Tackling Semantic Shift in Industrial Streaming Data Over Time

Lisa Ehrlinger*†, Christian Lettner*, Johannes Himmelbauer*
*Software Competence Center Hagenberg, Softwarepark 21, 4232 Hagenberg, Austria
†Johannes Kepler University Linz, Altenberger Straße 69, 4040 Linz, Austria
email: lisa.ehrlinger@jku.at, christian.lettner@scch.at, johannes.himmelbauer@scch.at

***Abstract*—Industrial production processes generate huge amounts of streaming data, usually collected by the deployed machines. To allow the analysis of this data (e.g., for process stability monitoring or predictive maintenance), it is necessary that the data streams are of high quality and comparable between machines. A common problem in such scenarios is *semantic shift*. For example, a sensor's weight unit might shift from tons to kilograms after a firmware update and still store the collected values to the same variable. In this paper, we discuss *semantic shift* theoretically and by means of an industrial case study from a production plant in Austria, where several hundred injection molding machines are employed. The data collected by these machines is used to monitor the stability of the production process with machine learning algorithms. In the following, we present and discuss the data preprocessing system we developed for the production plant to handle semantic shift for huge amounts of streaming data.**

Keywords–Semantic Shift; Streaming Data; Data Quality; Process Stability Monitoring; Data Preprocessing.

I. INTRODUCTION

Semantic shift originally describes the evolution of word meaning over time [1]. In this paper, we observe the semantic shift of industrial data streams, where the meaning of variables (also: attributes, features, column names) changes over time. Semantic shift has a negative effect on data analysis and thus, needs to be tackled strategically. We claim that semantic shift can be seen as a Data Quality (DQ) problem, which however, has been little discussed in this context so far.

The awareness for this problem has been raised by different research projects with company partners from industry. In this paper, we specifically describe the use case from one production plant in Austria where injection molding machines are employed to produce plastic products. Such industrial production processes have natural fluctuations due to the physical conditions of the machines, as well as the variability of the used materials [2]. To guarantee the production of high-quality products, it is essential to continuously monitor process signals and to ensure it moves within the specified limits [2]. To support stability monitoring of the production process with statistical measures, we developed L* (pronounce: L-star) a data preprocessing infrastructure that overcomes semantic shift in the process variables.

Therefore, we make the following twofold contribution: (1) a discussion of *semantic shift* as a data quality problem and outlook for future research direction, and (2) a case study from an industrial plant where we deployed the data preprocessing system L*, which tackles the problem of semantic shift in industrial data streams. The system has been deployed at our company partner and is currently under ongoing evaluation.

In Section II, we describe semantic shift as it appears in literature and related work. The case study at the production plant, which highlights the practical relevance of the concept is discussed in Section III. In Section IV, we present L*: a process data preprocessing system where we specifically describe the components that tackle semantic shift. We conclude with an outlook on future work in Section V.

II. SEMANTIC SHIFT – A DATA QUALITY PROBLEM

Historically, *semantic shift* (also: *semantic change*, *semantic drift*) is a term stemming from linguistics and describes the evolution of word meaning over time [1]. According to Bloomfield [1], it can have different triggers and different development. Although used interchangeably in linguistics, we explicitly want to highlight the focus of our research on *shift* (i.e., changes that can be attributed to a specific point in time [3]) in contrast to *drift* (i.e., continuous transformation [3]). The reason is that semantic changes in process data can usually be traced back to specific triggers, e.g., firmware update of a machine, or a change in the production process.

A similar and intensively studied term from Machine Learning (ML) research is *concept drift*, which refers to a drift in the target variable predicted by a ML model [4][5]. Such drifts are usually caused by changes in the hidden context and can be handled with regular updates of the ML model to ensure that the properties of the variable remain stable over time [5]. Klenner and Hahn [6] discuss the problem of semantic shift under the term *concept versioning* for technical standards.

