On Short Textual Value Column Representation Using Symbol Level Language Models

Ron Begleiter
Independent Researcher
ron@4t2.pw

Nathan Roll
Stanford University
nroll@stanford.edu

Abstract

String-type database columns containing short textual values are crucial for storing and managing a wide range of information in various applications. For example, they store categories, labels, enumerations, code, and abbreviations.

Here, we discuss a string column representation using symbol level language models that grasps the symbol level "distribution" of the column textual values. These language models are known for their good prediction quality, memory-footprint and runtime efficiency, while being theoretically justified. We focus on a column matching application, and provide empirical indication for their usefulness.

1 Introduction

String-type columns are essential for storing and managing textual data, providing a foundation for various applications and analyses. They offer flexibility, support for unstructured data, and enable efficient search, retrieval, and processing of human-readable information. In many cases, string type columns store short texts such as categories, labels, status information, enumerations, code and abbreviations. Modeling such columns is important for performing table representation learning downstream tasks such as detecting table natural keys [Bornemann et al., 2020], inferring pathless joins [Gong et al., 2023], near-duplicates detection [Pattara et al., 2023], data integration [Cappuzzo et al., 2020], grounding values [Deng et al., 2021], and data imputation [Deng et al., 2020].

Representation learning of short text columns can be challenging due to several factors. Short texts often suffer from data sparsity, ambiguity, noise, and contextual dependence. Their limited length makes it difficult to extract meaningful representations, while their inherent ambiguity, idiosyncrasies, and potential for errors can hinder the learning process. Moreover, the scale of real-world data sources necessitates highly resource-efficient models to ensure practical implementation.

Here, we describe a symbol level language model representation of short textual columns. The discussed language modeling belongs to a lossless compression information theory framework termed Variable order Markov Models, see [Begleiter et al., 2004]. These models are known to have competitive accuracy in practice, require limited memory overhead and has "fast" runtime footprints. For example such an algorithm is ranked 10th on the latest *Large Text Compression Benchmark*, see, algorithm "PPM" in Mahoney [2024]. [1] We cover that algorithm in Section 2.1.

In this work we consider a column matching downstream application. Database column matching is the process of identifying and aligning columns from different datasets that represent the same underlying entity or concept. It is a crucial task in data integration, data warehousing, and data analysis. Yet, it is challenging due to data quality issues such as noise, missing values, and inconsistencies, computational complexity due to the magnitude of the data, and semantic similarity that goes beyond similar column names or data types. In Section 3 we outline a utilization of symbol level language

[1] Referring to the benchmark version from June 4, 2024.

Table Representation Learning Workshop at NeurIPS 2024

models for performing column matching. We conclude with a few numerical examples in Section 4 and concluding remarks in Sections 5.

2 Symbol Language Models

In this work we focus on a family of symbol level language models, SLMs, that emerged from lossless compression models [Begleiter et al., 2004]. These language models operate on sequences, $s_1^n = s_1 s_2 \dots s_n$ over a finite alphabet of symbols $\Sigma = \{\sigma_1, \sigma_2, \dots, \sigma_k\}$. The models hypothesis space includes all up to D order Markov models. Given a training sequence $x_1^m \in \Sigma^m$ these language models learn a conditional distribution $\hat{P}(\sigma|s_1^d)$ over symbols $\sigma \in \Sigma$ condition on all $d \leq D$ sized sequences $s_1^d \in \Sigma^{\leq D}$. The quality measure of SLMs inference is the expected log-loss over some fixed length sequences $s_1^n \in \Sigma^n$, $\mathbb{E}\left[-\log(P(\sigma|s_1^n))\right]$. This turns to the empirical average log-loss $\frac{1}{n}\sum_1^n -\log(\hat{P}(\sigma_i|s_1^{i-1})$ over some test sequence s_1^n.

