RESEARCH-ARTICLE

stEELlm: An LLM for Generating Semantic Annotations of Tabular Data

MARCO CREMASCHI, University of Milan, Milan, MI, Italy

FABIO D'ADDA, University of Milan, Milan, MI, Italy

ANDREA MAURINO, University of Milan, Milan, MI, Italy

Open Access Support provided by:

University of Milan

Accepted: 01 February 2025
Revised: 10 December 2024
Received: 17 February 2024

stEELlm: An LLM for Generating Semantic Annotations of Tabular Data

MARCO CREMASCHI, FABIO D'ADDA, and ANDREA MAURINO, University of Milan-Bicocca, Italy

The capabilities of LLMs represent a pivotal step in transforming how we manage and interact with information and data. We witness an increasingly pervasive use of such models in various computational tasks. In some preliminary works, attempts to integrate Knowledge Graphs and Large Language Models (LLMs) can be identified, in particular, to perform the classic tasks related to the construction of Knowledge Graphs through semantic annotation of texts. Nowadays, tables are widely used and play a crucial role in creating, organising, and sharing information that could be used to produce factual knowledge to be integrated into a Knowledge Graph. However, table-to-KG techniques through LLM have not been extensively investigated. This paper presents stEELlm, an innovative Semantic Table Interpretation approach obtained by fine-tuning the Mixtral 8x7B model. Conducted experiments demonstrate the capabilities of our model to successfully create semantic annotations of heterogeneous datasets, a scenario where classic approaches based on heuristics tend to fail.

CCS Concepts: • **Computing methodologies** → **Learning paradigms**; **Semantic networks**; **Machine learning**.

Additional Key Words and Phrases: Large Language Models, Knowledge Graphs, Pre-training, Fine-tuning, Prompt Engineering, Semantic Table Interpretation

1 INTRODUCTION

The rise and widespread popularity of generative Artificial Intelligence (AI) led to a new era in the digital world. The advent of platforms like Dall-E[1] and ChatGPT[2], developed by OpenAI in 2022, marked a pivotal moment in society, igniting curiosity with their ability to artificially generate texts, videos, and images based on textual inputs provided by the user. Notably, supplying ChatGPT with a reasonably detailed request called "prompt" can generate a highly accurate and coherent text. However, beyond the innovative features of such platforms, what captured global attention was their accessibility, user-friendly interface and the calibre of the generated output, making them an incredible tool across various domains.

The release on March 14, 2023 of GPT-4, which is OpenAI's new version of their Large Language Model (LLM), marked a significant change in the AI landscape. According to OpenAI, while GPT-4, is recognised to be less proficient than humans in many real-world scenarios, it demonstrates human-level performance across a wide range of professional and academic tasks. In fact, GPT's performance scores in various fields, including macroeconomics, writing, maths, and oenology, rank prominently in at least 34 tests of ability [76]. Notably,

[1]OpenAI: DALL·E - openai.com/index/dall-e-2/
[2]OpenAI: ChatGPT - chatgpt.com

Authors' address: Marco Cremaschi, marco.cremaschi@unimib.it; Fabio D'Adda, fabio.dadda@unimib.it; Andrea Maurino, andrea.maurino@unimib.it, University of Milan-Bicocca, viale Sarca, 336 - Edificio U14, Milan, Italy, Italy, 20126.

ACM 2157-6912/2025/2-ART
https://doi.org/10.1145/3719206

GPT-4 achieved scores within the top 10% of test-takers in a simulated bar exam[3]. This success garnered massive attention, revealing the incredible possibilities of this product.

We are witnessing an increasingly pervasive use of LLM in various computational tasks [88]. Despite their success in many applications, LLMs have been criticised for lacking factual knowledge. Specifically, LLMs memorise facts and knowledge in the training corpus [83]. Further studies reveal that LLMs are not able to recall facts and often experience hallucinations by generating statements that are factually incorrect [45, 105]. Additionally, LLMs use a probability model for reasoning, which can be indecisive [109]. This severely limits the use of LLMs in high-stakes scenarios, such as medical diagnosis or legal judgment. A potential solution to address the above issues is incorporating Knowledge Graphs (KGs) into LLMs [5].

KGs, storing vast amounts of data as triples, constitute a structured and effective method of knowledge representation [80]. The possibility of combining LLMs with KGs has attracted increasing attention among researchers and practitioners. LLMs and KGs are inherently interconnected and can mutually enhance each other. For example, in *LLM-augmented KGs*, LLMs have been used in various Knowledge Graph (KG) related tasks, including KG embedding [101], KG completion [107], KG construction [23, 24, 66], KG-to-text generation [53], and KGQA [47], to improve the performance and facilitate the application of KGs.

Attempts to integrate KGs and LLMs can be found in some preliminary works [96, 108, 114][4]. They propose methods to perform the classic tasks related to the construction of KG through **text-to-KG** generation, such as entity discovery, co-reference resolution and relation extraction [80]. In this scenario, the goal is to create semantic annotations on text to generate new structured knowledge.

Although the state-of-the-art (SOTA) includes several contributions on **text-to-KG** techniques, **table-to-KG** approaches through LLM remain relatively unexplored. Tables are widely used and play a crucial role in creating, organising, and sharing information. They have long been essential in both business and scientific sectors, and their usage as a format for publishing information on the web has grown further with the uptake of the Open Data movement. As a result, many tabular data sources are published online, covering a wide range of domains such as finance, mobility, tourism, sports, or cultural heritage [71]. The relevance and diversity of tabular data can be assessed by looking at the number of available tables and/or users of tabular data manipulation tools:

- Web Tables: in 2008, 14.1 billion HTML tables were extracted, and it was found that 154 million were high-quality tables (1.1%). In the Common Crawl 2015 repository, there are 233 million content tables[5];
- Wikipedia Tables: the 2022 English snapshot of Wikipedia contains 2 803 424 tables from 21 149 260 articles [65];
- Spreadsheets: there are 750 million to 2 billion people in the world who use either Google Sheets or Microsoft Excel[6].

These statistics evidence the importance of tables as a source of information. Despite the simplicity of tabular data, understanding its meaning remains a challenge [26]. In this scenario, the **table-to-KG** matching problem, also referred to as Semantic Table Interpretation (STI), has recently collected considerable attention in the research communities like the Semantic Web, Data Management, AI, and Natural Language Processing (NLP) ones [28, 36, 49, 51] and is a key step to enrich data [29, 78] and construct and extend KGs from semi-structured data [54, 103].

[3]Forbes: GPT-4 Beats 90% Of Lawyers Trying To Pass The Bar - www.forbes.com/sites/johnkoetsier/2023/03/14/gpt-4-beats-90-of-lawyers-trying-to-pass-the-bar/

[4]GitHub: Awesome-LLM-KG - github.com/RManLuo/Awesome-LLM-KG

[5]Commoncrawl - commoncrawl.org

[6]Askwonder: Number of Google Sheets and Excel Users Worldwide - askwonder.com/research/number-google-sheets-users-worldwide-eoskdoxav

The growing interest is proved by the international *Semantic Web Challenge on Tabular Data to KG Matching* (SemTab), an annual challenge proposed since 2018[7] [28, 49, 51]. Over the years, several approaches, datasets, and related Gold Standards (GSs) on the topic of STI have been released.

In the SOTA, it is possible to identify different conceptualisations of the table annotation problem. Considering the formalisation proposed in the SemTab challenge as one of the formulations that has obtained the greatest consensus, we can define the STI as follows.

Given:

- a relational table T (Fig. 1);
- a Knowledge Graph (entities + statements) and an ontology (types + properties) (Fig. 2).

T is annotated when:

- each column is associated with one or more KG-types [Column-Type Annotation (CTA) task] (*e.g.*, the column *Name* in the Fig. 1 is annotated with the type `Mountain`; the column *Height* is annotated with datatype `xsd:integer`);
- each cell in "entity columns" is annotated with a KG-entity or with *Not In Lexicon (NIL)* (if it is not in the KG) [Cell-Entity Annotation (CEA) task] (*e.g.*, the cell *Le Mount Blanc* is annotated with `Mount_Blanc`; the cell *Hohtälli* is annotated with *NIL* since it has not yet been included in the KG);
- some pair of columns are annotated with a binary KG-property [Columns-Property Annotation (CPA) task] (*e.g.*, the pair composed by the columns *Name* and *Height* is annotated with `dbo:elevation`).

Approaches in the SOTA include a first step where candidate entities for the input string are collected, *i.e.*, Entity Retrieval (ER) [86], and a second step where the mention is disambiguated by eventually selecting one or none of the candidate entities, *i.e.*, Entity Disambiguation (ED) [85]. The result of the annotation process can be seen in Fig. 3. The annotations can generate new elements within the KG (*i.e.*, the entity `Hohtälli`). This conceptualisation covers most of the proposed definitions by generalising some aspects, such as the selection of column pairs for the CPA and the inclusion of NIL—strings that refer to entities without existing representations in the KG.

An in-depth analysis of the SOTA shows how current STI approaches are tailored to specific datasets (Gold Standards) to have optimal performances. As an example, considering the winning approaches of the various SemTab challenge editions, such as MTab (winner in 2019 with [72] and in 2020 with MTab4Wikidata [73]), Dagobah (winner in 2021 with [42] and in 2022 with [40]), KGCODE-Tab [59] (winner in 2022), and TorchicTab [30] (winner in 2023), it can be deduced how these were built to specifically work on the different datasets used in the corresponding editions of the SemTab challenge (Section 8). At the same time, by analysing the different Gold Standards, it is possible to see how these approaches, even the most recent ones, are designed for one or two specific datasets[8]. An experiment was conducted using Dagobah [42], as described in Section 8.

However, few "exploratory investigations" have attempted to tackle STI tasks by leveraging LLMs. One notable example is TabLLM [38], which introduces a methodology wherein the LLM is prompted by serialising a row in natural language and a concise classification problem description. In [63], the researchers delve into the challenges that LLMs encounter in comprehending and reasoning over tabular data. The authors specifically address the unique hurdles posed by the structural nature of tables, emphasising the impact on LLMs ability to perform complex statistical analyses and to interpret and localise information precisely. Other approaches [41, 56, 82] employ LLMs to perform tasks more closely related to the SemTab challenge. Specifically, [56] focuses solely on the CTA task, annotating columns by providing a set of types to the model alongside a table. In contrast, [82] undertakes a kind of Entity Linking (EL) task, identifying descriptions in diverse data sources that correspond to

[7] SemTab Challenge - cs.ox.ac.uk/isg/challenges/sem-tab/
[8] STI resources: datasets - unimib-datai.github.io/sti-website/datasets/

the same real-world entity. Lastly, in [41], the authors investigate the utilisation of LLMs across different STI tasks.

These studies can be considered as "preliminary explorations" since they focus on specific sub-issues linked to STI rather than the main STI tasks depicted above. Only Dagobah [41] tries to devise a solution capable of addressing the underlying challenges of semantic annotation tasks using two types of prompts: a textual and a pseudocode. Despite promising premises, the authors do not provide numerical results regarding the performance of the CEA task. A more concrete investigation on how Machine Learning (ML) and LLM based approach has been conducted by [12].

In the domain of STI, and within the broader context of annotation generation tasks, the primary objective transcends the simple identification of word sequences, an area in which generative models are proficient. Instead, the primary purpose is to create nuanced representations of data, facilitating the disambiguation process, which is fundamental for discerning entities (to obtain CEA) to enrich such data semantically.

Given this premise, we identified the following research questions: i) Are LLMs a "tool" capable of performing CEA task? ii) How can they be adapted to perform this task?

To address them, the main contributions of this paper are:

(1) **Creation of a Gold Standard (GS)**: This research presents a dataset of tables with well-structured semantic annotations for the CEA task, focusing on entity disambiguation. The dataset information can also be used to create prompts for fine-tuning or testing a language model;
(2) **Prompt engineering for STI**: we provide a comprehensive analysis of different prompts for CEA, aimed at identifying the structure that maximises the quality of the annotations;
(3) **Fine-tuning for STI**: we perform a fine-tuning of the Mixtral 8x7B model for CEA.

These contributions led to the creation of stEELlm (Semantic TablE IntErpretation LLM), an innovative STI approach focused on the CEA task, obtained by fine-tuning the Mixtral 8x7B model. The stEELlm approach, its repository[9] and documentation[10] are publicly available. stEELlm is released under the GNU Affero General Public License[11].

This paper is organised as follows: Section 2 formally defines the STI process and summarises its related challenges. Section 3 proposes an examination of the different techniques used by STI approaches in the SOTA. Section 4 describes the creation of a dataset for training and fine-tuning the LLMs for the semantic annotation of tabular data. Section 5 illustrates how the proposed dataset was used to create prompts. Section 6 introduces an in-depth study of prompts outputs using GPT models for STI purposes. Section 7 defines a strategy to adapt LLMs to solve STI. Section 8 shows the results obtained in both prompts and fine-tuning. The paper concludes, followed by a discussion of future developments in Section 9.

2 SEMANTIC TABLE INTERPRETATION

This Section formally describes the task we aim to solve and outlines the key challenges associated with STI tasks. The input of STI is i) a *well-formed and normalised* relational table (*i.e.*, a table with headers and cells filled with string values, which are referred to as *mentions* in this paper), and ii) a reference *KG* describing real-world entities in the domain of interest (*i.e.*, a set of concepts, datatypes, properties, instances, and relationships among them). The two inputs are represented in Fig. 1 and Fig. 2. The returned output is a semantically annotated table, as illustrated in Fig. 3.

[9]GitHub: stEELlm repository - github.com/unimib-datAI/steellm/
[10]GitHub: stEELlm documentation - github.com/unimib-datAI/steellm-docs/
[11]GNU: GNU Affero General Public License - www.gnu.org/licenses/agpl-3.0.en.html

Name	Coordinates	Height	Range
Le Mout Blanc	45° 49′ 57″ N, 06° 51′ 52″ E	4808	M. Blanc massif
Hohtälli	45° 59′ 20″ N, 7° 48′ 10″ E	3275	Pennine Alps
Monte Cervino	45° 58′ 35″ N, 07° 39′ 31″ E	4478	Pennine Alps

Input Data

Fig. 1. Example of a well-formed relational table.

