Column Property Annotation using Large Language Models

Keti Korini$^{1[0000-0002-2158-0070]}$ and Christian Bizer$^{1[0000-0003-2367-0237]}$

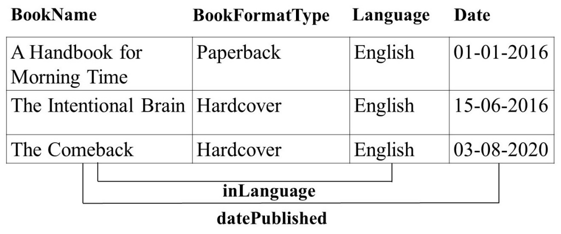
Data and Web Science Group, University of Mannheim, Mannheim, Germany
kkorini@uni-mannheim.de, christian.bizer@uni-mannheim.de

Abstract. Column property annotation (CPA), also known as column relationship prediction, is the task of predicting the semantic relationship between two columns in a table given a set of candidate relationships. CPA annotations are used in downstream tasks such as data search, data integration, or knowledge graph enrichment. This paper is the first to explore the usage of large language models (LLMs) for the CPA task. We experiment with different zero-shot prompts for the CPA task which we evaluate using two OpenAI models (GPT-3.5, GPT-4) and the open-source model SOLAR. We find GPT-3.5 to be quite sensitive to variations of the prompt, while GPT-4 reaches a high performance independent of the variation of the prompt. We further explore the scenario where training data for the CPA task is available and can be used for selecting demonstrations or fine-tuning the model. We show that a fine-tuned GPT-3.5 model outperforms a RoBERTa model that was fine-tuned on the same data by 11% in F1. Comparing in-context learning via demonstrations and fine-tuning shows that the fine-tuned GPT-3.5 performs 9% F1 better than the same model given demonstrations. The fine-tuned GPT-3.5 model also outperforms zero-shot GPT-4 by around 2% F1 for the dataset on which is was fine-tuned, while not generalizing that good to other CPA datasets.

Keywords: Table Annotation · Large Language Models · Column Property Annotation.

1 Introduction

Table annotation is the task of annotating elements of a table using pre-defined vocabularies in order to discover their semantics [14]. It consists of several sub-tasks that aim at discovering the semantics of different elements of the table. Two of the sub-tasks are column property annotation (CPA) which focuses on discovering the semantic relationship between two columns, and column type annotation (CTA) which aims at discovering the semantic types of entities contained in a column. An example of both tasks is presented in Figure 1. The example table contains book entities, the names of which are contained in the first column and some of its attributes are contained in the other three columns. The goal of a CTA system is to discover the types of each column separately, for example the last column contains dates therefore the CTA label assigned to this column by the system would be “Date”. On the other hand, the goal of a CPA system is to annotate the relationship of the columns with the first column of the table, also referred to as the subject column [21]. As the last column contains the dates of publishing of the book entities listed in the first column, a CPA system would annotate this relationship with the label “datePublished”.



BookName	BookFormatType	Language	Date
A Handbook for Morning Time	Paperback	English	01-01-2016
The Intentional Brain	Hardcover	English	15-06-2016
The Comeback	Hardcover	English	03-08-2020

inLanguage
datePublished

Fig. 1. Example of CTA and CPA labels. CTA labels are shown on top of the table columns, while CPA labels are shown below the table.

Early statistical approaches [12, 21] use a maximum likelihood or maximize joint probabilities to assign CTA and CPA labels to columns or pair of columns. Later systems [7, 15, 16] use a Knowledge Base (KB) such as DBpedia [1] to first match the entities in columns to entities in the KB and consider the KB class of the entity as its column type while considering the KB properties of the entities as potential CPA labels. Cannaviccio et al. [2] use a combination of language modeling and using a KB. They train a language model on a Web corpus to help in re-ranking the candidate KB properties that can be used to classify the relationships between the columns. Pre-trained language model-based methods use a pre-trained language model (PLM) such as BERT [4] and fall into two groups of approaches: methods that learn tabular embeddings such as TURL [3] where the authors propose an architecture that learns cell representations that can be used for predicting CTA and CPA labels, or works that fine-tune PLMs such as the state-of-the-art DODUO [19] which experiments with table serialization and a multi-task learning architecture for the CTA and CPA task.

