was biased against large models. We trained all models for either 500K steps or 7 days using the largest possible batch size. This means smaller models learn more frequently (more epochs) than larger models. However, we do not believe this severely impacted the outcome of our experiments. Our resources were enough to recover well-known trends: improved model performance with model size and context-length.

References

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, Daniele Mazzotta, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. 2023. The falcon series of open language models.

Fan Bai, Junmo Kang, Gabriel Stanovsky, Dayne Freitag, and Alan Ritter. 2024. Schema-Driven Information Extraction from Heterogeneous Tables. ArXiv:2305.14336 [cs].

Julia Bingler, Mathias Kraus, Markus Leippold, and Nicolas Webersinke. 2022. How Cheap Talk in Climate Disclosures relates to Climate Initiatives, Corporate Emissions, and Reputation Risk.

Vadim Borisov, Tobias Leemann, Kathrin Sessler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. 2022. Deep Neural Networks and Tabular Data: A Survey. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–21.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*,

volume 33, pages 1877–1901. Curran Associates, Inc.

Xi Fang, Weijie Xu, Fiona Anting Tan, Jiani Zhang, Ziqing Hu, Yanjun Qi, Scott Nickleach, Diego Socolinsky, Srinivasan Sengamedu, and Christos Faloutsos. 2024. Large Language Models(LLMs) on Tabular Data: Prediction, Generation, and Understanding – A Survey. ArXiv:2402.17944 [cs].

Witold Henisz, Tim Koller, and Robin Nuttall. 2019. Five ways that ESG creates value. *McKinsey Quarterly*.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. TaPas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333, Online. Association for Computational Linguistics.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7B. ArXiv:2310.06825 [cs].

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, L'elio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of experts. *ArXiv*, abs/2401.04088.

Arlind Kadra, Marius Lindauer, Frank Hutter, and Josif Grabocka. 2021. Well-tuned Simple Nets Excel on Tabular Datasets.

Marcin Kardas, Piotr Czapla, Pontus Stenetorp, Sebastian Ruder, Sebastian Riedel, Ross Taylor, and Robert Stojnic. 2020. AxCell: Automatic Extraction of Results from Machine Learning Papers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8580–8594, Online. Association for Computational Linguistics.

V. I. Levenshtein. 1966. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707.

Qian Liu, Bei Chen, Jiaqi Guo, Morteza Ziyadi, Zeqi Lin, Weizhu Chen, and Jian-Guang Lou. 2021. TAPEX: Table Pre-training via Learning a Neural SQL Executor.

Zhaocheng Liu, Qiang Liu, Hao Zhang, and Yuntian Chen. 2020. Dnn2lr: Interpretation-inspired feature crossing for real-world tabular data. *ArXiv*, abs/2008.09775.

Yaojie Lu, Qing Liu, Dai Dai, Xinyan Xiao, Hongyu Lin, Xianpei Han, Le Sun, and Hua Wu. 2022. Unified Structure Generation for Universal Information Extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5755–5772, Dublin, Ireland. Association for Computational Linguistics.

Lokesh Mishra, Cesar Berrospi, Kasper Dinkla, Diego Antognini, Francesco Fusco, Benedikt Bothur, Maksym Lysak, Nikolaos Livathinos, Ahmed Nassar, Panagiotis Vagenas, Lucas Morin, Christoph Auer, Michele Dolfi, and Peter Staar. 2024. ESG Accountability Made Easy: DocQA at Your Service. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(21):23814–23816. Number: 21.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mo Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa

Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, C. J. Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2023. GPT-4 Technical Report. ArXiv:2303.08774 [cs].

Giovanni Paolini, Ben Athiwaratkun, Jason Krone, Jie Ma, Alessandro Achille, Rishita Anubhai, Cicero Nogueira dos Santos, Bing Xiang, and Stefano Soatto. 2020. Structured Prediction as Translation between Augmented Natural Languages.

Mateusz Pawlik and Nikolaus Augsten. 2016. Tree edit distance: Robust and memory-efficient. *Information Systems*, 56:157–173.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Tobias Schimanski, Andrin Reding, Nico Reding, Julia Bingler, Mathias Kraus, and Markus Leippold. 2024. Bridging the gap in ESG measurement: Using NLP to quantify environmental, social, and governance communication. *Finance Research Letters*, 61:104979.

Stefan Schwarz, Mateusz Pawlik, and Nikolaus Augsten. 2017. A New Perspective on the Tree Edit Distance. In *Similarity Search and Applications*, Lecture Notes in Computer Science, pages 156–170, Cham. Springer International Publishing.

Baohua Sun, Lin Yang, Wenhan Zhang, Michael Lin, Patrick Dong, Charles Young, and Jason Dong. 2019. SuperTML: Two-Dimensional Word Embedding for the Precognition on Structured Tabular Data. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2973–2981. ISSN: 2160-7516.

Hugo Touvron, Louis Martin, and Kevin Stone. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models.

Chenguang Wang, Xiao Liu, Zui Chen, Haoyun Hong, Jie Tang, and Dawn Song. 2021. Zero-Shot Information Extraction as a Unified Text-to-Triple Translation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1225–1238, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Chenguang Wang, Xiao Liu, Zui Chen, Haoyun Hong, Jie Tang, and Dawn Song. 2022a. DeepStruct: Pretraining of Language Models for Structure Prediction. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 803–823, Dublin, Ireland. Association for Computational Linguistics.

