Understanding Cryptic Schemata in Large Extract-Transform-Load Systems

Alexander Albrecht, Felix Naumann

Technische Berichte Nr. 60

des Hasso-Plattner-Instituts für Softwaresystemtechnik an der Universität Potsdam

Technische Berichte des Hasso-Plattner-Instituts für Softwaresystemtechnik an der Universität Potsdam

Technische Berichte des Hasso-Plattner-Instituts für
Softwaresystemtechnik an der Universität Potsdam | 60

Alexander Albrecht | Felix Naumann

Understanding Cryptic Schemata in Large Extract-Transform-Load Systems

Universitätsverlag Potsdam

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.de/ abrufbar.

Universitätsverlag Potsdam 2012
http://verlag.ub.uni-potsdam.de/

Am Neuen Palais 10, 14469 Potsdam
Tel.: +49 (0)331 977 2533 / Fax: 2292
E-Mail: verlag@uni-potsdam.de

Die Schriftenreihe **Technische Berichte des Hasso-Plattner-Instituts für Softwaresystemtechnik an der Universität Potsdam** wird herausgegeben von den Professoren des Hasso-Plattner-Instituts für Softwaresystemtechnik an der Universität Potsdam.

ISSN (print) 1613-5652
ISSN (online) 2191-1665

Online veröffentlicht auf dem Publikationsserver der Universität Potsdam
URL http://pub.ub.uni-potsdam.de/volltexte/2012/6125/
URN urn:nbn:de:kobv:517-opus-61257
http://nbn-resolving.de/urn:nbn:de:kobv:517-opus-61257

Zugleich gedruckt erschienen im Universitätsverlag Potsdam:
ISBN 978-3-86956-201-8

Understanding Cryptic Schemata in Large Extract-Transform-Load Systems

Alexander Albrecht and Felix Naumann

Hasso Plattner Institute for Software Systems Engineering,
Prof.-Dr.-Helmert-Straße 2-3, 14482 Potsdam, Germany
{alexander.albrecht,felix.naumann}@hpi.uni-potsdam.de

Abstract. Extract-Transform-Load (Etl) tools are used for the creation, maintenance, and evolution of data warehouses, data marts, and operational data stores. Etl workflows populate those systems with data from various data sources by specifying and executing a DAG of transformations. Over time, hundreds of individual workflows evolve as new sources and new requirements are integrated into the system. The maintenance and evolution of large-scale Etl systems requires much time and manual effort. A key problem is to understand the meaning of unfamiliar attribute labels in source and target databases and Etl transformations. Hard-to-understand attribute labels lead to frustration and time spent to develop and understand Etl workflows.
We present a schema decryption technique to support Etl developers in understanding cryptic schemata of sources, targets, and Etl transformations. For a given Etl system, our recommender-like approach leverages the large number of mapped attribute labels in existing Etl workflows to produce good and meaningful decryptions. In this way we are able to decrypt attribute labels consisting of a number of unfamiliar few-letter abbreviations, such as `UNP_PEN_INT`, which we can decrypt to `UNPAID_PENALTY_INTEREST`. We evaluate our schema decryption approach on three real-world repositories of Etl workflows and show that our approach is able to suggest high-quality decryptions for cryptic attribute labels in a given schema.

Keywords: Etl, Data Warehouse and Repository, Data Integration

1 Cryptic Schemata

Etl systems are visual programming tools that allow the definition of complex workflows to extract, transform, and load heterogeneous data from one or more sources into a target database. Designing and maintaining Etl workflows requires significant manual work; the effort is up to 70% of the development cost in a typical data warehouse environment [8]. Etl workflows are stored in repositories to be executed periodically, e.g., daily or once a week. In the course of a complex data warehousing project up to several hundred Etl workflows are created by different individuals [1] and stored in such repositories. Moreover, the created ETL workflows get larger and more complex over time. Cryptic schemata are a well-known problem in the context of data warehousing. The main reason for cryptic schemata is the tendency to assign compact attribute labels consisting of a number of domain-specific abbreviations and acronyms.

Example 1 (Cryptic Attribute Labels). Consider a repository of ETL workflows to extract, transform, and load data of an OLTP system with attribute labels from the well-known TPC-E schema [19]. With the to-be-generated decryption pairs $\langle$`CO` $\approx$ `COMPANY`$\rangle$ and $\langle$`SP` $\approx$ `STANDARD, POOR`$\rangle$, it would be easier for a developer who is unfamiliar with this schema to identify the semantics of attribute labels, such as `CO_SP_RATE`.

Manually finding decryption pairs is ineffective and time consuming. To illustrate this problem, consider the attribute label `CO_SP_RATE` from the previous example. As this attribute label is too specific to have a directly mapped attribute label as decryption in the given ETL repository, the developer has to look up all pairs of mapped attribute labels that give a hint on an appropriate decryption of tokens `CO` and `SP`. With over ten thousand pairs of mapped attribute labels in the evaluated ETL repositories, manual schema decryption becomes infeasable. Readers are referred to Sec. 4 for a comprehensive overview of schema and ETL workflow characteristics in the given real-world ETL repositories.

In this paper, we regard ETL workflows as transformation graphs of the well-known model introduced by Cui and Widom [7]. This model is generally applicable to ETL workflows from common ETL tools: An ETL workflow is a directed acyclic transformation graph (DAG) and the topologically ordered graph structure determines the execution order of the connected transformations. In ETL, most transformations are a generalization of relational operators supporting multiple inputs and outputs. Two transformations are connected in the graph if one transformation is applied to the output obtained by the other transformation. Accordingly, attributes in the output schema of a transformation are connected to the corresponding attributes in the input schema of the subsequent transformations. We leverage these *connected* attribute labels in the existing ETL workflows as valuable source of information for automated schema decryption. We have observed that connected attributes with different labels often contain reasonable decryptions – often not for the entire label but for tokens within the labels. As cryptic attribute labels are often too specific to have a connected attribute label as decryption in the given ETL repository, the problem is to pair portions of the cryptic attribute label with portions of more descriptive attribute labels to produce reasonable decryptions.

Example 2 (Connected Attribute Labels). Consider the ETL repository from Example 1. Within some ETL workflows, extracted source attributes were renamed in the succeeding transformation to provide a better readability. For example, the attribute label `CP_COMP_CO_ID` was renamed to `COMPETITOR_COMPANY_ID` and `CO_CEO` to `COMPANY_CEO`. Thus, labels `CO_CEO` and `COMPANY_CEO` and labels `CP_COMP_CO_ID` and `COMPETITOR_COMPANY_ID` are connected, respectively.

As ETL tools allow the developer to drag-and-drop attribute labels from output to input schemata, there are many connected attributes with identical labels. But in large ETL repositories there is also a large number of connected attributes having different labels. There are several reasons for this, such as (1) source-, lookup-, and target-schemata used in an ETL workflow are often created independently and thus contain different attribute labels; (2) a data warehouse schema based on instances of cryptic source schemata uses attributes consisting of more descriptive tokens to provide a better readability; (3) copy-and-paste of entire transformations is a common practice in ETL development, which results in ETL sub-workflows connected to intermediate attributes with different labels.