Although there is a lot of research into DQ dimensions (cf. [7]–[10]), there is little discussion on the specific topic "semantic shift". In terms of DQ assessment in ontologies, Guarino and Welty [11] introduce the properties "identity" and "rigity", which are related to the stability of a variable. A similar DQ dimension is *timeliness*, which can be described as "how current data is for the task at hand" [12]. Semantic shift generalizes this dimension since the validity of data depends on the context within it appears (e.g., on the respective machine and the point in time). Thus, we define *semantic shift* in the context of DQ as the circumstance when "the meaning of data evolves depending on contextual factors". Consequently, when these factors are modeled accordingly (e.g., described with rules), it is possible to handle semantic shift even in very complex environments as outlined in the following case study.

III. SEMANTIC SHIFT IN INDUSTRIAL DATA STREAMS

In this section, we motivate the problem of semantic shift with the description of an industrial case study. Due to confidentiality, we are not allowed to publish details of the production process.

Our Austrian manufacturing company partner works in the field of plastics industry with injection molding machines. These machines are tools being able to produce plastic products and multi material-parts by the injection molding process

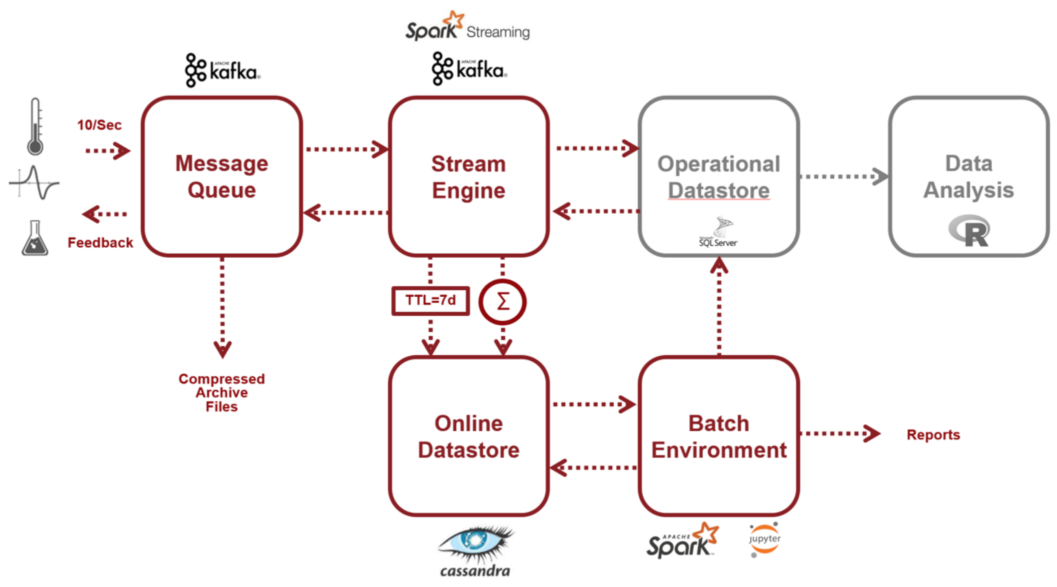

Figure 1. Architecture of the L* Data Infrastructure to Handle Semantic Shift

with a clear focus on mass-production. Injection molding represents a very complex physical-chemical process and there exists a wide variety of situations that can lead to a bad, or at least unstable, condition of a machine, as well as its production process, often ending up in (increased) production rejects. Considering that there are usually more than one hundred machines simultaneously in operation explains the company's aim for a monitoring system that can automatically send alerts where process instabilities that are potentially relevant for production quality show up. The benefit of such a system is manifold. At first, detecting unstable process situations is a prerequisite to actuate countermeasures in order to decrease scrap rates or to avoid machine damage. Moreover, the company operates in the field of massive production of very small pieces and thus complete quality inspection is unfeasible. Time-related knowledge about the process stability for each machine enables us to focus the quality inspection on produced pieces from critical production time periods.

