

Attribute Ambiguity Discovery: A Deep Learning Approach via Unsupervised Learning

(Discussion Paper)

Enzo Veltri¹, Gilbert Badaro², Mohammed Saeed² and Paolo Papotti²

¹Università degli Studi della Basilicata (UNIBAS), Potenza, Italy

²EURECOM, Biot, France

Abstract

Several applications, such as text-to-SQL and computational fact checking, exploit the relationship between relational data and natural language text. However, state of the art solutions simply fail in managing “data-ambiguity”, i.e., the case when there are multiple interpretations of the relationship between text and data. Given the ambiguity in language, text can be mapped to different subsets of data, but existing training corpora only have examples in which every sentence/question is annotated precisely w.r.t. the relation. This unrealistic assumption leaves the target applications unable to handle ambiguous cases. To tackle this problem, we present a deep learning method that identifies every pair of data ambiguous attributes and a label that describes both columns. Such metadata can then be used to generate examples with data ambiguities for any input table.

Keywords

Unsupervised Text Generation, Data to Text Generation, Data Ambiguity

1. Introduction

Ambiguity in natural language comes in many forms [1]. Factual sentences or questions can be “data-ambiguous” w.r.t the data available in a table. A simple example is the question “Is Curry the best shooter in NBA history?”. Based on the NBA official data, the answer changes depending on the interpretation of *shooting* in terms of table attributes [2]. Indeed, in the context of querying relational databases using natural language (text-to-SQL), “ambiguity of natural language queries is one of the most difficult challenges” [3]. The problem of data ambiguities in text is also relevant for many natural language processing (NLP) applications that use relational data. These span from computational fact checking, i.e., verify if a given claim holds w.r.t. a table [4, 5], to question answering in general [6, 7], and document classification [8, 9].

Consider a fact checking application that verifies a textual claim, such as “Carter LA has higher shooting than Smith SF”, against a relational D as in Table 1. Even as humans, it is hard to state if the sentence is true or false w.r.t. the data in D . The challenge is due to the two


SEBD 2023: 31st Symposium on Advanced Database Systems, July 02–05, 2023, Galzignano Terme, Padua, Italy

✉ enzo.veltri@unibas.it (E. Veltri); gilbert.badaro@eurecom.fr (G. Badaro); mohammed.saeed@eurecom.fr (M. Saeed); papotti@eurecom.fr (P. Papotti)

🆔 0000-0001-9947-8909 (E. Veltri); 0000-0002-7277-617X (G. Badaro); 0000-0001-6815-9374 (M. Saeed); 0000-0003-0651-4128 (P. Papotti)



© 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

	Player	Team	FG%	3FG%	fouls	apps
t_1	<u>Carter</u>	<u>LA</u>	<u>56</u>	<u>47</u>	4	5
t_2	<u>Smith</u>	<u>SF</u>	<u>55</u>	<u>50</u>	4	7

Table 1

A data-ambiguous example contains the sentence “Carter LA has higher shooting than Smith SF” and the evidence underlined.

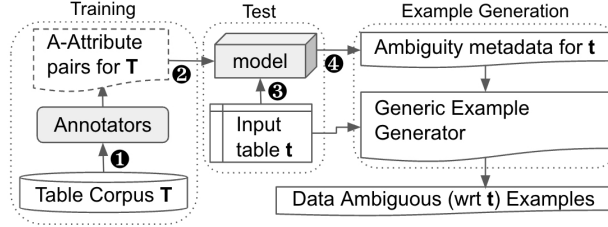


Figure 1: Overview of the solution for ambiguity metadata discovery. In the training phase, a model is fine-tuned for the prediction task. Given a table t , the model generates the metadata that is then used to generate examples.

different meanings that can be matched to *shooting*: the claim can refer to attribute *Field Goal* (FG%) or to *3-point Field Goal* (3FG%). The same challenge applies with a SQL query expressed in natural language such as “Did Carter LA has higher shooting than Smith SF?”. We refer to this issue as *data ambiguity*, i.e., the existence of more than one interpretation of a text w.r.t. the data for a human reader.