In this paper, we focus only on attribute pairs between connected transformations for schema decryption. We ignore the connections among attribute labels within a single ETL transformation, because we observed that developers use no synonyms within a single transformation. Furthermore, our approach overcomes weaknesses in existing approaches, such as string and schema matching techniques. These methods lead to poor decryption results due to domain-specific abbreviations, acronyms, and tokens in ETL schemata. Moreover, it is infeasible to exploit data redundancies between different schemata to find pairs of corresponding attribute labels: The data created in the intermediate ETL processing steps is not persisted and we lack this helpful information. Re-executing and storing data from intermediate processing steps is an unrealistic assumption in a typical data warehouse scenario.

We make the following contributions: First, we introduce the concept of decryption pairs, as a practical approach for schema decryption. Second, we identify desirable characteristics of decryption pairs to efficiently find decryption pairs leveraging the large number of mapped attribute labels in a given ETL repository. Third, experiments on three real-world ETL repositories show the accuracy and efficacy of our approach. Finally, we introduce a generalized technique for tokenization of attribute labels.

2 Using Connected Attributes for Decryption

To illustrate our approach upfront we introduce a toy example of an ETL workflow in Fig. 1. The ETL workflow loads company data into a dimension table of a data warehouse. The extracted source data is the input of a lookup transformation. There, a company record is assigned a country from a lookup table using the company identifier as lookup key. Finally, the data is loaded into the data warehouse (DWH).

We observe that (1) attributes can be tokenized based on special characters. We also observe that (2) no two connected attributes have the same label. This is a typical situation if source, lookup, and target schemata were developed independently or for different purposes. Finally, we observe that (3) some attributes use abbreviations that appear in extended form in connected attribute labels. These observations were made repeatedly in our analysis of three real-world ETL

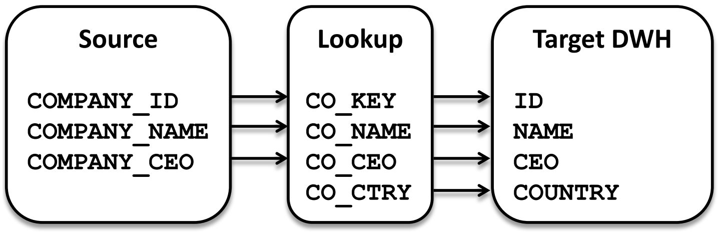

Fig. 1. An exemplary ETL workflow

repositories, each with up to several hundred Etl workflows containing thousands of connected attribute pairs with different labels.

Our decryption approach finds reasonable decryptions within the given Etl repository by making use of all three observations: For each Etl workflow in the Etl repository, we first break all labels into tokens, based on case-change or non-alphabetical separators. In the example, we tokenize using the underscore as separator. The second observation allows us to identify attribute labels with same or similar semantics. If data from an attribute in the source or preceding transformation is used as input for some attribute in the target or subsequent transformation, it is reasonable to assume that their two labels are semantically related – in most cases they are semantically equivalent. For instance, `CO_CTRY` and `COUNTRY` in Fig. 1 are such *connected* attribute labels. Finally, using the third observation, we realize that the tokens `CO` and `COMPANY` co-occur in multiple pairs of connected attribute labels, leading us to believe that they are synonymous (and not for instance `CO` and `COUNTRY`).

With the identified decryptions from the Etl repository, we can suggest decryptions for cryptic attribute labels of a given schema. For instance, given a schema with the cryptic attribute label `CO_ID`, it is decrypted to `COMPANY_ID` using the decryption $\langle \texttt{CO} \approx \texttt{COMPANY} \rangle$ derived from Etl workflow in Fig. 1.

3 Schema Decryption

Our goal is to suggest "decryption pairs" to provide developers with a better understanding of cryptic schemata and Etl workflows.

Definition 1 (ETL Workflow). *An* Etl workflow *comprises a set of transformations T with input and output schemata, interconnected with each other forming a directed acyclic graph (DAG). Let $W = (V, E)$ be a DAG representing an* Etl *workflow consisting of a set of vertices V representing the involved transformations. The edges $e \in E \subseteq V \times V$ connect the output schema of one transformation with the input schema of another transformation, i.e., an edge e represents an ordered pair of transformations.*

Input Schema	DEBT_RT, RESID_VAL_AT_RISK, UNP_PEN_INT
Top-5 Decryption Pairs	⟨UNP ≈ UNPAID⟩, ⟨INT ≈ INTEREST⟩, ⟨PEN ≈ PENALTY⟩, ⟨RT ≈ RATE⟩, ⟨RESID, VAL ≈ RESIDUAL, VALUE⟩

Table 1. Sample results for a Spanish ETL repository from the finance industry domain.

In this section we explain how to find decryptions for cryptic schemata leveraging the large number of connections among attribute labels in the given ETL repository.

Definition 2 (Connected Attribute Labels). *Two attribute labels are* connected *if there exists at least one* ETL *workflow in which a* direct link *is established between the corresponding attributes in the output and input schemata of two connected transformations.*

3.1 Our Schema Decryption Approach

For a given schema consisting of a set of cryptic attribute labels, our algorithm returns a ranked list of decryptions in descending order of their frequency of occurrence in the given ETL repository. We regard an attribute label as a set of tokens and represent a decryption as a pair of corresponding token sets that appear to be used synonymously within the ETL repository. The algorithm iterates over all attribute labels l from the given schema and returns the set of all applicable decryptions to decrypt l. Thus, for each attribute label l, we create all possible decryptions leveraging the large number of connected attribute labels in the given ETL repository (see Sec. 3.2). Each decryption is then added to the result. Finally, the algorithm returns a compact list of decryptions, ranked in descending order of their frequency of occurrence in the ETL repository.

Let T_i and T_j be disjoint sets of tokens, i.e., $T_i \cap T_j = \emptyset$. We define a decryption pair $\langle T_i \approx T_j \rangle$, where T_i and T_j are synonyms. We regard token sets and not single tokens, because a decryption often applies to multiple tokens or even contains multiple tokens. Table 1 shows a sample schema decryption in which individual tokens but also token sets are decrypted. A decryption pair is applicable to an attribute label only if either all tokens from T_i, or all tokens from T_j occur in the (tokenized) attribute label. Tokens from T_i or T_j may occur in any order in the attribute label.

Definition 3 (Decryption Pair). *Let T_i and T_j be disjoint sets of tokens. We call $\langle T_i \approx T_j \rangle$ a* decryption pair *if the token set denoted by T_i is synonymous to the token set denoted by T_j.*

Finally, to suggest a compact list of decryption pairs, we remove all subsumed decryption pairs from the result list, retaining only maximal decryption pairs.

