ActiveDeeper: A Model-based Active Data Enrichment System

Liang Zhao[†] Qingcan Li[◇] Pei Wang[◇] Jiannan Wang[◇] Eugene Wu[‡]
Xi'an Jiaotong University[†] Simon Fraser University[◇] Columbia University[‡]
zhaoliang0415@xjtu.edu.cn {qingcanl, peiw, jnwang}@sfu.ca ew2493@columbia.edu

ABSTRACT

Deep Web (e.g., Yelp, IMDb) is an invaluable external data source for enriching a local database with new attributes. In this paper, we present ActiveDeeper, a novel model-driven data enrichment system powered by deep web. ActiveDeeper treats deep web as "a labeler" and uses it to train a data enrichment model. We show that this model-based approach significantly outperforms the state-of-the-art system in real-world scenarios. We implemented ActiveDeeper as a Google Sheets add-on and made a demo video at http://tiny.cc/activedeeper.

PVLDB Reference Format:
Liang Zhao, Qingcan Li, Pei Wang, Jiannan Wang, and Eugene Wu. ActiveDeeper: A Model-based Active Data Enrichment System. *PVLDB*, 13(12): 2885-2888, 2020.
DOI: https://doi.org/10.14778/3415478.3415500

1. INTRODUCTION

Data enrichment extends local datasets with new attributes from external data sources [3, 2, 4, 5]. It is a crucial part of data science pipelines because it helps collect and integrate extra information to better identify new insights from local data. As a real-world scenario, banks maintain the profile data of their business members (e.g., business name, address, phone). Enriching the business category attribute for each member will help the bank understand the distribution of industries of its members and better target their financial services.

The *deep web*, as a popular web-based data source, provides great opportunities for data enrichment. Deep web (or *Hidden DB*) refers to the database (DB) that can only be accessed through a query interface (e.g., keyword search interface). Websites like Yelp[1] and YellowPages[2] maintain a large and rich entity collections that can be accessed through their web APIs. The state-of-the-art data enrichment system, Deeper [2, 3], uses these APIs to enrich the local dataset while minimizing the number of API calls.

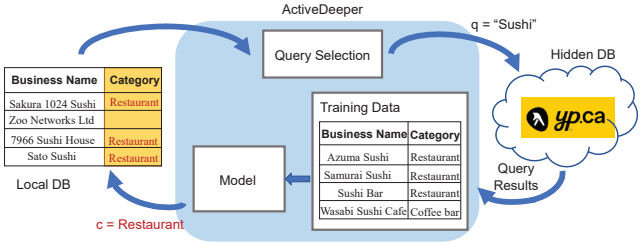

Figure 1: An Illustration of ActiveDeeper.

However, the hidden DB may not have all the data in the user's local data (the *local DB*). In our real experience, since many small businesses do not keep registration information online, only a very small number of business members (around 10%) can be found at deep websites like Yellow Pages and Google Maps. The state-of-the-art system, Deeper [2, 3], would fail under this low-overlap situation, as it relies on entity matches to enrich the local DB.

Is it possible to enrich a record when it does not have a deep web match? Figure 1 illustrates our idea. Consider "Sakura 1024 Sushi" in the local DB. If it does not exist in the YellowPages (`yp.ca`), Deeper [2] will fail to enrich it. However, YellowPages may find four results—three Restaurants and one Coffee Bar—for the keyword "Sushi". Although none of them exactly match "Sakura 1024 Sushi", we can infer that business names containing "Sushi" are more likely to be a Restaurant, and that "Sakura 1024 Sushi" is likely also categorized as a Restaurant.

Based on this idea, we propose ActiveDeeper, a novel model-based active data enrichment system powered by deep web. Figure 2 depicts the system architecture. The key innovation is to treat deep web as "a labeler" and use it to construct a training dataset to train a multi-class classification model. The model is then applied to enrich the local data with a categorical attribute. Since the data stored in deep web can only be accessed through a restrictive keyword-search interface, ActiveDeeper faces two technical challenges: i) which queries should be selected and issued to the deep web, and ii) how to utilize the returned query results to train a model to enrich the local data.