While existing corpora of examples come from extensive and expensive manual efforts, they do not contain examples with ambiguous text. Existing applications fail in these scenarios: the two examples above would lead to a single interpretation, which is incorrect in 50% of the cases.

In this work, we focus on the discovery of ambiguities over a relational schema, i.e., on *attribute ambiguities*. Ambiguities over data, and the combination of attributes and data ambiguities are explored in the full paper [10, 11].

Given a relation, the key idea is to identify the set of attributes involved in ambiguity over the data. To handle these challenges we introduce deep learning models that predict if two attributes are ambiguous and if so, it also predicts a text or “label” that makes them ambiguous. Such information can be used to generate a large corpus of ambiguous claims [10, 12].

Ambiguity Discovery is presented in Section 2 and the Experimental Evaluation in Section 3.

2. Ambiguity Discovery

From our analysis of an online fact-checking application’s log, we estimate that 40% of the user-submitted sentences present attribute ambiguity.

We describe the overview of the Attribute Discovery (Figure 1) to obtain the ambiguity metadata for a given relational table.

- 1) In the training phase, a transformer pre-trained model is fine-tuned with examples of pairs

Table 2

Examples of the aliases for two input attributes from five annotators: synonyms (Syn), related to (RelTo), derived (der), subtypes (isA), and Wikipedia results (Wiki).

Input	syn	relTo	der	isA	Wiki
silver medal	silver	runner up	medal	trophy	medal
earnings	wage	profit	earn	giving	income

of ambiguous attributes¹. As training data does not exist for this task, we use six unsupervised, noisy annotator functions. Given a pair of attributes for a table, the annotators produce a label for them if they are ambiguous. For example, given the attributes ‘length’ and ‘width’, an annotator function should produce labels such as ‘dimension’ or ‘magnitude’ that yield some ambiguity w.r.t. the input attributes. We run these noisy annotators on a large collection of tables, such as the WebTables corpus [13]. The unsupervised functions are detailed in Section 2.1.

2) The noisy output of the annotator functions is used as training examples for a fine-tuning task on an encoder-decoder language model [14]. This text-to-text architecture achieves state-of-the-art results on several NLP tasks. We introduce the novel task of predicting (1) if two attributes are ambiguous and (2) the label for such ambiguity. We design two variants that take into consideration different levels of access to the tables. One considers as input the schema only, while the second assumes that a sample of the data can be fed to the model. We discuss this component in Section 2.2.

3) Once the model has been trained, we use it at test time with any unseen input table t . We test all the possible pairs of non-key attributes in t and, if a label is predicted, we consider the input attribute pair in question as ambiguous.

4) Such ambiguous metadata can be used in a generic Text-Generator for the target application [10].

2.1. Annotator Functions

Our goal is to create training data for a model that, given two attributes alongside table schema (or table schema with a data sample), either produces a label if both attributes are ambiguous or abstains otherwise. Unfortunately, we cannot count on a large amount of manually annotated examples. To obtain such data in an unsupervised fashion, we start the process with a set of basic annotator functions that exploit external resources and heuristics to find candidates for ambiguity.

The first set of our weak supervision annotators [15] are designed in a two-step approach. First, an *alias function* automatically finds possible *aliases* for a given word. For example, a set of possible aliases for the word length are {measurement, measure, range}. Then the alias for two candidate attributes are compared and eventually selected. We design alias functions that use different external datasets. Four alias functions use the ConceptNet graph [16]. Given the input word (attribute label), we look for its relationships in the graph that are alternative

¹We focus on pair of attributes as we found it effective for example generation in our target application. The method can be extended with a post-processing step that analyzes pairs to compute larger sets, e.g., 1pt%, 2pt%, 3pt%.

representations of the input: a) *syn* synonyms; b) *relTo* related words; c) *der* words it derived from; and d) *isA* subtypes. For a second external resource, we use the Wikipedia API to search top page titles as other possible aliases with the *wiki* function. Table 2 shows examples of the generated aliases.