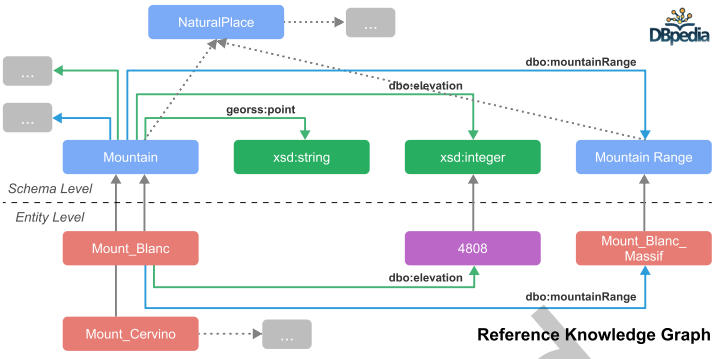

Fig. 2. A sample of a Knowledge Graph.

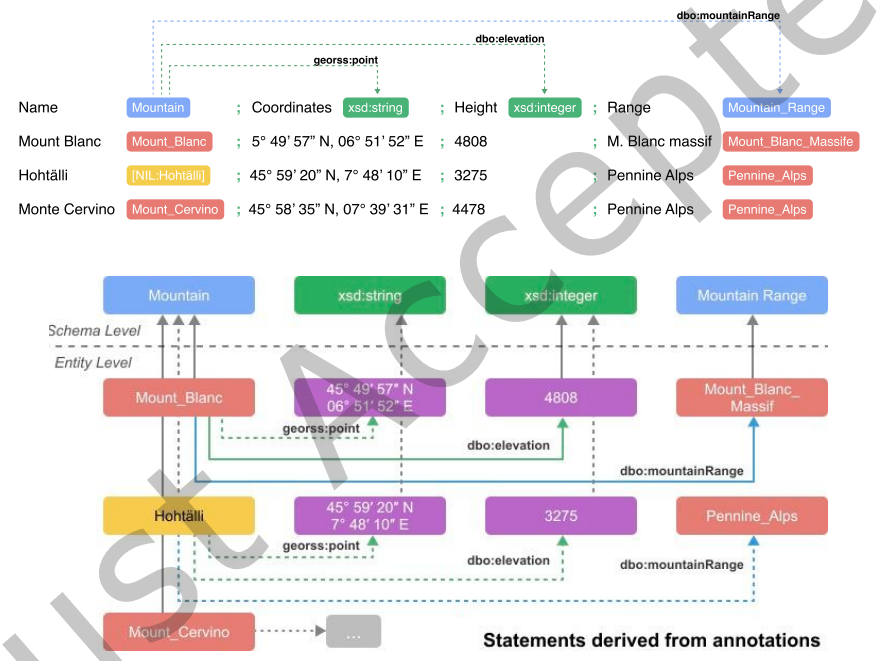

Fig. 3. The Table in Fig. 1 annotated with the KG.

The STI process is typically divided into the following four tasks:

- Column-Type Annotation (CTA): it concerns the prediction of the semantic types (*i.e.*, KG classes) for every column in a table;
- Columns-Property Annotation (CPA): it concerns the prediction of semantic properties (*i.e.*, KG properties) that represent the relationship between some pair of columns;
- Cell-Entity Annotation (CEA): it aims to predict the entity (*i.e.*, instance) represented by a cell in the table;
- Cell-New Entity Annotation (CNEA): it aims to predict which cell in the table represents an entity that does not occur in the KG and should therefore be labelled as NIL.

An ideal STI approach must consider several aspects that need to be taken into consideration to extract accurate annotations. These aspects pertain to all the tasks performed in a STI process. As previously described, this article focuses solely on the CEA. For this reason, we have identified the main challenges associated with this task, which will be addressed in this article:

(1) *Detecting the type of columns*: in a table, there can be columns that contain references to name entities (*NE-column*), known as entity mentions, and columns that contain numbers, dates, and in general, values that are instances of specific data types, which we refer to as literals (LIT-column). Distinguishing between the two types of columns is crucial to support the annotation process;
(2) *Matching tabular values against the KG*: matching the values extracted from the table with the data in the KG is a typical approach for collecting evidence to interpret the table. The mention in a table often differs from the entity's label in a KG (*i.e.*, use of acronyms, aliases, and typos). For instance, *High Tauern* in a table refers to the *Johannisberg mountain (High Tauern)* entity in DBpedia[12]; literal values, *e.g.*, height of the mountains, may vary because of different factors, which include outdated information, different measuring methodologies and so on;
(3) *Disambiguation of named entities*: the KG may contain many entities with similar or even identical names (homonyms). Furthermore, they may pertain to different or the same concept within the KG. For example, the mention Mont_Blanc in Fig. 1, which indicates the famous mountain located on the French-Italian border[13], can match labels of different entities associated with various types, including a tunnel, a poem, and a dessert[14]. Additionally, the same mention can match the label of a mountain on the Moon[15].

3 RELATED WORK

Numerous attempts to leverage LLM on tables can be found in the SOTA [26], with efforts falling into two main categories that can be associated with STI tasks: i) **table understanding** and ii) **table annotation** [74, 90]. The first group, i) **table understanding**, involves gaining insights, knowledge, and comprehension of tabular data's structure, content, and patterns. The ultimate objective of tabular understanding is to grasp the nature of the data, which can be crucial for making informed decisions, building models, or extracting meaningful information. The second group, ii) **table annotation**, refers to the process of clarifying the semantic meaning of a table by mapping its elements (*i.e.*, cells, columns, rows) to semantic tags (*i.e.*, entities, classes, properties) from KGs (*e.g.*, Wikidata, DBpedia) [1].

In the first group, i) **table understanding**, related work includes research conducted by [63], which highlights challenges in understanding tabular data by using LLMs. The study explores three main aspects: i) the resilience of LLMs to changes in table structure, ii) a comparison between textual and symbolic reasoning on tables, and iii) the potential improvement in model performance by combining multiple reasoning approaches. The research reveals a significant decline in performance, especially in symbolic reasoning tasks, when tables with the same content exhibit structural variations. This study on STI is valuable as it indicates how to represent tables to an LLM.

In the second group, ii) **table annotation**, multiple STI approaches can be identified in the current SOTA (more than 100)[16]. These approaches employ different techniques to retrieve and disambiguate entities for annotation.

Three main methods can be identified for **entity retrieval**: a) ***custom index***, b) ***external lookup*** services, and c) ***hybrid***. Custom index refers to building a specialised index for specific requirements or use cases. When building a custom index, there is flexibility in defining mappings, analysers, and other configurations based on

[12] dbpedia.org/page/Johannisberg_(High_Tauern)
[13] dbpedia.org/page/Mont_Blanc
[14] en.wikipedia.org/wiki/Mont_Blanc_(disambiguation)
[15] dbpedia.org/page/Mont_Blanc_(Moon)
[16] STI resources: approaches - unimib-datai.github.io/sti-website/approaches/

specific needs [6, 14–17, 19, 21, 33, 34, 40, 42, 43, 57, 61, 67, 73–75, 94, 95, 106]. ***External lookup*** services use a separate service or system to perform lookups or queries to retrieve specific information or data. The external lookup service usually employs entity recognition, entity disambiguation, or semantic matching techniques. It may consider factors like textual similarity, context, or other relevant information [1, 3, 4, 10, 11, 13, 14, 16, 20, 22, 31, 35, 55, 68–70, 72, 87, 89, 91, 92, 95, 99, 100, 106, 111–113]. ***Hybrid*** methods use both custom index and external lookup services [14, 16, 95, 106].

Regarding **disambiguation**, a greater variation in the techniques used can be found. We can identify approaches that employ: a) ***embedding***, b) ***similarity measures***, c) ***contextual information***, d) ***ML techniques***, e) ***probabilistic models***, and f) ***language models***. In graphs and natural language, the concept of ***embedding*** refers to the representation of nodes in a graph, or words in a text, as dense vectors in a continuous vector space. These embeddings capture semantic and structural relationships between nodes or words. Some approaches use embedding techniques to create vector representations of entities [14, 33, 35, 62, 91, 111, 112]. The entity disambiguation sub-task could involve the computation of some ***similarities*** among textual data. This type of score is usually adopted by lookup services to retrieve a ranked list of candidates. The disambiguation step often involves the selection of the winning candidate by considering the string similarity between the entity label and the mention. Some approaches use similarity, such as Levenshtein distance, Jaccard similarity, Cosine similarity, and similarity based on RegEx [4, 6, 17, 19–21, 21, 22, 25, 42, 43, 57, 61, 75, 89, 92, 95, 99, 113]. ***Contextual information*** during the CEA task considers the surrounding context of a table cell, such as neighbouring cells, column headers, or header row. Moreover, it provides additional clues or hints about the meaning and intent of the mention. By analysing the context, a system can better understand the semantics of the cell and make more accurate annotations [3, 6, 8, 10, 11, 15–17, 19, 21, 33, 40, 42, 43, 62, 67, 72–74, 87, 94, 95]. Other methods employed are ***ML techniques***. These techniques typically involve training an ML model on a labelled dataset where cells are annotated with their corresponding entities. The model learns patterns and relationships between the cells' content and associated entities. To predict the most appropriate entity, ML techniques consider various cell features, such as the textual content, context, neighbouring cells, and other relevant information [8, 69, 95, 112]. **Probabilistic models** are frameworks for representing and reasoning under uncertainty using probability theory. These models vary in their representation of dependencies and use diverse graphical structures [13, 57, 68, 70, 106].

The advent of ***LLMs***, whose timeline is displayed in Fig. 4, has led to a new category of promising approaches for table annotation. Based on the architecture of LLMs, these approaches can be categorised into three groups: a) ***encoder-decoder*** LLMs, b) ***encoder-only*** LLMs, and c) ***decoder-only*** LLMs [81]. Indeed, shortly after the first edition of SemTab, some works [31, 60, 93] applied encoder-only LLMs to table interpretation. Although they did not participate in or compare with the SemTab challenge, they created a different experimental setting. During the SemTab2022 challenge, a BERT-based [32] model was combined with a more traditional approach [40]. More recently, after the release of GPT-3.5 [77] as well as open-source decoder-only LLMs such as LLAMA [97] and LLAMA 2 [98], some works began applying encoder-based LLMs to table interpretation [58, 109]. In SemTab2023, a new decoder-only model utilising BERT was presented [30]. Starting from encoder-based approaches, Ditto [60] utilises Transformer-based language models to perform a slightly different task; the goal is entity-matching between different tables. TURL [31] leverages a pre-trained TinyBERT [48] model to initialise a structure-aware Transformer encoder. Doduo [93] performs CTA using a pre-trained language model, precisely fine-tuning the BERT model on serialised tabular data. Dagobah SL 2022 [40] employs an ELECTRA-based cross-encoder, a variant of the BERT model. The Cross Encoder takes a concatenated input, including left-side table headers, the target table header, right-side table headers, and the entity description. TorchicTab [30] is composed of two sub-systems: TorchicTab-Heuristic and TorchicTab-Classification. The classification model utilises Doduo [93]. Regarding decoder-based approaches, TableGPT [58] performs several tasks, including entity linking using GPT. TableLlama [109] performs CEA, along with several other tasks, creating a multi-task dataset for tabular data in which the entity linking sub-dataset derives from the TURL [31] dataset, and using it to fine-tune LLama2 [98].

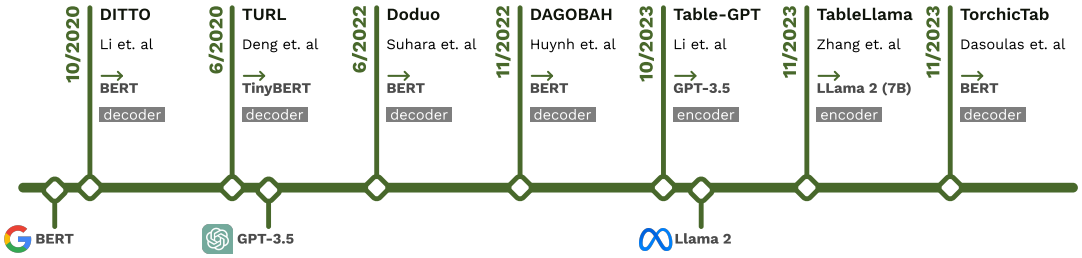

Fig. 4. Timeline of STI approaches based on Transformers.

Building on the insights presented in these last approaches, our investigation delves into the challenges faced by LLMs in effectively handling tabular data across various STI tasks. Specifically, we direct our focus towards the adaptation of LLMs to address the issues posed by SemTab challenge tasks. As explained in Section 2, these are: i) *detecting the type of columns*, ii) *matching tabular values against the KG*, and iii) *disambiguation of named entities*.

While some early work on STI focused on annotating table attributes (*i.e.*, the headers of the table) [110], most approaches today start from CEA and then, from its results, derive the CTA and CPA using heuristic-based algorithms. For example, CTA and CPA can be performed by counting the frequency of the types for the entities in the same column or the frequency of properties among entities in the same row. For this reason, our experiments will specifically explore the task of CEA, which, as indicated in Section 2, involves the resolution of several challenges.

4 CREATION OF THE DATASET (GOLD STANDARD) FOR PROMPTING AND FINE-TUNING

The dataset we created to be used as a Gold Standard for prompting and fine-tuning an LLM on the CEA is built using some datasets from the SemTab challenges. Particularly, it includes:

Table 1. Statistics of the datasets used to select prompts and fine-tune Language Model (LM).