Chenguang Wang, Xiao Liu, and Dawn Song. 2022b. IELM: An Open Information Extraction Benchmark for Pre-Trained Language Models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8417–8437, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Xiao Wang, Weikang Zhou, Can Zu, Han Xia, Tianze Chen, Yuansen Zhang, Rui Zheng, Junjie Ye, Qi Zhang, Tao Gui, Jihua Kang, Jingsheng Yang, Siyuan Li, and Chunsai Du. 2023. InstructUIE: Multi-task Instruction Tuning for Unified Information Extraction. ArXiv:2304.08085 [cs].

Jianshu Chen Yunkai Zhang Hong Wang Shiyang Li Xiyou Zhou Wenhu Chen, Hongmin Wang and William Yang Wang. 2020. Tabfact : A large-scale dataset for table-based fact verification. In *International Conference on Learning Representations (ICLR)*, Addis Ababa, Ethiopia.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. TaBERT: Pretraining for joint understanding of textual and tabular data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8413–8426, Online. Association for Computational Linguistics.

Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Bailin Wang, Yi Chern Tan, Xinyi Yang, Dragomir Radev, Richard Socher, and Caiming Xiong. 2020. GraPPa: Grammar-Augmented Pre-Training for Table Semantic Parsing.

Yitan Zhu, Thomas Brettin, Fangfang Xia, Alexander Partin, Maulik Shukla, Hyunseung Yoo, Yvonne A. Evrard, James H. Doroshow, and Rick L. Stevens. 2021. Converting tabular data into images for deep learning with convolutional neural networks. *Scientific Reports*, 11(1):11325. Number: 1 Publisher: Nature Publishing Group.

A ESG Keywords

Environment

1. **Scope 1 GHG Emissions**
   Scope 1 are all direct emissions from the activities of an organization under their control. This includes fuel combustion on site such as gas boilers, fleet vehicles and air-conditioning leaks.
2. **Scope 2 GHG Emissions Market Volume**
   Scope 2 are indirect emissions from electricity purchased and used by the organization. Emissions are created during the production of the energy and eventually used by the organization. A market-based method reflects emissions from electricity that companies have actively chosen to purchase or reflects their lack of choice.
3. **Scope 2 GHG Emissions Location Volume**
   Scope 2 emissions are indirect emissions from the generation of purchased energy. A location-based method reflects the average emissions intensity of grids on which energy consumption occurs (using mostly grid-average emission factor data)
4. **Scope 2 GHG Emissions Other Volume**
   Scope 2 emissions are indirect emissions from the generation of purchased energy. Overall, if not clearly defined whether it is market-based calculation or location-based calculation
5. **Scope 3 GHG Emissions**
   Scope 3 emissions are all other indirect emissions (excluding Scope 2) that occur in the value chain of the reporting company, including both upstream and downstream emissions.
6. **Environmental Restoration and Investment Initiatives Monetary Value**
   The fields represent the monetary value spent on environmental initiatives.
7. **Total Water Discharged**
   The fields represent the overall volume of water discharged by a company.
8. **Total Water Withdrawal**
   The fields represent the total volume of water withdrawn by a company.
9. **Total Water Recycled**
   The fields represent the total volume of water recycled or reused by a company.
10. **Toxic Air Emissions - NOx**
    The fields represent the total amount of nitrous oxide (NOx)emissions emitted by a company.
11. **Toxic Air Emissions - SOx**
    The fields represent the total amount of sulfur oxide (Sox) emissions emitted by a company.
12. **Toxic Air Emissions - Overall**
    The fields represent the total amount of air emissions emitted by a company.
13. **Toxic Air Emissions - VOC**
    The fields represent the total amount of volatile organic compound (VOC) emissions emitted by the company.
14. **Hazardous Waste - Disposed to Aquatic**
    The fields represent the total amount of hazardous waste disposed to aquatic environment.
15. **Hazardous Waste - Disposed to Land**
    The fields represent the total amount of hazardous waste disposed to non aquatic or land environment.
16. **Hazardous Waste - Total Recycled**
    The fields represent the total amount of hazardous waste recycled.
17. **Hazardous Waste - Total Amount Generated**
    The fields represent the total amount of hazardous waste generated by a company.
18. **Hazardous Waste - Total Amount Disposed**
    The fields represent the total amount of hazardous waste disposed.
19. **Non-Hazardous Waste - Disposed to Aquatic**
    The fields represent the total amount of non-hazardous waste disposed to the aquatic environment.
20. **Non-Hazardous Waste - Disposed to Land**
    The fields represent the total amount of non-hazardous waste to non aquatic or land environment
21. **Non-Hazardous Waste - Total Recycled**
    The field represents the total amount of non-hazardous waste recycled.
22. **Non-Hazardous Waste - Total Amount Generated**
    The fields represent the total amount of non-hazardous waste Generated by a company.