Once the aliases have been collected, given a pair of attributes represented by their names, we say that two attributes are ambiguous if the intersection of their aliases values is non-empty. The ambiguous labels are the intersection. If the intersection is empty, then the two pairs of attributes are not ambiguous. An annotation function makes use of the intersection for every external resource. If the attribute names are not meaningful, the annotator functions output empty results, e.g., attribute with name “A12” has empty results for all annotator functions.

As a sixth function, we find the Longest Common Substring (*LCS*) between two attribute names. Since the result may contain sequences of characters without meaning, we filter the generated word with a dictionary.

2.2. Fine-Tuning the Language Model

Pre-trained transformer based language models (LMs) such as BERT [17] or T5 [14] have shown to perform well in different NLP tasks such as question answering and text classification. Typically, these models are pre-trained on large text corpora such as articles from Wikipedia, news articles, or Common Crawl. The model is pre-trained in an unsupervised manner, for example by predicting a missing token or the next-sentence in a paragraph. Unlike conventional word embedding techniques, pre-trained transformer-based LMs learn better the semantics of words and provide different representations for the same word when utilized in different contexts. LMs can then be further *fine-tuned* for supervised, specific tasks. Tuning with task specific data is one of the advantages of pre-trained LMs, thanks to the advances in transfer learning. Fine-tuning is performed in a supervised manner by providing the LM the labeled input. Fine-tuning allows to generate tasks with new prompts while effectively exploiting transfer-learning for natural language understanding.

We define two variants of the same fine-tuning task with the objective of identifying a pair of ambiguous attributes and their common label, in case such label exists. We consider a sequence generation task where the goal is to learn a function $f : x \rightarrow y$, where x is a sequence of text containing the pair of attributes and y is a sequence of text corresponding to the label (where *none* means no ambiguity). The training data comes from the annotator functions discussed above. The difference between the two tasks lies in the prompt provided to the LM.

In the *schema-task*, function f takes a table schema T_s and a pair of its attribute with labels P_a as input, the goal is to predict a text label l in case a similarity exists between the two attributes or *None*. Hence $x = (T_s, P_a); y \in (l, None)$.

In the *data-task*, the table header and randomly selected rows are concatenated with the attributes along with the corresponding label. In this case, f' assumes as input a sample of the table data cells T_d , in addition to T_s and P_a . We denote the schema and the data together as T_{sd} . Hence, $x' = (T_{sd}, P_a); y \in (l, None)$. For this task, we discuss alternative representations of the table cells, namely a row serialization and a column serialization. Our decision of the two alternatives is based on the typical serialization in the literature to create neural representations for database tables [18]. While other contextual features are sometimes encoded, we keep our

<p>Schema-task prompt: concatenate T_s, P_a, l Player Team FG% 3FG% fouls apps, attr1: FG% attr2: 3FG%, [y: shooting]</p> <p>Data-task prompt (Row): concatenate T_{sd}, P_a, l Player Team FG% 3FG% fouls apps Carter LA 56 47 4 5 Smith SF ... 7 Carter ... , attr1: FG% attr2: 3FG%, [y: shooting]</p> <p>Data-task prompt (Column): concatenate T_{sd}, P_a, l Player Carter Smith Carter Team LA SF SF FG% 56 55 60 3FG% ... fouls 4 ... , attr1: FG% attr2: 3FG%, [y: field goal]</p>

Figure 2: Examples of input prompts for fine-tuning T5 model for the two tasks. “|” represents a cell separator token and “||” represents row separator.

model simple and generic.

Figure 2 shows a sample of the prompt for the fine tuning of the model based on the basket data in Table 1. The special tokens are used to help the model distinguish the start and the end of a cell, of a row, and of a column depending on the configuration.