Definition 4 (Maximal Decryption Pair). *Let $L = \{(l_m, l_n)\}$ be the set of pairs of connected attribute labels containing decryption pair $\langle T_i \approx T_j \rangle$. We call $\langle T_i \approx T_j \rangle$ a* maximal decryption pair *if there is no decryption pair $\langle T_i' \approx T_j' \rangle$ for every $\{(l_m, l_n)\} \in L$ with $T_i \subseteq T_i'$ and $T_j \subseteq T_j'$.*

Example 3 (Maximal Decryption Pair). Consider the three created decryption pairs ⟨`SP` $\approx$ `STANDARD`⟩, ⟨`SP` $\approx$ `POOR`⟩, ⟨`SP` $\approx$ `STANDARD, POOR`⟩ derived from the same pairs of connected attribute labels. We only suggest ⟨`SP` $\approx$ `STANDARD, POOR`⟩ and remove the other two subsumed decryption pairs from the result list.

3.2 Finding Decryption Pairs

We now describe how we identify decryption pairs $\langle T_i \approx T_j \rangle$: Given an attribute label and some contained tokens T_i, we want to find all applicable decryption pairs for T_i. To this end, we search among all connected attribute labels in the given Etl repository for those that contain T_i. More formally, we consider all attribute labels l that contain T_i and are connected to some attribute label containing no subset of T_i. Using these pairs of connected attribute labels, we choose candidate decryption pairs $\langle T_i \approx T_j \rangle$, where T_j is some subset of tokens from the other attribute label. For the given real-world Etl repositories, tokens are delimited by the underscore character. For the general case, it may happen that tokens of attribute labels are not always delimited with a special character. We will consider the general problem of attribute label tokenization later in Sec. 5. A candidate decryption is added to the result if all three of the following observations hold.

Our first observation is that connected attribute labels often share tokens, i.e., such tokens appear in both connected attribute labels. In Example 2 in Sec. 1, connected attribute labels `CP_COMP_CO_ID` and `COMPETITOR_COMPANY_ID` share token `ID` and connected attribute labels `CO_CEO` and `COMPANY_CEO` share token `CEO`. Considering shared tokens for decryption makes no sense, since their counterpart is the same token in the other label. Thus, we do not create decryption pairs containing a shared token. In the example we would not create a decryption pair such as ⟨`CO` $\approx$ `CEO`⟩; the token `CEO` is already 'taken' by its counterpart `CEO` in the other attribute label.

Our second observation (and assumption) is that synonymous token sets are never used together in a single attribute label, as it would be useless to label a single attribute with synonyms. That is, if tokens x and y appear together in one attribute label, there is no decryption pair $\langle T_i \approx T_j \rangle$ with $x \in T_i$ and $y \in T_j$ or vice versa. Considering the attribute labels in Example 2 in Sec. 1, we do not create decryption pair ⟨`CO` $\approx$ `COMP`⟩ from a corresponding pair of connected attribute labels, because both tokens appear together in the attribute label `CP_COMP_CO_ID` and thus are very unlikely to represent synonyms.

Our last observation is that a decryption is consistently used between two connected transformations. To determine the consistency of a decryption pair derived from a pair of connected attribute labels, we determine its correctness (*confidence*) and frequency of occurrence (*hit-ratio*) throughout the corresponding schemata of the two connected transformations: Let $L_{T_i} = \{(l_m, l_n)\}$ be the set of pairs of connected attribute labels in which all tokens of T_i appear either in l_m or l_n (but not both, as these are the trivial cases). These pairs represent the *positive class* for the decryption of T_i. Further, let L_{T_i,T_j} be the set of pairs of connected attribute labels in which T_i appears in one label and T_j in the other label. These pairs represent the *true positive class* for the decryption. Note that $L_{T_i,T_j} \subseteq L_{T_i}$. Then we define confidence as

$$\mathit{confidence}_{T_i,T_j} = \frac{|L_{T_i,T_j}|}{|L_{T_i}|}$$

and we define the hit-ratio for decryption pair $\langle T_i \approx T_j \rangle$ as

$$\mathit{hit\text{-}ratio}_{T_i,T_j} = \frac{|L_{T_i,T_j}|}{|L_{T_j}|}.$$

Note that both confidence and hit-ratio have to be considered. A high hit-ratio may result in a poor confidence, i.e., the decryption from T_i to T_j may occur frequently, but T_i also occurs frequently with other tokens in connected attribute labels. Similarly, a high confidence, e.g., achieved by returning only correct decryptions producing no false positives, may result in a poor hit-ratio.

Example 4 (Quality of Decryption Pairs). Consider the connected attribute labels from Example 2 in Sec. 1. Decrypting `CO` to `COMPETITOR` might have a high hit-ratio in the corresponding schemata of the two connected transformations, if `COMPETITOR` often co-occurs with `CO`. As `CO` also occurs frequently with tokens different from `COMPETITOR`, such as `COMPANY`, decrypting `CO` to `COMPETITOR` results in a poor confidence. On the other hand, decrypting `COMP` to `COMPANY` might have a high confidence: labels with the token `COMP` are almost always connected to labels containing `COMPANY`, but labels containing `COMPANY` might also often be connected with labels containing `CO` (but not `COMP`). Thus the decryption of `COMP` to `COMPANY` has a low hit-ratio.

As the quality of a decryption pair depends on both measures, we choose the harmonic mean of confidence and hit-ratio to determine the quality of a decryption pair. The harmonic mean is a typical way to aggregate measures:

$$\mathit{harmonicMean} = \frac{2 \cdot \mathit{confidence} \cdot \mathit{hit\text{-}ratio}}{\mathit{confidence} + \mathit{hit\text{-}ratio}}$$

As the reverse decryption of T_j to T_i results in the same harmonic mean value, we can ignore order. In our experiments, we choose a threshold value of 80% for the harmonic mean to suggest consistent decryptions from pairs of connected attribute labels.

4 Experimental Study

We evaluated our schema decryption approach on three real-world ETL repositories. These repositories were created separately by different departments of a banking organization in Switzerland (CH), Germany (DE), and Spain (ES) using Informatica PowerCenter[1]. Informatica provides ETL workflow specifications in

[1] `www.informatica.com`

Repository	**CH**	**DE**	**ES**
Etl workflows	45	131	167
Schemata	1591	3687	1994
Avg. attributes / schema	14	12	9
Min. attributes / schema	1	1	1
Max. attributes / schema	155	270	143
Pairs of connected attribute labels	11108	23076	10183
Distinct pairs of connected attribute labels	2502	3589	2483
Distinct pairs of connected attribute with different labels	1343	1798	984

Table 2. Characteristics of test ETL repositories

Repository	**CH**	**DE**	**ES**
Distinct attribute labels	2015	3045	2211
Distinct tokens used in attribute labels	1015	1393	1237
Avg. tokens / attribute label	2	3	3
Min. tokens / attribute label	1	1	1
Max. tokens / attribute label	4	10	6

Table 3. Characteristics of attribute labels in test ETL repositories

a proprietary Xml format, which our schema decryption algorithm takes as input. Schemata and connections between attribute labels are pre-indexed offline and are used to compute schema decryptions in an online fashion. Our algorithm operates efficiently and typically returns a ranked list of decryption pairs for a given schema in under 1 second.