23. **Non-Hazardous Waste - Total Amount Disposed**
    The fields represent the total amount of non-hazardous waste disposed.
24. **Total Waste Produced**
    The fields represent the total amount of waste produced by a company.
25. **Total Waste Recycled**
    The fields represents the total amount of waste recycled by a company.
26. **Total Waste Disposed**
    This fields represent the total amount of waste disposed by a company.
27. **Number of Sites in Water Stress Areas**
    The field represents the number of sites located in water stress areas.
28. **E-Waste Produced**
    The field identifies the mass volume of f E-waste produced which are electronic products that are unwanted, not working, and nearing or at the end of their life. Examples of electronic waste include, but not limited to : computers, printers, monitors, and mobile phones
29. **E-Waste Recycled**
    The field identifies the mass volume of E-Waste Recycled.
30. **E-Waste Disposed**
    The field identifies the mass volume of E-waste disposed.
31. **Number of Sites Operating in Protected and/or High Biodiversity Areas**
    The field identifies the number of sites or facilities owned,leased, managed in or adjacent to protected areas and areas of high biodiversity value outside protected areas.
32. **Impacted Number of Species on International Union of Conservation of Nature (IUCN) List**
    The field identifies the number of impacted species on International Union of Conservation of Nature (IUCN) red list.
33. **Impacted Number of Species on National listed Species**
    The field identifies the number of impacted species on National Listed Species.
34. **Baseline Level**
    The field identifies the value at baseline or year that target is set against.
35. **Target Year**
    The field identifies the year in which the renewable energy goal is set to be completed.
36. **Target Goal**
    The field identifies the target goal for renewable energy.
37. **Actual Achieved**
    The fields identifies the actual value achieved for the renewable energy goal.
38. **Baseline Level**
    The field identifies the baseline emissions value.
39. **Target Year**
    The field identifies the year in which GHG emission goal is set to be completed.
40. **Target Goal**
    The field identifies the target goal for GHG emission reduction.
41. **Actual Achieved**
    The field identifies the value achieved of GHG emissions reduced compare - in metric tons.

Social

1. **Training Hours Per Employee**
   The fields identifies the numerical value of training hours per employee.
2. **Training Hours Annually**
   The fields identifies the numerical values of training hours conducted within a year.
3. **Lost Time Injury Overall Rate**
   The fields identifies the total number of injuries that caused the employees and contractors to lose at least a working day.
4. **Lost Time Injury Rate Contractors**
   The fields identifies the number of injuries that caused the contractors to lose at least a working day.
5. **Lost Time Injury Rate Employees**
   The fields identifies the number of injuries that caused the employees to lose at least a working day.
6. **Employee Fatalities**
   The fields identifies the number of employee fatalities during a one year period.
7. **Contractor Fatalities**
   The fields identifies the number of contractor fatalities during a one year period.
8. **Public Fatalities**
   The fields identifies the number of general public fatalities during a one year period.
9. **Number of Other Fatalities**
   The fields identifies the number of fatalities during a one year period not broken down by employee, contractor, or public.
10. **Total Incident Rate Overall Workers**
    The field identifies the number of work-related

injuries per 100 overall workers during a one year period for both employees and contractors.

11. **Total Incident Rate Contractors**
    The field identifies the number of contractor work-related injuries per 100 overall workers during a one year period.
12. **Total Incident Rate Employees**
    The field identifies the number of work-related injuries per 100 overall workers during a one year period for employees.
13. **Employee Turnover - Gender Male Rate**
    The field identifies the absolute number turnover rate by males in a company .
14. **Employee Turnover - Gender Female Rate**
    The field identifies the absolute number turnover rate by females in a company.
15. **Employee Turnover Overall Rate**
    The field identifies the absolute number turnover rate for overall employees in a company.
16. **Median Gender Pay Gap - Global**
    The field identifies the gender pay gap median value of the company at a global level.
17. **Mean Gender Pay Gap - Global**
    The field identifies the gender pay gap mean or average value of the company at a global level.
18. **Median Gender Pay Gap by Location**
    The field represents the gender pay gap median value of the company at a location or country level.
19. **Mean Gender Pay Gap by Location**
    The field represents the gender pay gap mean/average value of the company at a location or country level.
20. **Employee Turnover by Age - Lower Value**
    The field Identifies the minimum age in a given range for employee turnover statistics.
21. **Employee Turnover by Age - Upper Value**
    The field identifies the maximum age in a given range for employee turnover statistics.
22. **Employee Turnover by Age - Rate**
    The field identifies the employee turnover rate.
23. **Employee Turnover by Location Rate**
    The field identifies the absolute number of employee turnover rate by location.
24. **Workforce Breakdown Rate**
    The field identifies the absolute number of employees of a company based on seniority, ethnicity or gender.
25. **Workforce Breakdown Job Category Data: Value (ABS)**
    The field represents the employee count absolute value at a category level within a workforce.
26. **Number Of Product Recalls**
    The fields identifies the number of product recalls.
27. **Product Recalls Annual Recall Rate**
    The fields identifies the product recall rate of a company.

Governance

1. **Percentage of Negative Votes on Pay Practices Year**
2. **Board of Director Term Limit**
   The field identifies maximum amount of years a board member can serve.
3. **Board of Director Term Duration**
   The field identifies number of years a board member can serve before reelection.
4. **Auditor Election Year**
   The field identifies when the current lead auditor elected.
5. **Independent Auditor Start Year**
   The field represents the start year the company started having the audit company as its independent auditor.
6. **Average/Mean Compensation of Company Employees-Global**
   The field represents the average or mean compensation for company employeesat a global level.
7. **Ratio Average Compensation of CEO to Employee - CEO- Global**
   The field represents the ratio between the compensation paid to the companies CEO and the average compensations received by employees at a global level.
8. **Compensation of Company Employees by Location**
   The field identifies the average compensation for company employees at a location level.
9. **Number of Suppliers Complying with Code of Conduct**
   The field identifies the number of suppliers that comply with companies supplier code of conduct.
10. **Share Class Numeric**
    The field identifies the share class numeric component.
11. **Voting Rights**
    The field identifies the number of voting rights per each share of stock within each class.