3. Experiments

We now analyze in detail the solution proposed to identify attributes with ambiguities and their corresponding labels. In the following, we refer to our methods described in Section 2.2 as SCHEMA and DATA.

Datasets. For the training, we take a sample of 500k tables from the Web Tables corpus [13]. We use relational tables with a header as first row and horizontal orientation. For the test, we use a crowd-annotated corpus of 13 tables from the popular UCI Dataset. We asked the crowd to annotate ambiguities for two tasks where (i) the schema or (ii) the schema and the data were provided. We presented pair of attributes with possible ambiguous labels generated from us. We asked to mark the ambiguous pairs, moreover, for all the ambiguous pairs, we asked also to report more ambiguous words if possible.

Metrics. We do not report accuracy, defined as the fraction of correct predictions. As the number of pairs of attributes defined as ambiguous is much smaller than the number of attribute combinations, the label distribution is skewed (output is dominated by true negative). We therefore report precision (P), recall (R), and their combination in the f-measure (F1) for the prediction of the models, where a prediction for a label is a true positive if such label is in the ground truth.

Baselines. The annotator functions introduced in Section 2.1 perform very poorly when naively applied to the test dataset. We therefore introduce two baseline methods. The first, is an unsupervised labeling heuristic function, uLABEL, that uses both ConceptNet and Wikipedia to find possible common words for the attributes with intersection. If the result is still empty, then it returns the output of the LCS function. The second is a supervised labeling solution (sLABEL), that also makes use of the pre-trained LM. This fine tuning task takes a single attribute as input and produces a list of possible labels. It starts with the examples from the same annotators, i.e., synonyms of the attribute, ConceptNet labels, Wikipedia page titles, and the least common sub-string between the attribute and every other attribute in the table. The resulting list of

Table 3

Results in % for all methods on the test dataset. Quality of predicting ambiguity and a label for a given pair of attributes. Our methods reported at the bottom.

	Ambiguity			Labeling		
	P	R	F1	P	R	F1
uLABEL	89.7	10.3	18.5	84.2	6.4	6.35
sLABEL	85.6	86.1	85.9	74.1	41.5	53.3
SCHEMA	88.8	84.9	86.8	87.8	77.4	82.3
DATA	80.4	91.3	85.5	79.2	84.5	81.8

possible labels is then used as training data. For testing, each attribute is submitted to the model and the pairs of attribute with non-empty intersection of their outputs are added to the results.

For the evaluation, we start by comparing the different methods. We then show how different parameters have an impact on our solution. All results are reported for the binary task of stating if two attributes are ambiguous (**Ambiguity**) and the task of predicting the correct label (**Labeling**).

Results w.r.t. the baselines. Table 3 shows the results for the four methods on the test corpus. We observe that the unsupervised baselines obtains good precision in both tasks, but very low recall. The other methods perform all well in terms of detecting ambiguous pairs (Ambiguity), with sLABEL close to our methods in terms of f-measure thanks to very high precision and recall. However, in the task of predicting the label, both our models clearly outperform both baselines with an f-measure of 82%, while sLABEL achieves only 53%. Interestingly, both models not using data (sLABEL and SCHEMA) achieve high precision, while the model that uses schema and data (DATA) achieves much higher recall. This is because the comparison of data values may lead to ambiguities that are not captured by looking at attribute label only. However, this may be misleading in some cases, such as those with numerical values with similar domains but different value distributions. For example, for attributes FG_PCT and FG3M, the human annotators agree on ‘FG’ as ambiguous label, but DATA returns *none* as they have different value distributions. For DATA, we found experimentally that the best quality results are achieved with the maximum number of rows to consider in the T_{sd} equals to five and that the row representation outperforms the column representation.

Results w.r.t. the number of training steps. Figure 3 (on the left) shows that both methods improve the quality of the results with an increasing number of training steps. Results nearly converge after 3k steps. Training SCHEMA and DATA models requires 1.5 hours.