Challenge I. Query Selection. The query selection process can be thought of as a type of active learning. The goal is to interactively query a labeler to construct a training dataset. In traditional active learning [1], the labeler

[1]https://www.yelp.com
[2]https://www.yellowpages.ca

Proceedings of the VLDB Endowment, Vol. 13, No. 12
ISSN 2150-8097.
DOI: https://doi.org/10.14778/3415478.3415500

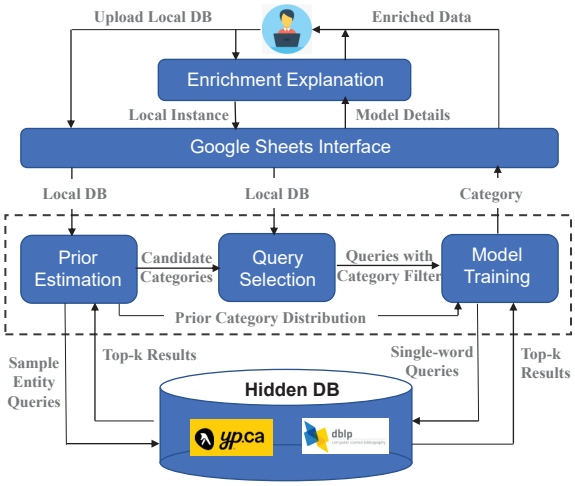

Figure 2: ActiveDeeper System Architecture.

is given an unlabeled instance (e.g, "Sakura 1024 Sushi") and returns its label (e.g., Restaurant). In ActiveDeeper, the labeler is given a keyword search query (e.g., "Sushi") and returns a set of labeled instances that match the query (e.g., see the four records in the training data in Figure 1). Due to the difference, it calls for the development of a new query selection strategy.

Challenge II. Model Training. The model training component aims to learn a *Naive Bayes* model from training data and then uses the model to enrich the data. *Naive Bayes* is a probabilistic classifier, which is composed of the prior probability $P(c)$ and a collection of conditional probabilities $P(w|c)$. One naive approach is to directly estimate these probabilities from the training data. For example, if 40% of the categories in the training data are Restaurant, then this approach will set $P(c = \text{Restaurant}) = 0.4$; if 70% of the Restaurant records in the training data contain "Sushi", then this approach will set $P(w = \text{Sushi} \mid c = \text{Restaurant}) = 0.7$. However, this approach ignores the fact that the training data (the union of the query results) and the test data (the local DB) may have different distributions. If Restaurant only appears in 10% records of the local DB, then $P(c = \text{Restaurant})$ should be set to 0.1 rather than 0.4. Thus, we need to train our model in a smarter way.

ActiveDeeper uses a frequency-based query selection approach for Challenge I and an effective model training approach for Challenge II. Both approaches are simple yet powerful. ActiveDeeper has been deployed in a leading financial cooperative (anonymized) in Vancouver and helped their business analysts enrich business categories.

In our demonstration, users will be able to use ActiveDeeper to enrich their own spreadsheet data, and compare its efficacy with the prior state of the art Deeper system. Users can upload a dataset to enrich. After selecting a web service to act as a hidden DB (we support YellowPages and DBLP for the demo), the system will populate a new column with enriched category information. In addition to enriching the dataset, users can double check the results by examining the model confidences for each enriched value. To make the system realistic, the demo wil be integrated as an add-on to Google spreadsheets, so users can benefit from formulas, spreadsheet manipulations, and visualizations before and after enrichment.

2. ACTIVEDEEPER DETAILS

To gain a deep understanding of ActiveDeeper internals, we first define the novel model-based active data enrichment problem and then present how ActiveDeeper solves the problem.

2.1 Problem Description

Let D_L denote a local DB and H denote a hidden DB. We model each local record in D_L as a set of words that describe a real-world entity. We consider that a hidden DB provides a keyword-search interface. It takes a keyword query as input and returns a set of entities that match the query. If the number of matched entities is very large, it will rank them based on an unknown ranking function and only return the top-k entities. Let q be a keyword-search query, $H(q)$ be the top-k records returned by q. We assume that the category column is missing in the local DB and the query results from the hidden DB contain the category column. The goal is to train a multi-class classification model to enrich the local DB with the category column.

PROBLEM 1 (Model-based Active Data Enrichment). *Given a local database D_L, a hidden database H, a query budget $N(\leq |D_L|)$ as the maximum number of queries issued to H, the model-based active data enrichment problem is to construct a model mapping between a local record d and its enriched data value v within N.*

2.2 ActiveDeeper Internals

Main Idea. Given a list of business names, human could tell the category of business names solely based on some keywords like "sushi", "clinic", and "supermarket". The intuition behind is that human have the knowledge that certain words are strongly correlated with some categories. For a business name "sakura 1024 sushi", consider it is generated by randomly selecting words from a large vocabulary. Assuming the words are independent, and we have the probability mapping from words "Sakura", "1024", "sushi" to restaurant category, we can infer the probability that the business is a restaurant. If "sushi" appears in a business title, the probability that the business is a restaurant is 0.75, which is very high, then a given business containing "Sushi" is very likely to be a restaurant. Inspired by this, we propose a Naive Bayes model to learn the mapping between the entity names and the data value to be enriched.