GS		Tables	Cols (min | max | $\bar{x}$)	Rows (min | max | $\bar{x}$)	Classes	Entities	Pred.	KG
WikidataTables 2023 R1 [37]		500	1,2K (2 | 4 | 2,46)	2,9K (2 | 10 | 5,952)	194	3,6K	177	Wikidata
SemTab 2022 [2]	**R1**	3,8K	9,9K (2 | 5 | 2,56)	22,4K (4 | 8 | 5,69)	240	1,4K	319K	Wikidata
	R2	5,1K	13,3K (2 | 5 | 2,56)	28,5K (4 | 8 | 5,57)	398	1,9M	348	
SemTab2020 [52]	**R1**	34,3K	170K (4 | 8 | 4,96)	249K (5 | 16 | 8,27)	136K	985K	136K	Wikidata
	R2	12,1K	55,9K (4 | 8 | 4,6)	84,9K (5 | 16 | 7,97)	438K	283K	43,8K	
	R3	62,6K	229K (3 | 7 | 3,66)	397K (3 | 16 | 7,34)	167K	768K	167K	
	R4	22,4K	79,6K (1 | 8 | 3,55)	670K (6 | 15,5K | 30,94)	32,5K	1,7M	56,5K	

- **WikidataTables 2023** [37]: a dataset of tables generated using SPARQL queries designed to create realistic-looking tables. It includes both *Test* and *Validation* tables, yet we exclusively employed the *Validation* tables as *Test* tables are not annotated. The target KG for this dataset was Wikidata, and the considered tasks were CEA, CTA, and CPA;

- **HardTables 2022** [2]: a dataset with tables generated using SPARQL queries [50]. The target KG for this dataset was Wikidata, and the considered tasks were CEA, CTA, and CPA;
- **SemTab 2020** [52]: SemTab 2020 was organised into four evaluation rounds, each strategically designed to incrementally increase the complexity and difficulty of the employed datasets. The initial three rounds were conducted with the assistance of AIcrowd[17], which enabled automated evaluation of the submitted solutions using an automatic dataset validator. In contrast, Round 4 was a blind round, meaning there was no AIcrowd evaluation. It featured a combination of two elements: i) an automatically generated dataset, similar to previous rounds' datasets, and ii) the Tough Tables dataset for CEA and CTA [27].

Table 1 reports statistics regarding the listed datasets. These datasets were selected to efficiently capture the salient features, patterns, and challenges related to the STI tasks. The diversity of the scenarios they cover allows the effective generalisation of the model to unfamiliar data and permits it to handle various situations. While conventional ML datasets typically aim for error-free data, detecting and managing misspelt data becomes crucial in this context, where tables may contain errors.

Hence, our Gold Standard dataset includes tables extracted from *WikidataTables 2023*, *HardTables 2022*, and *SemTab 2020*, amounting to a total of 140800 tables distributed among training, test, and validation sets with proportions of 70%, 20%, and 10%, respectively. The datasets for training (102910 tables) is publicly available in the GitHub repository[18] and testing (14000 tables)[19] are publicly available.

All the tables are in *csv* (Comma-Separated Values) format, which is characterised by a straightforward structure where each row represents a record and columns are separated by commas, as shown in Listing 1. This format allows widespread support and compatibility with various applications and programming languages.

Listing 1. Example of Table in csv format.

```
Mountain Name,Altitude (m),Location,Mountain Range
Mont Blanc,4810,Alps,Graian Alps
K2,8611,Himalayas,Karakoram
Kilimanjaro,5895,Tanzania,East African Mountains
Mount Everest,8848,Himalayas,Himalayas
```

As described in Section 2, the required input for a STI task includes a table (in *csv* format) and the knowledge related to the KG.

Creating a custom dataset is essential for structuring the desired responses, as it serves as a benchmark for evaluating the performance of the LLM model under test. In this case, two distinct fine-tuning approaches have been employed: i) the first method evaluates the model's ability to perform both ER and ED by excluding candidates from the prompts, and ii) the second method focuses specifically on the ED task by including candidates in the prompts. The dataset construction process, illustrated in Fig. 5, allows the generation of those two prompts.

The dataset construction involved the following steps: i) **Data Preprocessing**, ii) **Lookup**, and iii) **Candidate Generation**. The pipeline for the generation of the dataset is publicly available[20].

The first step, i) **Data Preprocessing**, was the loading of a) the GS of each dataset into a Python dictionary to provide direct access to all Wikidata annotations when building dataset instances, and b) the tables in *csv* format from the three datasets, employing the *csv* library available in Python. Following the data loading, the preprocessing phase focused on improving the data quality, making it more accurate, reliable, and suitable for annotation. Specifically, three cleaning strategies were applied:

(1) **Remove Wikidata URL**: the GS stores each annotation in the format

TABLE_ID, ROW, COLUMN, http://www.wikidata.org/entity/Q84

[17] AIcrowd - www.aicrowd.com

[18] stEELlm: training data - github.com/unimib-datAI/steellm/tree/main/datasets/dataset_train

[19] stEELlm: test data - github.com/unimib-datAI/steellm/tree/main/datasets/dataset_test

[20] GitHub: repository Git of the pipeline for prompt engineering - github.com/unimib-datAI/steellm

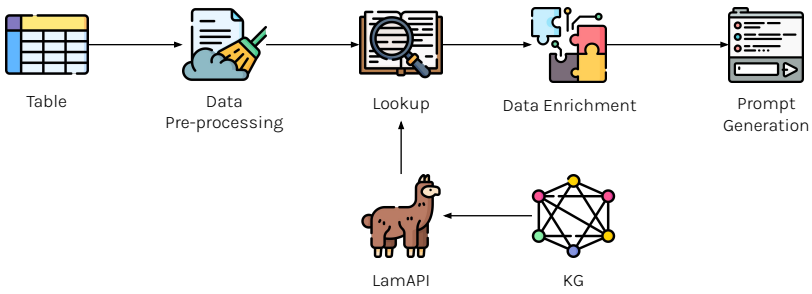

Fig. 5. Process of dataset creation.

but the entity id prefix (http://www.wikidata.org/entity/) is not relevant and can be removed;

(2) **Remove Unicode Characters**: removing Unicode characters, such as "0", is essential to prevent potential confusion for the model during the inference process. This process was implemented using *regular expressions*[21] patterns to efficiently remove extraneous text;

(3) **Lowercasing**: converting all text to lowercase, using string built-in functions, ensures coherence and eliminates variations due to capitalisation.

The second step in the dataset construction is the ii) **Lookup**, which consisted of retrieving a set of entities from LamAPI[22] [7], that is an ER system developed for querying and filtering entities in a KG by applying complex string matching algorithms. Using LamAPI's lookup service, accessible through the endpoint */lookup/entity-retrieval*, a set of candidates (Wikidata entities) is obtained for each mention to be annotated.

The third step, iii) **Candidate Generation**, combines and unifies previously cleaned tables with information extracted from LamAPI. Since the ultimate task is the execution of CEA, the objective of this step was to enrich the prompt with additional information to improve the LLM's ability to recognise and disambiguate entities in the candidate set. This enhancement involves associating entity labels with nearby Wikidata IDs (*e.g.*, (1,0) = Q108195506 [fidelidade seguros]). Such information is crucial as LLMs are particularly adept at tasks centred around natural language processing. Incorporating the entity label and entity ID helps the LLM in its reasoning process, enabling it to cultivate a profound comprehension of the contextual associations between Wikidata IDs and entity labels. Given that LLMs operate as generative models, their capacity to accurately generate the appropriate Wikidata ID during inference is enhanced.

5 PROMPT ENGINEERING

As previously indicated, the proposed Gold Standard was also used to generate prompts for fine-tuning an LLM on the CEA. The information in the dataset is organised to ensure comprehensive data inclusion for prompt generation during model training. Once the data is saved, a prompt is generated for each table, and it is successively fed to the LLM.

Listing 2. Dataset instance example.

```
{
  "dataset": "hardtables_2022_r1",
  "table": "WWQXGBG9",
  "instruction": "given a table T, it is possible to identify each of its elements through ...",
  "input": "fidelidade;portugal|nzi;new zealand|de centrale;netherlands|cigna;taiwan| ...",
  "output": "(1,0)=Q108195506 [fidelidade seguros];(1,1)=Q45 [portugal]; ...",
  "entities": "Q108195506 [fidelidade seguros]|Q67204954 [a fidelidade]| ..."
}
```

[21] Python Docs: Re - docs.python.org/3/library/re.html
[22] LamAPI: Swagger description - lamapi.datai.disco.unimib.it

The final dataset is in JSON format and contains the following fields:

- **Dataset**: the table is sourced from a dataset recognised by its name, a crucial piece of information given the integration of multiple tables;
- **Table**: the dataset's table ID is employed to extract the GS from the original dataset;
- **Instruction**: the instruction is used to train the model, therefore guiding it in determining the specific task it has to perform;
- **Input**: the *csv*-formatted table employs vertical bars ("|") as row separators, enhancing compactness for efficient utilisation in training the LLM;
- **Output**: the expected LLM result is generated after providing the instruction and the table. This field can be used to validate the annotations produced by the model;
- **Entities**: the set of entities is essential to supply the model with a range of choices during annotation, assisting it with the disambiguation task.

Listing 2 presents an example of a dataset instance, displaying the above-illustrated fields.

6 PROMPTING FOR STI USING GPT-4

Various prompts have been crafted to elicit the most appropriate output from GPT-4, subsequently evaluating it against the GS. Given that GPT is a decoder-only model, significant textual variations in the results generated from two slightly distinct inputs are to be expected. The formulation of a prompt with a predetermined structure is an essential requirement to generate a parsable response, facilitating the evaluation of annotations. Various prompts have been tested using APIs provided by OpenAI[23] to interact with GPT-4. The OpenAI APIs enable interaction with the GPT-4 model and provide the ability to configure parameters that control response generation: i) **Temperature**, ii) **Max Tokens**, iii) **Frequence Penalty**, and iv) **Presence Penalty**.

The i) **Temperature** parameter, used to control the randomness of predictions during text generation, was set to 0.1. In this way, for each prompt, similar outputs can be generated and parsed similarly. The ii) **Max Tokens** parameter, which restricts the maximum number of tokens in the output generated by an LLM, was set to 100 to avoid generating unnecessary text. This value, tailored to the utilised dataset, appeared to be a good trade-off between the number of tokens needed for the generated text and the potential generation of out-of-context tokens by the model. Nonetheless, a cut in the output of long tables is imposed. iii) **Frequence Penalty** is a parameter that controls how often an LLM repeats similar phrases or sequences. In this case, it was set to 0.2 to help generate the correct Wikidata ID from a provided candidates set, ensuring that each table mention in the input prompt is accurately annotated. The iv) **Presence Penalty**, which is a parameter that regulates the frequency of words present in the input compared to those generated in the output text, was set to 0.2. This value is specifically useful for those prompts including the candidate set (which is described later in this Section), as, in this case, the entities to be used for the CEA annotations are already in the prompt. Otherwise, there is the risk of generating random entity IDs.

After configuring the API parameters, various prompts were designed based on an initial categorisation of possible input formulations. These prompts were then tested to evaluate the structure of GPT-4's output, and the most effective one was selected.

Different types of prompts allow an analysis of how an LLM responds to the same question when phrased differently. Generating an answer is closely linked to the input, so it is crucial to identify a format that elicits the best response. In this case, four prompt templates have been identified:

(1) **Execution** prompt: it effectively represents a question that directly explains the task, providing all the necessary data as input. It is structured as follows: "perform the CEA task on the table T, where csv is the

[23] OpenAI: OpenAI API - openai.com/blog/openai-api

format of the table" (see Listing 3). This case study assumes that the LLM knows the steps necessary to complete the task;

(2) **Request** prompt: it is structured as a question that inquires about the result of performing a specific task (CEA) on the provided data. Essentially, it seeks information about the consequences of that action. The prompt is structured as "what is the result of performing the CEA task on the table T, where csv is the format of the table" (see Listing 4). Also, in this case, it is assumed that the model knows how to perform the requested task;

(3) **Completion** prompt: it provides the model with a portion of the sentence and requires completion, *e.g.*, "the CEA task performed on csv format of the table is:" (see Listing 5). This mode seeks to exploit the LLM's ability to generate text sequences;

(4) **Programming** prompt: it reports a series of instructions in Python that the LLM reads and executes to carry out the requested task step by step, such as "for each i in table T ..." (see Listing 6). This type of prompt is longer and verbose but, at the same time, better suited for a language model that can understand the logical instructions of a high-level language. A similar approach has also been adopted by Dagobah [41], as converting structured data into code is more manageable for this task and provides a more informative representation than transforming it into free-form text.

In the listings below, we highlighted in teal colour the textual part (or code) in the different types of prompts. Black was used for tabular data.

Listing 3. Execution prompt.

```
T =
Mont Blanc,4810,Alps,Graian Alps
K2,8611,Himalayas,Karakoram
Kilimanjaro,5895,Tanzania,East African Mountains
Mount Everest,8848,Himalayas,Himalayas

Each element in the table T is identified by the indexes couple (i, j) where i is the row index and j is the
column index. Perform a Cell Entity Annotation task on the table T using Wikidata as a Knowledge Graph.
```

Listing 4. Request prompt.

```
T = [..]

Given a table T where each cell is identified by the pair (i, j) with i as the row index and j as the column
index, what is the result of performing the task of Cell Entity Annotation for each cell (i, j) in table T
using Wikidata as a Knowledge Graph?
```

Listing 5. Completion prompt.

```
Given a table T, it is possible to identify each of its elements through a pair of indices (i, j), where
i is the row index and j is the column index.
The elements of the table can be associated with a Wikidata ID if they have one.
Given the table

T = [..]

the association between the elements of the table and the Wikidata IDs is:
```

Listing 6. Programming prompt.

```
T =
Mont Blanc;4810;Alps;Graian Alps
K2;8611;Himalayas;Karakoram
Kilimanjaro;5895;Tanzania;East African Mountains
Mount Everest;8848;Himalayas;Himalayas

For each row r in table T:
    For each field c in row r:
        # Assume that the field contains an entity mention
        entity_mention = r[c]

        # Initialize a variable to track the best match
        best_match = null
```

```
        max_score = 0

        # assume the get_entities_from_wikidata return entities from Wikidata that can be suitable for the entity_mention
        candidate_entities = get_entities_from_wikidata(entity_mention)

        # Search for matches among candidate entities
        For each wikidat_entity in candidate_entities:
             # Calculate a similarity score between the mention in the table and the entity in Wikidata
             score = calculate_similarity_score(entity_mention, wikidata_entity)

             # Update the best match if the score is higher than the current maximum
             If score > max_score:
                  max_score = score
                  best_match = wikidata_entity

        # Assign the best match to the mention in the entity in the table
        r[c] = best_match
```

Each type of prompt was subsequently tested based on two additional criteria: i) *zero-shot and few-shots learning* and ii) *the use or omission of a set of entities retrieved from Wikidata*.