12. **Shares Outstanding**
    The field identifies the number of shares outstanding within a companies common stock.
13. **Chairman Effective Begin Year**
    The field indicates the year when the current chairman assume his or her position. This field is used if a full effective date is not available.
14. **Chairman Effective End Year**
    The field indicates the year when the chairman left the position.
15. **CEO Effective Begin Year**
    The field identifies the year the CEO assumed his or her position.
16. **CEO Effective End Year**
    The field indicates the year when the CEO left the position.
17. **CEO Compensation Salary**
    The field identifies the current CEO salary.
18. **CEO Compensation Overall**
    The field identifies the CEO's overall compensation including salary, bonuses and all awards.
19. **CEO Cash Bonus**
    The field identifies the cash bonus value for the CEO.
20. **CEO Stock Award Bonus**
    The CEO Stock Award Bonus value
21. **CEO Option Awards**
    The CEO Option Awards bonus value
22. **CEO Other Awards**
    The fields identifies other compensation outside of salary, cash bonus, stock award bonus and option awards. This could include change in pension and values categorized as "all other compensation"
23. **CEO Pension**
    The fields identifies the CEO pension amount.
24. **Cash Severance Value**
    The fields identifies the amount of cash the severance policy for each category.
25. **Total Severance Value**
    The fields identifies the total value amount of the severance policy.
26. **CEO Share Ownership**
    The field identifies the number of shares the CEO owns in the company.
27. **CEO Share Class Numeric**
    The field identifies the share class numeric component.
28. **Board Member Age**
    The field identifies the age of the members of the board.
29. **Board Member Term in Years**
    The fields identifies how long the individual board member has been on the board which is determined in years.
30. **Board Member Effective Year (Director Since)**
    The fields identifies the year the individual board member started serving on the board.
31. **Board Profile As of Year**
    The field identifies the year of the board information. An example would be the year of the proxy statement.
32. **Participation On Other Company Board**
    The field identifies the number of boards a member is part of outside of the organization.
33. **For Value Negative Votes on Directors**
    The field identifies the number of for value votes the director received.
34. **Against Value Negative Votes on Directors**
    The field identifies the number of against votes the director received.
35. **Abstain Value Negative Votes on Directors**
    The field identifies the number of votes that were abstained for a given director.
36. **Broker Non Vote Value Negative Votes on Directors**
    The field identifies the number of broker non votes for given director.
37. **Number of Board Meetings Attended by Board Member**
    The field identifies the number of board meetings attended by a board member.
38. **Number of Board Meetings Held by Company**
    The field identifies the number of board meetings held by a company while member was on the board.
39. **Total Members on Board per Skill Set**
    The field identifies the number of board members within a specific skillset type.

B Examples of Statements

A `statement` is complete when it contains all the predicates needed to completely specify objective knowledge pertaining to a subject, i.e. a `statement` includes all co-dependent predicates. We borrow this notion of completeness from the fields of natural science. An important implication of these definitions is that within a single statement, multiple predicates cannot carry information about the same 'property'. This implies, for example, multiple measurements of the same variable in *n* different conditions will lead to *n* different statements. While complete `statements` are extremely valuable, we find that incomplete `statements` are quite resourceful, especially as we apply our ideas to domains outside of natural science.

Examples of statements from other domains are shown below.

Basic Sciences: Consider the following piece of text or unstructured data. "At a pressure of one atmosphere (atm), water freezes (solidifies) at 0 °C and water boils at 100 °C." We note that to completely describe the phase changes of water, we need to specify both temperature and pressure. Leaving any one of temperature or pressure out makes the information regarding phase change incomplete. This information is presented as `statements` in the Tables table 3 and table 4. This example demonstrates that multiple statements can be extracted from even single sentences.

Table 3: Example Statement from Material Science: Phase change of water from solid to liquid.

Subject	Subject Value	Property	Property Value	Unit
Chemical	Water	freezing temperature	0	°C
Chemical	Water	pressure	1	atmosphere

Table 4: Example Statement from Material Science: Phase change of water from liquid to gas.

Subject	Subject Value	Property	Property Value	Unit
Chemical	Water	boiling temperature	100	°C
Chemical	Water	pressure	1	atmosphere

Physics:

Table 5: Example Statement from Physics: Speed of light.

Subject	Subject Value	Property	Property Value	Unit
Boson	Light	speed	299 792 458	ms^{-1}
Boson	Light	medium	vacuum	

Independent properties make independent statements, as shown below.

Table 6: Example Statement from Physics: Mass of electron.

Subject	Subject Value	Property	Property Value	Unit
Fermion	Electron	Mass	$9.1093837015 \times 10^{-31}$	kg

Table 7: Example Statement from Physics: Charge of electron.

Subject	Subject Value	Property	Property Value	Unit
Fermion	Electron	Electric Charge	$-1.602176634 \times 10^{-19}$	C

Table 8: Example Statement from Physics: Charge of electron.

Subject	Subject Value	Property	Property Value	Unit
Fermion	Electron	Spin	1/2	h

C TED Similarity Score

C.1 Creating Trees

The `statement` data structure can be viewed in many representations: hypergraphs, tree, table, records, and transforming the representation of this data structure in other formats is trivial.

In our setup, when represented as a tree, all nodes in a `statement` has four attributes: *name*, *type*, *value*, and *parent*. We start a tree with the root node with name as '/root', type as 'root', and no value. This node does not have any parent node. Next, the statement nodes emerge as branches from the root. Each statement node has a name like '/root/s0' or '/root/s2' (here, 's' indicates that this is a statement node and the number acts as an index), type as 'statement', no value and the root node as its parent. Further, attached to each statement node are predicate node(s) with names like '/root/s1/p0' or '/root/s0/p3', type as 'predicate', no value and a statement node as its parent. Finally, in our current implementation, each predicate node has five children nodes attached to it. These leaf nodes can be of type: subject, subject-value, property, property-value, unit and the value attribute is populated with the actual value. The leaf nodes may have names like '/root/s2/p1/subject' or '/root/s0/p3/property-value'. In this representation, the name of a node completely determines the location of the node in a tree.