4.1 Real-world ETL repositories

Table 2 shows the characteristics of the three Etl repositories to emphasize that Etl development results in many and complex Etl workflows. Each Etl workflow consists of many source, target, and transformation schemata, some of which contain several hundred attributes. There are many connected attribute labels, but for schema decryption we are only interested in pairs of connected attributes with different labels. The number of distinct pairs is shown in the last line of Table 2 and confirms our observation from the beginning that there is indeed a large number of connected attributes with different labels.

Table 3 gives an overview of the characteristics of the attribute labels in the Etl repositories. It is apparent that in all three Etl repositories there is a large number of distinct tokens used for the large number of distinct attribute labels. Figure 2 shows the distribution of the number of tokens used in distinct attribute labels. In the DE and ES repository for more than fifty percent of all distinct labels more than two tokens were used by the developers. The attribute labels in the CH repository contain almost always fewer than three tokens. Figure 3 gives an overview on the average number of tokens used to label attributes in a schema.

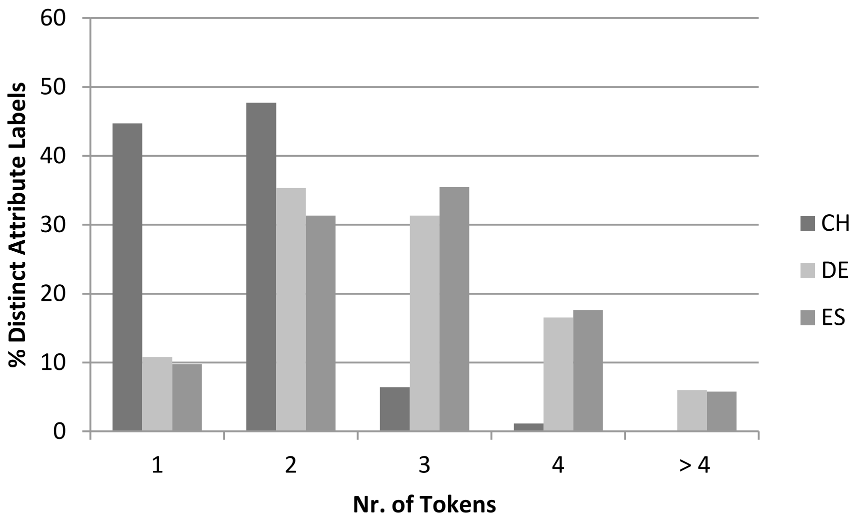

Fig. 2. No. of tokens in distinct attribute labels

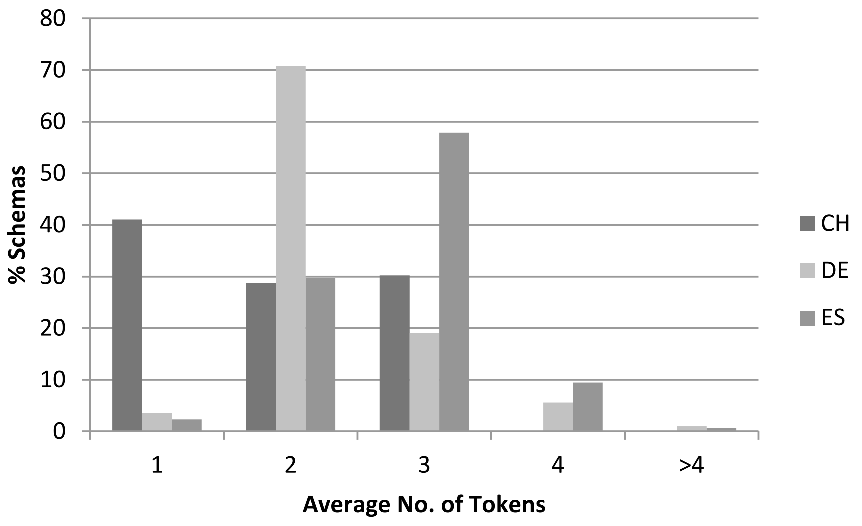

Fig. 3. Average No. of tokens per attribute in schemata

4.2 Evaluation Technique

We have successfully tested our schema decryption approach on all ETL work-flows from the three given ETL repositories. To evaluate the accuracy of our schema decryption, we randomly selected from each repository three schemata

Rank	**Decryption Pair**	***rel(i)***	**Precision**
1	$\langle$`UNP` $\approx$ `UNPAID`$\rangle$	1	1
2	$\langle$`INT` $\approx$ `OUTPUT`$\rangle$	0	1/2
3	$\langle$`PEN` $\approx$ `PENALTY`$\rangle$	1	2/3

Table 4. Calculating average precision for top-3 decryption pairs

consisting of at least 20 attribute labels and use schema decryption to generate ranked lists of decryption pairs for the selected nine schemata.

In our evaluation we consider the top-k decryption pairs p_i in the ranked list. Let i be the position of p_i in the ranked list, i.e., $i \leq k$. Then, we manually determine whether p_i is relevant/correct or not for the given schema, i.e., we set $rel(i)$ to 0 or 1, respectively. We consider a decryption pair to be accurate if it helps to understand the underlying semantic domain of the original attribute label. Then, we calculate the average precision for the top-k decryption pairs. The average precision is the average of the precision values for the seen accurate decryption pairs [3]:

Definition 5 (Average Precision). *Let $P(i)$ be the precision of the first i suggested decryption pairs. Then, the average precision at position k is*

$$AvP_k = \frac{\sum_{i=1}^{k} P(i) \cdot rel(i)}{\sum_{i=1}^{k} rel(i)}$$

where precision is defined as $P(i) = \frac{\sum_{j=1}^{i} rel(j)}{i}$

Example 5 (Average Precision). Table 4 shows an illustrative top-3 example of ranked decryption pairs. The examples are from the Etl repository from Spain (ES). The precision values after each new accurate decryption is observed are 1 and $\frac{2}{3}$. Thus, the *average precision* of the top 3 results (with two seen accurate decryptions) is given by $(1 + \frac{2}{3})$ / 2 = 83%.