In the collaboration with our company partner, we have worked towards the design of a data-driven solution being able to automatically recognize such critical situations. Thus, it is necessary to note that the machines cyclically supply status values to a machine data acquisition system. These machine statuses are recorded shot by shot and currently stored for several months. By analyzing this data, machine states should be determined by our data-driven solution. The process conditions depend on the following factors, which partly influence each other: machine condition, tool condition, material condition, environmental influences, and processor operating point setting. Our data-driven solution can find diverse known error patterns (ranging from occasionally occurring, isolated critical shots to slowly (in terms of weeks) deteriorating machine conditions, e.g., due to wearing of machine parts) in all machine data. For this purpose, firstly, we analyze in which data sources it is possible to find relevant information from which we can benefit. Based on that information, our solution tries to learn recurrent error patterns. For these tasks, we use different methods including stream data processing, classical machine learning algorithms, outlier detection, robust learning algorithms, and causal discovery.

For use in real production, these developed applications should be easy to integrate in the existing operation system and they should be applicable to as many machines as possible without specific adaptions. Applying certain stability checks to only a few out of many machines is unsatisfactory. Here, we want to point out that all these applications are based on making use of the data that a machine provides and each algorithm expects to be fed with data in a predefined standardized format. Fortunately, the injection molding machines of our customer are almost exclusively from the same vendor; shipped with a standardized data Application Programming Interface (API), which logs data about the injection molding process (in the following indicated with `MD`, short for measurement data) in a system called "MES system". However, there exist different machine types and machine versions. Moreover, machines with identical machine type and version can still provide differences with respect to provided data as different firmware might include also changes in the data schema. Due to the fact that all the machines come from the same vendor, all in all, we found a high level of data consistency; in the sense that variable names remain the same and major changes in newer versions mainly consist in extensions with additional variables. However, there exist cases when certain variables undergo a semantic shift. For example, a variable that represents the measurements of some pressure sensor for one machine might be stored in bar while for another machine or even for the same in a later version the same variable is recorded in millibar. Ignoring such semantic shifts would result in situations when algorithms produce wrong results.

IV. L* System Architecture

The entire data preprocessing system has been implemented on four nodes with Linux Ubuntu 16.04 installed, respectively. Figure 1 illustrates the system architecture of the implementation, where the red components (which are are part of L*) are described in the following subsections. Each component refers to exactly one node in the system infrastructure. The system is designed for linear scalability and therefore employs tools from the Big Data ecosystems. The gray components illustrate the original data analysis infrastructure at the production plant used for process stability monitoring.

A. Data Loading

Initially, process data collected at the injection molding machines is loaded every 10 seconds with a message queue to the stream engine. The message queue has been implemented with Apache Kafka [13] and aims at a robust transmission of huge amount of messages. One advantage of using an asynchronous solution here is that the message queue represents a buffer, which is why messages are not lost even if L* is offline temporarily. We installed Apache Kafka through the Confluent platform [14], which is an event streaming platform that allows to manage and organize data streams (from different sources) for industry applications with high-performance requirements.

B. Online Datastore

The online datastore has been implemented with Apache Cassandra [15], a column-based NoSQL DB that is optimized to manage large amounts of measurement data. Since we deployed the system for our company partner, we selected Cassandra also due to its popularity [16] in comparison to other NoSQL DBs that have similar features. Figure 2 shows the creation statements of the two tables used for storing the process data.

```
create table MDavro (
  jahr int,
  seriennummer int,
  interval int,
  zeitpunkt timestamp,
  value blob,
  primary key((jahr, seriennummer, interval),
      zeitpunkt)
);

create table MD (
  jahr int,
  seriennummer int,
  metric text,
  zeitpunkt timestamp,
  value text,
  primary key((jahr, seriennummer, metric),
      zeitpunkt)
);
```

Figure 2. Cassandra Tables to Store Process Data

C. Data Preprocessing

We used Spark [17] to implement the data preprocessing system, which specifically tackles the problem of semantic shift for our use case. Three different Spark jobs have been implemented: (1) `LoadMD- Avro`, (2) `PreProMDStream`, and (3) `PreProMDBatch`, where the first two are implemented as Spark streaming jobs, and the last one as batch job. Figure 3 displays the three data streams between Cassandra and the streaming platform Confluent.