As an example, we show the tree structure for the statements shown in fig. 2:

```
Node('/root', type='root', value=None)
|-- Node('/root/s0', type='statement', value=None)
|   |-- Node('/root/s0/p0', type='predicate', value=None)
|   |   |-- Node('/root/s0/p0/Subject', type='Subject', value='Organization')
|   |   |-- Node('/root/s0/p0/Subject Value', type='Subject Value', value='XYZ')
|   |   |-- Node('/root/s0/p0/Property', type='Property', value='scope 1 emissions')
|   |   |-- Node('/root/s0/p0/Property Value', type='Property Value', value='3.3')
|   |   |-- Node('/root/s0/p0/Unit', type='Unit', value='million metric tons of CO2e')
|   |-- Node('/root/s0/p1', type='predicate', value=None)
|       |-- Node('/root/s0/p1/Subject', type='Subject', value='Organization')
|       |-- Node('/root/s0/p1/Subject Value', type='Subject Value', value='XYZ')
|       |-- Node('/root/s0/p1/Property', type='Property', value='time')
|       |-- Node('/root/s0/p1/Property Value', type='Property Value', value='2020')
|       |-- Node('/root/s0/p1/Unit', type='Unit', value='year')
|-- Node('/root/s1', type='statement', value=None)
    |-- Node('/root/s1/p0', type='predicate', value=None)
    |   |-- Node('/root/s1/p0/Subject', type='Subject', value='Organization')
    |   |-- Node('/root/s1/p0/Subject Value', type='Subject Value', value='XYZ')
    |   |-- Node('/root/s1/p0/Property', type='Property', value='scope 1 emissions')
    |   |-- Node('/root/s1/p0/Property Value', type='Property Value', value='2.5')
    |   |-- Node('/root/s1/p0/Unit', type='Unit', value='million metric tons of CO2e')
    |-- Node('/root/s1/p1', type='predicate', value=None)
        |-- Node('/root/s1/p1/Subject', type='Subject', value='Organization')
        |-- Node('/root/s1/p1/Subject Value', type='Subject Value', value='XYZ')
        |-- Node('/root/s1/p1/Property', type='Property', value='time')
        |-- Node('/root/s1/p1/Property Value', type='Property Value', value='2021')
        |-- Node('/root/s1/p1/Unit', type='Unit', value='year')
```

C.2 Computing Tree Similarity Score

For comparing two statement trees, we setup strict costs for each edit operation. The predictions are maximally punished for any structural deviation from the ground truth, i.e. deletion and insertion each have a cost of 1. For renaming of the node's value attribute, we only allow two nodes to be renamed if they are of the same type. If both nodes' value attribute is of type string, then we calculate a normalized Levenshtein edit distance between the two strings.

If both nodes' value attribute is of numerical type, then the two values are directly compared. In this case, the cost is 0 if the two values are the same, and 1 in all other cases. If the value attribute of both the

ground truth and the prediction node is empty, then the cost operation is also 0. We denote TED with t. We define normalized TED (nTED or $\bar{t}$) as the ratio of the distance to the number of edits between two trees. Using the normalized TED, a normalized Tree Similarity score can be computed as $t_s = 1 - \bar{t}$.

Consider comparing the trees for the two statements $s0$ and $s1$, from the example above. These two trees differ only in their numeric value but are otherwise similar to each other. Two edits are required to convert one tree into another: one corresponding to the property-value of 'time' and the other corresponding to the property-value of 'scope 1 emissions'. If the numeric values are interpreted as floats, then our strict setup will maximally punish for each edit giving an edit distance of 2 renaming, 0 deletions, and 0 insertions. The normalized tree edit distance (ratio of distance to total number of edits) would be 2 / 2 = 1. Thus, the TED similarity score would be 1 - 1 = 0.

However, our model outputs numeric values as strings, which can be compared via normalized Levenshtein distance. Then, the first rename edit of year values will give a distance of 1/4 = 0.25, and the other rename edit will give a distance of 2/3 = 0.66. In this case, the total tree edit distance is 0.9166, the normalized tree edit distance is 0.4583. This gives a TED similarity score of 0.54. We will interpret this by saying that "the two tree (when the numeric value are interpreted as strings) are 54% similar to each other". Given that the two trees are similar in their structure and only differ in their numeric values, this shows that our setup of TED similarity score is very strict.

For illustrative purposes, let us consider another example. We consider that the $s0$ in the above example is the ground truth statement:

```
Node('/root', type='root', value=None)
|-- Node('/root/s0', type='statement', value=None)
|   |-- Node('/root/s0/p0', type='predicate', value=None)
|   |   |-- Node('/root/s0/p0/subject', type='subject', value='Organization')
|   |   |-- Node('/root/s0/p0/subject_value', type='subject_value', value='XYZ')
|   |   |-- Node('/root/s0/p0/property', type='property', value='scope 1 emissions')
|   |   |-- Node('/root/s0/p0/property_value', type='property_value', value='3.3')
|   |   |-- Node('/root/s0/p0/unit', type='unit', value='million metric tons of CO2e')
|   |-- Node('/root/s0/p1', type='predicate', value=None)
|       |-- Node('/root/s0/p1/subject', type='subject', value='Organization')
|       |-- Node('/root/s0/p1/subject_value', type='subject_value', value='XYZ')
|       |-- Node('/root/s0/p1/property', type='property', value='time')
|       |-- Node('/root/s0/p1/property_value', type='property_value', value='2020')
|       |-- Node('/root/s0/p1/unit', type='unit', value='year')
```