4.3 Results

Figure 4 shows the accuracy of our schema decryption approach. We measure the mean average precision for each of the experiments and show the top-5, top-10, top-15, and top-20 results. For all three repositories the algorithm achieves an accuracy of above 90%. For the CH repository the algorithm provides the best accuracy. This is expected, because if there is a pair of connected attributes with different labels in the CH repository, it often contains an accurate decryption. The experiments demonstrate the advantages of identifying decryption pairs based on tokens and based on their characteristics. Additional experiments confirmed that our approach results in a significantly lower number of incorrect, conflicting, and redundant decrpytion pairs compared to other approaches. We compared our approach to a straightforward alternative of choosing entire labels

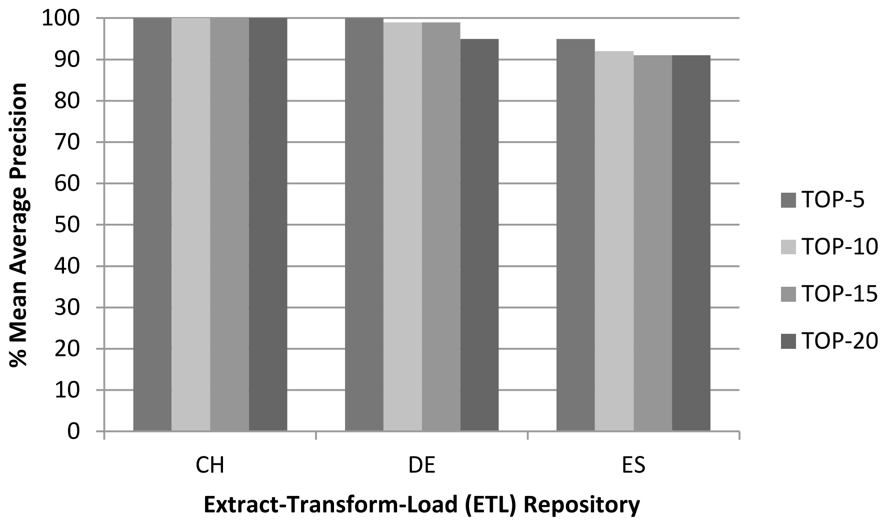

Fig. 4. Schema Decryption Accuracy for Etl Repositories (CH), (DE), (ES)

of two connected attributes as decryption pair. In addition, we compared our approach against different string-similarity measures, such as Levenshtein distance [10]. Figure 5 shows the accuracy of the simple strategy of choosing entire labels of two connected attributes as decryption. This simple approach often leads to a large number of incorrect and conflicting suggestions. Furthermore, a large number of correct suggestions contain redunant decryptions, as frequent pairs of connected attributes with different labels often contain the same correct decryption for the same portion of tokens.

5 A Generalized Technique for Tokenization

For our given real-world Etl repositories, the tokens are delimited by the underscore character. The experiments have shown that the simple tokenization strategy based on special characters already yields good results. For the general case, it may happen that tokens of attribute labels are not always delimited with a special character. Our schema decryption solution needs a set of tokens that belong to some standard vocabulary to perform effectively. Therefore, we consider in this section the general problem of attribute label tokenization, apart from the simple tokenization strategy.

Our tokenization approach segments attribute labels of a given schema into a set of "meaningful" tokens. A meaningful token has a semantics of its own or it modifies the semantics of another token. For instance, the attribute label `ORDERSTATUS` from the well-known Tpc-h schema [20] should be tokenized to the meaningful tokens `ORDER` and `STATUS`. We supply a more formal definition of

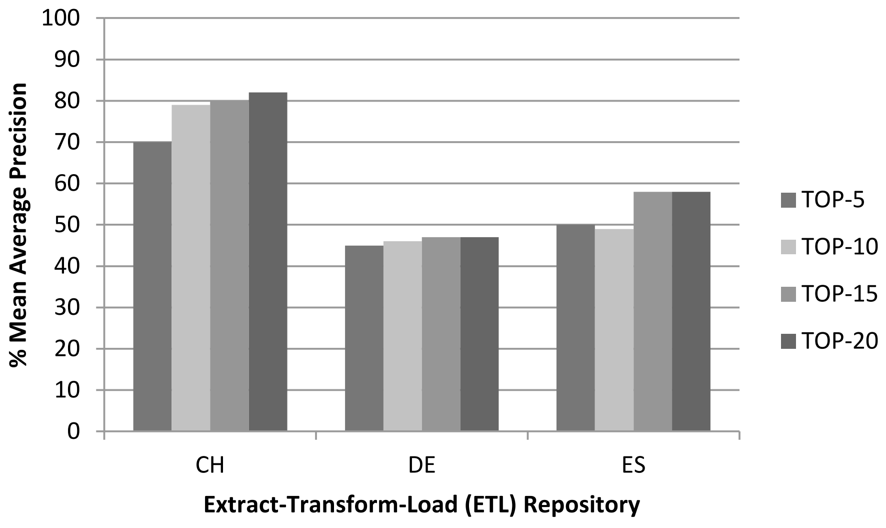

Fig. 5. Schema Decryption Accuracy for using entire attribute labels as decryption

a good tokenization later. Our tokenization approach is based on the discovery of token patterns for attribute labels making use of any pre-existing separators and tokens in only some of the attributes of the given schemata.

5.1 Discovery of Token Patterns

Attribute labels almost always contain tokens that belong to some standard vocabulary. When choosing an attribute label for a relational schema, often several tokens from this standard vocabulary are chosen, describing the underlying attribute domain. We call these tokens *frequent tokens*. These tokens frequently occur in different attribute labels, which means that the same token, or more generally the same pattern containing this token (and potentially more tokens), is used in several different contexts. For instance, `ORDER`, `STATUS`, and `PRICE` are frequent tokens often used for labeling attributes of transactional data records.

Definition 6 (Frequent token). *Given a set A of attribute labels and a minimum support $s \in \mathbb{N}$, a token t is* frequent *if there exists a subset $A' \subseteq A$, such that t is a token in all attribute labels $a_i \in A'$ and $|A'| \geq s$.*

Definition 7 (Standard vocabulary). *Given a set A of attribute labels and a minimum support $s \in \mathbb{N}$, the* standard vocabulary V_s *is the set of all frequent tokens for A and s.*

There are also tokens that do not have the minimum-support. These tokens are used during schema creation when no combination of standard tokens is suitable or known to unambiguously describe an attribute label. We call these tokens *non-standard tokens*; an example is the attribute label `MFGR` from the well-known TPC-H schema. `MFGR` is an abbreviation for manufacturer and appears in only one attribute label of the TPC-H schema.

It should be noted that, in analogy to tables in relational databases, processed data records in ETL workflows may contain only distinct attribute labels. Thus, multiple tokens are possibly concatenated with a delimiter character to create the final attribute label, such as `C_AREA_1`, `C_AREA_2`, and `C_AREA_3` from the TPC-E schema. In other cases tokens are concatenated without a delimiter, such as `EXTENDEDPRICE` in the TPC-H schema. The simple form of tokenizing attribute labels is to parse them into tokens based on known delimiter characters. The experiments have shown that this simple tokenization strategy already yields good results. In general, however, this naïve strategy may result in an inappropriate standard vocabulary: The delimiter character may be used inconsistently in a schema, i.e., the delimiter character is omitted when tokenization is intuitively clear.