Note 1 (Relation to auto-regressive LLMs) *Observe that the symbol level language models description matches auto-regressive large language model (LLMs) description. The differences are subtle, as LLMs alphabet cover sub-sequences (or "words") and SLMs symbols, and the hypothesis space of LLMs is much more expressive than the limited $\leq D$ Markov models.*

Note 2 (Log-loss and Perplexity) *Log-loss and perplexity are closely related metrics used in language modeling. Both metrics measure the model's ability to predict the correct outcome. Perplexity, the exponentiated average log-loss, is often used in language modeling tasks where the goal is to predict the next word in a sequence. Log-loss is more general and can be used for classification or regression tasks.*

2.1 Prediction by Partial Match (PPM)

Prediction by Partial Match, PPM, is a statistical character-level language modeling technique that aims to utilize the longest context matched to predict the next symbol, but drops to lower context sizes for symbol counts of zero [Bell et al., 1990]. There are a few PPM variants upon the definitions of how to estimate the next symbol probability and penalizing zero count contexts. PPM starts computing the next symbol probability, $\hat{P}(\sigma|s_{n-D+1}^n)$, using the maximal context size D symbol counts whenever $s_{n-D+1}^n\sigma$ appears in the training set. Otherwise, σ does not appear in the context of s_{n-D+1}^n and the next symbol probability is defined recursively as a penalized next symbol probability of the shorter context s_{n-D+2}^n, as follows $\hat{P}_D(\texttt{penalty}|s_{n-D+1}^n) \cdot \hat{P}_D(\sigma|s_{n-D+2}^n)$.[2]

3 Column Matching

By carefully comparing database columns, organizations can gain valuable insights into their data and make informed decisions about how to manage and use it effectively. Here are some specific applications of column matching:

Data Optimization: Identifying and removing duplicates improves data quality and efficiency, reducing storage, errors, and query time.

Data Understanding: Comparing columns can help organizations gain a deeper understanding of the relationships between different data elements, identify potential data anomalies, and uncover hidden patterns. This can lead to valuable insights and informed decision-making.

Data Integration: When integrating data from multiple sources, column matching is essential to ensure that data is aligned correctly and that there are no conflicts. This helps maintain data integrity and consistency across different systems.

Data Governance: Understanding the similarities and differences between columns can help organizations establish effective data governance policies and standards. This can improve data quality, security, and compliance.

[2]For the empty context, the probability of the next symbol can be assumed to be independent and identically distributed (i.i.d.).

By leveraging column matching techniques, organizations can normalized and fix their data, drive data-driven decision-making, and improve overall business outcomes.

3.1 Column Matching via Distribution Similarity

SLMs capture the "type" of column values by counting the "frequency" of each unique symbol that appears in it. This is known as the Method of Types (see, e.g., Chapter 11.1 in Cover and Thomas [2006]). In turn, we can quantify *relatedness* between columns via the distance between their corresponding (empirical) distributions. In other words, given two columns, C_1, C_2, and their models, $\hat{P}_1, \hat{P}_2$, quantify the columns matching via $\texttt{distance}(\hat{P}_1, \hat{P}_2)$ for some distance measure.

We take advantage of corresponding Information Theory framework. Notice that the average log-loss accounts for the extra bits per symbol on top of the "true" probabilistic source's Entropy. This in turn, is equivalent to the Kullback–Leibler (KL) divergence [e.g., Section 2 in Begleiter et al., 2004]

$$D_{KL}(P, \hat{P}) = \sum_{X \in \Sigma^n} P(X) \log \left(\frac{P(X)}{\hat{P}(X)} \right)$$

for the true unknown distribution, P (defining optimal Entropy) and its estimation $\hat{P}$ (e.g., via PPM). Note, however, that the KL-divergence is not symmetric and unbounded.[3] Instead we use the Jensen-Shannon Divergence (JSD) which is a smoothed variant of the Kullback–Leibler (KL) divergence (see, Lin [1991]). The KL-divergence over distributions P and Q is defined as

$$\texttt{JSD}(P, Q) = \frac{1}{2} D_{KL}(P, M) + \frac{1}{2} D_{KL}(Q, M) \tag{1}$$

where $M = \frac{1}{2}(P + Q)$ is a mixture model of the two sources.