The former aims to provide the GPT model with guidance on how to generate an appropriate output given a request in the form: "for example, if we had the mention x, then through CEA task, we associate (i,j) = Q..." (see Listing 7). This is akin to giving the model a specific scenario to consider, which can improve the relevance and specificity of the generated responses.

Listing 7. Prompt with example.

```
T =
Mont Blanc,4810,Alps,Graian Alps
K2,8611,Himalayas,Karakoram
Kilimanjaro,5895,Tanzania,East African Mountains
Mount Everest,8848,Himalayas,Himalayas

Each element in the table T is identified by the indexes couple (i, j) where i is the row
index and j is the column index.
Perform a Cell Entity Annotation task on the table T using Wikidata as a Knowledge Graph
For example, if we had the cell (2, 5) in table T containing the entity
'Milano', and in Wikidata, we have the entity 'Q490 Milano' then through CEA, we associate (2, 5)=Q490
```

The latter, *i.e.*, the use of a set of entities, is devised to overcome the restrictions imposed on the GPT model's ability to access the Internet or initiate calls to Wikidata services to acquire the entities needed for the task. In fact, LLMs are trained on a large corpus of texts taken from the internet and other sources, but they can only process and generate text based on what has been included in their training set. Given this limit, we provided a set of identifiers, denoted as P in the prompt, for each mention, along with the correct identifier in the dataset. Supplying the model with such a predefined set ensures that the entities relevant to the task are effectively available during inference, thereby enhancing its capacity to handle entity-related queries. As shown in Listing 8, to include the set of entities, the prompt can be expanded in this way: "in the set P there is a group of Wikidata IDs ...". In the set, each Wikidata ID is followed by the corresponding label to provide additional information to the model for the annotation task. In the SOTA, the attempt to add a set of possible responses was tested with good results on the CTA task [56].

Including the candidate set in the prompt makes it possible to assess the language model's capability to accurately execute ED on mentions within the table. Conversely, by discarding a candidate set from the prompt, one can evaluate the overall CEA pipeline, which comprises both Entity Retrieval (ER), where candidate entities for the mention are collected, and Entity Disambiguation (ED), where the mention is disambiguated by selecting one or none of the candidate entities.

Listing 8. Prompt with pool.

```
T =
Mont Blanc,4810,Alps,Graian Alps
K2,8611,Himalayas,Karakoram
Kilimanjaro,5895,Tanzania,East African Mountains
Mount Everest,8848,Himalayas,Himalayas

```

```
Each element in the table T is identified by the indexes couple (i, j) where i is the row
index and j is the column index.

P =
Q513 [Everest], Q524 [Vesuvio], Q583 [Mont Blanc]
Q1024 [Triglav], Q1248 [Viso], Q1374 [Gran Paradiso], Q1372 [Grivola]
Q1373 [Matterhorn], Q18869 [Caucasus], Q1451 [Pinatubo], Q1484 [Mayon Volcano]
Q5469 [Karakoram], Q17009782 [East African Mountain], Q1286 [Alps], Q1637969 [K2]
Q2353 [Topfer], Q265406 [Kilimanjaro], Q3309 [Pec], Q3375 [Zugspitze], Q1262 [Graian Alps]
Q3388 [Grossglockner], Q43105 [Mount Elbrus], Q655495 [Tanzania], Q3403 [Dufourspitze]
Q3428 [Bungsberg], Q3660 [Mount Davis], Q686902 [Himalayas]

In the pool P there are a group of wikidata ids. Perform a Cell Entity Annotation task on the
table T using the pool
```

By supplying the prompt with both a few-shots learning approach and a set of entities, the task becomes more comprehensive to the model. Not only does this approach guide the model by providing a specific context, but it also equips it with the necessary identifiers, creating a more robust and effective input for the desired task (Listing 9).

Listing 9. Prompt with example and pool.

```
T =
Mont Blanc,4810,Alps,Graian Alps
K2,8611,Himalayas,Karakoram
Kilimanjaro,5895,Tanzania,East African Mountains
Mount Everest,8848,Himalayas,Himalayas

Each element in the table T is identified by the indexes couple (i, j) where i is the row
index and j is the column index.
P =
Q513 [Everest], Q524 [Vesuvio], Q583 [Mont Blanc]
Q1024 [Triglav], Q1248 [Viso], Q1374 [Gran Paradiso], Q1372 [Grivola]
Q1373 [Matterhorn], Q18869 [Caucasus], Q1451 [Pinatubo], Q1484 [Mayon Volcano]
Q5469 [Karakoram], Q17009782 [East African Mountain], Q1286 [Alps], Q1637969 [K2]
Q2353 [Topfer], Q265406 [Kilimanjaro], Q3309 [Pec], Q3375 [Zugspitze], Q1262 [Graian Alps]
Q3388 [Grossglockner], Q43105 [Mount Elbrus], Q655495 [Tanzania], Q3403 [Dufourspitze]
Q3428 [Bungsberg], Q3660 [Mount Davis], Q686902 [Himalayas]

In the set P there are a group of wikidata ids.
Perform a Cell Entity Annotation task on the table T using the pool P.
For example, if we had the cell (2, 5) in table T containing the entity 'Milano' and in pool P, we
have the entity 'Q490 Milano' then through CEA, we associate (2, 5)=Q490.
```

If the set of entities is not given, such as in Listings (3,4 and 5), the GPT model is instructed to use Wikidata as a reference KG. In the specific case of the iv) **Programming** prompt (Listing 6), the extraction of Wikidata entities is simulated by a function that retrieves a list of entities.

We will now discuss the results of these experiments, which can be found in the code repository[24].

The i) **Execution** and ii) **Request** prompts do not provide valuable responses for the CEA, as the model does not yield a response that can be parsed to extract mention annotations. The model generates a response that contains a lot of Python-like pseudocode. The LLM faces challenges in effectively addressing a particular task and delivering a valuable response due to inadequate coverage of the task in its training data or a lack of diversity in relevant examples. Moreover, the CEA is inherently complex, demanding a profound comprehension of tabular data and KG.

The iii) **Completion** prompt consistently delivers impressive, easily interpretable results. The model can accurately select the appropriate annotation by incorporating a defined set of entities. The effectiveness of the *Completion* prompt in achieving desired outcomes lies in the structure of its template design. These templates not only guide the generation process but also mirror how LLMs learn during training. Through predicting and completing text based on input patterns, LLMs are trained to recognise and produce coherent, contextually appropriate outputs. *Completion* templates present the model with partial sentences, phrases, or cues that provide essential context for the task. This context enables the model to generate more relevant and logically consistent

[24] stEELlm: documentation pages with prompts descriptions - unimib-datai.github.io/steellm-docs/docs/prompts/

completions. Moreover, completion templates act as a framework, guiding the model on how and where to integrate the missing information.

Given the successful responses of GPT-4 with Python pseudocode in earlier responses, the final prompt, *i.e.*, the iv) **Programming** one, aims to guide the model in generating accurate pseudocode for annotating tables in the context of the CEA. GPT-4's output tries following instructions to annotate each mention using Wikidata IDs. The resulting annotations are presented in a matrix format, mirroring the structure of the input table specified in the prompt. Additionally, the model's responses remain accurate when incorporating entities, ensuring correct information retrieval from the KG.

The best result was obtained with the **Programming** prompt, but due to the extensive length of the prompt, the execution time was huge. Therefore, the **Completion** template, which is the second best-performing prompt, was chosen as *baseline* for fine-tuning.

7 FINE-TUNING OF A LLM FOR CEA TASK

LLMs are typically pre-trained on a diverse and large corpus of text data using a pre-training objective such as language modelling. During the pre-training phase, the model learns to predict the next word in a sentence or to fill in missing words. While this process allows the model to capture general language patterns, it does not explicitly train the model for task-specific operations. Consequently, LLMs are designed to generalise well across a broad range of language tasks, learning to understand language's underlying structure and semantics. This generalisation makes them versatile but may lead to less optimal performance on highly specific tasks. In contrast, fine-tuning allows the model to learn task-specific patterns.

7.1 Mixtral 8x7B

Mixtral 8x7B [46], developed by Mistral AI[25], is the chosen LLM to perform STI. The main advantages of employing **Mixtral 8x7B** model are:

- **Licensing**: Mixtral 8x7B is licensed under the Apache 2.0 license, which allows users to freely employ, modify, and distribute the model under certain conditions;
- **Performance**: Mixtral 8x7B outperforms Llama 2 70B on most benchmarks [46] and offers 6x faster inference. It is regarded as the strongest open-weight model with a permissive license and considered the best model overall in terms of cost/performance trade-offs;
- **Capabilities**: Mixtral 8x7B can handle a context of 32k tokens, support multiple languages, and be fine-tuned into an instruction-following model that achieves a score of 8.3 on MT-Bench[26] [115].

Mixtral 8x7B is a decoder-model featuring a sparse **Mixture of Experts** [44] network with a unique architectural design. The feedforward block draws from a pool of 8 parameter groups, with a router dynamically selecting two "experts" per layer and a token to handle different aspects of data (Fig. 6). This approach efficiently increases the parameters count while managing costs and latency. Furthermore, Mixtral 8x7B has been pre-trained with a $32k$ token context using multilingual data.

This paper uses the **Mixtral 8x7B – Instruct** model to perform the fine-tuning on CEA. The *Mixtral 8x7B – Instruct* model imposes a precise prompt format that must be strictly respected; otherwise, the model generates sub-optimal outputs [84].

[25]Mistral AI - mistral.ai

[26]Hugging Face: LMSYS Chatbot Arena Leaderboard - huggingface.co/spaces/lmsys/chatbot-arena-leaderboard. Mixtral 8x7B was in 12th position on February 17, 2024.

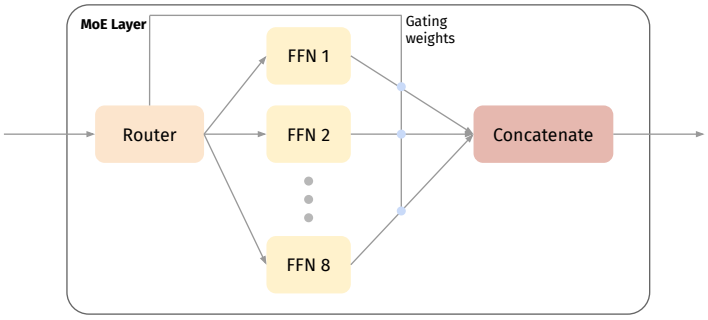

Fig. 6. Mixtral 8x7B Mixture of Experts Architecture.

7.2 Fine-tuning

To effectively utilise the instructions for fine-tuning, enclosing the prompt within [INST] and [/INST] tokens when feeding it to Mixtral 8x7B is recommended[27]. The initial instruction should begin with a beginning-of-sentence identifier (<s>). The end-of-sentence (</s>) token identifier will conclude the assistant's generation. An example is provided in Fig. 7.

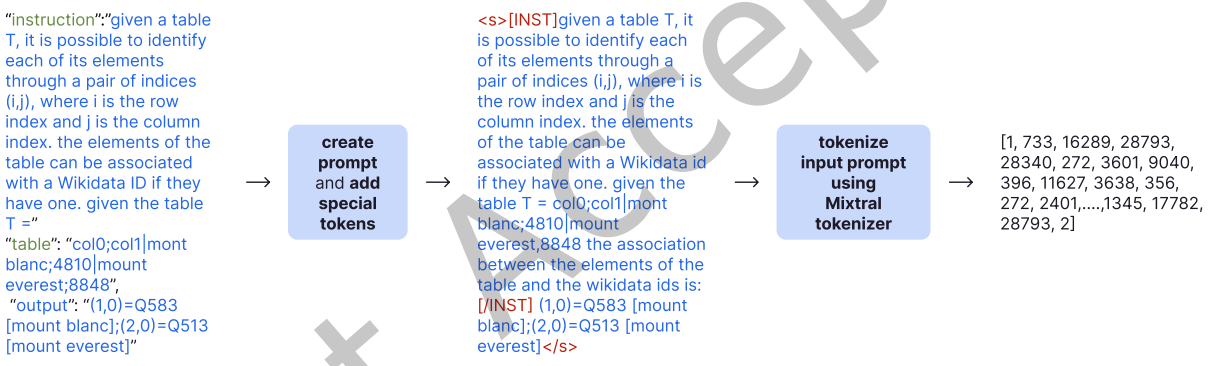

Fig. 7. Mixtral 8x7B Tokenizer.

The table has been compacted by replacing the new line, which separated each row, with the pipe character "|". The output has been specified between the closing INST token ([/INST]) and the end of sentence token </s>. Once each prompt is created, the input string is passed to the tokenizer, which breaks it down and transforms each token or sub-token into an integer value representation.

Tokenisation into sub-tokens offers several advantages, such as: i) managing rare or uncommon words by breaking them into more frequent parts; ii) having a smaller vocabulary of sub-tokens compared to whole words, therefore reducing model complexity and memory requirements, and improving training and inference efficiency; iii) benefiting languages with complex morphologies, such as Turkish, Finnish, or Hungarian, by breaking words with multiple affixes into more manageable parts; and iv) compressing information into more compact representations while preserving the meaning of the original words. This process can also introduce certain challenges and issues in language models. One is the potential increase in the overall sequence length,

[27] Hugging Face: Mixtral-8x7B-Instruct-v0.1 - huggingface.co/mistralai/Mixtral-8x7B-Instruct-v0.1

resulting in longer input sequences for the model. This can impact the duration of training and inference, as well as the memory requirements of the model [18].

Rare words might get split into overly specific sub-tokens, making it difficult for the model to learn meaningful representations. This situation is particularly evident when Wikidata IDs are provided to the model, as they are split into single characters. This behaviour stems from the intrinsic nature of Wikidata IDs, which consist of alphanumeric strings. For instance, the entity "Paris" has the ID "Q90", which is tokenised in this way: ["Q", "9", "0"]. This issue arises when the model is required to generate the ID of the annotated entity at inference time, as it must accurately predict the sequence of the numbers that compose the entity ID, which is not handled as a single token.

This paper discusses the application of multiple fine-tuning strategies to test how the Mixtral 8x7B model learns to perform STI. Specifically, the dataset whose construction was described in Section 4 contains instances for performing CEA. The three fine-tunings that have been conducted are: i) **Fine-tuning on Wikidata triples**, ii) **Table-Annotated fine-tuning**, and iii) **Table-Annotated with the candidate set fine-tuning**.