Example 6 (Tokenization). Given a schema with attribute set A. Assume that the set A of given attribute labels consists of only the three TPC-H attributes `ORDERKEY`, `ORDER_STATUS`, and `LINESTATUS`. With a minimum support $s > 1$, by Definition 7 the standard vocabulary is empty. Therefore, all four tokens are regarded as non-standard tokens.

For tokenization, we can regard `ORDER_STATUS` as a reliable source of information, since this tokenization is meaningful to some human. The whole attribute label is of no use for tokenization in this example, but the tokens {`ORDER`, `STATUS`} are applicable in different contexts, i.e., the adjacent substrings or tokens might differ. The idea of using these two tokens as *token patterns* allows us to deduce a meaningful tokenization using the underscore as separator character: `ORDER_KEY`, `ORDER_STATUS`, and `LINE_STATUS`.

Given a schema with attribute set A, we want to iteratively populate an attribute set A^T, which contains tokenized attribute labels. We initialize A^T with the set of attribute labels that contain at least one separator character. Additionally, A^T contains all frequent tokens that also appear as an individual attribute label.

To derive meaningful tokenizations, we use so-called *token patterns*, which consist of frequent tokens and their well-defined matching positions in an attribute label, defined below. We derive these patterns from the set A^T of already tokenized attribute labels and attribute labels representing a standard token. Whenever a pattern match is found in a non-standard token of an attribute label, the attribute label is tokenized accordingly. In the example above, the patterns $[\texttt{(ORDER)}^F]$ and $[\texttt{(STATUS)}^L]$ are derived from the tokenized attribute label `ORDER_STATUS`. To indicate the position of the matched tokens in the attribute label, we use superscripts: F, I, or L represent the matched token pattern to be prefix, infix, or suffix respectively. Tokens that are neither first nor last are defined as being in-between. In our example, the pattern matching results in the tokenized attribute labels `ORDER_KEY` and `LINE_STATUS`. With the following definition, we formalize how to obtain candidate token patterns in general:

Definition 8 (Candidate Token Pattern). *We obtain the set of candidate token patterns $[\mathtt{p}^l]$ from all possible token sequences in an attribute label $a \in A^T$ and we set $l = F$, $l = I$ or $l = L$, if* $\mathtt{p}$ *occurs as prefix, as infix, or as suffix in A^T, respectively. Note that up to 4 token patterns can be produced from one token sequence, i.e., $[\mathtt{p}^F]$, $[\mathtt{p}^I]$, $[\mathtt{p}^L]$ and $[\mathtt{p}^{F,L}]$.*

We take into account that some ETL workflows already contain a number of tokenized attribute labels. So we incorporate this valuable source of information into our approach to obtain candidate patterns. To obtain all possible candidate patterns $[\mathtt{p}^l]$ from a given attribute label consisting of k tokens, we generate all possible $2^k - 1$ token sequences and create candidate patterns comparable to a regular expressions.

Example 7 (Candidate Token Pattern). Consider the attribute label for the start date of a quarter from the TPC-E schema, `FI_QTR_START_DATE`. From this attribute label we create candidate token patterns, such as $[(\mathtt{FI})^F]$, $[(\mathtt{DATE})^L]$, $[(\mathtt{QTR})(\mathtt{START})^I]$, and $[(\mathtt{FI})(.+)(\mathtt{DATE})^{F,L}]$. Among these candidates the next step is to choose the most meaningful ones.

5.2 Choosing Token Patterns

To reduce the very large number of candidate patterns to a set of *meaningful* token patterns, we require a token pattern to occur in different contexts. We search for matches of a token pattern p in all non-standard tokens. For each match we collect the token pattern-context, i.e., all substrings adjacent to the tokens specified in the token pattern. For a given pattern p all these contexts form a set N_p of neighborhood strings.

Additionally, there are already neighborhood tokens for $a' \in A^T$, if a' contains token pattern p. From the set of candidate token patterns, we consider only those token patterns p with no neighborhood t that frequently occurs in N_p. All candidate token patterns with at least one frequent neighborhood are regarded as incomplete. We model this property with the conditional probability of t given p. We favor patterns with a low probability, i.e., they occur in many different contexts. We discard all candidates with a probability higher than a 50% threshold. All other are regarded as meaningful token patterns.

Example 8 (Choosing Token Patterns). Consider the pattern $p_1 = [(\mathtt{L})^I]$ extracted from attribute label `C_L_NAME` from the TPC-E schema in Fig. 6. The neighborhood tokens of p_1 are {`AP, C, NAME`}, because the pattern occurs not only in `C_L_NAME`, but also in `AP_L_NAME`. As the conditional probability $P(\mathtt{NAME}|p_1) = 1$, the candidate pattern p_1 is discarded: Apparently, the token `L` always occurs together with a token `NAME` and is thus not meaningful on its own. But $p_2 = [(\mathtt{NAME})^L]$ is considered a meaningful token pattern, because the pattern occurs next to 14 different tokens (`CO`, `IN`, `TX`, etc.) in the TPC-E schema, and thus no conditional probability exceeds the value of 12.5%.