With the advancements in large language models (LLMs) and the release of ChatGPT [17] and LLaMa-2 [20] models, research has started to explore prompt designs for table tasks as well as fine-tuning these models on table tasks. In [9], prompt designs on the CTA task are explored, where adding instructions to the prompt for the GPT-3.5 model gave the model some reasoning directions and boosted its performance on the task. In this work, we also test instruction-based prompt designs on the task of CPA. In Chorus [8], different table tasks are explored, including CTA. The authors test instructions in their prompts as well and introduce the concept of *anchoring* to remap to the original label space the answers generated by the model that are not part of the label space. ArcheType [5] fine-tunes a LLaMa-7B model on the CTA task and compares its results to two

PLM baselines. Table-GPT [11] fine-tunes the *text-davinci-002* GPT-3.5 model using a combination of unsupervised table tasks such as row/column filtering, row/column sorting as well as supervised table tasks such as schema matching and entity matching. They show that their fine-tuned model generalizes to other unseen tasks. In our work, we also fine-tune a GPT-3.5 model, *gpt-3.5-0613*, on singular tasks CTA and CPA as well as their combination to test if fine-tuning on both tasks simultaneously provides a better generalization ability than fine-tuning on the tasks separately. In TableLlama [23] a LLaMa-7B model is fine-tuned on 6 table tasks two of which are CTA and CPA and the resulting model is evaluated on seen and unseen tasks. Some initial steps towards the exploration of the CPA task using LLMs are taken in TableLlama, however the zero-shot scenario still remains unexplored. We aim to fill this gap in this paper. The main contributions of this paper are:

1. We are the first to explore prompt designs for the CPA task in a zero-shot and few-shot setting in contrast to existing work which explore fine-tuning LLMs for this task amongst other tasks.
2. Using different zero-shot prompt designs, we analyze the performance and prompt sensitivity of GPT-3.5, GPT-4, and SOLAR-70B for the CPA task.
3. Existing research has only fine-tuned for the CPA task as one task amongst others in a multi-task learning setting. In contrast, we explore the effect of fine-tuning *gpt-3.5-0613* exclusively for the CPA task and explore how the fine-tuned model generalizes to other datasets as well as to the CTA task.

2 Experimental Setup

This section introduces our experimental setup. We make the code and data available on GitHub[1] so that all our experiments can be replicated.

Datasets. We use two datasets for the experiments on the CPA task. The first dataset is SOTAB V2 CPA [10] which consists of tables whose topics range across 17 domains, including books, products, local businesses etc. Its test set consists of 595 tables with 2,340 columns annotated using 108 Schema.org[2] terms which are manually verified. The second dataset is the T2Dv2 CPA dataset. The dataset was originally published by Ritze et al. [18] and we use the manually verified version[3], the test set of which consists of 80 tables from domains such as animals, book, country etc. labeled using 48 terms from the DBpedia properties.

Additionally, for the fine-tuning experiments we use two more datasets for the evaluation of the CTA task. SOTAB V2 CTA [10] consists of tables with topics ranging over 17 domains including movies, music albums, events etc. Its test set consists of 609 tables where 1851 columns are labeled using 82 Schema.org terms and the annotation is manually verified. Lastly, we build the T2Dv2 CTA dataset by using T2Dv2 CPA's tables where we map the DBpedia properties to

[1] https://github.com/wbsg-uni-mannheim/TabAnnGPT
[2] https://schema.org/
[3] https://webdatacommons.org/structureddata/smb/

DBpedia classes to generate the CTA labels. Statistics about all datasets used can be found in Table 1. For training in our experiments, we do not use the original large training sets for SOTAB V2 CTA and CPA, but we use sampled train sets to explore the scenario where less training data is available.

Table 1. Statistics of datasets used.

Dataset	**Original Train**		**Sampled Train**		**Test**		**Labels**
	Tables	Columns	Tables	Columns	Tables	Columns	
SOTAB V2 CTA	44,769	116,887	1199	1640	609	1,851	82
T2Dv2 CTA	74	146	-	-	71	145	16
SOTAB V2 CPA	29,158	109,994	1264	2160	565	2,340	108
T2Dv2 CPA	81	170	-	-	82	166	48

Language models. The LLMs that we test are two of OpenAI's GPT models[4], *gpt-3.5-turbo-0125* and *gpt-4-0125-preview* and one open-source model SOLAR-70B[5], which is a LLaMa-2-70B [20] fine-tuned model. In our experiments, we will refer to these models as GPT-3.5, GPT-4 and SOLAR respectively. To build our prompt templates and to access OpenAI's models we use the *Langchain*[6] library, while for using the open-source model we use the *Huggingface transformers*[7] library. In order to make our experiments reproducible, we set the temperature of the models to 0.