The former, i) **Fine-tuning on Wikidata triples**, aimed at providing knowledge regarding the structure of the KG to the LLM. While LLMs can learn about general knowledge and facts extensively in their training data, they do not inherently know specific entities or facts stored in a KG. Additionally, they do not have direct access to databases or external knowledge bases like Wikidata during their inference. Prompting an LLM about specific entities relies on the patterns and associations learnt during training. The model may provide accurate or pertinent information if the knowledge exists within the training data. For this purpose, the idea for the first fine-tuning approach was to infuse knowledge in the LLM through a form of "pre-training", where the model is instructed to learn Wikidata triples, which are inherent only to the entities present in the GS. This approach is motivated by the high computational complexity of the training, given that Wikidata comprises approximately 100 million entities. Thus, reducing the number of triples helps decrease the computational workload, the economic effort, and the energy consumption. We must emphasise that this method can import the entirety of Wikidata. Following the methodology chosen in [107], which treats triples as textual sequences and turns KG completion into a sequence classification problem, we opted for a training strategy that entails the creation of a dataset of Wikidata triples. For each pair of subject and predicate, the model was trained to predict the correct object to complete the triple. An example is shown in Listing 10. Each prompt instance provided to the model corresponds to a triple in Wikidata. The subject "QID [label]" and the predicate "QID [label]" are given to the model, while the object "QID [label]" is generated by the LLM. Consequently, the output is positioned between the "[/INST]" token and the "<s>" token, which, as already explained, marks the end of the prompt.

Listing 10. Wikidata triples prompts.

```
<s>[INST]which wikidata object complete triple:
<Q107184 [cupric sulfate], P462 [color]>[/INST] <Q107184 [cupric sulfate], P462 [color], Q23444 [white]></s
    ↪ >

<s>[INST]which wikidata object complete triple:
<Q18845221 [zinc-69], P279 [subclass of]>[/INST] <Q18845221 [zinc-69], P279 [subclass of], Q758 [zinc]> </s
    ↪ >

<s>[INST]which wikidata object complete triple:
<Q52033848 [Garfo], P17 [country]>[/INST] <Q52033848 [Garfo], P17 [country], Q155 [Brazil]> </s>
```

In this case, the model could not perform STI as the purpose of the training was only to memorise facts inside the LLM. The training has been executed on 1154611 triples.

The second fine-tuning is applied to the model aligned with Wikidata triples, as it already knows the KG. In ii) **Table-Annotated fine-tuning**, the model is instructed to annotate the mentions of the tables (CEA). In this case, the label of the correct entity in the output is composed of the ID and the label itself. The choice to maintain the pair ID-label was made so the model could learn this mapping. An example of input prompts can be found in Listing 11.

Listing 11. Table-Annotated fine-tuning.

```
<s>[INST] perform the cell entity annotation (cea) task on this table where each row
is separated by '|' and each cell is separated by ';' and the first row in position 0
is the header of the table:

darabani;7453|dealu mare;414|tirici;56

the association between the elements of the table and the wikidata ids is:[/INST]

(1,0)=Q899467 [darabani]|(2,0)=Q12103079 [dealu mare]|(3,0)=Q1321408 [tirici]</s>
```

The final fine-tuning, iii) **Table-Annotated with the candidate set fine-tuning**, consists of the integration of the set of entity IDs and the labels extracted from LamAPI into the training process. This additional data helps the model to annotate mentions accurately. The correct entity is always available in the provided set (extracted from the GS) because the purpose is to evaluate if the model can perform disambiguation by selecting the correct entity for each name entity mention. This strategy has been applied to enhance the annotation process and improve the model's overall accuracy. An example of input prompts is shown in Listing 12.

During the fine-tuning process, the model learns the fine details and the specific terminology of the STI tasks, improving its ability to generate the intended output. As a result, it is no longer necessary to include an example of how to link mention with entities as it occurs in few-shot learning (*e.g.*, see Listing 7, lines 10-11).

Listing 12. Table-Annotated with candidate set fine-tuning.

```
<s>
[INST] given a table T, it is possible to identify each of its elements through a pair of
indices (i,j), where i is the row index and j is the column index.
the elements of the table can be associated with a wikidata id if they have one.
given the table T =

shyamchand temple;11;1660-01-01;11|
gokulchand temple;13.7;1643-01-01;13.7|
sareswar temple;5.8;1346-01-01;5.8|
saileswar temple;4.5;1346-01-01;4.5.

use the following pool of entities to annotate the table:

Q110991313 [sareswar mahadev temple];Q110622204 [shyamchand temple];Q56159226 [saileswar temple];
Q66439262 [shyamchand temple];Q243999 [temple];Q56160151 [gokulchand temple];
Q56159477 [sareswar temple];Q58783429 [shyamchand temple];Q44539 [temple];[/INST]

the association between the elements of the table and the wikidata ids is:[/INST]

(1,0)=Q58783429 [shyamchand temple]|(2,0)=Q56160151 [gokulchand temple]|
(3,0)=Q56159477 [sareswar temple]|(4,0)=Q56159226 [saileswar temple]
</s>
```

The strategy for fine-tuning the model is the same for each training dataset. The predominant method for fine-tuning in contemporary AI research is Supervised Fine-Tuning (SFT). This strategy directly fine-tunes the model over the examples present in the training dataset; then, the model learns to replicate the style of these examples during fine-tuning. SFT closely resembles language model pre-training, as both methods employ next token prediction as their fundamental training objective [77].

Fine-tuning an LLM is a challenging computational task. In addition to accommodating the model, adequate storage space is required to hold vital parameters for the training process. This is where Parameter-Efficient Fine-Tuning (PEFT) [64] becomes a key aspect. While full LLM fine-tuning updates every model's weight during the supervised learning process, PEFT methods only update a small set of parameters. Our approach uses **LoRA** [39], an improved method that fine-tunes two smaller matrices of weights instead of updating all the weights in the model. In **LoRA**, the r parameter, which represents the rank of the matrices learned during the fine-tuning process, is set to 16. This value allows for a suitable trade-off between the computational resources required for the training and the quality of the model outputs.

In Python, this process is facilitated by TRL library[28], as it offers a user-friendly API that enables the effortless creation of SFT models. With just a few lines of code, it is possible to train LLMs on a custom dataset.

In terms of time, fine-tuning Mixtral 8x7B remains relatively expensive. For this reason, the training using Wikidata triples has been done for 4 epochs, which took around 65 hours. The other three fine-tunings took around 40 hours for 20 epochs. The employed VM was created on Azure Platform with the following characteristics: 220GB RAM, 248GB Disk, and 80GB GPU (size Standard_NC24ads_A100_v4)[29]. The final cost for the computation amounted to 3028€.

8 EVALUATION

The responses given by GPT-4 and Mixtral 8x7B for STI must be parsable to automate the validation process. For this reason, the outputs are passed through a regular expression module, which extracts the annotations and converts them in *csv* format, as summarised in Fig. 8.

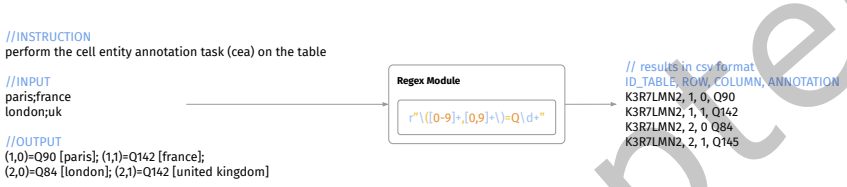

Fig. 8. Parser of prompt response.

The metrics used to evaluate the CEA are [50]:

- **Precision**: a measure of the accuracy of a model's positive annotations, calculated as the ratio of correct annotations to the total number of annotations generated by the system (GPT-4 and Mixtral 8x7B);

$$Precision = \frac{\#correct\_annotations}{\#system\_annotations} \tag{1}$$

- **Recall**: a metric that considers the coverage of annotations for each table. It is calculated as the ratio of true positive annotations to the total number of target annotations in the GS;

$$Recall = \frac{\#correct\_annotations}{\#target\_annotations} \tag{2}$$

- **F1-Score**: the F1-score is the harmonic mean of precision and recall.

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{3}$$

Table 2 presents the results from the first set of experiments, *i.e.*, those where GPT-4 was employed in the prompting phase. Results demonstrate that the CEA task cannot be completed through prompting a general pre-trained LLM alone. Two key reasons account for this outcome: i) the model lacks the knowledge needed to perform the task, and ii) it cannot generate the required annotations in the output.

Listing 13 provides an example of a request prompt and GPT-4's response, demonstrating how the pre-trained model generates annotations.

[28]Hugging Face: Transformer Reinforcement Learning - huggingface.co/docs/trl/index
[29]Microsoft Azure: NC_A100_v4 sizes series - learn.microsoft.com/en-us/azure/virtual-machines/nc-a100-v4-series

Table 2. Results obtained for CEA task using different prompts templates.

Prompt		Zero-Shot	Few-Shot	Zero-Shot + Entity Set	Few Shot + Entity Set
Execution	**P**	0.08	0.018	0.329	0.304
	R	0.005	0.014	0.048	0.213
	F1	0.009	0.016	0.084	0.250
Request	**P**	0	0.038	0	0.488
	R	0	0.039	0	0.216
	F1	0	0.038	0	0.299
Completion	**P**	0	0.020	0.329	0.391
	R	0	0.017	0.055	0.312
	F1	0	0.019	0.094	0.347
Programming	**P**	0.056	-	0.349	-
	R	0.017	-	0.257	-
	F1	0.026	-	0.296	-

Listing 13. Request prompt response example.

```
Given a table T, it is possible to identify each of its elements through a pair of indices (i,j), where i
    ↪ is the row index and j is the column index.
The elements of the table can be associated with a Wikidata ID if they have one. The Wikidata IDs are in a
    ↪ pool P. Given the table:

T =
Mont Blanc,4810,Alps,Graian Alps
K2,8611,Himalayas,Karakoram
Kilimanjaro,5895,Tanzania,East African Mountains
Mount Everest,8848,Himalayas,Himalayas

the association between the elements of the table is:

To associate the elements of the table with Wikidata IDs, you need to identify the Wikidata entity
corresponding to each element where possible. The table you provided includes names of mountains, heights, ranges,
and other descriptors. Here's how the association would look if each element corresponds to an entity in Wikidata:

Table (T) Row (i) Column (j) Element Wikidata ID
Mont Blanc 1 1 Mont Blanc Q334
4810 1 2 4810 (height in m) - (No Wikidata ID, numerical)
Alps 1 3 Alps Q14683
Graian Alps 1 4 Graian Alps Q29416
K2 2 1 K2 Q131454
8611 2 2 8611 (height in m) - (No Wikidata ID, numerical)
Himalayas 2 3 Himalayas Q35657
...
```

While GPT-4 can identify which table cells should be annotated as KG entities or datatypes, it struggles to assign accurate Wikidata QIDs. For example, the QIDs in Listing is entirely incorrect.

A second example is shown in Listing 14; in this case, the presence of candidate entities (pool) allows for correct annotation. However, the model output is very verbose, making it difficult to parse.

Listing 14. Request prompt response example.

```
Given a table T, it is possible to identify each of its elements through a pair of indices (i,j), where i
    ↪ is the row index and j is the column index.
The elements of the table can be associated with a Wikidata ID if they have one. The Wikidata IDs are in a
    ↪ pool P. Given the table:

T =
Mont Blanc,4810,Alps,Graian Alps
K2,8611,Himalayas,Karakoram
Kilimanjaro,5895,Tanzania,East African Mountains
Mount Everest,8848,Himalayas,Himalayas

and the pool:
```

```
P =
Q513 [Everest], Q524 [Vesuvio], Q583 [Mont Blanc]
Q1024 [Triglav], Q1248 [Viso], Q1374 [Gran Paradiso], Q1372 [Grivola]
Q1373 [Matterhorn], Q18869 [Caucasus], Q1451 [Pinatubo], Q1484 [Mayon Volcano]
Q5469 [Karakoram], Q17009782 [East African Mountain], Q1286 [Alps], Q1637969 [K2]
Q2353 [Topfer], Q265406 [Kilimanjaro], Q3309 [Pec], Q3375 [Zugspitze], Q1262 [Graian Alps]
Q3388 [Grossglockner], Q43105 [Mount Elbrus], Q655495 [Tanzania], Q3403 [Dufourspitze]
Q3428 [Bungsberg], Q3660 [Mount Davis], Q686902 [Himalayas]

the association between the elements of the table and the IDs in the pool is:

To associate the elements of the table T with the Wikidata IDs from the pool P, we match each element in T to the
appropriate ID in P, if one exists. Here's the association:

Table (T) Row (i) Column (j) Element Wikidata ID
Mont Blanc 1 1 Mont Blanc Q583 [Mont Blanc]
4810 1 2 4810 (height in m) - (No matching ID)
Alps 1 3 Alps Q1286 [Alps]
Graian Alps 1 4 Graian Alps Q1262 [Graian Alps]
K2 2 1 K2 Q1637969 [K2]
...

Notes:
Numerical entries: Heights (e.g., 4810, 8611, 5895, 8848)
are literal values and do not have
Duplicate entries: The same element (e.g., "Himalayas") is mapped to the same
Wikidata ID wherever it appears in the table.
Unmatched elements: If no corresponding Wikidata ID exists in P, the entry is noted as No matching ID.
This ensures each element with a potential Wikidata ID in P is correctly associated.
```

Table 3 shows the results achieved by the *baseline with Programming Prompt* and *Mixtral 8x7B*.

Table 3. Results obtained for CEA task.

Model	Precision	Recall	F1-Score
Baseline with Programming Prompt	0.0565	0.0166	0.0257
Baseline with Programming Prompt + Entity Set	0.3495	0.2575	0.2965
Mixtral 8x7B Fine-Tuned	0.0241	0.0235	0.0238
Mixtral 8x7B Fine-Tuned + Entity Set	**0.9235**	**0.9086**	**0.9160**

The execution of the *Baseline with Programming Prompt*, which uses the LLM trained to understand natural language text, is unsuitable for performing specific tasks, such as CEA. The main issue, as depicted in Listing 13, is that the model cannot access the KG, resulting in entirely random annotations. The model remembers the entity identification as one of its parameters, which accounts for the few correctly annotated mentions. Even when using the *Mixtral 8x7B Fine-tuned* model, the annotations are subject to **hallucinations**, *i.e.*, to the generation of plausible content that is not accurate or based on factual information. Hallucinations occur when the model has learnt patterns from the training data but not the semantics of the annotations.