And we have a model which makes the following prediction:

```
Node('/root', type='root', value=None)
|-- Node('/root/s1', type='statement', value=None)
    |-- Node('/root/s1/p0', type='predicate', value=None)
        |-- Node('/root/s1/p0/subject', type='subject', value='Organization')
        |-- Node('/root/s1/p0/subject_value', type='subject_value', value='XYZ')
        |-- Node('/root/s1/p0/property', type='property', value='scope 2 emissions')
        |-- Node('/root/s1/p0/property_value', type='property_value', value='3.3')
        |-- Node('/root/s1/p0/unit', type='unit', value='million metric tons of CO2e')
```

We observe that the predicted tree is missing an entire predicate with time property. This happens when models stop generating new tokens. Compared to the previous example, the ground truth and model prediction have a major structural deviation. In addition, the model also made a mistake in the value of the 'property' node. Instead of 'scope 1 emissions' as in ground truth, the model predicted 'scope 2 emissions'.

To convert one tree into another, we need a total of 7 edits: six nodes need to be deleted (or inserted) (5 leaf nodes and 1 predicate node) and 1 renaming edit. All deletions or insertions have equal score of 1 each, and the renaming costs 1/17 $\approx$ 0.0588. The total tree edit distance becomes 6.0588, the normalized tree edit distance is 0.8655. This gives us a tree similarity score of 0.1344. We interpret that the two trees are only 13% similar to each other.

D Baseline Experiments

Example of successful statement extraction:

Table 9: Table with simple layout from page 68 of the 2022 ESG report from Splunk Inc.

0	1	2
Emissions Scope	FY21	FY22
Scope 1 Direct Emissions	24	374
Scope 2 Indirect Emissions	3,686	3,257
Scope 3 Other Indirect Emissions	11,430	7,938
Total	15,140	11,569

Consider the above table, with a simple layout, from the 2022 ESG report of Splunk Inc.

We prompt Mixtral with the above table using the following prompt. For rendering, we replace our line-break token '
' with actual line-breaks and remove some aspect of the example statement for brevity.

```
<table>
| 0                                             |        1 |        2 |        3 |
|-----------------------------------------------|----------|----------|----------|
| Public Metrics                                | 2019     | 2020     | 2021     |
| Public Fatalities Due to Electrical Contacts  |    4     |    2     |    5     |
| Employee Metrics                              | 2019     | 2020     | 2021     |
| Employee DART Rate                            |    0.412 |    0.31  |    0.43  |
| Total Employee Recordable Incident Rate (TRIR) |    0.673 |    0.576 |    0.648 |
| Employee Lost Time Incident Rate (LTIR)       |    0.329 |    0.245 |    0.333 |
| Employee Severity Rate                        |   22.719 |   15.284 |   19.113 |
| Total Employee OSHA Recordable Events         |  129     |  106     |  113     |
| Employee Fatalities                           |    1     |    0     |    0     |
</table>
<response>
...
| property                                                  |   property_value | unit   | subject   | subject_value   |
|-----------------------------------------------------------|------------------|--------|-----------|-----------------|
| Employee Metrics : Employee Lost Time Incident Rate (LTIR) |            0.329 |        |           |                 |
| time                                                      |         2019     |        |           |                 |
<sep>
| property                                                  |   property_value | unit   | subject   | subject_value   |
|-----------------------------------------------------------|------------------|--------|-----------|-----------------|
| Employee Metrics : Employee Lost Time Incident Rate (LTIR) |            0.245 |        |           |                 |
| time                                                      |         2020     |        |           |                 |
<sep>
| property                                                  |   property_value | unit   | subject   | subject_value   |
|-----------------------------------------------------------|------------------|--------|-----------|-----------------|
| Employee Metrics : Employee Lost Time Incident Rate (LTIR) |            0.333 |        |           |                 |
| time                                                      |         2021     |        |           |                 |
<sep>
| property                                  |   property_value | unit   | subject   | subject_value   |
|-------------------------------------------|------------------|--------|-----------|-----------------|
| Employee Metrics : Employee Severity Rate |           22.719 |        |           |                 |
| time                                      |         2019     |        |           |                 |
<sep>
| property                                  |   property_value | unit   | subject   | subject_value   |
|-------------------------------------------|------------------|--------|-----------|-----------------|
| Employee Metrics : Employee Severity Rate |           15.284 |        |           |                 |
| time                                      |         2020     |        |           |                 |
<sep>
| property                                  |   property_value | unit   | subject   | subject_value   |
|-------------------------------------------|------------------|--------|-----------|-----------------|
| Employee Metrics : Employee Severity Rate |           19.113 |        |           |                 |
| time                                      |         2021     |        |           |                 |
...
</response>

<table>
| 0                                | 1       | 2       |
|----------------------------------|---------|---------|
| Emissions Scope                  | FY21    | FY22    |
| Scope 1 Direct Emissions         | 24      | 374     |
| Scope 2 Indirect Emissions       | 3,686   | 3,257   |
| Scope 3 Other Indirect Emissions | 11,430  | 7,938   |
| Total                            | 15,140  | 11,569  |
</table>
<response>
```