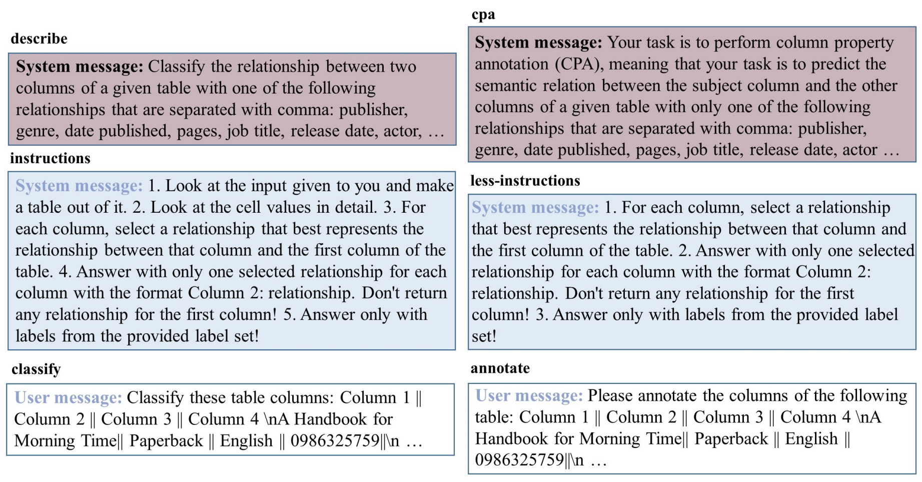
Evaluation Setup. For all the selected data we use a multi-class classification setup, and following previous work we report Micro-F1 as evaluation metric due to the imbalance that exists in the different classes that are present in the datasets. For the answers that the model generates that cannot be directly mapped to the label set, we consider them as errors and we do not re-map them to our label space. We refer to these answers as OOV (out of vocabulary).

3 Comparison of Zero-shot Prompts

We start designing the CPA prompts by distinguishing three main parts in the prompts: task description, instructions, and classification sentence. The task description part aims at describing the CPA task to the model. In the instructions part we aim at writing some simple instructions that can help the model follow the CPA tasks' steps and inform the model of a preferred format for generating its answer. In the last part, we test how classification words can influence the answer of the model. Two example prompts are shown in Figure 2. The prompt on the left contains as its first message a formulation of the task part where we only *describe* the CPA task without mentioning the name of the task. It is

[4] https://platform.openai.com/docs/models
[5] https://huggingface.co/upstage/SOLAR-0-70b-16bit
[6] https://www.langchain.com/
[7] https://github.com/huggingface/transformers

followed by a five-step *instructions* part where we inform the model that the input is a table and the answer should be returned in a required format. In the final message we ask the model to classify the table columns and pass the first five rows of a table in a markdown format. If in the first five rows some missing values are present, we fill the cells using values from the rest of the rows. The prompt on the right contains in the first message a task formulation where the *cpa* task is mentioned and explained. It is followed by a second message that contains *less instructions* than the prompt on the left, where we have removed the first two steps. Finally in its last message we test the keyword *annotate* to ask the model to return the labels for the CPA task. For the last message, we also test the words *determine* and classification of *relationships*.



describe

System message: Classify the relationship between two columns of a given table with one of the following relationships that are separated with comma: publisher, genre, date published, pages, job title, release date, actor, …

instructions

System message: 1. Look at the input given to you and make a table out of it. 2. Look at the cell values in detail. 3. For each column, select a relationship that best represents the relationship between that column and the first column of the table. 4. Answer with only one selected relationship for each column with the format Column 2: relationship. Don't return any relationship for the first column! 5. Answer only with labels from the provided label set!

classify

User message: Classify these table columns: Column 1 || Column 2 || Column 3 || Column 4 \nA Handbook for Morning Time|| Paperback || English || 0986325759||\n …

cpa

System message: Your task is to perform column property annotation (CPA), meaning that your task is to predict the semantic relation between the subject column and the other columns of a given table with only one of the following relationships that are separated with comma: publisher, genre, date published, pages, job title, release date, actor …

less-instructions

System message: 1. For each column, select a relationship that best represents the relationship between that column and the first column of the table. 2. Answer with only one selected relationship for each column with the format Column 2: relationship. Don't return any relationship for the first column! 3. Answer only with labels from the provided label set!

annotate

User message: Please annotate the columns of the following table: Column 1 || Column 2 || Column 3 || Column 4 \nA Handbook for Morning Time|| Paperback || English || 0986325759||\n …

Fig. 2. Example of two prompt designs.