To address the problem of accessing the KG, a set of entities has been incorporated in the prompt, as in the *Baseline with Programming Prompt + Candidate Set*. This solution uses GPT-4, which can disambiguate more effectively when annotating mentions, resulting in a higher precision score compared to the *Baseline with Programming Prompt* and the *Mixtral 8x7B Fine-tuned*, both of which lack a candidate set.

The best results are achieved by *Mixtral 8x7B Fine-tuned + Candidate Set* (see Listing 15 for a response example), resulting in a precision of 0.9235. Training the model to disambiguate the correct entity from those extracted by an information retrieval system, such as LamAPI, yields better performance. The key strategies learnt during the training for performing ED include:

(1) Choosing the entity with the label that most closely matches the mention;
(2) Learning to perform annotations by observing the context within the table from both column and row perspectives;
(3) Selecting the entity with greater relevance, which in the case of Wikidata, generally, are the numerical IDs with lower values.

Listing 15. Example of response of fine-tuned model with candidate set.

```
<s>[INST] perform the cell entity annotation (cea) task on this table where each row
is separated by '|' and each cell is separated by ';' and the first row in position 0
is the header of the table:

col0;col1;col2|
Austrian Grand Prix;1973-08-19;Red Bull Ring|
1971 Austrian Grand Prix;1971-08-15;Red Bull Ring.

use the following pool of entities to annotate the table:
Q66890685 [red bull ring layout];
Q15503 [1971 austrian grand prix];
Q15513 [1973 austrian grand prix];
Q3774652 [1973 austrian motorcycle grand prix];
Q3465217 [grand prix 1973];
Q3465199 [grand prix 1971];
Q171566 [red bull ring];
Q2734894 [bull ring];
Q3774647 [1971 austrian motorcycle grand prix];

the association between the elements of the table and the wikidata ids is :[/INST]

(1,0)=Q15513 [1973 austrian grand prix]|
(1,2)=Q171566 [red bull ring]|
(2,0)=Q15503 [1971 austrian grand prix]|
(2,2)=Q171566 [red bull ring]|
(3,0)=Q3774652 [1973 austrian motorcycle grand prix]|
(3,2)=Q171566 [red bull ring]|
(4,0)=Q3774647 [1971 austrian motorcycle grand prix]|
(4,2)=Q171566 [red bull ring]|
(5,0)=Q3465217 [grand prix 1973]|
(5,2)=Q171566 [red bull ring]|
(6,0)=Q3465199 [grand prix 1971]|
(6,2)=Q171566 [red
```

Additionally, as shown in Table 3, in every run of the CEA algorithm, the Recall score is lower than the Precision score. This discrepancy arises due to the LLM's occasional failure to generate annotations. Sometimes, the model annotates all the cells, even those containing literals. Consequently, tables with many rows experience a limitation: mentions located at the end of the table are frequently overlooked, reaching the maximum length of the model output. Conversely, for tables having a limited number of rows, the model endeavours to label cells that are, in reality, absent. This behaviour suggests that, even if the model was fine-tuned on CEA, it does not always understand which mentions need to be annotated.

8.1 Comparison with SOTA approaches

The comparison with SOTA approaches has been performed using a downsampled version of the original dataset. This reduced dataset consists of around 10000 tables for training (used exclusively for stEELlm) and 300 tables for testing, with a total of 1278 cells. The dataset used and responses given by each model are publicly available in the GitHub Repository[30].

Four approaches have been selected as representative of different categories of algorithms that solve semantic tasks on tables, especially CEA: TableLlama is the first autoregressive LLM that is specifically instruction-tuned on tabular data and reports SOTA results [109]; s-elBat is a recent feature-based supervised and unsupervised algorithm [9]; Dagobah, a heuristic-based algorithm, has been the winner of various rounds of the SemTab challenge in 2020 [43], 2021 [42], and 2022 [40], and is publicly available[31]. We also considered Blink [104] as a SOTA approach related to entity linking. This model uses a two-stage zero-shot algorithm to retrieve entities from dense spaces and re-ranks candidates based on mention-entity text concatenation.

Table 4 shows the results achieved by each approach, along with their corresponding execution times. TableLlama achieves the best performance in terms of Precision, Recall, and F1-score. Its prompt design offers an advantage for Recall because processing one prompt per cell ensures no annotation is missed. Regarding Precision,

[30] stEELlm: comparison results - github.com/unimib-datAI/steellm/tree/main/datasets/Comparison
[31] Orange: Table Annotation - github.com/Orange-OpenSource/Table-Annotation

Table 4. Comparison of different CEA approaches.

Model	Precision	Recall	F1-Score	Inference Time (s)
TableLLama	0.913	0.913	0.913	1267 (22 minutes)
s-elBat	0.914	0.772	0.837	467 (8 minutes)
BLINK	0.597	0.594	0.595	426 (7 minutes)
Dagobah	0.630	0.594	0.611	7344 (2 hours)
stEELlm	0.874	0.860	0.867	1520 (25 minutes)

TableLlama also excels by treating label, description, and entity types as annotations, all of which are textual. This makes it easier for an LLM to generate the correct annotation, as it only needs to produce text rather than alphanumeric QIDs. Additionally, TableLlama has been fine-tuned on a vast collection of annotated tables using a cluster of 48 A100 GPUs. The second approach tested is s-elBat; it employs a supervised method using a two-step Neural Network (NN) pipeline to perform the CEA task on tables. This system delivers strong performance on the provided test set, as it has been trained on several SemTab datasets. The third model, Blink, performs entity linking using Wikipedia entities. To accommodate this, a mapping API between Wikipedia and Wikidata has been integrated into LamAPI[32]. The gap between Precision and Recall is attributed to the absence of corresponding mappings for six annotations between Wikipedia and Wikidata. As highlighted by the Precision scores, Blink is not well-suited for entity linking in tables, as it was developed for natural language text. It may need to be adapted for the tabular domain to bring its performance closer to SOTA algorithms. The last compared model is Dagobah, which is a heuristic-based algorithm developed during the 2021/2022 SemTab editions. Like s-elBat, the approach was created to work on tabular data, annotating the cells using the row and column context. In summary, based on the results of these approaches, stEELlm achieves performance comparable to SOTA methods, positioning itself just below TableLlama. At the same time, it is important to note that the proposed model obtains good, if not excellent, scores in the original dataset created from the various editions of the challenge (Table 3).

The score achieved by Dagobah should not be surprising since the techniques and heuristics applied in Dagobah were created ad hoc after long and in-depth analyses of the specific characteristics of the dataset to be annotated, as stated in the introduction. This results in minimal, if any, applicability of this kind of approach to other data. This is empirically demonstrated in Table 5, which presents the results of Dagobah 2021/2022 [40] on various challenge datasets.

Table 5. Dagobah system (2021/2022) on previous SemTab challenge dataset (2020).

Dataset	Precision	Recall	F1-Score
SemTab 2020 R1	0.889	0.813	**0.849**
SemTab 2020 R2	0.889	0.821	**0.854**
SemTab 2020 R3	0.630	0.594	**0.611**
SemTab 2021 R1 [42]	0.926	-	0.926
SemTab 2021 R2 [42]	0.976	-	0.976
HT 2022 R1 [40]	0.955	-	0.954
HT 2022 R2 [40]	0.905	-	0.904

Dagobah has been evaluated on a sample of approximately 450 tables from SemTab 2020 Round 1, Round 2, and a sample of 300 tables from Round 3[33]. Despite the tables being generated similarly in 2021/2022, the

[32]LamAPI - lamapi.datai.disco.unimib.it
[33]stEELlm: Dagobah evaluation - github.com/unimib-datAI/steellm/tree/main/datasets/Dagobah-Inference

system's performance showed a decline compared to recent years. Although the core algorithm of Dagobah remains unchanged, many finer details have been optimised to achieve top results in the current year's challenge, which accounts for the reduced performance on older datasets.

8.2 Transferability of the proposed approach

To demonstrate the transferability of the proposal in the paper, we used the same prompt for fine-tuning the meta-llama/Llama-2-7b-hf model[34], developed by Meta. The model was aligned using the same template as stEELlm. Below are the obtained results:

Precision: 0.704, **Recall**: 0.556 and **F1**: 0.621

The superior performance of Mixtral 8x7B can be attributed to its optimised architecture and efficient utilisation of parameters. Despite having fewer parameters than larger models like Llama 2 7B, Mixtral 8x7B achieves superior results, as evidenced by its comparatively high precision score in this study. This indicates the model's proficiency in accurately identifying and selecting the correct candidate entities for annotation.

However, the lower recall score reveals limitations in the model's ability to identify all mentions requiring annotation comprehensively. This difficulty stems from challenges in recognising mentions and, critically, in generating the precise coordinates, specifically, row and column identifiers that locate each mention (*e.g.*, (1,2)=Q123), for these mentions. Notably, both models demonstrate commendable precision despite the inherent complexities of the task, particularly the accurate generation of corresponding Wikidata QIDs. This observation underscores the potential of these models for CEA, even with the current challenges in achieving complete recall.

8.3 Comparative analysis of execution times

The inference time of LLMs is faster than traditional algorithms due to the efficient parallel processing of input sequences, taking advantage of GPUs or TPUs. Additionally, they can handle multiple input sequences simultaneously in batches for increased parallelism and throughput. The execution times, affected by the length of the input, the number of tokens the model should generate, and the set of entities in the prompt, are shown in Fig. 6 and in Table 7.

In Table 4, the execution times of the different approaches are reported on the test dataset, which consists of 300 tables created to compare these methods. An example of a table of this kind is reported in Listing 16. The tests have been performed on a server with the following characteristics: Intel(R) Xeon(R) CPU E5-2650 0 @ 2.00GHz with 32 cores and 96GB of RAM. As can be observed, the model with the shortest execution time is Blink, though it does not achieve strong performance in terms of Precision, Recall, and F1 score. Analysing the models developed during the challenge, Dagobah exhibits significantly longer annotation times, while s-elBat demonstrates a good running time performance. s-elBat was previously tested few months, but now the tool has been improved in terms of performances and inference time. However, the neural networks employed by s-elBat involve far fewer hyper-parameters than generative models such as TableLlama and stEELlm. Both TableLlama and Mixtral 8x7B exhibit increased inference times, particularly Mixtral 8x7B, due to its significantly larger parameter count, which surpasses that of LLama 2 7B. However, despite the additional inference time, TableLlama and stEELlm demonstrate enhanced annotation performance, as evidenced by their higher F1 scores.

Listing 16. Table used for computing time complexity.

```
Charles and Theresa Roper House;historic house;1912-01-01;620 SW Alder Street, Newport, OR 97365, USA;
    ↪ Castellated Gothic
Frederick Armbruster Cottage;single-family detached home;1898-01-01;502 NE Tillamook Street, Portland, OR
    ↪ 97212, USA;Queen Anne style architecture
Charles and Theresa Roper House;folly;1912-01-01;620 SW Alder Street, Newport, OR 97365, USA;Castellated
    ↪ Gothic
```

[34]Huggingface: Meta LLama - huggingface.co/meta-llama/Llama-2-7b-hf

Table 6. Time Complexity of Mixtral 8x7B.

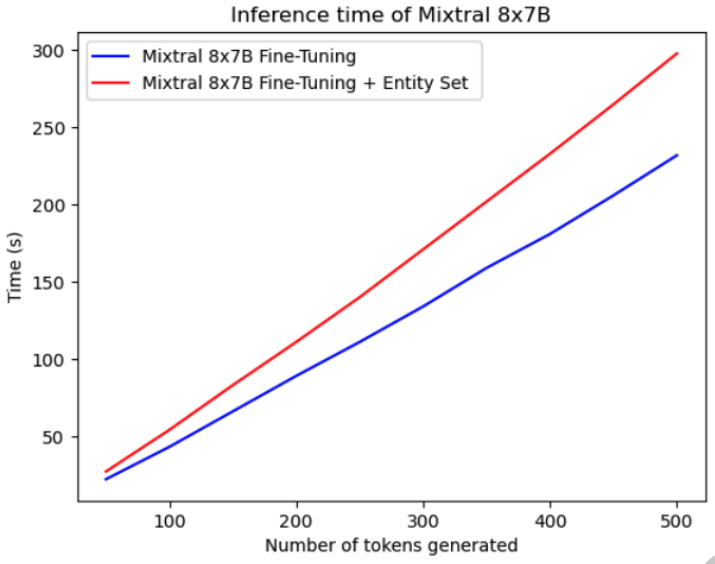

Table 7. Time Complexity in seconds.

Number of Tokens Generated	Mixtral 8x7B Fine-Tuning	Mixtral 8x7B Fine-Tuning + Entity Set
50	22	27
100	43	54
150	66	83
200	89	111
250	111	140
300	134	171
350	159	202
400	181	233
450	206	265
500	232	298

```
Charles T. Holt House;single-family detached home;1897-01-01;228 Holt St., Haw River, North Carolina;Queen Anne style architecture
```

8.4 Ablation study

To understand the impact of prompt perturbations, the stEELlm model has been fine-tuned by adding entity information in the prompt. The fine-tuning was performed by downsampling the initial dataset to 10000 tables for the training and 300 tables for the test set. This experimental setting was chosen to have many training and test examples without significantly impacting the cost of the A100 machine in Azure. The datasets and model responses are publicly available in the GitHub Repository[35]. The difference between each prompt is found in the single entity representation, reported in the Listing 17. The entity taken into account to show the different prompt representation is:

Listing 17. Entity example for different prompt templates.

```
{
    "entity": "Q4201789",
    "label": "Institute of Solid State Physics",
    "description": "research institution, located in the small town of Chernogolovka",
    "types": ["institute of the Russian Academy of Sciences"]
}
```

The entity representations are:

(1) **stEELlm**: *e.g., Q4201789 [Institute of Solid State Physics];*
(2) **stEELlm + Description**: *e.g., Q4201789 [Institute of Solid State Physics [DESCRIPTION] research institution, located in the small town of Chernogolovka];*

[35] Ablation experiments - https://github.com/unimib-datAI/steellm/tree/main/datasets/Steellm-Prompt-Perturbation

(3) **stEELlm + Types**: *e.g., Q4201789 [Institute of Solid State Physics [TYPES] institute of the Russian Academy of Sciences];*
(4) **stEELlm + Description + Types**: *e.g., Q4201789 [Institute of Solid State Physics [DESCRIPTION] research institution, located in the small town of Chernogolovka [TYPES] institute of the Russian Academy of Sciences].*

Table 8. Comparison of different enriched prompts in performing CEA task.