TPC-E	**TPC-H**
Meaningful token patterns [(1)^L], [(2)^L]	**Meaningful token patterns** [(DATE)^L], [(KEY)^L], [(LINE)^F], [(ORDER)^F], [(PRICE)^L], [(PRIORITY)^L], [(SHIP)^F], [(STATUS)^L], [(SUPP)^F]
AD_CTRY, AD_ID, $\boxed{\texttt{AD\_LINE\_1}}$, $\boxed{\texttt{AD\_LINE\_2}}$, AD_ZC_CODE, AP_ACL, AP_CA_ID, AP_F_NAME, AP_L_NAME, AP_TAX_ID, B_COMM_TOTAL, B_ID, B_NAME, B_NUM_TRADES, B_ST_ID, CA_BAL, CA_B_ID, CA_C_ID, CA_ID, CA_NAME, CA_TAX_ST, CH_CHRG, CH_C_TIER, CH_TT_ID, CO_AD_ID, CO_CEO, CO_DESC, CO_ID, CO_IN_ID, CO_NAME, CO_OPEN_DATE, CO_SP_RATE, CO_ST_ID, CP_COMP_CO_ID, CP_CO_ID, CP_IN_ID, CR_C_TIER, CR_EX_ID, CR_FROM_QTY, CR_RATE, CR_TO_QTY, CR_TT_ID, CT_AMT, CT_DTS, CT_NAME, CT_T_ID, CX_C_ID, CX_TX_ID, C_AD_ID, C_AREA_1, C_AREA_2, C_AREA_3, C_CTRY_1, C_CTRY_2, C_CTRY_3, C_DOB, C_EMAIL_1, C_EMAIL_2, C_EXT_1, C_EXT_2, C_EXT_3, C_F_NAME, C_GNDR, C_ID, C_LOCAL_1, C_LOCAL_2, C_LOCAL_3, C_L_NAME, C_M_NAME, C_ST_ID, C_TAX_ID, C_TIER, DM_CLOSE, DM_DATE, DM_HIGH, DM_LOW, DM_S_SYMB, DM_VOL, EX_AD_ID, EX_CLOSE, EX_DESC, EX_ID, EX_NAME, EX_NUM_SYMB, EX_OPEN, FI_ASSETS, FI_BASIC_EPS, FI_CO_ID, FI_DILUT_EPS, FI_INVENTORY, FI_LIABILITY, FI_MARGIN, FI_NET_EARN, FI_OUT_BASIC, FI_OUT_DILUT, FI_QTR, FI_QTR_START_DATE, FI_REVENUE, FI_YEAR, HH_AFTER_QTY, HH_BEFORE_QTY, HH_H_T_ID, HH_T_ID, HS_CA_ID, HS_QTY, HS_S_SYMB, H_CA_ID, H_DTS, H_PRICE, H_QTY, H_S_SYMB, H_T_ID, IN_ID, IN_NAME, IN_SC_ID, LT_DTS, LT_OPEN_PRICE, LT_PRICE, LT_S_SYMB, LT_VOL, NI_AUTHOR, NI_DTS, NI_HEADLINE, NI_ID, NI_ITEM, NI_SOURCE, NI_SUMMARY, NX_CO_ID, NX_NI_ID, SC_ID, SC_NAME, SE_AMT, SE_CASH_DUE_DATE, SE_CASH_TYPE, SE_T_ID, S_52WK_HIGH, S_52WK_HIGH_DATE, S_52WK_LOW, S_52WK_LOW_DATE, S_CO_ID, S_DIVIDEND, S_EXCH_DATE, S_EX_ID, S_ISSUE, S_NAME, S_NUM_OUT, S_PE, S_START_DATE, S_ST_ID, S_SYMB, S_YIELD, TH_DTS, TH_ST_ID, TH_T_ID, TR_BID_PRICE, TR_B_ID, TR_QTY, TR_S_SYMB, TR_TT_ID, TR_T_ID, TT_ID, TT_IS_MRKT, TT_IS_SELL, TT_NAME, TX_ID, TX_NAME, TX_RATE, T_BID_PRICE, T_CA_ID, T_CHRG, T_COMM, T_DTS, T_EXEC_NAME, T_ID, T_IS_CASH, T_LIFO, T_QTY, T_ST_ID, T_S_SYMB, T_TAX, T_TRADE_PRICE, T_TT_ID, WI_S_SYMB, WI_WL_ID, WL_C_ID, WL_ID, ZC_CODE, ZC_DIV, ZC_TOWN	ACCTBAL, ADDRESS, AVAILQTY, BRAND, CLERK, COMMENT, $\boxed{\texttt{COMMIT\_DATE}}$, CONTAINER, $\boxed{\texttt{CUST\_KEY}}$, DISCOUNT, EXTENDED_PRICE, $\boxed{\texttt{LINE\_NUMBER}}$, LINE_STATUS, MFGR, MKTSEGMENT, NAME, $\boxed{\texttt{NATION\_KEY}}$, $\boxed{\texttt{ORDER\_DATE}}$, $\boxed{\texttt{ORDER\_KEY}}$, $\boxed{\texttt{ORDER\_PRIORITY}}$, $\boxed{\texttt{ORDER\_STATUS}}$, $\boxed{\texttt{PART\_KEY}}$, PHONE, QUANTITY, $\boxed{\texttt{RECEIPT\_DATE}}$, $\boxed{\texttt{REGION\_KEY}}$, $\boxed{\texttt{RETAIL\_PRICE}}$, RETURNFLAG, $\boxed{\texttt{SHIP\_DATE}}$, $\boxed{\texttt{SHIP\_INSTRUCT}}$, $\boxed{\texttt{SHIP\_MODE}}$, $\boxed{\texttt{SHIP\_PRIORITY}}$, SIZE, $\boxed{\texttt{SUPP\_KEY}}$, $\boxed{\texttt{SUPP\_LYCOST}}$, TAX, $\boxed{\texttt{TOTAL\_PRICE}}$, TYPE

Fig. 6. Boxes highlight new deduced tokenizations using a minimum-support greater than 1 for standard tokens. TPC-E is already well-tokenized to begin with, but TPC-H is greatly improved.

Given a candidate token pattern p and its set N_p of neighborhood substrings and tokens, we more formally define its choice as token pattern as follows:

Definition 9 (Meaningful Patterns). *Let $P(p)$ be the fraction of attribute labels containing the pattern p, and let $P(p\|t)$ be the fraction of attribute labels containing the pattern p adjacent to string $t \in N_p$. Token pattern p is meaningful if and only if*

$$\arg\max_{t \in N_p} P(t|p) = \arg\max_{t \in N_p} \frac{P(p\|t)}{P(p)} \leq 50\% \tag{1}$$

To summarize, given the initial set of tokenized attribute labels A^T, we are able to use it as a bootstrap corpus for iteratively deriving token patterns. At every iteration new meaningful token patterns are created considering attribute labels in A^T. Then, meaningful token patterns are used for tokenizing non-standard tokens. The resulting tokenizations are added to A^T. We stop the iteration when no new tokenization is added to A^T. At the end, A^T contains all meaningful tokenization of the given schema.

Fig. 6 shows the result using the whole Tpc-e schema as A^T and the sample set of already tokenized attribute labels $A^T = \{\texttt{LINE\_STATUS}, \texttt{EXTENDED\_PRICE}\}$ for the Tpc-h schema. For the Tpc-e schema, tokenization stops after two iteration steps. Tokenization of the Tpc-h schema requires five iteration steps. In Fig. 6, boxes highlight all created tokenizations and the first line shows all meaningful token patterns that were applied to at least one attribute label during iterations.

6 Related work

Although the practical importance of Etl in data integration is significant [22], only little research on Etl at a meta-level has been performed. Most related research results improve Etl workflow modeling [13, 14, 21], and there are only a few implementations that support further processing on Etl workflows. Examples include the optimization of Etl workflows [16, 17] and the generation of Etl workflow reports [15].

Our work is mainly related to research on schema normalization in the field of data integration [18] and attribute-synonym finding for relational tables and spreadsheet data in web pages [6]. There is also some work on schema normalization in the area of schema matching [4]. The authors of [11] introduced schema normalization as an important pre-processing step in schema matching to improve the discovery of semantically similar schema elements. Therefore, labels of schemata are tokenized based on case-change or non alphabetical characters. For tokens, an approximate lookup in a global dictionary, such as WordNet [12], is performed to find a common representation. String matching techniques, such as Levenshtein distance [10], are used to perform the approximate lookups.

The authors of [6] point out that distance metrics and global dictionaries, as used in schema matching, are not appropriate to automatically find synonyms for arbitrary attribute labels. This observation is supported by our experimental results: Common distance metrics result in poor decryptions and global dictionaries lead to a relatively poor coverage of domain-specific abbreviations, acronyms, and tokens. Furthermore, we propose a generalized technique for tokenization. In contrast, schema matching approaches perform tokenization only with the simple strategy of tokenizing attribute labels based on case-change or known delimiters.