The model output for the above prompt with greedy decoding was:

```
| property                                                  |   property_value | unit   | subject   | subject_value   |
|-----------------------------------------------------------|------------------|--------|-----------|-----------------|
| Emissions Scope : Scope 1 Direct Emissions                |               24 |        |           |                 |
| time                                                      |             FY21 |        |           |                 |
<sep>
| property                                                  |   property_value | unit   | subject   | subject_value   |
|-----------------------------------------------------------|------------------|--------|-----------|-----------------|
| Emissions Scope : Scope 1 Direct Emissions                |              374 |        |           |                 |
| time                                                      |             FY22 |        |           |                 |
<sep>
| property                                                  |   property_value | unit   | subject   | subject_value   |
|-----------------------------------------------------------|------------------|--------|-----------|-----------------|
| Emissions Scope : Scope 2 Indirect Emissions              |            3,686 |        |           |                 |
| time                                                      |             FY21 |        |           |                 |
<sep>
| property                                                  |   property_value | unit   | subject   | subject_value   |
|-----------------------------------------------------------|------------------|--------|-----------|-----------------|
| Emissions Scope : Scope 2 Indirect Emissions              |            3,257 |        |           |                 |
| time                                                      |             FY22 |        |           |                 |
<sep>
| property                                                  |   property_value | unit   | subject   | subject_value   |
|-----------------------------------------------------------|------------------|--------|-----------|-----------------|
| Emissions Scope : Scope 3 Other Indirect Emissions        |           11,430 |        |           |                 |
| time                                                      |             FY21 |        |           |                 |
<sep>
| property                                                  |   property_value | unit   | subject   | subject_value   |
|-----------------------------------------------------------|------------------|--------|-----------|-----------------|
| Emissions Scope : Scope 3 Other Indirect Emissions        |            7,938 |        |           |                 |
| time                                                      |             FY22 |        |           |                 |
<sep>
| property                                                  |   property_value | unit   | subject   | subject_value   |
|-----------------------------------------------------------|------------------|--------|-----------|-----------------|
| Emissions Scope : Total                                   |           15,140 |        |           |                 |
| time                                                      |             FY21 |        |           |                 |
<sep>
| property                                                  |   property_value | unit   | subject   | subject_value   |
|-----------------------------------------------------------|------------------|--------|-----------|-----------------|
| Emissions Scope : Total                                   |           11,569 |        |           |                 |
| time                                                      |             FY22 |        |           |                 |
</response>
```

This is an example of correct statement extraction. For the same table with a different example in the prompt, the output of the same model was:

```
| property
```

This is an invalid output without any correct markdown structure or content. This shows that the in-context approach is sensitive to the prompt and thus is not robust.

E Algorithm for Statement Extraction

We present the algorithm we used to extract statements. For this algorithm, the inputs are the original table and the labels table.

Algorithm 1 Extract Statements

procedure EXTRACT STATEMENTS(Table, LabelsTable)
    Input: Table, LabelsTable: Table and Table of cell annotations
    AllStatements ← empty list
    for all row in LabelsTable **do**
        for all column in LabelsTable **do**
            if LabelsTable[row][column] = Property Value **then**
                Search in the same row and column for (Sub)-Property
                if Property is found **then**
                    Append Headers in hierarchy to Property, if any, starting from the minimum level
                    Construct Statement with Property, Row and Column
                else if SubProperty is found **then**
                    Append Property to the SubProperty
                    Append Headers in hierarchy to SubProperty, if any, starting from the maximum level
                    Construct Statement with SubProperty, Row and Column
                else
                    Property is not found, continue to the next iteration
                end if
                Append Statement to AllStatements
            end if
        end for
    end for
    Return AllStatements
end procedure

```
procedure CONSTRUCT STATEMENT(Row, Column, Property)
    Input: Row, Column, Property: Row and Column of the Property Value, with its related Property
    Output: Statement: list
    Statement ← empty list
    Predicate ← empty dictionary
    Predicate [Property Value] ← Table[Row][Column]
    Predicate [Property] ← Property
    Search in the same row and column(Unit Value)
    Predicate[Unit] ← Table[row_uv][column_uv]
    Search for a Subject - Subject Value pair
    Predicate[Subject] ← Table[row_s][column_s]
    Predicate[Subject_Value] ← Table[row_sv][column_sv]
    Add Predicate to the Statement
    Search in the same row and column(Time Value)
    if Time Value is found then
        Predicate ← empty dictionary
        Predicate [Property Value] ← Table[row_tv][column_tv]
        Predicate [Property] ← "Time"
        Add Predicate to the Statement
    end if
    Search for all Key - Key Value pairs
    for all Key - Key Value pairs found do
        Predicate ← empty dictionary
        Predicate[Property] ← Table[row_k][column_k]
        Predicate[Property Value] ← Table[row_kv][column_kv]
        Add Predicate to the Statement
    end for
    Return Statement
end procedure
```

Algorithm 2 Utility function for appending section header.

```
procedure APPEND HEADERS(Row, Column, Propery, Level)
    Input: Row, Column, Property, Level: Row, Column, value of a Property cell and the level of the header to search for.
    Output: Property: string
    for all Row_a above Row do
        for all Column_l on the left of Column do
            if LabelsTable[Row_a][Column_l] is a header with a higher level than Level then
                Append Table[Row_a][Column_l] on top of Property
                if the level of LabelsTable[Row_a][Column_l] is maximum then
                    Return Property
                else
                    Append Headers in hierarchy to Property starting from the level of LabelsTable[Row_a][Column_l]
                    Return Property
                end if
            end if
        end for
    end for
    Return Property
end procedure
```