Model	Precision	Recall	F1-Score	Inference Time (s)
stEELlm	0.874	0.860	0.867	1620 (27 minutes)
stEELlm + Description	0.861	0.461	0.600	1756 (29 minutes)
stEELlm + Description + Types	0.866	0.117	0.206	1800 (30 minutes)
stEELlm + Types	0.926	0.128	0.224	1759 (29 minutes)

Table 8 shows that enriching the model increases annotation accuracy at the expense of a drop in recall and, consequently, in the F1 score. This behaviour may occur as the prompt becomes longer and more complex while the task, and therefore the expected output, remains equally challenging to generate. This is because the model must create one token for each character at a time to create the position (i,j) and the QID (Q1234). Despite the task's complexity, when the model succeeds in understanding the correct format for the output response and the proper position of the mention to annotate, it can reach SOTA results.

9 CONCLUSION AND FUTURE WORK

This paper presented stEELlm, an innovative STI approach that can efficiently and effectively annotate a set of tables with entities that refer to a KG by means of an LLM. Mainly, stEELlm enhances the performance of the LLM through targeted prompt engineering and integrating a candidate set of entities.

Initial results from prompting GPT-4 alone demonstrated that general pre-trained LLM struggled to handle the CEA task due to two key limitations: i) the LLMs intrinsic lack of factual knowledge in the context of STI, and the consequent ii) inability to generate accurate semantic annotations. While the task of CEA can be understood, further experiments involving the *Programming Prompt* template and Mixtral 8x7B produced sub-optimal results, often suffering from hallucinations and randomly generated IDs.

To address these challenges, we set a GS and employed various few-shot learning prompt techniques alongside fine-tuning the Mixtral 8x7B model approach to providing Wikidata IDs to the LLM. With the introduction of a candidate set of entities into the prompting phase. In this way, we were able to guide the LLMs more effectively, allowing them to disambiguate mentions with greater accuracy.

Our evaluation demonstrates that our model produces good annotations even on unseen datasets, unlike the heuristic-based approaches present in the SOTA, and in an efficient timeframe. For this reason, the use of LLMs for STI tasks is particularly advantageous in gaining a deeper understanding of tabular context. Compared to heuristic-based approaches, which implement ad hoc rules to achieve high performance on a specific GS, the proposed solution is more versatile, extending beyond the confines of the SemTab challenge.

Additionally, our ablation study revealed that enriching the model prompts with additional entity information (*e.g.*, descriptions and types) led to further improvements in annotation accuracy, albeit at the cost of recall. This reflects the balance between increasing model input complexity and ensuring the quality of the output.

Since stEELlm was trained to recognise mentions in tables using Wikidata entities, testing it with other KGs can lead to performance drops due to variations in ID formats, complicating the extraction of annotations. Achieving optimal performance would require fine-tuning the model to generate correct entity IDs for different KGs. Additionally, like other LLMs, stEELlm struggles with larger tables as it is designed for smaller contexts.

Enhancing its performance may involve using techniques like Retrieval-Augmented Generation (RAG) to process smaller data chunks.

In summary, in the context of addressing our research question regarding the employment of LLMs to perform the CEA task, our developed proposal stEELlm demonstrates the potential of fine-tuned LLMs for CEA tasks, particularly when supplemented with external knowledge of candidate entities sets. While the model still struggles with recall and large-scale table annotation, the results show promise for further exploration in improving generative models' ability to understand how to annotate correctly.

Future endeavours in this research direction involve exploring the use of Retrieval-Augmented Generation architecture, such as the approach described in [79]. Another potential avenue is the application of more sophisticated prompting techniques, such as chains of thought, which have demonstrated significant results in various domains [102].

REFERENCES

[1] N. Abdelmageed, J. Chen, V. Cutrona, V. Efthymiou, O. Hassanzadeh, M. Hulsebos, E. Jiménez-Ruiz, J. Sequeda, and K. Srinivas. Results of semtab 2022. *Semantic Web Challenge on Tabular Data to Knowledge Graph Matching*, 3320, 2022.

[2] N. Abdelmageed, J. Chen, V. Cutrona, V. Efthymiou, O. Hassanzadeh, M. Hulsebos, E. Jiménez-Ruiz, J. Sequeda, and K. Srinivas. Results of semtab 2022. *Semantic Web Challenge on Tabular Data to Knowledge Graph Matching*, 3320, 2022.

[3] N. Abdelmageed and S. Schindler. Jentab: Matching tabular data to knowledge graphs. In *SemTab@ ISWC*, pages 40–49, 2020.

[4] N. Abdelmageed and S. Schindler. Jentab meets semtab 2021's new challenges. In *SemTab@ ISWC*, pages 42–53, 2021.

[5] G. Agrawal, T. Kumarage, Z. Alghamdi, and H. Liu. Can knowledge graphs reduce hallucinations in LLMs? : A survey. In K. Duh, H. Gomez, and S. Bethard, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3947–3960, Mexico City, Mexico, June 2024. Association for Computational Linguistics.

[6] R. Avogadro and M. Cremaschi. Mantistable v: A novel and efficient approach to semantic table interpretation. In *SemTab@ ISWC*, pages 79–91, 2021.

[7] R. Avogadro, M. Cremaschi, F. D'adda, F. De Paoli, M. Palmonari, et al. Lamapi: a comprehensive tool for string-based entity retrieval with type-base filters. In *17th ISWC workshop on ontology matching (OM)*, 2022.

[8] R. Avogadro, F. D'Adda, and M. Cremaschi. Feature/vector entity retrieval and disambiguation techniques to create a supervised and unsupervised semantic table interpretation approach. *Knowledge-Based Systems*, 304:112447, 2024.

[9] R. Avogadro, F. D'Adda, and M. Cremaschi. Feature/vector entity retrieval and disambiguation techniques to create a supervised and unsupervised semantic table interpretation approach. *Knowledge-Based Systems*, 304:112447, 2024.

[10] R. Azzi, G. Diallo, E. Jiménez-Ruiz, O. Hassanzadeh, V. Efthymiou, J. Chen, and K. Srinivas. Amalgam: making tabular dataset explicit with knowledge graph. In *SemTab@ ISWC*, pages 9–16, 2020.

[11] W. Baazouzi, M. Kachroudi, and S. Faiz. Kepler-asi at semtab 2021. In *SemTab@ ISWC*, pages 54–67, 2021.

[12] F. Belotti, F. Dadda, M. Cremaschi, R. Avogadro, R. Pozzi, and M. Palmonari. Evaluating language models on entity disambiguation in tables. *arXiv e-prints*, pages arXiv–2408, 2024.

[13] C. S. Bhagavatula, T. Noraset, and D. Downey. Tabel: Entity linking in web tables. In *The Semantic Web - ISWC 2015*, pages 425–441, 2015.

[14] Y. Chabot, T. Labbé, J. Liu, and R. Troncy. Dagobah: An end-to-end context-free tabular data semantic annotation system. In *SemTab@ ISWC*, pages 41–48, 10 2019.

[15] J. Chen, E. Jiménez-Ruiz, I. Horrocks, and C. Sutton. Colnet: Embedding the semantics of web tables for column type prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 29–36, 2019.

[16] S. Chen, A. Karaoglu, C. Negreanu, T. Ma, J.-G. Yao, J. Williams, A. Gordon, and C.-Y. Lin. Linkingpark: An integrated approach for semantic table interpretation. In *SemTab@ ISWC*, pages 65–74, 2020.

[17] S. Chen, A. Karaoglu, C. Negreanu, T. Ma, J.-G. Yao, J. Williams, F. Jiang, A. Gordon, and C.-Y. Lin. Linkingpark: An automatic semantic table interpretation system. *Journal of Web Semantics*, 74:100733, 2022.

[18] N. Chirkova and S. Troshin. Codebpe: Investigating subtokenization options for large language model pretraining on source code, 2023.

[19] M. Cremaschi, R. Avogadro, A. Barazzetti, D. Chieregato, and E. Jiménez-Ruiz. Mantistable se: an efficient approach for the semantic table interpretation. In *SemTab@ ISWC*, pages 75–85, 2020.

[20] M. Cremaschi, R. Avogadro, and D. Chieregato. Mantistable: an automatic approach for the semantic table interpretation. *SemTab@ ISWC*, 2019:15–24, 2019.

[21] M. Cremaschi, R. Avogadro, D. Chieregato, et al. s-elbat: A semantic interpretation approach for messy table-s. In *SemTab@ ISWC*, pages 59–71, 2022.

[22] M. Cremaschi, F. De Paoli, A. Rula, and B. Spahiu. A fully automated approach to a complete semantic table interpretation. *Future Generation Computer Systems*, 112:478 – 500, 2020.

[23] M. Cremaschi, F. D'Adda, and S. Nocco. Mantistable ui: A web interface for comprehensive semantic table interpretation management. 2024.

[24] M. Cremaschi, A. Rula, A. Siano, and F. De Paoli. Mantistable: a tool for creating semantic annotations on tabular data. In *The Semantic Web: ESWC 2019 Satellite Events: ESWC 2019 Satellite Events, Portorož, Slovenia, June 2–6, 2019, Revised Selected Papers 16*, pages 18–23. Springer, 2019.

[25] M. Cremaschi, A. Rula, A. Siano, F. De Paoli, et al. Semantic table interpretation using mantistable. In *OM@ ISWC*, pages 195–196, 2019.

[26] M. Cremaschi, B. Spahiu, M. Palmonari, and E. Jimenez-Ruiz. Survey on semantic interpretation of tabular data: Challenges and directions. *arXiv preprint arXiv:2411.11891*, 2024.

[27] V. Cutrona, F. Bianchi, E. Jiménez-Ruiz, and M. Palmonari. Tough tables: Carefully evaluating entity linking for tabular data. In *International Semantic Web Conference*, pages 328–343. Springer, 2020.

[28] V. Cutrona, J. Chen, V. Efthymiou, O. Hassanzadeh, E. Jimenez-Ruiz, J. Sequeda, K. Srinivas, N. Abdelmageed, M. Hulsebos, D. Oliveira, and C. Pesquita. Results of semtab 2021. In *20th International Semantic Web Conference*, volume 3103, pages 1–12. CEUR Workshop Proceedings, March 2022.

[29] V. Cutrona, M. Ciavotta, F. D. Paoli, and M. Palmonari. ASIA: a tool for assisted semantic interpretation and annotation of tabular data. In *Proceedings of the ISWC 2019 Satellite Tracks*, volume 2456 of *CEUR Workshop Proceedings*, pages 209–212. CEUR-WS.org, 2019.

[30] I. Dasoulas, D. Yang, X. Duan, and A. Dimou. Torchictab: Semantic table annotation with wikidata and language models. In *CEUR Workshop Proceedings*, pages 21–37. CEUR Workshop Proceedings, 2023.

[31] X. Deng, H. Sun, A. Lees, Y. Wu, and C. Yu. Turl: Table understanding through representation learning. *ACM SIGMOD Record*, 51(1):33–40, 2022.

[32] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[33] V. Efthymiou, O. Hassanzadeh, M. Rodriguez-Muro, and V. Christophides. Matching web tables with knowledge base entities: From entity lookups to entity embeddings. In *The Semantic Web – ISWC 2017*, pages 260–277, 2017.

[34] B. Ell, S. Hakimov, P. Braukmann, L. Cazzoli, F. Kaupmann, A. Mancino, J. Altaf Memon, K. Rother, A. Saini, and P. Cimiano. Towards a large corpus of richly annotated web tables for knowledge base population. In *5th International Workshop on Linked Data for Information Extraction*, pages 2–13, 2017.

[35] Y. Eslahi, A. Bhardwaj, P. Rosso, K. Stockinger, and P. Cudré-Mauroux. Annotating web tables through knowledge bases: A context-based approach. In *2020 7th Swiss Conference on Data Science (SDS)*, pages 29–34, 2020.

[36] O. Hassanzadeh, N. Abdelmageed, M. Cremaschi, V. Cutrona, F. D'Adda, V. Efthymiou, B. Kruit, E. Lobo, N. Mihindukulasooriya, and N. H. Pham. Results of semtab 2024. In *CEUR Workshop Proceedings*, volume 3889, pages 1–11, 2024.

[37] O. Hassanzadeh, N. Abdelmageed, V. Efthymiou, J. Chen, V. Cutrona, M. Hulsebos, E. Jiménez-Ruiz, A. Khatiwada, K. Korini, B. Kruit, et al. Results of semtab 2023. In *CEUR Workshop Proceedings*, volume 3557, pages 1–14, 2023.

[38] S. Hegselmann, A. Buendia, H. Lang, M. Agrawal, X. Jiang, and D. Sontag. Tabllm: Few-shot classification of tabular data with large language models. In *International Conference on Artificial Intelligence and Statistics*, pages 5549–5581. PMLR, 2023.

[39] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[40] V.-P. Huynh, Y. Chabot, T. Labbé, J. Liu, and R. Troncy. From heuristics to language models: A journey through the universe of semantic table interpretation with dagobah. *Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab)*, 2022.

[41] V.-P. Huynh, Y. Chabot, and R. Troncy. Towards generative semantic table interpretation. In *VLDB Workshops*, 2023.

[42] V.-P. Huynh, J. Liu, Y. Chabot, F. Deuzé, T. Labbé, P. Monnin, and R. Troncy. Dagobah: Table and graph contexts for efficient semantic annotation of tabular data. *SemTab@ ISWC*, 2021.

[43] V.-P. Huynh, J. Liu, Y. Chabot, T. Labbé, P. Monnin, and R. Troncy. Dagobah: Enhanced scoring algorithms for scalable annotations of tabular data. In *SemTab@ ISWC*, pages 27–39, 2020.