Results w.r.t. the number of T5 parameters (model size). Figure 3 (on the right) reports the impact of the number of training parameter on the quality of the results. Increasing the size of the model parameters (sizes: Small < Base < Large < 3B) increases the quality of the predictions in the final model. LMs with a small number of parameters after 2000 training steps start to converge to low-quality results. The number of parameters also influences the inference time. The biggest model (3B) takes on average two seconds per prediction on our machine.

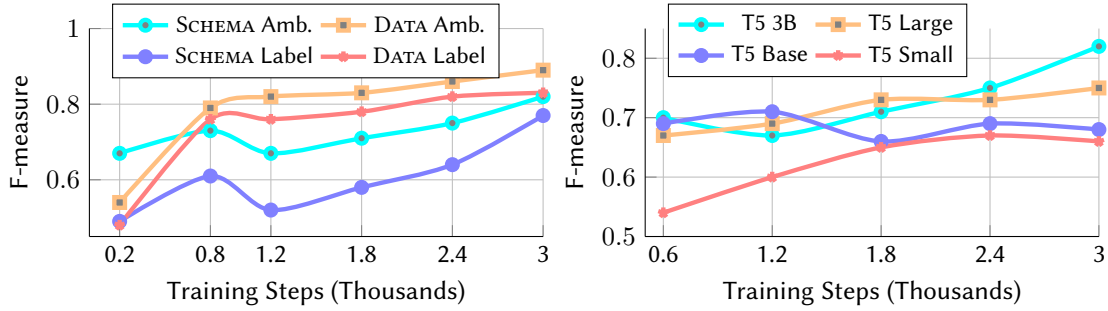


Figure 3: On the left the impact of the number of training steps and on the right the impact of the number of T5 parameters on prediction quality.

3.1. End-To-End User Evaluation

We conduct a second user study to evaluate the text generated by an end-to-end system [10] over 11 datasets. The system is configured to use our ambiguity discovery module. We generate at least four ambiguous sentences and two not ambiguous ones for each table. We ask eleven participants to manually annotate the generated text in two ways. First, *ambiguity detection*: state if the text is ambiguous w.r.t. the associated schema and a sample of data. Second, *attribute ambiguity detection*: if the text is ambiguous, mark the ambiguous attributes. We instruct the participants with data ambiguity definitions and examples. We give each participant three datasets to annotate, so that every dataset has three annotations. Then, for every annotator and dataset, we measure P, R and F1 for both ambiguity and attribute ambiguity detection. For ambiguity, we count a match if the annotation agrees with the binary label for the text in the ground truth. For the attribute ambiguity, we count a match if at least one of the annotated attributes is in the ground truth of the text. Results per dataset (averaged over three participants)

Table 4
Quality Results of the End-To-End experiment.

Dataset	Ambiguity			Attr. Ambiguity		
	P	R	F1	P	R	F1
Abalone	1.000	0.807	0.893	1.000	0.807	0.893
Adults	0.923	0.889	0.906	0.885	0.885	0.885
Basket Acronyms	0.750	0.813	0.780	0.731	0.809	0.768
Basket	1.000	0.619	0.765	1.000	0.619	0.765
Heart Diseases	0.875	0.656	0.750	0.833	0.645	0.727
Iris	1.000	0.963	0.981	0.981	0.962	0.971
Superstore	0.950	0.679	0.792	0.900	0.667	0.766
Wine Quality	0.950	0.792	0.864	0.950	0.792	0.864
Laptop	0.923	0.667	0.774	0.846	0.647	0.733
Mushroom	1.000	0.905	0.950	1.000	0.905	0.950
Soccer	0.762	0.800	0.780	0.714	0.789	0.750
AVG	0.921	0.781	0.840	0.895	0.775	0.825

are shown in Table 4. We observe an average F1 (over all datasets) of 84% and 82.5% for ambiguity and attribute ambiguity detection, respectively. Such results show that (1) humans recognize text with and without data ambiguity, (2) they also recognize the right attributes when there is ambiguity. As expected, the second action has lower F1 as it is an harder task.