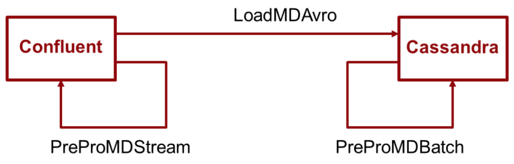

Figure 3. Spark Data Streams

1) Stream Engine: `PreProMDStream` receives data, which is encoded with the Apache Avro [18] data serialization from the machines. The data is decoded and preprocessed according the the defined rules (cf. Table I) to handle semantic shifts. Eventually, the task returns the encoded data back to Confluent.

2) Batch Environment: `PreProMDBatch` basically has the same functionality as `PreProMDStream`, only that it is conducted as Spark batch job. Thus, it requires a defined time interval (start and end point) to load and process the data.

L* supports linear transformations and the application of a time offset (lag). The current version does not allow to represent calculated values, which needs to be done with an external program.

Table I shows an excerpt of preprocessing rules defined to handle semantic shift. Depending on the machine type (`machinetype`), the table maps a machine internal parameter name to a consolidated, meaningful parameter name. In addition, a simple linear transformation (`scale` and `offset`), as well as a time delay (`lag`) may be applied. In the example provided in Table I, a semantic shift has happened on `process_value_3` for machine type `T3`. Starting with machine type `T3`, the production mode is divided into two phases. Further, in machine type `M3`, the temperature values are measured in degrees Fahrenheit. To consolidate these values to `M1` and `M2` values, which are measured in degree Fahrenheit, the values must be multiplied by 1.8 and shifted by 32. The process parameter `Process Temperature1 Previous` makes use of a time delay functionality to provide the previously measured temperature.

D. Performance Metrics

Since L* should be capable for deployment in productive environments, we calculated a few performance metrics to verify its suitability to handle Big Data.

In a test, 28.8 million records have been processed from the MES system, which contained a total of 1,216 million measurement values. This yielded an average of 42.2 measurement values per record. Table II summarizes the processed records or measurement values (short "values") per Spark job. In total, `LoadMDAvro` generated disk storage of 5.01 GB for the Cassandra table `MDavro` and `PreProMDBatch` disk space of 6.49 GB for the table `MD` in the 2.5 weeks time period.

V. Conclusion and Outlook

In this paper, we presented the data preprocessing system L*, which tackles semantic shift in data streams used for process stability monitoring. The rule-based solution is a first attempt to systematically overcome shift in process variables and aligns with the predominant idea how to solve DQ issues in practice (cf. [8]). In the future, we would like to extent

TABLE I. DATA PREPROCESSING DEFINITION

MD_paramname	**process_paramname**	**machinetype**	**scale**	**offset**	**lag**	**datatype**
process_value_1	Mode Stopped	T1, T2, T3	1	0	0	bool
process_value_2	Mode Starting	T1, T2, T3	1	0	0	bool
process_value_3	Mode Production	T1, T2	1	0	0	bool
process_value_4	Product Counter	T1, T2, T3	1	0	0	long
process_value_5	Process Temperature1	T1, T2	1	0	0	float
process_value_6	Process Preasure	T1, T2	1	0	0	float
process_value_3	Mode Production Phase 1	T3	1	0	0	bool
process_value_7	Mode Production Phase 2	T3	1	0	0	bool
process_value_5	Process Temperature1	T3	1.8	32	0	float
process_value_6	Process Temperature2	T3	1.8	32	0	float
process_value_5	Process Temperature1 Previous	T3	1.8	32	1	float
process_value_8	Process Preasure	T3	1	0	0	float

TABLE II. PERFORMANCE METRICS

Spark Data Stream	**Unit**	**Throughput (unit/sec)**	**Storage (byte/unit)**
`LoadMDAvro`	Records	358	182
`PreProMDBatch`	Values	174,343	5.6
`PreProMDStream`	Values	4,816	-

this rule-based system with a semantic solution that takes into account the context (e.g., of the respective machine) since it allows to reach a higher degree of automation.