[44] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

[45] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung. Survey of hallucination in natural language generation. *ACM Comput. Surv.*, 55(12), mar 2023.

[46] A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. d. l. Casas, E. B. Hanna, F. Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

[47] J. Jiang, K. Zhou, W. X. Zhao, and J.-R. Wen. Unikgqa: Unified retrieval and reasoning for solving multi-hop question answering over knowledge graph, 2023.

[48] X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, and Q. Liu. TinyBERT: Distilling BERT for natural language understanding. In T. Cohn, Y. He, and Y. Liu, editors, *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, Online, Nov. 2020. Association for Computational Linguistics.
[49] E. Jiménez-Ruiz, O. Hassanzadeh, V. Efthymiou, J. Chen, and K. Srinivas. Semtab 2019: Resources to benchmark tabular data to knowledge graph matching systems. In *The Semantic Web*, pages 514–530, Cham, 2020. Springer International Publishing.
[50] E. Jiménez-Ruiz, O. Hassanzadeh, V. Efthymiou, J. Chen, and K. Srinivas. Semtab 2019: Resources to benchmark tabular data to knowledge graph matching systems. In *The Semantic Web: 17th International Conference, ESWC 2020, Heraklion, Crete, Greece, May 31–June 4, 2020, Proceedings 17*, pages 514–530. Springer, 2020.
[51] E. Jimenez-Ruiz, O. Hassanzadeh, V. Efthymiou, J. Chen, K. Srinivas, and V. Cutrona. Results of semtab 2020. *CEUR Workshop Proceedings*, 2775:1–8, January 2020.
[52] E. Jiménez-Ruiz, O. Hassanzadeh, V. Efthymiou, J. Chen, K. Srinivas, and V. Cutrona. Results of semtab 2020. In *CEUR Workshop Proceedings*, volume 2775, pages 1–8, 2020.
[53] P. Ke, H. Ji, Y. Ran, X. Cui, L. Wang, L. Song, X. Zhu, and M. Huang. Jointgt: Graph-text joint representation learning for text generation from knowledge graphs, 2021.
[54] M. Kejriwal, C. A. Knoblock, and P. Szekely. *Knowledge graphs: Fundamentals, techniques, and applications.* MIT Press, 2021.
[55] D. Kim, H. Park, J. K. Lee, W. Kim, E. Jiménez-Ruiz, O. Hassanzadeh, V. Efthymiou, J. Chen, and K. Srinivas. Generating conceptual subgraph from tabular data for knowledge graph matching. In *SemTab@ ISWC*, pages 96–103, 2020.
[56] K. Korini and C. Bizer. Column type annotation using chatgpt. *arXiv preprint arXiv:2306.00745*, 2023.
[57] B. Kruit, P. Boncz, and J. Urbani. Extracting novel facts from tables for knowledge graph completion. In *The Semantic Web – ISWC 2019*, pages 364–381, Cham, 2019. Springer International Publishing.
[58] P. Li, Y. He, D. Yashar, W. Cui, S. Ge, H. Zhang, D. R. Fainman, D. Zhang, and S. Chaudhuri. Table-gpt: Table-tuned gpt for diverse table tasks, 2023.
[59] X. Li, S. Wang, W. Zhou, G. Zhang, C. Jiang, T. Hong, and P. Wang. Kgcode-tab results for semtab 2022. In *SemTab@ ISWC*, pages 37–44, 2022.
[60] Y. Li, J. Li, Y. Suhara, A. Doan, and W.-C. Tan. Deep entity matching with pre-trained language models. *VLDB*, 2020.
[61] G. Limaye, S. Sarawagi, and S. Chakrabarti. Annotating and searching web tables using entities, types and relationships. *Proc. VLDB Endow.*, 3(1-2):1338–1347, Sept. 2010.
[62] J. Liu, V.-P. Huynh, Y. Chabot, and R. Troncy. Radar station: Using kg embeddings for semantic table interpretation and entity disambiguation. In *ISWC 2022, October 23–27, 2022*, pages 498–515. Springer, 2022.
[63] T. Liu, F. Wang, and M. Chen. Rethinking tabular data understanding with large language models. *arXiv preprint arXiv:2312.16702*, 2023.
[64] S. Mangrulkar, S. Gugger, L. Debut, Y. Belkada, S. Paul, and B. Bossan. Peft: State-of-the-art parameter-efficient fine-tuning methods. *URL: https://github. com/huggingface/peft*, 2022.
[65] M. Marzocchi, M. Cremaschi, R. Pozzi, R. Avogadro, and M. Palmonari. Mammotab: a giant and comprehensive dataset for semantic table interpretation. *Proceedings of the SemTab2022*, 2022.
[66] I. Melnyk, P. Dognin, and P. Das. Grapher: Multi-stage knowledge graph construction using pretrained language models. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.
[67] H. Morikawa. Semantic table interpretation using lod4all. *SemTab@ ISWC*, 2019:49–56, 2019.
[68] V. Mulwad, T. Finin, and A. Joshi. Semantic message passing for generating linked data from tables. In *The Semantic Web – ISWC 2013*, pages 363–378. Springer Berlin Heidelberg, 2013.
[69] V. Mulwad, T. Finin, Z. Syed, and A. Joshi. T2ld: Interpreting and representing tables as linked data. In *ISWC Posters & Demonstrations Track*, ISWC-PD'10, pages 25–28, Aachen, Germany, Germany, 2010. CEUR-WS.org.
[70] V. Mulwad, T. W. Finin, and A. Joshi. Automatically generating government linked data from tables. In *AAAI 2011*, 2011.
[71] S. Neumaier, J. Umbrich, J. X. Parreira, and A. Polleres. Multi-level semantic labelling of numerical values. In *The Semantic Web – ISWC 2016*, pages 428–445, Cham, 2016. Springer International Publishing.
[72] P. Nguyen, N. Kertkeidkachorn, R. Ichise, and H. Takeda. Mtab: Matching tabular data to knowledge graph using probability models. *arXiv preprint arXiv:1910.00246*, 2019.
[73] P. Nguyen, I. Yamada, N. Kertkeidkachorn, R. Ichise, and H. Takeda. Mtab4wikidata at semtab 2020: Tabular data annotation with wikidata. *SemTab@ ISWC*, 2775:86–95, 2020.
[74] P. Nguyen, I. Yamada, N. Kertkeidkachorn, R. Ichise, and H. Takeda. Semtab 2021: Tabular data annotation with mtab tool. In *SemTab@ ISWC*, pages 92–101, 2021.
[75] D. Oliveira and M. d'Aquin. Adog-annotating data with ontologies and graphs. *SemTab@ ISWC*, 2019:1–6, 2019.
[76] OpenAI. Gpt-4 technical report, 2023.
[77] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. F. Christiano, J. Leike, and R. Lowe. Training language models to follow

instructions with human feedback. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc., 2022.

[78] M. Palmonari, M. Ciavotta, F. De Paoli, A. Košmerlj, and N. Nikolov. Ew-shopp project: Supporting event and weather-based data analytics and marketing along the shopper journey. In *Advances in Service-Oriented and Cloud Computing*, pages 187–191, Cham, 2020. Springer International Publishing.

[79] F. Pan, M. Canim, M. Glass, A. Gliozzo, and J. Hendler. End-to-end table question answering via retrieval-augmented generation. *arXiv preprint arXiv:2203.16714*, 2022.

[80] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, and X. Wu. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–20, 2024.

[81] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, and X. Wu. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering*, page 1–20, 2024.

[82] R. Peeters and C. Bizer. Using chatgpt for entity matching. *arXiv preprint arXiv:2305.03423*, 2023.

[83] F. Petroni, T. Rocktäschel, P. Lewis, A. Bakhtin, Y. Wu, A. H. Miller, and S. Riedel. Language models as knowledge bases?

[84] R. Rafailov, A. Sharma, E. Mitchell, S. Ermon, C. D. Manning, and C. Finn. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*, 2023.

[85] D. Rao, P. McNamee, and M. Dredze. Entity linking: Finding extracted entities in a knowledge base. In *Multi-source, Multilingual Information Extraction and Summarization*, pages 93–115, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[86] L. Ratinov and D. Roth. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 147–155, Boulder, Colorado, June 2009. Association for Computational Linguistics.

[87] D. Ritze, O. Lehmberg, and C. Bizer. Matching html tables to dbpedia. In *5th International Conference on Web Intelligence, Mining and Semantics*, WIMS '15, pages 10:1–10:6, New York, NY, USA, 2015. ACM.

[88] D. Schicchi, A. Upadhyaya, M. Fisichella, and D. Taibi. Using chatgpt to enhance students' behavior in social media via the moral foundation theory. In *Proceedings of the First International Workshop on High-performance Artificial Intelligence Systems in Education co-located with 22nd International Conference of the Italian Association for Artificial Intelligence (AIxIA 2023)*, volume 3605 of *CEUR Workshop Proceedings*, 2023.

[89] R. Shigapov, P. Zumstein, J. Kamlah, L. Oberländer, J. Mechnich, and I. Schumm. bbw: Matching csv to wikidata via meta-lookup. In *CEUR Workshop Proceedings*, volume 2775, pages 17–26. RWTH, 2020.

[90] A. Shigarov. Table understanding: Problem overview. *WIREs Data Mining and Knowledge Discovery*, 13(1):e1482, 2023.

[91] B. Steenwinckel, F. De Turck, and F. Ongenae. Magic: Mining an augmented graph using ink, starting from a csv. In *SemTab@ ISWC*, pages 68–78, 2021.

[92] B. Steenwinckel, G. Vandewiele, F. De Turck, and F. Ongenae. Csv2kg: Transforming tabular data into semantic knowledge. *SemTab, ISWC Challenge*, 2019.

[93] Y. Suhara, J. Li, Y. Li, D. Zhang, Ç. Demiralp, C. Chen, and W.-C. Tan. Annotating columns with pre-trained language models. In *Proceedings of the 2022 International Conference on Management of Data*, pages 1493–1503, 2022.

[94] Z. Syed, T. Finin, V. Mulwad, and A. Joshi. Exploiting a web of semantic data for interpreting tables. In *Proceedings of the Second Web Science Conference*, volume 5, 2010.

[95] A. Thawani, M. Hu, E. Hu, H. Zafar, N. T. Divvala, A. Singh, E. Qasemi, P. A. Szekely, and J. Pujara. Entity linking to knowledge graphs to infer column types and properties. *SemTab@ ISWC*, 2019:25–32, 2019.

[96] H. Tian, C. Gao, X. Xiao, H. Liu, B. He, H. Wu, H. Wang, and F. Wu. Skep: Sentiment knowledge enhanced pre-training for sentiment analysis. *arXiv preprint arXiv:2005.05635*, 2020.

[97] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample. Llama: Open and efficient foundation language models, 2023.

[98] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

[99] S. Tyagi and E. Jimenez-Ruiz. Lexma: Tabular data to knowledge graph matching using lexical techniques. In *CEUR Workshop Proceedings*, volume 2775, pages 59–64, 2020.

[100] J. Wang, H. Wang, Z. Wang, and K. Q. Zhu. Understanding tables on the web. In *Proceedings of the 31st International Conference on Conceptual Modeling*, ER'12, pages 141–155. Springer-Verlag, 2012.

[101] X. Wang, T. Gao, Z. Zhu, Z. Zhang, Z. Liu, J. Li, and J. Tang. KEPLER: A Unified Model for Knowledge Embedding and Pre-trained Language Representation. *Transactions of the Association for Computational Linguistics*, 9:176–194, 03 2021.
[102] J. Wei, X. Wang, D. Schuurmans, M. Bosma, b. ichter, F. Xia, E. Chi, Q. V. Le, and D. Zhou. Chain-of-thought prompting elicits reasoning in large language models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc., 2022.
[103] G. Weikum, X. L. Dong, S. Razniewski, and F. M. Suchanek. Machine knowledge: Creation and curation of comprehensive knowledge bases. *Found. Trends Databases*, 10(2-4):108–490, 2021.
[104] L. Wu, F. Petroni, M. Josifoski, S. Riedel, and L. Zettlemoyer. Scalable zero-shot entity linking with dense entity retrieval. In B. Webber, T. Cohn, Y. He, and Y. Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6397–6407, Online, Nov. 2020. Association for Computational Linguistics.
[105] L. Yang, H. Chen, Z. Li, X. Ding, and X. Wu. Give us the facts: Enhancing large language models with knowledge graphs for fact-aware language modeling. *IEEE Transactions on Knowledge and Data Engineering*, 36(07):3091–3110, jul 2024.
[106] L. Yang, S. Shen, J. Ding, and J. Jin. Gbmtab: A graph-based method for interpreting noisy semantic table to knowledge graph. In *SemTab@ ISWC*, pages 32–41, 2021.
[107] L. Yao, C. Mao, and Y. Luo. Kg-bert: Bert for knowledge graph completion, 2019.
[108] L. Yao, C. Mao, and Y. Luo. Kg-bert: Bert for knowledge graph completion. *arXiv preprint arXiv:1909.03193*, 2019.
[109] H. Zhang, H. Song, S. Li, M. Zhou, and D. Song. A survey of controllable text generation using transformer-based pre-trained language models. *ACM Comput. Surv.*, 56(3), oct 2023.
[110] M. Zhang and K. Chakrabarti. Infogather+: semantic matching and annotation of numeric and time-varying attributes in web tables. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, page 145–156, New York, NY, USA, 2013. Association for Computing Machinery.
[111] S. Zhang and K. Balog. Ad hoc table retrieval using semantic similarity. In *International World Wide Web Conference*, WWW '18, page 1553–1562, Republic and Canton of Geneva, CHE, 2018.
[112] S. Zhang, E. Meij, K. Balog, and R. Reinanda. Novel entity discovery from web tables. In *Proceedings of The Web Conference 2020*, WWW '20, page 1298–1308, New York, NY, USA, 2020. Association for Computing Machinery.
[113] Z. Zhang. Effective and efficient semantic table interpretation using tableminer+. *Semantic Web*, 8(6):921–957, 2017.
[114] Z. Zhang, X. Han, Z. Liu, X. Jiang, M. Sun, and Q. Liu. Ernie: Enhanced language representation with informative entities. *arXiv preprint arXiv:1905.07129*, 2019.
[115] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*, 2023.

Received 17 February 2024; revised 10 December 2024; accepted 1 February 2025