References

- [1] [n.d.], Notes on ambiguity, <https://cs.nyu.edu/~davise/ai/ambiguity.html>, 2021.
- [2] BleacherReport, Stephen Curry is ‘the best shooter who ever lived’ per Warriors HC Steve Kerr, <https://bleacherreport.com/articles/10009994>, 2021.
- [3] G. Katsogiannis-Meimarakis, G. Koutrika, A deep dive into deep learning approaches for text-to-sql systems, in: SIGMOD, ACM, 2021, pp. 2846–2851.
- [4] R. Aly, Z. Guo, M. S. Schlichtkrull, J. Thorne, A. Vlachos, C. Christodoulopoulos, O. Co-carascu, A. Mittal, FEVEROUS: Fact extraction and VERification over unstructured and structured information, in: NIPS, 2021.
- [5] G. Karagiannis, M. Saeed, P. Papotti, I. Trummer, Scrutinizer: A mixed-initiative approach to large-scale, data-driven claim verification, *Proc. VLDB Endow.* 13 (2020) 2508–2521.
- [6] W. Chen, M. Chang, E. Schlinger, W. Y. Wang, W. W. Cohen, Open question answering over tables and text, in: ICLR, OpenReview.net, 2021.
- [7] J. Thorne, M. Yazdani, M. Saeidi, F. Silvestri, S. Riedel, A. Y. Levy, From natural language processing to neural databases, *Proc. VLDB Endow.* 14 (2021) 1033–1039.
- [8] P. Lapadula, G. Mecca, D. Santoro, L. Solimando, E. Veltri, Humanity is overrated. or not. automatic diagnostic suggestions by greg, ml (extended abstract), *Communications in Computer and Information Science* 909 (2018).
- [9] P. Lapadula, G. Mecca, D. Santoro, L. Solimando, E. Veltri, Greg, ml – machine learning for healthcare at a scale, *Health and Technology* 10 (2020).
- [10] E. Veltri, G. Badaro, M. Saeed, P. Papotti, Data ambiguity profiling for the generation of training examples, in: ICDE, IEEE, 2023.
- [11] E. Veltri, D. Santoro, G. Badaro, M. Saeed, P. Papotti, Pythia: Unsupervised generation of ambiguous textual claims from relational data, in: SIGMOD 2022, ACM, 2022, p. 2409–2412. doi:10.1145/3514221.3520164.
- [12] E. Veltri, D. Santoro, G. Badaro, M. Saeed, P. Papotti, Ambiguity detection and textual claims generation from relational data, in: SEBD, volume 3194, CEUR-WS.org, 2022, pp. 333–340.
- [13] O. Lehmborg, D. Ritze, R. Meusel, C. Bizer, A large public corpus of web tables containing time and context metadata, in: WWW, ACM, 2016, pp. 75–76.
- [14] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P. J. Liu, Exploring the limits of transfer learning with a unified text-to-text transformer, *J. Mach. Learn. Res.* 21 (2020) 140:1–140:67.
- [15] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, C. Ré, Snorkel: Rapid training data creation with weak supervision, in: VLDB, volume 11, 2017, p. 269.
- [16] R. Speer, J. Chin, C. Havasi, Conceptnet 5.5: An open multilingual graph of general knowledge, in: Thirty-first AAAI conference on artificial intelligence, 2017.

- [17] J. Devlin, M. Chang, K. Lee, K. Toutanova, BERT: pre-training of deep bidirectional transformers for language understanding, in: NAACL-HLT, ACL, 2019, pp. 4171–4186. URL: <https://doi.org/10.18653/v1/n19-1423>. doi:10.18653/v1/n19-1423.
- [18] G. Badaro, M. Saeed, P. Papotti, Transformers for Tabular Data Representation: A Survey of Models and Applications, TACL (2023). URL: <https://hal.science/hal-03877085>.