Technical Perspective: Unicorn: A Unified Multi-Tasking Matching Model

AnHai Doan
Department of Computer Science
University of Wisconsin-Madison, USA
anhai@cs.wisc.edu

Data integration has been a long-standing challenge for data management. It has recently received significant attention due to at least three main reasons. First, many data science projects require integrating data from disparate sources before analysis can be carried out to extract insights. Second, many organizations want to build knowledge graphs, such as Customer 360s, Product 360s, and Supplier 360s, which capture all available information about the customers, products, and suppliers of an organization. Building such knowledge graphs often requires integrating data from multiple sources. Finally, there is also an increasing need to integrate a massive amount of data to create training data for AI models, such as large language models.

Integrating data from different sources often requires matching: deciding whether two data elements are "semantically the same", where "data elements" can be strings, tuples, columns, etc. Numerous matching tasks have been studied. Well-known examples include string matching, entity matching (which matches tuples), schema matching (which matches columns), ontology matching, entity linking, entity alignment, and column type annotation.

Over the past 40 years, these matching tasks have been intensively studied in many research communities (including databases, AI, Web, Semantic Web, and KDD). Significant progress has been made, and today there is a general consensus on the overall architecture to solve these tasks.

Consider the problem of matching two sets A and B, which contain data elements of the same type (such as strings, tuples, or columns). State-of-the-art solutions typically solve this problem in two steps: *blocking* and *matching*. Considering all pairs in the Cartesian product of A and B is often impractical, so the blocking step uses heuristics to quickly discard pairs $(a \in A, b \in B)$ that are unlikely to match (e.g., people living in different cities). The matching step focuses on the remaining pairs, and applies a rule- or learning-based matcher to each pair to predict match/no-match.

The highlighted Unicorn paper focuses on the matching step. It starts with the observation that, despite decades of work, today within an organization people are still likely to solve the matching tasks *separately*. For example, they may use three separate labeled data sets to train three matchers to match strings, tuples, and columns, respectively.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Copyright 2024 ACM 0001-0782/24/0X00 ...$5.00.

Inspired by recent progress in deep learning, the paper asks: can we *unify* these tasks, i.e., train a single matcher to match all kinds of data elements, such as strings, tuples, and columns, as well as new unseen types of data elements?

The paper shows that this is indeed possible, and describes an elegant solution architecture based on recent progress in deep learning. To match a pair $(a \in A, b \in B)$, the solution first serializes the pair into a textual sequence, then uses a pre-trained language model to map the sequence into an embedding vector. Next, the solution transforms the vector using a neural-network mixture-of-expert model, then applies a binary classifier to output a probability that the input pair is a match. The parameters of the neural-network based architecture are trained using the labeled data sets of "all" available matching tasks. Extensive experiments with 20 data sets over seven matching tasks show the promise of the proposed solution.

Thus, the Unicorn paper introduces the first solution that can solve the matching step of a wide variety of matching tasks. The paper correctly observes that this can enable learning across the matching tasks (because the solution is trained on the union of the labeled data sets of all tasks), and that the solution can be used to solve new matching tasks where no labeled data is yet available (a.k.a., zero-shot matching).

In addition, the paper raises a number of interesting questions. First, does this mean we now just need a single matching solution per organization? Probably not, because organizations often have use cases that require the data to be labeled differently. For example, two products may be declared a match in the context of customer service, yet not a match in the context of marketing. Yet, the possibility of being able to unify most matching solutions is intriguing and worth exploring further.

Second, even if we assume that an organization still needs multiple matching solutions, there still seems to be significant value in developing a single solution that can match a wide variety of tasks and can perform zero-shot matching. Perhaps this solution can be quickly customized (e.g., fine tuned) to help match a particular use case. How to do this is a topic of potentially great interest to both academia and industry.

Finally, while the paper has demonstrated the promise of a single unifying solution, how to make such a solution scalable and accurate for a very large amount of data – a scenario that is very common in practice – is still unclear, and is likely to motivate a lot of follow-up work.