Results. Table 2 reports the results for prompting the models with each combination of the three building blocks described in the above paragraph. In the cases where *cpa* is not mentioned the *describe* task description is used. The results show that including the definition of the CPA task into the prompt does not help the GPT-3.5 model. This can be seen from the combinations of the *cpa* prompt which are all below the score of 61%. In opposite, SOLAR seems to benefit from including the task name and achieves the highest score in this case. Considering the instructions, including more or less instructions seems to not have much effect as the gap between using *instructions* and *less-instructions* is small, and the results seem to be influenced more by the classification message. We can observe, mostly from the results of T2Dv2, that simply changing the wording in the classification message ranges the Micro-F1 score by up to 13% when comparing the *classify* and the *relationships* keyword. In SOTAB V2, this gap ranges from 2-4% for GPT-3.5. Regarding SOLAR, for both datasets

Table 2. Micro-F1 zero-shot CPA prompt designs results.

Prompt	**SOTAB V2 CPA**			**T2Dv2 CPA**		
	GPT-3.5	**GPT-4**	**SOLAR**	**GPT-3.5**	**GPT-4**	**SOLAR**
instr-classify	64.99	80.31	49.56	56.64	80.36	75.23
cpa-instr-classify	60.14	80.38	53.59	52.55	79.76	69.72
instr-annotate	66.06	79.88	50.56	65.08	80.00	77.30
instr-determine	64.76	80.19	50.25	64.43	82.35	77.68
instr-relationships	66.42	79.30	50.58	73.12	81.48	72.56
less-instr-classify	62.86	80.19	52.57	60.50	78.55	76.83
less-instr-annotate	65.67	79.78	52.61	62.33	81.57	75.38
less-instr-determine	64.82	79.51	52.02	56.34	81.48	73.94
less-instr-relationships	66.49	80.07	52.13	73.58	79.76	71.60
cpa-less-instr-classify	56.73	81.36	53.60	50.56	80.86	78.29
Prompt sensitivity	3.95	0.64	1.30	7.49	1.06	2.60

we can observe a slightly higher score when the *cpa* task formulation is used in combination with less instructions. Overall, the results show a prompt sensitivity of 3.95 and 7.49 in the case of GPT-3.5, while when looking at the results of SOLAR and GPT-4, we notice a lower sensitivity to the different formulations, especially for GPT-4 which is the most stable model amongst the three.

Table 3. Few-shot results using two methods for selecting demonstrations.

Method	**shots**	**SOTAB V2 CPA**			**T2Dv2 CPA**		
		GPT-3.5	**GPT-4**	**SOLAR**	**GPT-3.5**	**GPT-4**	**SOLAR**
random	1	68.99	81.24	53.42	76.69	84.24	78.55
random	5	70.23	82.45	-	78.55	83.38	-
similar	1	70.68	82.62	58.21	76.13	84.77	82.67
similar	5	74.39	84.10	-	76.82	86.09	-

4 In-Context Learning via Demonstrations

There exist some works that explore different ways of selecting demonstrations for in-context learning [13, 22]. To conduct few-shot experiments for CPA we employ two methods for selecting demonstrations: randomly and based on similarity. In the random method, we simply pick randomly a number of demonstrations to pass to the model. In the similarity method, using the embedding model *text-embedding-ada-002*[8], we embed the test examples without labels as well as the available training examples and select for each test example a number of most similar examples determined using cosine similarity. The prompt that we use for running the few-shot experiments is the *less-instr-relationships* prompt

[8] https://platform.openai.com/docs/models/embeddings

which overall gave a decent score for all models. To perform few-shot, we pass the demonstrations with the help of a message with the role of user that contains the demonstration table an an assistant role message that contains the classification output for the demonstration. These messages are passed to the model before passing the last user message which contains the test table. The results of the few-shot experiments are listed in Table 3. Due to memory issues, we could run SOLAR only in a one-shot setting. From the table, we can observe that in the case of GPT-3.5 the similarity method outperforms the random method by at least 2%, while when using GPT-4 this difference again becomes very small.