The authors of [6] propose a large-scale discovery method to automatically find synonyms for attribute labels. The source of information is a corpus of 125 million independently created relational tables extracted from 14.1 billion Html tables. Their approach is based on pairs of attribute labels co-occurring with same context attributes. As already pointed out in the introduction, this type of web-scale analysis is infeasible for Etl systems: With our approach we can identify accurate decryptions from a substantially smaller corpus of examples compared to approaches that rely on a large set of web-scale example data. In addition, we regard tokens and not entire attribute labels in order to achive a high quality for schema decryption.

Sorrentino et al. present a semi-automatic technique for schema normalization and motivate the importance of incorporating individual examples in the process of schema normalization [18]. This work describes the importance of using labels from corresponding attribute labels in schema normalization. Pairs of corresponding attribute labels are extracted from *complementary schemata* connected by primary key to foreign key relationships. In contrast, ours is the first work that incorporates corresponding attribute labels as source of information for fully-automated, token-based schema decryption.

The methods in [2] use the Minimum Description Length (MDL) principle [9] to capture regularities between two matching strings. In particular, the authors address the related scenario of matching textual dissimilar strings motivated by the fact that common distance metrics, such as Levenshtein distance, are inappropriate in such a scenario. Similar to [6], the introduced techniques rely on a large set of web-scale example data. Hence, these techniques to identify synonyms cannot be applied to the schema decryption problem we consider in this paper. There is also MDL based work on word segmentation [5] that is related to our tokenization approach. In general, we consider a comparison of our approach with MDL based approaches to be very promising for further work.

7 Conclusion

With this paper we presented a fully-automated schema decryption method leveraging the large number of mapped attribute labels in a given Etl repository. Our work is motivated by observing the need of easy-to-understand schemata during Etl development and maintenance. Cryptic schemata significantly increase the amount of time to understand unfamiliar data, as many readers might have experienced themselves.

We introduced a novel approach for schema decryption to find high-quality decryptions for cryptic attribute labels. Our suggested approach is intended to support and improve manual Etl workflow development and maintencance: An Etl developer is now able to quickly grasp the underlying semantics of data records in cryptic schemata.

We demonstrated that our schema decryption approach provides helpful suggestions for three different real world Etl repositories. An experimental study shows the high average precision of our schema decryption approach.

Acknowledgment

This research was funded by InfoDyn AG; we want to thank Benjamin Böhm and Dieter Radler of InfoDyn AG for supplying real-world data.

References

1. Agrawal, H., Chafle, G., Goyal, S., Mittal, S., Mukherjea, S.: An Enhanced Extract-Transform-Load System for Migrating Data in Telecom Billing. In: Proceedings of the International Conference on Data Engineering (ICDE). Cancún, México (2008)
2. Arasu, A., Chaudhuri, S., Kaushik, R.: Learning String Transformations from Examples. In: Proceedings of the International Conference on Very Large Databases (VLDB). Lyon, France (2009)
3. Baeza-Yates, R.A., Ribeiro-Neto, B.: Modern Information Retrieval. Addison-Wesley, Boston, MA, USA (1999)
4. Bernstein, P.A., Madhavan, J., Rahm, E.: Generic Schema Matching, Ten Years Later. VLDB Journal 4 (2011)
5. Brent, M.R., Cartwright, T.A.: Distributional Regularity and Phonotactic Constraints are Useful for Segmentation. In: Cognition, vol. 61, pp. 93–125. Elsevier Science Publishers (1996)
6. Cafarella, M.J., Halevy, A., Wang, D.Z., Wu, E., Zhang, Y.: WebTables: Exploring the Power of Tables on the Web. In: Proceedings of the International Conference on Very Large Databases (VLDB). Auckland, New Zealand (2008)
7. Cui, Y., Widom, J.: Lineage Tracing for General Data Warehouse Transformations. VLDB Journal 12(1) (2003)
8. Dayal, U., Castellanos, M., Simitsis, A., Wilkinson, K.: Data Integration Flows for Business Intelligence. In: Proceedings of the International Conference on Extending Database Technology (EDBT). Saint Petersburg, Russia (2009)
9. Grünwald, P.: A Minimum Description Length Approach to Grammar Inference. In: Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language Processing, Lecture Notes in Computer Science, vol. 1040, pp. 203–216. Springer Verlag (1996)
10. Levenshtein, V.I.: Binary Codes Capable of Correcting Deletions, Insertions and Reversals. Soviet Physics Doklady. 10(8), 707–710 (1966)
11. Madhavan, J., Bernstein, P.A., Rahm, E.: Generic Schema Matching with Cupid. In: Proceedings of the International Conference on Very Large Databases (VLDB). Rome, Italy (2001)
12. Miller, G.A.: WordNet: A Lexical Database for English. Communications of the ACM 38(11), 39–41 (1995)
13. Poole, J., Chang, D., Tolbert, D.: Common Warehouse Metamodel, Developer's Guide (OMG). Wiley & Sons, Indianapolis, IN, USA (2003)

14. Simitsis, A.: Mapping Conceptual to Logical Models for ETL Processes. In: Proceedings of the International Workshop on Data Warehousing and OLAP (DOLAP). Bremen, Germany (2005)
15. Simitsis, A., Skoutas, D., Castellanos, M.: Natural Language Reporting for ETL Processes. In: Proceeding of the International Workshop on Data Warehousing and OLAP (DOLAP). Napa Valley, CA, USA (2008)
16. Simitsis, A., Vassiliadis, P., Sellis, T.: Optimizing ETL Processes in Data Warehouses. In: Proceedings of the International Conference on Data Engineering (ICDE). Tokyo, Japan (2005)
17. Simitsis, A., Wilkinson, K., Dayal, U., Castellanos, M.: Optimizing ETL workflows for Fault-Tolerance. In: Proceedings of the International Conference on Data Engineering (ICDE). Long Beach, CA, USA (2010)
18. Sorrentino, S., Bergamaschi, S., Gawinecki, M., Po, L.: Schema Normalization for Improving Schema Matching. In: Proceedings of the International Conference on Conceptual Modeling (ER). Gramado, Brazil (2009)
19. TPC Benchmark E. TP Council. (2010), `http://www.tpc.org/tpce/`
20. TPC Benchmark H. TP Council. (2012), `http://www.tpc.org/tpch/`
21. Trujillo, J., Luján-Mora, S.: A UML Based Approach for Modeling ETL Processes in Data Warehouses. In: Proceedings of the International Conference on Conceptual Modeling (ER). Chicago, IL, USA (2003)
22. Vassiliadis, P., Karagiannis, A., Tziovara, V., Simitsis, A.: Towards a Benchmark for ETL Workflows. In: Proceedings of the International Workshop on Quality in Databases (QDB). Vienna, Austria (2007)

ISBN 978-3-86956-201-8
ISSN 1613-5652