Formally, suppose $d \in D_L$ is a local entity name, $W(d) = \{w_1, w_2, \ldots, w_m\}$ are the m independent words of d. The probability of d belonging to category c_i is calculated as

$$P(c_i|d) = P(c_i|\{w_1, w_2, \ldots, w_m\}) = \frac{\Pi_{j=1}^{m} P(w_j|c_i) \cdot P(c_i)}{P(d)} \quad (1)$$

$P(w_j|c_i)$ represents the category-conditional probability where word w_j appears under c_i. $P(c_i)$ is the prior probability of c_i in local DB. $P(d)$ is the probability of d which could be seen as a constant, so that $P(c_i|d) \propto \Pi_{j=1}^{m} P(w_j|c_i) \cdot P(c_i)$. Then the enriched category of d, denoted as c_d^* is decided by

$$c_d^* = \underset{c_i \in C}{\text{argmax}} \, \Pi_{j=1}^{m} P(w_j|c_i) \cdot P(c_i) \quad (2)$$

Now the key becomes learning such a mapping from words to category. Specifically, we need to compute $P(w_j|c_i)$ and $P(c_i)$ for each word w_j and each category c_i. There are two challenges we must address. One is to decide which queries

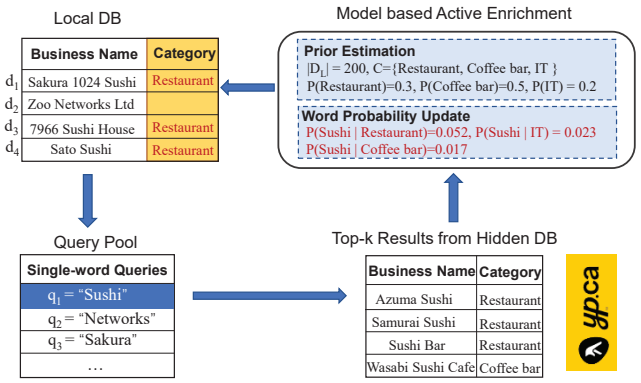

Figure 3: A Running Example of ActiveDeeper.

to issue to the keyword interface to obtain the labeled data. The other is after we have the query words and their corresponding results, how to construct the model.

Dive Into Query Selection. For the first challenge, there are so many distinct words in the local table. We generate a query pool from D_L and select the most informative queries issuing to H. As the model is based on word probability, each single word in D_L is considered as a candidate query. We propose a frequency-based query selection strategy, i.e., every time the most frequent word in local DB is issued as the query. The intuition is that we want to issue a query whose query results have an impact on as many local records as possible. See the running example in Figure 3. "Sushi" is the most frequent word in local DB. By issuing "Sushi" to hidden DB, once its probabilities under each category from the returned results are acquired (see below for how these probabilities are computed), 75% examples in local DB get enriched with only one query.

Dive Into Model Training. For the second challenge, as the hidden DB and the local DB may come from different distributions, we manipulate the probabilities from the training data in hidden DB to local DB to enable the Naive Bayes model. We need a range of categories C as supervised labels and $P(c_i)$ $(c_i \in C)$ in the local DB, which are the priors. A small sample of D_L, denoted as D_{sample} $(|D_{sample}| \ll N)$ is randomly selected to approximate the priors. Each record $r \in D_{sample}$ is issued as a keyword-search query to H and get the most frequent category among the query results $H(r)$ as its category. The set of the categories obtained in this step is considered as an estimation of C. For $c_i \in C$, $P(c_i) = |S(c_i, D_{sample})|/|D_{sample}|$ is consequently determined, where $S(c_i, D_{sample})$ depicts the set of records in D_{sample} with category c_i. In the example (Figure 3), suppose the whole local DB contains 200 records (we only show an example containing 4 records), by sampling 10 with the above processing we get C={Restaurant, Coffee bar, IT}, where 3 sampled records are Restaurant, 5 and 2 belong to the other two categories, respectively, then $P(c_i)$ $(c_i \in C)$ is calculated as shown in Figure 3.