Direct computation of JSD and Entropy of a Variable Order Markov-model, like PPM, is challenging. Mainly, because the computation of the stationary process is non-trivial due to the varying order. Thus, we estimate the JSD by utilizing the elegant method of El-Yaniv et al. [1997] for measuring the statistical similarity of two sequences. Their method defines a sampling procedure over two input sequences that emits an empirical "most likely" sequence of their mixture distribution (i.e., M from Equation 1). Then, estimating the JSD between sources of the input sequences using the log-loss of the model learned from M and each of the original sequences [See, Figure 1 in El-Yaniv et al., 1997].

Recall we want to quantify the matching between two database textual columns. To this end, we turn each column into a character sequence by concatenating its values.[4] This results with the sequences $\mathbf{x}_1$ and $\mathbf{x}_2$. Then, we apply the statistical similarity method of El-Yaniv et al. [1997] to compute the JSD estimation $\widehat{\texttt{JSD}}(\mathbf{x}_1, \mathbf{x}_2)$, and compute the following matching score:

$$\texttt{matching}(\mathbf{x}_1, \mathbf{x}_2) = \max_{i \in \{1,2\}} \frac{\left| \widehat{\texttt{JSD}}(\mathbf{x}_1, \mathbf{x}_2) - D_{KL}(P_{\mathbf{x}_i}, \hat{P}_{\mathbf{x}_i}) \right|}{D_{KL}(P_{\mathbf{x}_i}, \hat{P}_{\mathbf{x}_i})} \tag{2}$$

wherein, $D_{KL}(P_{\mathbf{x}_i}, \hat{P}_{\mathbf{x}_i})$ is estimated using the average log-loss of an SLM estimator learnt from $\mathbf{x}_i$.

4 Numerical Examples

In this section we provide a glimpse of the properties of utilizing SLMs for over the column matching challenge. Our goal here is to provide motivating indications for the usefulness and challenges of the method. We use the Prediction by Partial Match version C sequential prediction algorithm [Bell et al., 1990] as our choice of SLM. See, Begleiter et al. [See, 2004, Section 3.2] for a full description.

4.1 Statistical Similarity Ingredients

The statistical similarity measure proposed in El-Yaniv et al. [1997], as detailed in Section 3.1, aims to quantify the degree of similarity between two sequences. The core idea involves constructing a

[3] Except for absolutely continuous source distributions.

[4] Using some arbitrary order.

"typical" mixture sequence and subsequently applying an SLM to the given pair of input sequences. The latter equals the excess redundancy of the mixture estimator, wherein "similar" sequences should be small (and in dissimilar high).

Sequence $\mathbf{x}$	Sequence $\mathbf{y}$	Mixture Sequence	Similarity	Matching Score
`abracadabra`	`arbadacarba`	`raaracara...caaracara`	3.23	0.23
`00000...`	`11111...`	`10001...`	4.04	34.36
`aabaab...`	`abbabba...`	`aaaaabbaab...`	1.37	4.44

Table 1: Example of sequence pairs and their statistical similarity ingredients.

4.2 Common Real-World Data

To experiment with common real-world values, we utilize a fast-food restaurant dataset [Datafiniti, 2019]. The data contains columns with websites, postal addresses, restaurant names, and geo-coordinates (lat/lon). Table 2 depicts the computed pairwise matching scores along with the corresponding generated mixture-source's sequences. Irrelevant columns such as geo-coordinate v.s. restaurant-name achieve high matching scores, while postal v.s. name columns induce a small score.