5 Fine-Tuning the LLM

We fine-tune the *gpt-3.5-0613* model using the sampled training sets of SOTAB V2 CTA and CPA. We fine-tune the model with different combinations of training sets: First, we fine-tune the model using only the SOTAB V2 CTA training set. Second, we fine-tune the model only on the CPA dataset therefore only on the CPA task. As third approach, we fine-tune the model using the CTA and CPA datasets from the two previous steps merged together to fine-tune on both tasks simultaneously. As the last approach, we fine-tune again on both tasks but with both datasets halved to make the number of fine-tuning examples more comparable with the first two steps. We refer to these models as *cta-ft*, *cpa-ft*, *cta-cpa-ft* and *cta-cpa-ft-small* respectively.

Table 4. Fine-tuning results on CTA and CPA compared to GPT-3.5 zero-shot results.

	SOTABV2 CTA	**SOTABV2 CPA**	**T2Dv2 CTA**	**T2Dv2 CPA**
zero-shot	64.35	66.49	69.85	73.58
cta-ft	81.22	63.98	69.43	72.34
cpa-ft	70.95	83.55	65.74	72.95
cta-cpa-ft	82.01	84.08	72.92	76.36
cta-cpa-small-ft	80.35	80.80	71.97	69.51

Results. The results of fine-tuning are shown in Table 4. From them we conclude that fine-tuning for a specific task greatly increases the Micro-F1 score for the task and dataset that was used for fine-tuning, while giving small increases to the dataset from the same domain, and decreasing the performance on both unseen datasets T2Dv2. When fine-tuning on both tasks with the larger set, we observe that the model generalizes better to all datasets and there is an increase in performance in all cases except for one. On the other hand, when this set of CTA and CPA examples are less, the fine-tuned model performs good on only the datasets that have been used for fine-tuning and not for the unseen T2Dv2 datasets. By comparing the results in Table 3 and Table 4, we can conclude that when using GPT-3.5 it is more beneficial to use the training set for fine-tuning the model rather than using the training set as a pool for demonstrations as fine-tuning reaches 9% more in F1 score. In addition, fine-tuning also helps reduce the number of OOV answers which are over 100 when GPT-3.5 is used in the zero-shot setting to only around 20-30.

5.1 Comparison to PLM-baselines

We compare the previous zero-shot prompts for GPT-3.5, GPT-4 and fine-tuned GPT-3.5 to three baselines. The first baseline is a fine-tuned RoBERTa model with the maximum token length set to 512 and a batch size of 32. The pairs of columns are concatenated together and passed to RoBERTa which we train for 30 epochs. The second baseline TURL [3] is pre-trained using table corpora and uses TinyBERT [6] in its architecture. For this baseline we use CrossEntropy as a loss function and train the model for 50 epochs. The last PLM baseline DODUO [19] uses a new serialization method where a single table is serialized into one sequence. For this model, we use a batch size of 32 and run training for 30 epochs. We run all the baselines three times with three different random seeds and report their average. For all models we use a learning rate of 5e-5.

Table 5. PLM baselines' results compared to CPA LLMs' results.

Method	shots	SOTAB V2 CPA	shots	T2Dv2 CPA
GPT-3.5	0	66.49	0	73.58
GPT-4	0	81.43	0	82.35
FT GPT-3.5	2160	83.55	2160	72.95
TURL	2160	60.75	170	59.23
DODUO	10,496	70.34	170	4.08
RoBERTa	2160	71.45	170	81.52

Results. The results of the PLM baselines are summarized in Table 5. Comparing RoBERTa and GPT-3.5 which were both fine-tuned on the SOTAB V2 CPA dataset, GPT-3.5 achieves 11% higher in Micro-F1 than RoBERTa. On the other hand, for the T2Dv2 dataset, GPT-3.5 does not generalize well and the score compared to RoBERTa fine-tuned with the full T2Dv2 training set is 8% lower. Comparing to TURL, fine-tuned GPT-3.5 achieves 23% higher Micro-F1, while comparing it to DODUO which is fine-tuned with more data, GPT-3.5 still achieves 11% more in F1.

6 Conclusion

In this work, we explore different prompt designs for the task of CPA and observe that LLMs reach a good performance of around 80% with GPT-4 for the datasets tested. Compared to the PLM baselines, the fine-tuned model and GPT-4 outperform their performance by up to 11% on the SOTAB V2 dataset. Finally, in the scenario where training data is available for GPT-3.5, we conclude it is better to use the available training data for fine-tuning rather than as a pool for choosing demonstrations in a few-shot setting with a 9% better Micro-F1.