Next we discuss how to compute local $P(w|c)$ from the training data. Initially, word probability to any $c_i \in C$ is set equally by $1/|C|$, such that a word equally belongs to any category at the beginning and the predicted category depends on the $P(c_i)$. Every time the most frequent local word w_j is issued as the query. Based on the top-k returned hidden records $H(w_j)$, the smoothing word probability under c_i in local DB is updated from the hidden DB as

$$P(w_j|c_i) = \frac{\frac{|S(c_i,H(w_j))|}{k} \cdot Freq_{w_j} + 1}{Num_{c_i} + |C|} \quad (3)$$

where $S(c_i, H(w_j))$ denotes the set of hidden records under c_i in the top-k returned query results, $Freq_{w_j}$ is the frequency of w_j in local DB, and $Num_{c_i} = P(c_i) * |D_L|$ is the estimated number of records with c_i in local DB. The Bayes model is thus updated, with the probabilities of the training data from hidden DB transferred to those of the local DB. The iterative process repeats until the budget runs out. Notice that only the probability of the query word is updated at each iteration, since the returned results are all highly related with the query word, which is more statistically significant than the other randomly appearing words.

Continue the running example shown in Figure 3. At the beginning, for any word w_j in local DB, $P(w_j|c_i) = 1/3$ $(c_i \in C)$. Suppose the currently most frequent word q_1="Sushi" $(Freq_{q_1} = 3)$ is selected as the query and the hidden DB returns 4 Sushi-related business entities. Then $P(Sushi|c_i)$ is updated with Equation 3. Recomputing the title probability by Equation 2, we have $c^*_{d_1,d_3,d_4}$ = Restaurant, and three in four example local records get correctly enriched.

3. THE ACTIVEDEEPER SYSTEM

We now describe the ActiveDeeper system architecture in detail (see Figure 2). The system consists of four main components. It can not only help users quickly enrich data with categories but also provide insights and explanations about enrichment results.

Prior Estimation. ActiveDeeper adopts the method described in Section 2.2 to compute the set of candidate categories C and the prior category distribution $P(c_i)$ $(c_i \in C)$. Furthermore, the system allows the user to fine-tune the candidate categories. As shown in Figure 4, the categories automatically obtained by our system are illustrated in an pop-up dialog, then the user can filter out those categories that she does not want to include in the enriched column.

Query Selection. Filtering out the stop words, we extract the remaining single words from the local DB, apply lemmatization (e.g., lemmatize "networks", "networking" into "network"), and rank them as candidate queries by frequency in descending order. We iteratively select the most frequent word and issue it to the hidden DB. One implementation detail is that we add the categories as conditional filters when issuing queries so that the top-k results returned are within the category set C and the word probability calculation will thus be more accurate in model training.

Model Training. Following Section 2.2, the category-conditional word probability (i.e., $P(w_j|c_i)$) is uniformly initialized. Once a word q is issued as the query, $P(q|c_i)$ $(c_i \in C)$ will be updated upon the top-k returned results by Equation 3, then the Naive Bayes model will get updated as Equation 2.

Enrichment Explanation. As a model-based enrichment tool, ActiveDeeper opens the black box and visualizes the model details as enrichment explanation to help the user double check the results. Displayed in a side bar (Figure 4 right side), once data is enriched, the category distribution will be presented by a pie chart. And for any row specified by the user, we show in an interactive bar chart the most likely

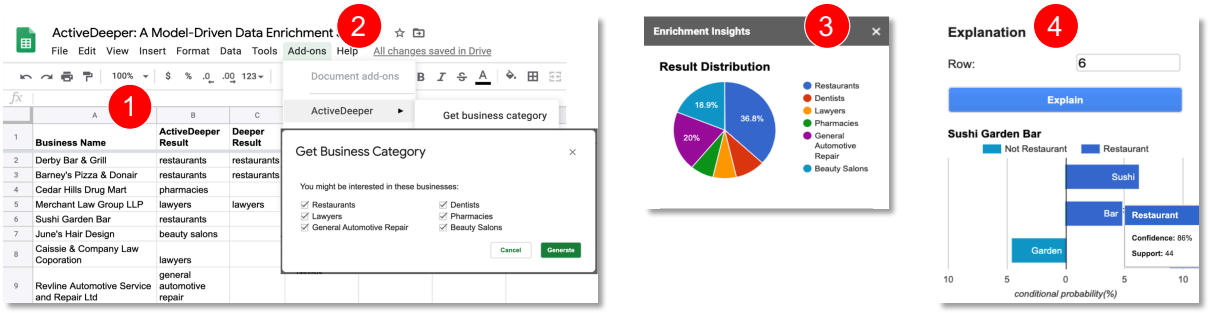

Figure 4: The User Interface of ActiveDeeper (see the demo video at http://tiny.cc/activedeeper).

category of each word (i.e., for word q, its most likely category $c = \text{argmax}\, P(q|c_i \in C)$) together with the confidence and support for reasonability validation when hovering over the word. Derived from the top-k deep web query results, the confidence represents the proportion of businesses containing q labeled as c among all the businesses with q, and the support depicts how many businesses containing q with category c appear within top-k results.