Algorithm 3 Utility function for appending property name to sub-property

```
procedure APPEND PROPERTY(Row, Column, SubProperty)
    Input: Row, Column, SubProperty: Row,Column and Value of a SubProperty cell
    Output: Subproperty: string
    for all Row_a above Row do
        for all Column_l on the left of Column do
            if LabelsTable[Row_a][Column_l] is a Property then
                Append Table[Row_a][Column_l] on top of SubProperty
                Return SubProperty
            end if
        end for
    end for
    Return SubProperty
end procedure
```

Algorithm 4 Utility function to search for related predicates

procedure SEARCH IN THE SAME ROW AND COLUMN(Row, Column, Key)
  Input: Row, Column, Key: Row and Column where to search the specified Key
  Output: Row_k, Column_k: Row and column of the designated Key, if found
  for all Cell respectively on the Left, Above, and Right to the cell at LabelsTable[Row][Column] **do**
    if Cell is Key **then**
      Return Row, Column of Cell
    end if
  end for
  Return Null
end procedure

Algorithm 5 Utility function for searching corresponding key-value.

procedure SEARCH FOR A PAIR(Row, Column, Key, Key Value)
  Input: Row, Column, Key: Row and Column where to search the specified Key
  Output: Row_k, Column_k: Row and column of the designated Key, if found
  for all Cell_{kv} respectively on the Left, Above, and Right to the cell at LabelsTable[Row][Column] **do**
    if Cell_{kv} is Key Value **then**
      for all Cell_k in the Orthogonal Direction with respect to Cell_{kv} from LabelsTable[Row][Column] **do**
        if Cell_k is Key **then**
          Return Coordinates of Cell_k, Cell_{kv}
        end if
      end for
    end if
  end for
  Return Null
end procedure

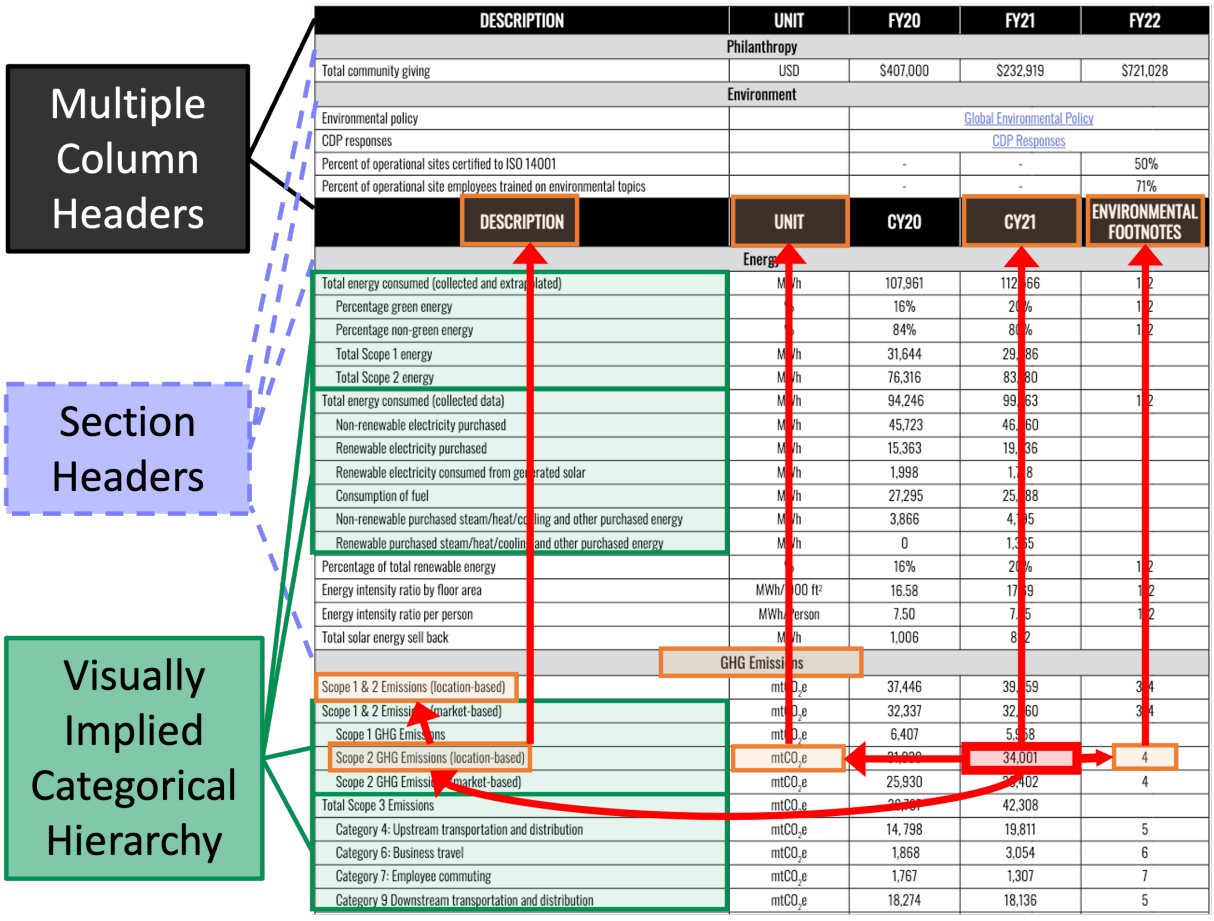

Figure 5: Example table from an ESG report with a complicated layout. To extract the information content of a single cell (highlighted in red), the content and relationships (lines drawn in red) to many other cells (highlighted in orange) also needs to be understood.