References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: Dbpedia: A nucleus for a web of open data. In: international semantic web conference. pp. 722–735. Springer (2007)
2. Cannaviccio, M., Barbosa, D., Merialdo, P.: Towards annotating relational data on the web with language models. In: Proceedings of the 2018 World Wide Web Conference. pp. 1307–1316 (2018)
3. Deng, X., Sun, H., Lees, A., Wu, Y., Yu, C.: TURL: Table understanding through representation learning. Proceedings of the VLDB Endowment **14**(3), 307–319 (Nov 2020)
4. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1. pp. 4171–4186 (Jun 2019)
5. Feuer, B., Liu, Y., Hegde, C., Freire, J.: Archetype: A novel framework for open-source column type annotation using large language models. arXiv preprint arXiv:2310.18208 (2023)
6. Jiao, X., Yin, Y., Shang, L., Jiang, X., Chen, X., Li, L., et al.: TinyBERT: Distilling BERT for natural language understanding. In: Findings of the Association for Computational Linguistics EMNLP 2020. pp. 4163–4174 (Nov 2020)
7. Jiménez-Ruiz, E., Hassanzadeh, O., Efthymiou, V., Chen, J., Srinivas, K.: SemTab 2019: Resources to Benchmark Tabular Data to Knowledge Graph Matching Systems. In: Proceedings of The Semantic Web. pp. 514–530 (2020)
8. Kayali, M., Lykov, A., Fountalis, I., Vasiloglou, N., Olteanu, D., Suciu, D.: Chorus: Foundation models for unified data discovery and exploration. arXiv preprint arXiv:2306.09610 (2023)
9. Korini, K., Bizer, C.: Column type annotation using chatgpt. In: Joint proceedings of workshops at the 49th International Conference on Very Large Data Bases (VLDB 2023) (2023)
10. Korini, K., Peeters, R., Bizer, C.: SOTAB: The WDC Schema. org Table Annotation Benchmark. Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab), CEUR-WS. org (2022)
11. Li, P., He, Y., Yashar, D., Cui, W., Ge, S., Zhang, H., et al.: Table-gpt: Table-tuned gpt for diverse table tasks. arXiv preprint arXiv:2310.09263 (2023)
12. Limaye, G., Sarawagi, S., Chakrabarti, S.: Annotating and searching web tables using entities, types and relationships. Proceedings of the VLDB Endowment **3**(1-2), 1338–1347 (2010)
13. Liu, J., Shen, D., Zhang, Y., Dolan, B., Carin, L., Chen, W.: What makes good in-context examples for gpt-3? arXiv preprint arXiv:2101.06804 (2021)
14. Liu, J., Chabot, Y., Troncy, R., Huynh, V.P., et al.: From tabular data to knowledge graphs: A survey of semantic table interpretation tasks and methods. Journal of Web Semantics **76** (2023)
15. Liu, J., Troncy, R.: Dagobah: an end-to-end context-free tabular data semantic annotation system. SemTab@ ISWC (2019)
16. Nguyen, P., Kertkeidkachorn, N., Ichise, R., Takeda, H.: Mtab: Matching tabular data to knowledge graph using probability models. arXiv preprint arXiv:1910.00246 (2019)

17. Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., et al.: Training language models to follow instructions with human feedback. Advances in Neural Information Processing Systems **35**, 27730–27744 (2022)
18. Ritze, D., Lehmberg, O., Bizer, C.: Matching html tables to dbpedia. In: Proceedings of the 5th international conference on web intelligence, mining and semantics. pp. 1–6 (2015)
19. Suhara, Y., Li, J., Li, Y., Zhang, D., Demiralp, Ç., Chen, C., et al.: Annotating columns with pre-trained language models. In: Proceedings of the 2022 International Conference on Management of Data. pp. 1493–1503 (2022)
20. Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., et al.: Llama 2: Open foundation and fine-tuned chat models (2023)
21. Venetis, P., Halevy, A.Y., Madhavan, J., Pasca, M., Shen, W., Wu, F., et al.: Recovering semantics of tables on the web (2011)
22. Ye, X., Iyer, S., Celikyilmaz, A., Stoyanov, V., Durrett, G., Pasunuru, R.: Complementary explanations for effective in-context learning. arXiv preprint arXiv:2211.13892 (2022)
23. Zhang, T., Yue, X., Li, Y., Sun, H.: Tablellama: Towards open large generalist models for tables. arXiv preprint arXiv:2311.09206 (2023)