4. DEMONSTRATION

Users will use ActiveDeeper to enrich a business listing, investigate how ActiveDeeper makes labelling decisions, and compare its efficacy with the prior Deeper system.

Scenario: Business Category Enrichment. We are by a financial institution that we work with. The company provides services to businesses and maintains a database of business members. They wish to enrich each business with its category in order to provide personalized service per business category. In their case, around 60% business names are not found in YellowPages. The workflow is shown in Figure 4.

(1) The Dataset: The audience member will load a business listing representative of the above company's data into Google Sheets. **Business Name** column shows a sample of the names. The other two columns will show enrichment data using ActiveDeeper and Deeper, respectively. Note that the Deeper column contains empty values when the business name cannot be found in YellowPages. In fact, the majority of the values are empty.

(2) Using ActiveDeeper: To enrich the listings, the user selects "Add-ons" → "ActiveDeeper" → "Get business category". She will specify the input data range (all rows) and the output data column (the ActiveDeeper Result column). Our system then suggests a set of candidate categories and she can select the ones of interest. Clicking "Generate" will run ActiveDeeper and populate the output column. Similarly, the user can choose to use Deeper instead to populate the Deeper Result column.

(3) Enrichment Statistics: We have integrated an Insights sidebar into Google Sheets. This will show statistics about the distribution of categories found (or predicted) by ActiveDeeper. Note that the user can also use Google Sheet's integrated functionality to create custom visualizations or formulas based on the enrichment data.

(4) Enrichment Explanation: Finally, the user can select any row in the sheet—the figure shows row 6: "Sushi Garden Bar". If the category was predicted, she can click the Explain button to examine why the row was categorized as Restaurant. We show a visualization where each bar represents the conditional probability of a word given its most likely category (the category with the biggest conditional probability, referring to Section 3). She can also hover over each bar to examine the Support and Confidence of the per-word prediction. For example, When issuing "Bar", 51 businesses containing "Bar" are returned from the YellowPages, and 44 out of them are labeled with Restaurant, such that we got its confidence 86% and support 44, which is quite reliable of its Restaurant prediction.

Superiority of ActiveDeeper Over Deeper. As a comparison, a user can use our demo to enrich data using the state-of-the-art system, Deeper [3, 2]. The result is shown in the "Deeper Result" column (See Figure 4). Since Deeper can only enrich business names which can be found in the YellowPages, there are many missing categories. Quantitative experimental comparison was also conducted. Varying the query budget from 100 to 800, we compared the enrichment accuracy between ActiveDeeper and Deeper on a local DB containing 1599 real business names with six business categories. Setting top-100 results returned per query, ActiveDeeper four times beats Deeper given each query budget, and could accurately enrich 83% business names with a quite small budget (200 queries).

5. ACKNOWLEDGMENTS

This work was supported in part by Mitacs through an Accelerate Grant, NSERC through a discovery grant and a CRD grant as well as China Postdoctoral Science Foundation 2018T111030 & 2017M62313, NSF 1527765 & 1564049 1845638, Google LCC and Amazon.com, Inc.. All opinions and findings in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

6. REFERENCES

[1] B. Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.

[2] P. Wang, Y. He, R. Shea, J. Wang, and E. Wu. Deeper: A data enrichment system powered by deep web. In *SIGMOD*, pages 1801–1804, 2018.

[3] P. Wang, R. Shea, J. Wang, and E. Wu. Progressive deep web crawling through keyword queries for data enrichment. In *SIGMOD*, pages 229–246, 2019.

[4] M. Yakout, K. Ganjam, K. Chakrabarti, and S. Chaudhuri. Infogather: entity augmentation and attribute discovery by holistic matching with web tables. In *SIGMOD*, pages 97–108, 2012.

[5] M. Zhang and K. Chakrabarti. Infogather+: semantic matching and annotation of numeric and time-varying attributes in web tables. In *SIGMOD*, pages 145–156, 2013.