In our ongoing work, we are going to generalize the problem of semantic shift by investigating DQ assessment for streaming data more broadly. Although there exist many context-independent DQ metrics for batch data sets (cf. [7]), so far, there is little research specifically on data streams. Thus, we would like to extract domain-independent properties that can be applied to measure the DQ of any data stream.

ACKNOWLEDGMENT

The research reported in this paper has been supported by the Austrian Ministry for Transport, Innovation and Technology, the Federal Ministry of Digital and Economic Affairs, and the Province of Upper Austria in the frame of the COMET center SCCH.

REFERENCES

[1] L. Bloomfield, *Language*. Allen & Unwin, 1933.

[2] R. D. Snee, "Crucial Considerations in Monitoring Process Performance and Product Quality," *Pharmaceutical Technology*, vol. 34, no. 10, 2010, pp. 38–40.

[3] Oxford University Press, "Oxford Dictionaries," https://www.lexico.com/?search_filter=dictionary [retrieved: April, 2020].

[4] A. Tsymbal, "The Problem of Concept Drift: Definitions and Related Work," *Computer Science Department, Trinity College Dublin*, vol. 106, no. 2, 2004, p. 58.

[5] G. Widmer and M. Kubat, "Learning in the Presence of Concept Drift and Hidden Contexts," *Machine Learning*, vol. 23, no. 1, 1996, pp. 69–101.

[6] M. Klenner and U. Hahn, "Concept Versioning: A Methodology for Tracking Evolutionary Concept Drift in Dynamic Concept Systems," in *ECAI*, vol. 94. PITMAN, 1994, pp. 473–477.

[7] B. Heinrich, D. Hristova, M. Klier, A. Schiller, and M. Szubartowicz, "Requirements for Data Quality Metrics," *Journal of Data and Information Quality*, vol. 9, no. 2, January 2018, pp. 12:1–12:32.

[8] L. Ehrlinger, E. Rusz, and W. Wöß, "A Survey of Data Quality Measurement and Monitoring Tools," 2019, https://arxiv.org/abs/1907.08138 [retrieved: April, 2020].

[9] R. Y. Wang and D. M. Strong, "Beyond accuracy: What data quality means to data consumers," *Journal of Management Information Systems*, vol. 12, no. 4, 03 1996, pp. 5–33.

[10] M. Scannapieco and T. Catarci, "Data Quality Under a Computer Science Perspective," *Archivi & Computer*, vol. 2, 2002, pp. 1–15.

[11] N. Guarino and C. Welty, "Evaluating Ontological Decisions with OntoClean," *Communications of the ACM*, vol. 45, no. 2, 2002, pp. 61–65.

[12] B. Heinrich and M. Klier, "A Novel Data Quality Metric for Timeliness Considering Supplemental Data," in *Proceedings of the 17th European Conference on Information Systems*. Verona, Italy: Università di Verona, Facoltà di Economia, Departimento de Economia Aziendale, 2009, pp. 2701–2713.

[13] Apache Software Foundation, "Apache Kafka – A Distributed Streaming Platform," Online, 2020, https://kafka.apache.org [retrieved: April, 2020].

[14] Confluent Inc., "Confluent," Online, 2020, https://docs.confluent.io [retrieved: April, 2020].

[15] Apache Software Foundation, "Apache Cassandra," Online, 2020, http://cassandra.apache.org [retrieved: April, 2020].

[16] solid IT gmbh, "DB-Engines Ranking of Wide Column Stores," Online, 2020, https://db-engines.com/en/ranking/wide+column+store [retrieved: April, 2020].

[17] Apache Software Foundation, "Apache Spark," Online, 2020, https://spark.apache.org [retrieved: April, 2020].

[18] Apache Software Foundation, "Apache Avro," Online, 2020, https://avro.apache.org [retrieved: April, 2020].