Column A	Column B	Matching-Score	Mixture-Sequence (Prefix)
lon	name	0.60	`3.1r8-8n2.6s-11e4418`
lat	name	0.49	`2.4S224r359n5.7-.36S`
lon	website	0.47	`474r3.1X444s83.c147:`
lon	postal	0.43	`90.0.82564590.7.2055`
lat	postal	0.33	`9.86811.5935.0378823`
lat	website	0.29	`681w931p842h469c1.5p`
lat	lon	0.29	`35.580742.697.5640.8`
name	website	0.23	`iver acomElexio B/ip`
postal	website	0.14	`017d920R044i301M3323`
name	postal	0.027	`7-E2le 0y's3 Sh9ven1`

Table 2: Matching common data sequences.

5 Concluding Remarks

We have introduced a novel statistical approach for quantifying the similarity between textual columns in databases. Our method leverages autoregressive symbol-level language models (SLMs), which demonstrate strong performance on the common short textual values typically found in database columns [see, e.g., Section 7.2 in van Renen et al., 2024].

The numerical examples presented in Section 4 highlight the effectiveness of our approach in capturing column-level content similarities and dissimilarities. While these empirical results provide promising evidence, further in-depth research is needed to fully validate the method's capabilities.

Beyond their application to database matching, symbol-level language models are a versatile tool for addressing various database-related challenges. For instance, the generative nature of these models can be exploited for tasks such as missing data completion, while their ability to compute statistical likelihood can be useful for determining the relevance of free-text values to specific columns.

In addition to the SLMs discussed in this paper, exploring other symbol-level models, such as symbol-level embedding models [e.g., Zhang et al., 2015], or combinations of symbol and word models could further enhance the performance and applicability of our approach.

References

Ron Begleiter, Ran El-Yaniv, and Golan Yona. On prediction using variable order markov models. *Journal of Artificial Intelligence Research*, 22:385–421, 2004.

T.C. Bell, J.G. Cleary, and I.H. Witten. *Text Compression*. Prentice Hall advanced reference series: Computer science. Prentice Hall, 1990. ISBN 9780139119910.

Leon Bornemann, Tobias Bleifuß, Dmitri V. Kalashnikov, Felix Naumann, and Divesh Srivastava. Natural key discovery in wikipedia tables. In *WWW*, pages 2789–2795, 2020.

Riccardo Cappuzzo, Paolo Papotti, and Saravanan Thirumuruganathan. Creating embeddings of heterogeneous relational datasets for data integration tasks. In *SIGMOD/PODS 2020, ACM International Conference on Management of Data, 14-19 June 2020, Portland, Oregon, USA*, 2020.

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory 2nd Edition (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, July 2006. ISBN 0471241954.

Datafiniti. Fast food restaurants across america, 2019. URL `https://data.world/datafiniti/fast-food-restaurants-across-america`.

Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. Turl: table understanding through representation learning. *Proc. VLDB Endow.*, 14(3):307–319, 2020.

Xiang Deng, Ahmed Awadallah, Chris Meek, Alex Polozov, Huan Sun, and Matthew Richardson. Structure-grounded pretraining for text-to-sql. In *NAACL 2021*, 2021.

Ran El-Yaniv, Shai Fine, and Naftali Tishby. Agnostic classification of markovian sequences. In *Advances in Neural Information Processing Systems*, volume 10. MIT Press, 1997.

Yue Gong, Zhiru Zhu, Sainyam Galhotra, and Raul Castro Fernandez. Ver: View discovery in the wild. In *ICDE*, pages 503–516, 2023.

Jianhua Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, 1991.

Matt Mahoney. Large text compression benchmark, 2024. URL `https://www.mattmahoney.net/dc/text.html`.

Sukprasert Pattara, Chan Gromit Yeuk-Yin, Rossi Ryan A., Du Fan, and Koh Eunyee. Discovery and matching numerical attributes in data lakes. In *IEEE Big Data*, pages 423–432, 2023.

Alexander van Renen, Dominik Horn, Pascal Pfeil, Kapil Eknath Vaidya, Wenjian Dong, Murali Narayanaswamy, Zhengchun Liu, Gaurav Saxena, Andreas Kipf, and Tim Kraska. Why TPC is not enough: An analysis of the amazon redshift fleet. In *VLDB 2024*, 2024.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28, 2015.